НЕ ПРАВЕТЕ КАТО ТЯХ, ЗА ДА НЕ СТАВАТЕ ЗА СМЯХ

Уважаеми читатели,

На ваше разположение са шедьоврите на вашите колеги, получени на изпитите по дисциплината „Програмиране“. Тази рубрика е създадена с мисълта, че разумните хора могат да се учат и от грешки (по-добре от чуждите, вместо от своите). Ако някой отговор не ви се стори достатъчно смешен за да попадне в тази рубрика, прочетете отново въпроса, на който той е даден, и не забравяйте, че всички термини се обсъждат от гледна точка на жаргона на дисциплината Програмиране. Всички отговори се публикуват без каквато и да е редакторска намеса от моя страна, както и без връзка с подреждането на въпросите във въпросника за изпит. В редки случаи за по-голяма яснота след отговора **в зелено** е даден моят **пояснителен коментар**.

Включването в рубриката „Не правете като тях, за да не ставате за смях“ (и резил) **е безплатно и анонимно**. **Рецидивите се наказват с обявяване на авторските права** на студента.

За да се включите сред студентите-съавтори на рубриката **не е необходимо** да не сте си взели изпита по „Програмиране“. Нещо повече, дори да сте получили оценка Отличен (6,00) на изпита все пак имате възможност да дадете и два отговора, достойни за включване в тази рубрика. При подобни случай поради невъзможност за рецидив не възможно и да бъдат обявени Вашите авторски права, освен ако авторът лично не се яви и помоли за това.

За да се включите в тази рубрика **не е достатъчно** да не си вземете изпита по „Програмиране“. Редица Ваши колеги дават напълно верни отговори на цял въпрос и дори на 2–3 въпроса. Това за нещастие не може да им помогне да си вземат изпита защото за оценка Среден (3,00) се изискват верни отговори на поне 6 въпроса.

Във фигурни скоби в началото на всеки отговор е изданието, в което той е присъствал. Освен това за да се привлече вниманието на читателя към величието на даден отговор, част от него е курсивирана с получер вариант на шрифта. Тъй като не е запазен архив за 1, 2, 5 и 6 издания, отговорите получени до 3 издание включително са отбелязани като {до 3}, а получените от 5 до 7 издание – като {5–7}.

В настоящето издание са включени отговорите, получени на следните изпити:

1. Редовен изпит за специалности „Електроенергетика и електрообзавеждане“ и „Комуникационна техника и технологии“, проведен на 20 януари 2005 г. в ТК в гр. Смолян.
2. Поправителен изпит за специалности „Електроенергетика и електрообзавеждане“ и „Комуникационна техника и технологии“, проведен на 29 януари 2005 г. в ТК в гр. Смолян.
3. Ликвидационен изпит за специалности „Електроенергетика и електрообзавеждане“ и „Комуникационна техника и технологии“, проведен на 12 септември 2005 г. в ТК в гр. Смолян.
4. Предварителен изпит за специалности „Електроенергетика и електрообзавеждане“ и „Комуникационна техника и технологии“, проведен на 16 декември 2005 г. в ТК в гр. Смолян.
5. Редовен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 24 февруари 2006 г. в ТК в гр. Смолян.
6. Поправителен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 2 март 2006 г. в ТК в гр. Смолян.
7. Ликвидационен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 5 септември 2006 г. в ТК в гр. Смолян.
8. Предварителен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 15 декември 2006 г. в ТК в гр. Смолян.
9. Редовен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 26 януари 2007 г. в ТК в гр. Смолян.

10. Поправителен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 3 февруари 2007 г. в ТК в гр. Смолян.

11. Ликвидационен изпит за специалности „Електроенергетика и електрообзавеждане“, „Комуникационна техника и технологии“ и „Компютърни и комуникационни системи“, проведен на 5 септември 2007 г. в ТК в гр. Смолян.

12. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 24 март 2010 г. във ФМИ в гр. Пловдив.

13. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 29 март 2010 г. във ФМИ в гр. Пловдив.

14. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 21 март 2011 г. във ФМИ в гр. Пловдив.

15. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 7 април 2011 г. във ФМИ в гр. Пловдив.

16. Ликвидационен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 11 септември 2011 г. във ФМИ в гр. Пловдив.

17. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 26 март 2012 г. във ФМИ в гр. Пловдив.

18. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 4 април 2012 г. във ФМИ в гр. Пловдив.

19. Редовен изпит за специалност „Приложна математика“, проведен на 13 декември 2012 г. във ФМИ в гр. Пловдив.

20. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 21 март 2013 г. във ФМИ в гр. Пловдив.

21. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 5 април 2013 г. във ФМИ в гр. Пловдив.

22. Ликвидационен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 10 септември 2013 г. във ФМИ в гр. Пловдив.

23. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 27 март 2014 г. във ФМИ в гр. Пловдив.

24. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 3 април 2014 г. във ФМИ в гр. Пловдив.

25. Ликвидационен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 16 септември 2014 г. във ФМИ в гр. Пловдив.

26. Редовен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 18 март 2015 г. във ФМИ в гр. Пловдив.

27. Поправителен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 2 април 2015 г. във ФМИ в гр. Пловдив.

28. Ликвидационен изпит за специалност „Бизнес информационни технологии“ – редовно обучение, проведен на 10 септември 2015 г. във ФМИ в гр. Пловдив.

Приятно четене (и учене) Ви желае вашият преподавател Вл. Шкуртов.

НАПЪЛНО ГРЕШНИ И СМЕШНИ ОТГОВОРИ

1. 001. Що е бройна система?

1. 1) {5–7} Съвкупност от знакове и правила за броене.
1. 2) {9} Бройна сима е съвкупност от знаци записани под формата на числа – част от тези числа се наричат цифри
1. 3) {15} Бройната система представлява система (съвкупност) от знакове и символи, с които се представят числата.
1. 4) {16} Бройна система е съвкупност от знакове и правила, чрез които се записват цифрите

1. 5) {17} Система от знакове с йерархична подредба. Бройните системи биват два вида: такива при които разположението на знаковете има значение и такива в които няма т. е позиционни и непозиционни
1. 6) {17} Съвкупност от правила и знакове означавани с числа
1. 7) {20} БС е съвкупност от знакове и правила, чрез които записваме **цифрите**
1. 8) {20} Бройната система е съвкупност от символи

2. 002. Що е цифра на БС?

2. 1) {до 3} Това е знак с който се описват стойностите на числото.
2. 2) {до 3} Цифра на БС се означават числата а с числата се означават възлите.
2. 3) {до 3} То се появява в резултат на необходимостта Цифрите не могат да си променят стойността си.
2. 4) {10} Символът от езика на бройната система който има определена стойност
2. 5) {12} В бройните системи има **краен брой величини**, всяко число се записва със знак, наречен цифра
2. 6) {14} Цифрата на БС е символ, който носи някаква стойност
2. 7) {14} Знак
2. 8) {17} Цифрата е знак, чрез който се представя (записва) едно число.
2. 9) {17} **Цифрата** на БС **е знак с** помощта, на **който се изписват** безкраен брой **цифри**.
2. 10) {17} Във всяка бройна система, има краен брой символи които образуват числата.
2. 11) {20} Цифра на бройна система е
2. 12) {20} Във всяка бройна система са определени краен брой възлови числа. Всяко възлово число се представя с отделен знак **нареченъ число**.
2. 13) {23} Всяка бройна система има краен брой възлови числа, всяко число се записва с определен знак, наречен цифра. Обикновено броя на едениците е равен на основата, но е възможно и да не са равни.
2. 14) {26} Цифрите са знакове, с които се изобразяват числата в бройните системи.
2. 15) {26} Цифра е възлово число представено с определен знак.
2. 16) {26} Бройна система е символичен метод за представяне на числата посредством ограничен брой символи, наречени цифри. Съществуват два вида бройни системи непозиционни и позиционни

3. 003. Какви видове БС съществуват?

3. 1) {5–7} адитивна и мултипликативна
3. 2) {15} Непозиционирани и позиционирани
3. 3) {20} Съществуват **двуични** и десетични БС

4. 004. Какви могат да бъдат НПБС и по какво се различават те?

4. 1) {26} Свойството формалност е от съществено значение, защото позволява изпълнителят на един алгоритъм да бъде лишен от разум автомат
4. 2) {26} Римска, йонийски, китайски.
4. 3) {26} Свойството формалност е от съществено значение, защото позволява изпълнителят на един алгоритъм да бъде лишен от разум автомат
4. 4) {26} Римска, йонийски, китайски.
4. 5) {27} Да всички клетки са еднакви и са ги различаваме се номерират с цели числа.
4. 6) {27} Могат да бъдат специализирани инеутрални.

5. 006. Какъв е основният принцип на ПБС?

5. 1) {5–7} В зависимост от позицията в записа.
5. 2) {8} Да представя едно число от *p*-ична ПБС в *n*-ична ПБС, по позиции от дясно наляво Пр: превръщане на двойчно число в шеснайсетично число. Като посредник м/у ПБС се използва двойчната БПС
5. 3) {8} В зависимост от позицията на записа.

5. 4) {9} В зависимост от позицията в записа
5. 5) {9} Преобразуване на цели числа от една бройна система в друга, като понякога за посредник се използва двойчната ПБС
5. 6) {15} Основният принцип на ПБС гласи, че в едно число, всяка цифра трябва да стои на точно дадена позиция (най-просто казано), иначе то се променя.
П-р: 10-та БС е позиционна
$12\underline{3} \neq 1\underline{3}2$

6. 007. Що е основа на ПБС?

6. 1) {9} Множества с които се изменя стойността при която се изменя стойността при всяко съседно изместване
6. 2) {10} В зависимост от позицията на записа
6. 3) {14} Основа на ПБС е **нейната „азбука“** – това са знаците, които участват в определянето на числата в нея. Обикновено **броят на елементите** от основата **съвпада с вида на ПБС**.
6. 4) {14} Когато стойностите на алгоритмичното число променя стойността в зависимост от това каква позиция заема.
6. 5) {14} **Множителят преместен** на съседна позиция на цифрата
6. 6) {14} Основа на ПБС може да е всяко число ≥ 2 **[И числото π ли?]**

7. 008. Формулирайте основната теорема на ПБС.

7. 1) {9} Основната теорема е че представянето на всяко естествено число N се получава само чрез неговите цифри.
7. 2) {12} При промяна на позицията на цифрата **стойността на числото се променя** със стойност каквато е неговата основа.
7. 3) {15} Всяко естествено число може да се представи във вида
$N = a_n p^n + a_{n-1}p^{n-1} + ... + a_1 p + a_0$, където $a_n \neq 0$, $0 \leq a_i \leq p-1 = 0,1,..., n$
7. 4) {15} $N = a_n p^n + a_{n-1}p^{n-1} + ... + a_1 p + a_0$, където $a_n \neq 0$, $0 \leq a_i \leq p-1 = 0,1,..., n$

8. 011. Какво е най-важното следствие от основната теорема на ПБС?

8. 1) {до 3} Множителят с който се изменя стойността на всяка цифра, при преместване на съседна позиция се нарича основа на ПБС.

9. 012. Може ли всяко цяло число да се запише в ПБС с произволна основа? Ако да – как, ако не – защо?

9. 1) {5–7} Всяко цяло число не може да се запише в ПБС с произволна основа защото зависи от позицията на основата (т. е. има еднозначност за разлика от НБС,където числата се представят чрез събиране, умножение).

10. 013. Еднозначен ли е записът на произволно цяло число в ПБС с дадена основа? Ако да – кога, ако не – защо?

10. 1) {14} Не е еднозначен, защото при ПБС не се допуска еднозначност
10. 2) {17} Невъзможно да се запише произволно цяло число, поради липса на еднозначност
10. 3) {20, 26} Записът не е еднозначен. Всеки запис представлява различно (естествено) число
10. 4) {20} Множителят с който се изменя стойността на цифра преместена на съседна позиция се нарича основа на ПБС. Обикновенно броя на цифрите е равен на основата, но е възможно и да не съвпада Не всеки запис представлява различно (естествено) число. **[Де го питаш, що ти отговаря в стил „И аз съм чувàл нещо по въпроса!“???]**
10. 5) {23} Не, всеки запис представлява различно число **[Така формулиран отговорът показва липса на всякакви знания у отговарящия!]**
10. 6) {23} Не е еднозначен. Всеки запис представлява различно естествено число, като основата е всяко цяло число ≥ 2. **[Тези три записа – 002, 02, и 2, очевидно представят три различни числа!]**

10. 7) {26, 26} Не, всеки запис представлява различно (естествено) число.
10. 8) {26} Не, всеки запис е разл. ест. число.
10. 9) {26} ПБС с дадена основа, неможе да има еднозначен запис представлява различно число
10. 10) {26} Не всеки запис е различно (естествено) число.

11. 014. Еднозначен ли е записът на произволно цяло число в НПБС? Защо?

11. 1) {14} Да, според основната теорема.
11. 2) {15} Невъзможно да се запише произволно цяло число, поради липса на еднозначност.
11. 3) {17} Да
11. 4) {17, 17} Невъзможността да се запише произволно цяло число, поради липса на еднозначност.
11. 5) {17} Да, защото стойността на числото не се изменя ако сменим позицията му.
11. 6) {20} Да
11. 7) {23} Да трябва да е еднозначен, защото липсва еднозначност при НПБС.
11. 8) {24} Да защото при НПБС липсва еднозначност.

12. 015. Колко и кои числа могат да се използват като основа на ПБС?

12. 1) {до 3} Могат да се използват цели числа 0, 1, 2, 3, n–1.
12. 2) {5–7} Броят на различните цифри които се използват в зададена ПБС се нарича основа на БС.
12. 3) {9} естествените числа
12. 4) {9} Броя на различните цифри, които се използват в дадена ПБС, се нарича основа на БС
12. 5) {9} Като осн. могат да се използват цели числа (2, 8, 10, 16 **[Числото –2 също е цяло!]**
12. 6) {14} **ПБС биват** два вида – **позиционни и непозиционни**

13. 016. Колко ПБС могат да съществуват и по какво се различават те?

13. 1) {14} адитивна и мултипликативна
13. 2) {15} Адитивни, в които стойността на числото се получава чрез събиране и мултипликативни, в които стойността се получава чрез умножаване
13. 3) {17} Съществуват два вида ПБС
1 адитивни – за да се получи алгоритмичното число цифрите се събират
2 мултипликативни – за да се получи алгоритмичното число се използва и умножение.
13. 4) {23} **Позиционните системи** могат да бъдат 2 вида: **Адитивни и мултипликативни**. При адитивните, **за да се получи даденото алгоритмично число използва действието събиране на стойностите на записа**, а при мултипликативните, за да се получи даденото алгоритмично число се използва действие умножение
13. 5) {23, 26} Могат да съществуват n на брой ПБС, които се различават по тяхната основа n
13. 6) {26} Имат два типа на ПБС (адитивни и мултипликативни едните се събират стойностите чрез стойностите чрез състения други чрез замени

14. 017. Посочете поне две предимства на двоичната ПБС пред останалите ПБС.

14. 1) {до 3} 1 Дали всяко (естествено) число може да се запише в съответната БС (съществуване)
2) дали всеки запис представя различно различно (естествено) число (единственост).
14. 2) {5–7} а) записването на числата става само с два символа. б) Аритметичните действия с двоични числа са изключително прости
14. 3) {14} ① Пести място
② Лесно за използване
14. 4) {14} Икономичност, лесна за изпълнение
14. 5) {14} Предимства на двоичната ПБС: икономична, лесна за преобразуване

14. 6) {14} 1) Използване на основа 2 (записите се състоят само от 1 и 0)
2) Възможни са само две състояния – на истиност и неверност
14. 7) {17} Икономичност и просто техническо записване
14. 8) {17} С краен брой елементи могат да се представят безкраен брой елементи.
14. 9) {17} а) Има само две стойности
б) Икономични са
14. 10) {20} двоичната е по компактна, с по малка основа
14. 11) {26} Двоичната бройна система е разбираема за компютъра, ако се използва например за преносими памети, ще събира по-голям обем информация и ще бъдат по евтини.
14. 12) {26} По бързо изпълнение на задачите и по-голям набор от възможни числа за записване.
14. 13) {27} Голям набор от числови комбинации и бързо извършване на действия (true/false).

15. 019. Кои аритметични операции са в основата на всеки от двата алгоритъма за преобразуване на записа на естествено число от ПБС с една основа в ПБС с друга основа? Коя операция при кой алгоритъм се използва?

15. 1) {9} Операциите са събиране и умножение Заместват се числата в задачата с едната основа а се решават с другата.
15. 2) {14, 14, 21} Умножение при първия алгоритъм, а деление при втория **[Иди че разбери кой е „първия“ и кой – „втория“, алгоритъм?]**
15. 3) {14} При преобразуване на естествено число от ПБС с по-голяма основа към ПБС с по-малка основа се използва деление. При обратния процес се използва умножение.
15. 4) {14} Първи алгоритъм умножение, втори деление.
15. 5) {17, 17, 17} Умножение при първия алгоритъм , деление при втория. **[Иди, че разбери, кой е първи и кой – втори алгоритъм!]**
15. 6) {17} събиране и деление.
15. 7) {19} събиране, умножение, изваждане, деление.
15. 8) {20} Аритметичната операция в основата на алгоритъма за преобразуване на записа на естествено число **от ПБС с по-малка основа към ПБС с по-голяма основа** е умножение, а от ПБС с по-голяма основа към ПБС с по-малка е деление.

16. 020. Коя операция – умножение или деление, ще използва човек при преобразуване на десетичен запис на естествено число в двоичен? Защо?

16. 1) {до 3} При преобразуване на десетичен запис на естествено число в двоичен е операцията – „умножение“.

17. 021. Коя операция – умножение или деление, ще използва компютър при преобразуване на десетичен запис на естествено число в двоичен? Защо?

17. 1) {8} Деление, защото
17. 2) {9, 9, 12} деление
17. 3) {12} Деление, защото числата в двоичен запис се записват само с два знака – 0 и 1.
17. 4) {14} Деление, защото така се преобразува от десетичен в двоичен запис.
17. 5) {14} Деление, защото е по-лесно изпълнимият начин.
17. 6) {14} Операция „Делене“ дели числото на 2 докато не стигне до краен резултат
17. 7) {14} Компютърът ще използва деление, защото човек извършва по-лесно преобразуването от десетичен в двоичен запис
17. 8) {17} Деление, защото е по-лесния начин за преобразуване на естествено число от 10-ична в двоична БС
17. 9) {17} Деление, защото това е по-лесния начин за преобразуване на дадено естествено число от десетична в **двуична** бройна система
17. 10) {17} Деление, защото ~~човек умее да извършва пресмятания в 10-ична ПБС~~ това е по-лесен начин за преобразуване на дадено естествено число от 10-ична в 2-ична бройна система.

17. 11) {17} Деление, защото е по-лесния начин за преобр. на дад. естествено число от дес. в двоична бр. с-ма.
17. 12) {18} Компютърът използва операция делене
17. 13) {18} Деление, защото **човек умее** да извършва пресмятания в десетична бройна система.
17. 14) {20} Деление. Защото такъв е алгоритъма за преобразуване на число от 10-тична в 2-чна.
17. 15) {20} При този вид преобразуване компютърът използва операцията **деление**, защото **тя по-лесно осъществимата** при този вид преобразуване
17. 16) {20} Деление, защото е по лесния начин за преобразуване на десетична в двоична бройна система
17. 17) {20} деление, защото това е по-лесния начин за преобразуване на десетичен запис на дадено естествено число в двоичен
17. 18) {21} **Компютърът ще използва операцията деление** при преобразуване на десетичен запис на естествено число в двоичен.
17. 19) {21} **Човек** би използвал операциата деление, за да може чрез остатъците от делението да сформира двоичния запис на естественото число. **[Само дето във въпроса не се говори за хора!]**
17. 20) {23} Ще **използваме** деление, защото двоичното число се образува от остатъка
17. 21) {23} Ще използва деление, защото чрез остатъците от делението ще се получи числото в двоичен запис.
17. 22) {26} Компютърът би използвал операцията деление, защото чрез остатъците от делението ще сформира двоичния запис на естественото число.
17. 23) {26} Операцията деление, за да може чрез остатъците от делението да сформира двоичния запис на естественото число
17. 24) {26} Компютъра ще използва операция делене, защото използва остатъците от деленето за да покаже резултата. Остатъците от деленето се взимат от долу нагоре.
17. 25) {26} би използвал операцията деление, за да може от остатъците от делението да образува двоичен запис.
17. 26) {26, 26} **Човек** би използвал операциата деление за да може чрез остатъци от делението да сформира двоичния запис на естественото число. **[Само дето във въпроса не се говори за хора!]**

18. 022. Как се получава записът на едно естествено число в ПБС с основа q, когато знаем неговия запис в ПБС с основа p и умеем да извършваме аритметични действия в ПБС с основа p?

18. 1) {10} Цифрите на p-ична БС и нейната основа p са числа a и p, в q-ична БС, заминеме със съответнити числа и извършваме пресмятане с основа q **[Само дето във въпроса изрично е казано, че такива пресмятания не можем да извършваме. Можем да смятаме само с числа, записани в ПБС с основа p!]**
18. 2) {10} Записът на едно естествено число в ПБС с основа q се получава **след извършване на необходимите** аритметични **действия** (умножение, деление) и след това запишем полученото число със основата, в която се изисква да се напише. **[Следва пример, от който по подобие на отговора нищо не може да се разбере.]**
18. 3) {14} $N = Q_n p^n + Q_{n-1} p^{n+1} + \ldots + Q_n p + Q_n$ **[И от този неправилен израз на теб читателю как се получава записът ти стана ясно като през гъста есенна мъгла!]**
18. 4) {14} $N = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0$, където $a_n \neq 0$, $0 \leq a_i \leq p-1$ $i = 0, 1, \ldots n$;
18. 5) {14} $\frac{N}{q} = b_m + q^{m-1} + b_{m-1}.q^{m-2} + \ldots + b_1 + \frac{b_0}{q}$
18. 6) {14, 17, 18, 23, 23, 24, 26} $N = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0$

18. 7) {17} $\frac{N}{q} = b_m q^{m-1} + b_{m-1} q^{m-2} + \ldots + b_1 + \frac{b_0}{q}$ **[Тази формула (наизустена от лекционния слайд) не означава нищо както и предишните, запомнени наизуст грешно. Знаещият студент може с няколко думи да формулира съдържанието на формулата!]**

18. 8) {17} $N = a_n p^n + a_{n-1} p^{n-1} + \ldots + к$

18. 9) {20} $N = a_n p^n + a_{n-1} p^{n-1} + к + a_1 p + a_0$

18. 10) {20} $N = a_n p^n, a_{n-1} p^{n-1}, \ldots, a_0 p$

18. 11) {23} Ако основа p е степен на основа q или обратното и имаме предварително изготвена таблица с числата в двоично бр. система и ги групираме ат по-голяма основа към по-малка чрез делене а от по-малка към по-голяма чрез умножене

18. 12) {26} $N = a_n p^n + a_n p^{-1}\ a_n p$

18. 13) {26, 26} $((\ldots(a_{i-1}:p + a_{-(i-2)}):p) + \ldots + a_{-2}):p + a_{-1}):p$

19. 024. Как се получава записът на едно естествено число в ПБС с основа q, когато знаем неговия запис в ПБС с основа p и умеем да извършваме аритметични действия в ПБС с основа q?

19. 1) {12, 12, 14, 17, 18, 23, 23, 26, 26, 26} $-\frac{N}{q} = b_m q^{m-1} + b_{m-1} q^{m-2} + \ldots + b_1 + \frac{b_0}{q}$

19. 2) {14, 17, 17, 20, 26} $N = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_n p^n + a_0$

19. 3) {17} $N = p + a p^n + a_1 p + a_0$

19. 4) {17} Чрез формулата $\frac{N}{q} = a_n q^{n-1} + a_{n-1} q^{n-2} \ldots + a_1 + \frac{a_0}{q}$

19. 5) {21} Записа на едно естествено число в ПБС с основа q, когато знаем записа му в ПБС с основа p и умеем да извършваме аритметични действия в ПБС с основа q е следния $N = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_n p^n + a_0$

19. 6) {26} $q_n p^n + q_{n-1} p^{n-1} + \ldots + q_1 p$

19. 7) {27} UNIVAC I

20. 025. Как според Вас се записват правилните дроби (числа между 0 и 1) в ПБС?

20. 1) {15} Като полином от вида 1 2 12 където 01за12;

20. 2) {15} Като полиноми **[Какви полиноми?]**

20. 3) {16} Стойността в числителя е по-малка от тази в знаменателя.

21. 026. Има ли някакви опасности при преобразуване на записа на правилна дроб от ПБС с една основа в ПБС с друга основа? Ако да – какви са те и как се решават, ако не – защо?

21. 1) {20} Няма опасност, когато се преобразува на правилна дроб от ПБС с една основа в ПБС с друга основа.

21. 2) {24} Да има опасности, защото новата дроб в ПБС, може да има период от 0 или p–1

21. 3) {26} Дробта в новата ПБС, може да има период от 0 или p–1, следователно има опасности.

21. 4) {26} Въпросът за вярност има два верни отговора Да и не. В нея съдържанието не може да бъде вярно и невярно.

22. 029. Как се записва осмичното число ... в десетична бройна система?

22. 1) {8} (55) с нули и единици

23. 031. Може ли да се опростят алгоритмите за намиране на записа на число в ПБС с основа q, когато знаем неговия запис в ПБС с основа p и p и q са точни степени на трето число r ($p = r^s$, $q = r^t$)? Ако да – как, ако не – защо?

23. 1) {до 3} Съвкупност от краен брой двоични функции, чрез които може да се изрази произволна двоична функция.

24. 032. Що е „съждение“?

24. 1) {17, 17, 17} Мисъл, чрез което се твърди или отрича нещо

24. 2) {20} Това е мисъл в която се твърди или отрича нещо още познато като изречение на естествена психика, **съждението на което може да се оценява** като вярно и невярно

25. 034. Какви видове съждения има?

25. 1) {16} Условни, Преки

25. 2) {26} верни, неверни

26. 033. Що е „двузначна логика“?

26. 1) {14} Има конфликт на значения

26. 2) {17} Логика в която въпросите имат само два възможни отговора. Да или не.

26. 3) {17} „Двузначна логика“ – твърдение, което **може да бъде едновременно вярно и невярно**

26. 4) {20} Двузначна логика:
Мисъл чийто отговор е или „да“ или „не“
пример: 2 + 2 = 4

26. 5) {20} Двузначна логика е логика **която има две значения**

26. 6) {20} логика, която съдържа в себе си 2 значения

26. 7) {23} Логика за въпроса на вярност която може да има само два отговора „да“ и „не“

27. 036. Дайте пример за две съставни съждения.

27. 1) {15} – 2 +2 = 4
– Децата ще ходят на разходка

27. 2) {26} Това е дърво
Времето е топло

28. 037. Дайте пример за две изречения, които не са съждения.

28. 1) {8} Аз се казвам Нели.
Той е мъж.

28. 2) {9} Времето е слънчево. Аз съм ученик.

28. 3) {9} Днес е слънчево → <u>просто</u> Искам да уча информатика ама нямам компютър → <u>сложно</u>

29. 038. Какво представляват логическите отношения?

29. 1) {17} Логическото отношения – това е отношение, което има в него съждение: то може да бъде сложно и просто

30. 039. Дайте пример за три логически отношения.

30. 1) {9}
```
Program XXX;
Var (A, B); Integer
Write(A><B). If A>B A := A–B
     Else B := B–A
  Writeln ('НОД е ', A);
  Readln
End.
```

30. 2) {9} – конюкция; – дезюкция; инверсия
– конюкция – логическо и (умножение)
– дезюкция – логическо или (събиране)
– инверсия – логическо не (отрицание)

31. 041. Какво представлява математическата логика?

31. 1) {10} Булева алгебра **[Авторът едва ли подозира, че отговорът е синоним на посоченото във въпроса.]**

31. 2) {10} Начина по който се извършват математическите действия/пресмятания.

32. 042. Кой формализира класическата аристотелева логика?

32. 1) {17} Класическата аристотелева логика е формализирана от Чарлз Бери

32. 2) {17} Класическата аристотелева логика е формализирана от английският професор по математика Чарлз Бебидж

33. 043. Коя функция наричаме двоична (логическа)?

33. 1) {до 3} Тъй като съжденията са две променливи то отношенията са функции на такива аргументи.

33. 2) {до 3} отрицание $\neg x = 1 - x$
конюнция x & y и min(x y)
Дизюкция x + y = max (x y)
Всяка аритметична операция между числа записвани в двоична ПБС може да се опише чрез операции между техните двоични цифри.

33. 3) {4} Двоични функции са тези функции с променливи 2 стойности.

33. 4) {5–7} Която има само две стойности 0 и 1.

33. 5) {10} Двоична е тази ф-я, която може да има само две решения „да“ или „не“ „вярно“ или „невярно“

33. 6) {10} Функция с дефиниционна област и област на стойностите.

33. 7) {12} Всяка функция $f: D^n \rightarrow D$ с дефиниционна област D^n и област на стойностите D ($n \in N$), наричаме двоична, логическа или още булева. **[Без обяснение що за звяр е D отговорът е абсолютно грешен!]**

33. 8) {12} Двоична функция наричаме всяка функция от вида $Д^n \rightarrow Д$, където $Д^n = Д.Д. \ldots Д$ е множеството допустими стойности към Д функцията.

33. 9) {14} Двоична функция е функция която приема само 2 стойности – вярно (истина), грешно (лъжа) – съответно тези стойности се представят с ,1 и 0

33. 10) {14} Двоична (логическа) функция е такава при която са възможни само два изхода – ДА и НЕ и се нарича двоична защото се използват само 0, 1. 1 – True и 0 – False. Задава се условие, което може да бъде истина или лъжа.

33. 11) {14} Всяка функция f D_n > D с D0 D_n и областта на стойностите D се нарича логическа функция на n аргумента.

33. 12) {17} Всяка функция ~~D^n~~ с дефиниционна област Dn и област на стойностите D, се нарича логическа.

33. 13) {17} Всяка функция $f: D^n \rightarrow D$ с дефиниционна област D^n и област на стойностите D, се нарича логическа функция на n аргумента **[Без обяснение що за звяр е D отговорът е абсолютно грешен!]**

33. 14) {17} Общата логическа функция е И–НЕТТЛ– елементите с прост инвертор имат голямо бързодействие от ОТЛ.

33. 15) {18} Всяка $f: D^n \rightarrow D$ функция, където D е множество от стойности.

33. 16) {20, 21, 21, 26} Всяка функция $f: D^n \rightarrow D$ с дефиниционна област D^n и област на стойностите D, се нарича логическа (двоична) функция на n аргумента. **[Без обяснение що за звяр е D отговорът е абсолютно грешен!]**

33. 17) {20} Всяка функция $f: Dn > D$ с дефиниционна област Dn и област на стойностите D, се нарича логическа (двоична) функция на n аргумента.

33. 18) {23} Всяка функция: $f: D^n \rightarrow D$ дефиниционна област D^n и на стойностите D, се нарича логическа (двуична) функция на n аргумента. **[Без обяснение що за звяр е D отговорът е абсолютно грешен!]**

33. 19) {23} „*f: $D^n \rightarrow D$*“ с дефиниционна област „D^n“ и област на стойностите „D“, нарича се логическа функция на „n“ аргумента. **[Без обяснение що за звяр е D отговорът е абсолютно грешен!]**
33. 20) {26} Функция с дефиниционна област и област са стойностите
33. 21) {27} Двоична функция наричаме логическото ИЛИ“.

34. 044. Как става определянето на една логическа функция?

34. 1) {до 3} Всяка функция *f*: D^n – *D* **с дефиниционна област** за някое естествено число **n** област на стойностите D се нарича логическа функция.
34. 2) {8} Чрез логически отрицание, и, или,
34. 3) {8} – Определянето на логически ф-ции става по два начина:
– двусмислена
– по общо приетия случай
И имаме две литерални операции – вярно или грешно.
34. 4) {9} По нейната дефиниционна област, и областта на нейните стойности.
34. 5) {9} чрез дефиниране **[Дефиниране и определяне не бяха ли синоними?]**
34. 6) {12} Има **2 възможни решения** и се избира едно от тях в зависимост от условието
34. 7) {14} Като се изгради съждени при което резултата е винаги да или не.
34. 8) {17} Определянето на една логическа функция става чрез логическите отношения – „и“, „или“, „дали“, „но“
34. 9) {20} Чрез дефиниране на променлива от даден тип.
34. 10) {20} За да бъде определена 1 логическа ф-я трябва да се знаят 2 неща: ① **какви стойности да приема алгоритъма** и на тези стойности какви функции отговарят. ② чрез таблицата на истинността.
34. 11) {23} При Определянето на една логическа функция трябва да се знаят 2 неща: какви стойности **може да приема документът** и на тези стойности на аргумента какви стойности отговарят
34. 12) {23} За да бъде определена една функция първо трябва да знаем какви стойности ще приема.
34. 13) {23} За да бъде определена една ф-я първо трябва да се знаят 2 неща: какви стойности може да приеме аргументът и на тези стойности на аргумента какви стойности на ф-та отговарят
34. 14) {26} Една лог. ф-ця се определя, когато се знаят какви4 стойности може да приема аргументът и каква е таблицата за истинност
34. 15) {26} Като се определи истинността на простите съждения, които я изграждат.

35. 045. Колко редици от нули и единици с дължина n съществуват?

35. 1) {20} Съществуват n на брой редици от нули и единици, като n = 8 бита
35. 2) {21} Безброй много.
35. 3) {22} Безброй.
35. 4) {23} Съществуват N редици
35. 5) {23} три
35. 6) {26, 26} Редици от n нули и единици – 2n броя съществуват
35. 7) {26} Редици от n нули и единици – 21 броя съществуват
35. 8) {2} EDSAC

36. 046. Колко е броят на логическите функции с n аргумента?

36. 1) {до 3} **Броят** на логическите функции **са 6**.
1. Логически съждения
2. Двоични функции
3. Функции на една променлива
4. Функции на две променливи

5. Пълни системи от функции
6. Приложение на логически функции.

36. 2) {до 3} Ако имаме две променливи 0 и 1 то броят на техните функции с *n* аргумента е $\mathbf{2^n}$.
36. 3) {12} Броят на логическите функции с n аргумента е 2^n
36. 4) {14, 14, 17, 20, 20} 2^n
36. 5) {14} Скоби, величини, знаци за сравняване знаци за операции (+, –, *, /)
36. 6) {14, 14} Броят на логическите функции с n аргумента е 2^n.
36. 7) {17} Броя логически функции е 2^n
36. 8) {17} два броя: 2^n
36. 9) {17, 20, 26} 2^{2n}
36. 10) {18} Всеки аргумент на дадена двоична функция може да приеме стойност 0 или 1, независимо от останалите аргументи. Следователно възможните **комбинации от стойности на n аргумента** ще е 2^{2n}
36. 11) {20} Всеки аргумент на дадена двоична функция може да приеме стойност 0 или 1, независимо от останалите аргументи. Поради това броят на възможните **комбинации от стойности на n аргумента** ще е 2_{2n}.
36. 12) {20} Неопределен
36. 13) {21} Всеки аргумент на **двуична** функция приема стойност n независимо от стойностите на другите аргументи. Броят на възможните комбинации от стойности на n е 2^{2n}.
36. 14) {23} Всеки аргумент на една двоична функция приема стойност 0 или 1 независимо от стойностите на останалите аргументи. Поради това броят на възможните комбинации от стойности на аргумента ще бъде 2^{2n}
36. 15) {23} Броят на възможните комбинации от стойностите на n аргумента е 2^{2n}
36. 16) {23} Броят на логическите функции с n аргумента е равен на 2^{2n}.
36. 17) {23, 26} Всеки аргумент на една двоична (логическа) функция приема стойност 0 или 1 независимо от стойностите на останалите аргументи. Поради това броят на възможните комбинации от стойности на n аргумента ще бъде 2^{2n}
36. 18) {26, 26} Всеки аргумент на една двоична (логическа) функция приема стойност 0 или 1

37. 047. Как става доказателството на равенства, в които участват само двоични функции? Защо?

37. 1) {14} $f(x_1, x_2, \ , x_n) = g(x_1, x_2, \ , x_n)$

x_1	x_2		x_n
0	0		0

37. 2) {14, 17, 17, 17, 17, 23} $F(x_1, x_2, \ , x_n) = G(x_1, x_2, \ , x_n)$ – изчисляваме поотделно и сравняваме.
37. 3) {20} Чрез сравняване на функционалните стойности на двете функции
37. 4) {20} $f(x_1, x_2, \ , x_n) = g(x_1, x_2, \ , x_n)$ Решаване поотделно и сравняване
37. 5) {20} Като направите проверка, защото така се проверяват.

38. 049. Определете таблично функцията конюнкция (логическо „И“).

38. 1) {9} Функцията конюкция на (логическоИ) представлява логическо умножение.
38. 2) {14}

x	y	$x \wedge y$
0	0	0
0	1	1
1	0	1

1	1	1

38. 3) {17}

x	y	x∧y
1	0	1
0	1	1
1	1	0
0	0	1

38. 4) {17}

x	y	x∧y
0	1	0
0	1	0
1	0	0
1	0	1

38. 5) {17}

x	y	x и y
0	0	1
0	1	0
1	0	0
1	1	1

38. 6) {20}

x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1

38. 7) {26} като

39. 050. Определете таблично функцията дизюнкция (логическо „ИЛИ“).

39. 1) {до 3}

x	y	x∧y
0	0	0
0	1	0
1	0	0
1	1	1

39. 2) {до 3} 1. синоними: логическо (включващо) ИЛИ, логическо събиране
2. Означаване x∨y, x+y.

39. 3) {9} Логическо събиране.

39. 4) {14}

x	y	
1	0	1
0	1	1
0	0	0
1	1	0

→ Логическото „или“ в този случай само 1 от двете е възможен. Никоило и2та.

39. 5) {14}

x	y	x∧y
0	0	0
0	1	1
1	0	1
1	1	0

40. 051. Определете таблично функцията сума по модул 2 (логическо изключващо „ИЛИ“).

40. 1) {до 3}

x	*y*	*x*⊕*y*	
0		0	0
0	1	1	
1	0	1	
1	1	1.	

40. 2) {до 3}

x	*y*	*x*⊕*y*
0	0	1
0	1	1
1	0	1
1	1	0.

40. 3) {5–7} **Невалидност**.

40. 4) {9} Неравнозначност

40. 5) {14}

x	y	x∧y
0	0	0
0	1	0
1	0	0
1	1	1

40. 6) {20}

x	y	x∨y
0	0	0
0	1	1
1	0	1
1	1	1

41. 052. Какви са законите на де Морган?

41. 1) {9} конюнкция над дезюнкция
дезюнкция над конюнкция

41. 2) {10} свързват три функции

41. 3) {14} $\overline{x \wedge y} = \bar{x} \wedge \bar{y}$
$\overline{x \vee y} = \bar{x} \vee \bar{y}$

41. 4) {14} $\overline{x \vee y} = x \wedge y$
$\overline{x \wedge y} = x \vee y$

41. 5) {20, 21} Законите на Август де Морган (1806–1971) са връзката между трите функции: конюнкция дизюнкция и отрицание и са следните: [горна черта за x и y] x ∨ y = x ^ y [горна черта за x и y отделно]: [горна черта за x и y] x ^ y = x ∨ y горна черта за x и y отделно].

41. 6) {20, 23, 23, 26} Законите на Август де Морган (1806–1971) са връзката между трите функции: конюнкция дизюнкция и отрицание

41. 7) {23} Законите на Август де Морган (1806–1971) са връзката между трите функции: конюнкция дизюнкция и отрицание и са следните (горна черта за x и y) x ∨ y = й ^ y (горна черта за x и y отделно), (горна черта за x и y) x ^ y = x ∨ y (горна черта за x и y отделно)

41. 8) {26} Случаен експеримент, фазово пространство, обединение, сечение.
Събитие пораждащо друго събитие.

41. 9) {26} Те са връзката между трите функции: конюнкция дизюнкция и отрицание. Те са следните: (горна черта за x и y y x ∨ y = x ^ y [горна черта за x и y отделна] [горна черта за x и y] и x ^ y = x ∨ y [горна черта за x и y отделно]

41. 10) {27} Законите на Де Морган са връзка между трите функции: Конюнкция Дизюнкция и отрицание и са следните: (горна черта за x и y) (горна черта за x и y отделно)

42. 053. Колко и кои са дистрибутивните закони за дизюнкция и конюнкция?

42. 1) {15} Два са дистрибутивните закони за дизюнкция и конюнкция $\overline{x \wedge y} = \bar{x} \vee \bar{y}$ $\overline{x \vee y} = \bar{x} \wedge \bar{y}$

43. 054. Определете таблично функцията еквивалентност.

43. 1) {14}

x	y	x=y
0	0	0
0	1	0
1	0	0
1	1	1

43. 2) {16}

x	y	x^y	x∨y
0	0	1	0
0	1	0	1
1	0	0	0

1	1	1	0

43. 3) {16}

x	y	x	y	x y
0	0	1	0	
0	1	0	1	
1	0	0	1	
1	1	1	0	

43. 4) {21}

x	y	x ≡ y	x ⊕ y (+)
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

[Прекрасна таблица!!]

44. 056. Определете таблично функцията щрих на Шефер (НЕ–И).

44. 1) {15}

x	y	x y
0	0	1
1	0	0
0	1	0
1	1	1

44. 2) {15}

x	y	x≡y	x⊕y
0	0	1	0
0	1	0	1
1	0	0	0
1	1	1	0

44. 3) {20}

0	0	0
0	1	0
1	0	0
0	1	1

45. 057. Що е функционално пълна система от двоични функции?

45. 1) {4} $D^n = D \times D \times \ldots \times D$

$\underbrace{\qquad\qquad}_{n \text{ пъти}}$

45. 2) {5–7} **Всяка функция в машинния език** има поредица от нули и единици, като нулата има стойност лъжа (false), а единицата истина (true) като така се представя информацията в машинния език. Като вземем две функции и редуваме възможните комбинации от нули и единици се получава пълна система от двоични функции. Пример : функция A(1,0)и B(0,1). И така се наблюдават функции като конюнкция, дизюнкция и др.

45. 3) {9} **Това се четири пълни функции** от двоичната функция
- конюкция и отрицание
- дизюнкция и отрицание
- модул 2
- дизюнкция 1 и сума по модул 2

45. 4) {9} **Един алгоритъм** представя структура от операции, които се извършват в определен последователност. Конюнкция, дизюнкция и отрицание образуват пълна система

45. 5) {12} Функционално пълна **система от двоични функции**, чрез които **можем да предс-тавим други** такива **функции**

45. 6) {17} тя е система от малко функции, които работят с 2-ична БС и **могат да изпълнят** всяка произволна функция **[И функцията синус ли?]**

45. 7) {17} Функциите дизюнкция, конюнкция и отрицание образуват функционално пълна система.

45. 8) {23} Съвкупност от множество двоични функции, където всяка една произволна двоична функция може да бъде изразена.

45. 9) {23} Съвкупност от краен брой двоични функции чрез, които се определят **числата**. Произволна двоична функция се определят **числата**.

45. 10) {23} Функциите конюнкция, дизюнкция и отрицание образуват една функционално пълна система.

45. 11) {26} Функционално пълна система – това е система, в която от формулирани двоични функции могат да се образуват произволни двоични функции.

46. 059. Посочете примери на поне 4 функционално пълни системи от двоични функции.

46. 1) {17} Йонийска, старобългарска, старогръцка, римска.

47. 060. Могат ли двоичните функции да се определят по аритметичен път? Ако не – защо, ако да – дайте пример за определяне на функциите конюнкция, дизюнкция и отрицание по аритметичен път.

47. 1) {16} **Двуичните** функции могат да се определят по аритметичен път.
конюнкция, дизюнкция, отрицание

x	x'	y
0	0	0
0	1	1

x	x'	y
0	0	1
0	1	0

x	x'	y
0	0	1
0	1	0

1	0	1	1	0	0	1	0	0
1	1	1	1	1	1	1	1	0

[Некоректният пример показва, че студентът си няма и хал хабер понятие що е аритметика и що е логика!]

48. 062. Какво е най-важното следствие от съществуването на пълни системи от двоични функции за създаването на изчислителни машини?

48. 1) {9, 9} Може да се изрази произволна двоична функция.
48. 2) {12} Най-важното следствие от съществуването на пълни система от двоични функции е, че благодарение на тях става изработването на логически винтили
48. 3) {12} Чрез тях може да изразим **произволна двоична бройна система**.
48. 4) {17} Чрез пълна система от двоични функции може да се изрази произволна двоична функция
48. 5) {17} Това, че може да се изрази произволна двоична функция чрез пълна система.
48. 6) {18} Функциите конюнкция, **дифункция** и отрицание образуват пълна система – щрих на Шефер, стрелка на Пирс.
48. 7) {20, 20, 26, 26} Важно следствие е че можем да изразим произволна двоична функция чрез използването на функционално пълна система.
48. 8) {20} Може да се изрази произволна двоична функция чрез използването на функционално пълна система.
48. 9) {20} Важното следствие е, че можем да изразим произволна десетична функция чрез използването на функционално пълна система
48. 10) {20} По-бързата и по-ефикасната работа
48. 11) {26} Важно следствие е, че можем да изразим произволна двоична функция чрез използване на функционално пълна система.
48. 12) {27} Изчислителните машини могат да извършват прости аритметични действия стига да са в двоична бройна система.

49. 063. Определете интуитивно понятието „алгоритъм“.

49. 1) {19} Понятието алгоритъм интуитивно се свързва с римският принцип „разделяй и владей“. Той се прилага успешно ако искаме да изпълним или да обясним някои сложно действие.
49. 2) {19} разделяй и владей
49. 3) {21} Алгоритъма е начин да се представи сложно действие, чрез по-прости с цел полесно изпълнение.
49. 4) {27} Начин за извършване на дадено действие стъпка по стъпка

50. 063. Определете интуитивно понятието „алгоритъм“.

50. 1) {до 3} Алгоритъм е един вход и един изход. Това позволява той да бъде разглеждан като ново, по сложно действие (с определено начало и край) и следователно да бъде използван като част от описанието на друг алгоритъм. В такива случаи първият алгоритъм се нарича подалгоритъм на втория.
50. 2) {до 3} точно определен ред който използваме за изпълнението на една програма.
50. 3) {14} Понятието „алгоритъм“ назовава последователното **изпълнение на елементарни действия** с цел достигане на краен резултат.
50. 4) {17} Алгоритъм е правило по което се извършват дадена група от действия. Пример за алгоритъм е една рецепта.

Как да приготвим кафе. 1. Подготвяме необходимите съдове. 2 Слагаме определените дози кафе и вода. 3. Поставяме на котлона. 4. Изчакваме да се свари кафето. 5. Прецеждаме 6. Поднасяме

51. 064. Посочете понятия, най-близки до интуитивната представа за алгоритъм.

51. 1) {до 3} 1 Да може да извършва елементарните действия
2 Да изпълнява тези действия в посочената от алгоритъма последователност.

51. 2) {16, 21} Алгоритмът представя сложно действие чрез редица от достатъчно прости действия, изпълняват и може да се извърши в последователни стъпки и без допълнителни обяснения **[Колко жалко е, че въпросът не е „Що за звяр е понятието алгоритъм“, а „Кое понятие е близко до него“?]**

51. 3) {17} Може да се изрази произволна двоична функция чрез използването на функционално пълна система.

51. 4) {20} Алгоритъма е сложно действие с начални входни данни и крайни изходни данни, представено чрез редица от достатъчно прости действия, за изпълнението на които не е необходимо допълнително обяснение

51. 5) {20} представлява сложно действие чрез редица от достатъчно прости действия, които изпълняващият може да се извърши в последователни стъпки и без допълнителни обяснения **[Колко жалко е, че въпросът не е „Що за звяр е понятието алгоритъм“, а „Кое понятие е близко до него“?]**

51. 6) {23, 23, 23, 26, 26} Алгоритмът представя сложно действие чрез редица от достатъчно прости действия, които изпълняващят ги може да ги извърши в последователни стъпки и без допълнителни обяснения **[Колко жалко е, че въпросът не е „Що за звяр е понятието алгоритъм“, а „Кое понятие е близко до него“?]**

51. 7) {26} Алгоритъм – метод начин за извършване на дадено действие

51. 8) {27} Алгоритъмът е изпълнение на последователност от прости действия, които нямат нужда от допълнително пояснение относно изпълнението им

52. 065. Що е „елементарно действие“ в теорията на алгоритмите?

52. 1) {до 3} Един алгоритъм представя дадено сложно действие чрез редица от достатъчно прости (елементарни) действия.

52. 2) {5–7} Съвкупност от елементарни прости действия, образуващи сложни действия, образуващи алгоритъм.

52. 3) {8} събират и изваждат

52. 4) {12} Всяко действие, което води до изпълнение на следващото. По принцип програмата изпълнява действията едно след друго, но и елементарното действие оказва кое точно това действие.

52. 5) {14} Всеки алгоритъм е сложно действие представено от поредица от достатъчно на брой (краен) по-прости (елементарни) действия които се изпълняват от изпълнителя за да се получи търсения резултат.

52. 6) {14} Елементарното действие съставя един алгоритъм, като **изпълнителят** на алгоритъма **не е длъжен да дава допълнителни указания**. Елементарното действие е **поредица от прости действия** в алгоритъма.

52. 7) {14} Алгоритмите се състоят от последователност от елементарни действия. **Това са прости действия** поредицата от които води до резултат.

52. 8) {14} „Елементарно действие“ е извършване на **просто действие, обяснено с** отделни **алгоритми**

52. 9) {20} Елементарно действие е действие което е недвусмислено.

52. 10) {20} Действие, което **изпълняващият може да изпълни в последователни стъпки и допълнителни указания**. Наборът от допълнителни елем. действия определя възможностите на изпълнителя

52. 11) {21} Събиране и изваждане или друг.

52. 12) {23} Алгоритмите представляват поредица от елементарни действия, които би трябвало да осигурят на човека или машината, които изпълняват алгоритъма да премине през всички стъпки и да стигне до решението без да му е нужно да навлиза в смисъла на самият алгоритъм.

52. 13) {26} Елементарно действие е това действие, което се извършва по последователни стъпки
52. 14) {26} Действие, което се извършва еднозначно.

53. 066. Какви лица са свързани с понятието алгоритъм?

53. 1) {5–7} Линейна последователност; разклоняване по поне 2 пъти;
53. 2) {14} оператор, променлива
53. 3) {15} Алгоритъм – това са прости действия които имат начало и завършек. Човек може сам да си състави алгоритъм след това да го следва и след като изпълни всички части да стигне до решението ако разбира действията на алгоритъма.
53. 4) {23} Прости (елементарни), които не съдържат като част от своя запис други оператори

54. 067. Какво извършва съставителят на един алгоритъм?

54. 1) {9} **Изпълнява** предписаните елементарни действия при конкретни начални условия.
54. 2) {12} **оптимизира**, описва, **тества** алгоритъма
54. 3) {14} Съставителят на алгоритъм задава на програмата как да се процедира, тоест какво да следва ~~след всяко продходно действие~~ и реда на действие
54. 4) {14} Съставителят определя условието на работа на алгоритъма
54. 5) {20} подрежда изчислителният ред
54. 6) {20} Съставителят на един алгоритъм извършва заделяне на клетки в оперативна памет за обработката на променливи данни, за оператори за пресвояване

55. 068. Какво извършва изпълнителят на един алгоритъм?

55. 1) {5–7} **Усмисля** пътя на алгоритъма.
55. 2) {5–7} **Подготвя алгоритъма** като подбира елементарните действия и реда на тяхното изпълнение
55. 3) {9} Кара алгоритъма да работи
55. 4) {14} Кара компютърът да върши работа.
55. 5) {20} Изпълнителят **изпълнява написаната програма** по реда в който е написана

56. 069. Какво извършва потребителят на един алгоритъм?

56. 1) {до 3} Алгоритъмът е средство да се възложи определена дейност на изпълнителя.
56. 2) {9} Извършва последователност от прости действия.
56. 3) {14} готовия продукт.
56. 4) {15} Въвежда данните
56. 5) {17} **изпълнява** последователността от прости стъпки и ползва крайния резултат.
56. 6) {18} Ако са два алгоритъм – първият алгоритъм – заменя цифрите и основата със съответните числа и извършва пресмятане в БС с основа q, а при вторият използваме схемата на Хорнер да изчисляваме стойността на полиномите **[Читателят неминуемо ще се съгласи, че това е повече от брилянтен отговор на зададения въпрос!]**
56. 7) {20} Потребителят **изпълнява** предписаните елементарни действия при конкретни начални условия.
56. 8) {23} Използва и се служи с посочените алгоритми
56. 9) {27} Потребител вече използва готовия алгоритъм за да си достигне нима си например написана без грешки програма след компилация е приготвен алгоритъм за използване на потребител

57. 070. Какво определя възможностите на един абстрактен изпълнител?

57. 1) {8} той изпълнява и следва посочения от съставителя алгоритъм
57. 2) {9} Да представя абстрактни модели.
57. 3) {10} Да създава абстрактни модели
57. 4) {10} Да се представят абстрактни модели

57. 5) {12} Абстрактният изпълнител изпълнява действията на алгоритъма, които са точно определени, последователни и ясни, затова при наличието на входни данни изпълнителят еднозначно може да определи изходните без да е нужно да разбира цялостния смисъл на алгоритъма. Поради това той може да бъде и автомат
57. 6) {14} Възможностите на ЕП.
57. 7) {17} позволените действия, които са определени от създателя
57. 8) {18} задава изпълнението на алгоритъма и ползва крайния резултат
57. 9) {20} Оперативната памет и бързината на ЦП.
57. 10) {20} Допустимите стойности, функции и операции
57. 11) {23} входни данни
57. 12) {26} Един абстрактен изпълнител определя възможностите на елементарни действиа.

58. 071. Определете детайлно и неформално понятието „алгоритъм“.

58. 1) {5–7} команди, план
58. 2) {5–7} понятието интуитивно се свързва с понятието множество от правила (инструкции и команди)
58. 3) {9} Съвкупност от знакове и правила, чрез които се записват числа
58. 4) {9} Последователност от правила за изпълнение на дадена програма

59. 072. Коя наука първа формулира алгоритми?

59. 1) {до 3} Алгоритъма е точно предписание който задава изчислителен процес (наричан алгоритмичен) започващ от произволни начални данни.
59. 2) {26} Криптографията

60. 073. Дайте пример за име на популярен алгоритъм и задачата, която се решава чрез него.

60. 1) {10} вход: 1 чаена лъжица кафе, 1 чаша вода, захар на вкус изход: чаша горещо кафе
60. 2) {12} Алгоритъм за приготвяне на кафе
    1. Приготвяме се за работа
    2. Смесваме всички нужни съставки
    3. Поставяме на котлона
    4. Изчакваме до кипване
    5. Поднасяме горещо
    6. Приключваме работа
60. 3) {14} Направа на кафе, Вход на задачата: кафе, чаша, вода Изход: Чаша кафе
60. 4) {20} Алгоритъма не е в клетка

61. 074. Какъв е произходът на думата „алгоритъм“?

61. 1) {9} Арабски **[Но от какво точно идва?]**
61. 2) {9} Произхода на думата „алгоритъм“ е **Гръцки**
61. 3) {14} **Произхода** на думата „Алгоритъм“ **е** арабски **произход**.
    От арабския математик **Мухамед** **[И какво общо има между името на пророка и алгоритмите?]**

62. 075. Какво представлява понятието „подалгоритъм“?

62. 1) {5–7, 5–7} подалгоритъм на втория
62. 2) {9} Подалгоритъм означава подалгоритъм на втория.
62. 3) {9} Даден Алгоритъм има вход и изход. Това води до усложняване на дадения алгоритъм. За да може да бъде изпълнен дадения алгоритъм има подалгоритъм. Предимството на подалгоритъма е в това че той опростява дадената задача.
62. 4) {12} Действия обособени като явен алгоритъм, използвани най-често при повтарящи се действия и са част от един общ алгоритъм
62. 5) {14} Подалгоритъмът е част от главния алгоритъм. Той се състои от опростени действия.

62. 6) {14} Част от алгоритъма
62. 7) {17} Подалгоритъмът е алгоритъм за изпълнение на определена и често повтаряща се, той може да съкрати до голяма степен съставянето на избрания (дадения) алгоритъм и да съкрати записването му.
62. 8) {18} Подалгоритъмът е алгоритъм, който
62. 9) {21} Подалгоритъм е **по-малък алгоритъм**, който е включен като **част от един голям алгоритъм**.
62. 10) {26} Подалгоритмите са последователност от елементарни действия на изпълнителят, които са обяснени в отделни алгоритми.

63. 076. Каква е ползата от подалгоритмите?

63. 1) {14} Улесняване на крайният потребител
63. 2) {16} Псевдо елементарно действие на изпълнителя които са му обяснени с отделни алгоритми **[Колко жалко е че въпросът не е „Що за звяр е то?“, а „Каква е файдата от него?“!]**
63. 3) {17} Подалгоритмите са псевдо-алгоритми, които се използват за да се улесни работата на основнит алгоритъм
63. 4) {17} Под алгоритмите са за дообяснение на алгоритмите. Лесни са за изпълнение.
63. 5) {27} Подалгоритмите (поддействия) могат да коригират главния алгоритъм

64. 079. Какво е компютърна програма?

64. 1) {до 3} Система от правила и знаци чрез които записваме информация и задаваме дани.
64. 2) {5–7, 5–7, 5–7, 5–7, 5–7, 5–7} Алгоритъм във форма и вид
64. 3) {5–7} Това което кара един компютър да върши полезна работа. (алгоритам във форма и вид.
64. 4) {9} Последователност от правила
64. 5) {12, 14} Компютърна програма е това, което кара компютъра да върши полезна работа.
64. 6) {14} Компютърна програма е действие или поредица от действия изпълнението на които води до постигането на полезен резултат.
64. 7) {14} Това, което кара компютъра да върши полезна работа.
64. 8) {14} Компютърна програма е **поредица от алгоритми**
64. 9) {17} Алгоритъм представен на компютърен език.
64. 10) {17} Компютърна програма е изрази с логическа последователност, които включва в себе си алгоритми, скоби и оператори чрез които се задава дадена операция на компютъра.
64. 11) {20} Алгоритъм във форма и вид, който **може да се чете от програмата**.

65. 080. Какви са параметрите на един алгоритъм (поне 5)?

65. 1) {14} Това са променливите. Фактически, указатели, формални
65. 2) {14} Параметрите на един алгоритъм са:
- – входна база от данни
- – цел на алгоритъма (краен резултат)
- – подалгоритми
- – прости действия, подредени в последователен ред
- – изходна база от данни

65. 3) {15} еднозначност, масовост, ефективност
65. 4) {17} Формалност, крайност, ефективност, масовост, резултатност
65. 5) {17} начални данни множител на резултати, обработка,

66. 081. Какви са свойствата на компютърните алгоритми (поне 5)?

66. 1) {11} Бързина, енерго зависимост, голяма памет, възможност за промяна.
66. 2) {14} – Множество на възможните начални данни
- – Множество на междинните резултати
- – Множество на възможните резултати

– Правило за започване
– Правило за непосредствена обработка
– Правило за край

66. 3) {17} Имат точно определено начало и винаги водят до даден край. Структурата им е праволинейна до достигане на резултат, определя се от автора на алгоритъма. Правят обработката на информация по-лека за човека. Могат да се представят в различни ЕП. **[Читателят естествено ще се съгласи, че изброените свойства са брилянтни!?!]**

66. 4) {18} Недвузначност; Кратки заповеди; могат да се разчетат от машина; могат да се изпълнат от машина

66. 5) {23} 1. Точност
2. Икономичност
3. По-лесно извършване на пресмятания чрез дадени
4. съкращения

67. 082. Какво представлява свойството „формалност“ на компютърните алгоритми?

67. 1) {5–7} формалност който съответства на дадени променливи на съответния блок-процедури – функция.

67. 2) {9} Това своиство е от съществено значение защото позволява изпълнението да бъде лишено от разум автоматично.

67. 3) {9} част от множеството от правила на алгоритмите

67. 4) {12} Свойството формалност е изпълняващия алгоритъма **може** да не разбира защо прави тези действия но с елементарни кратки инструкции да се справи, това позволява (програмата) алгоритъма да бъде изпълнен и от машина

67. 5) {14} Формалност – Възможността в един алгоритъм файловете и въведените данни, да вземат по-висок или по-нисък приоритет, да бъдат взимани под предвид или не, както и при изпълнението на алгоритъма – данните и информацията от които е съставен да могат да се заменят с други.

67. 6) {17} Свойството „формалност“ – позволява **изпълнението** на един алгоритъм **да бъде** и **автомат** и не е необходимо изпълнителят да има представа какво ще се случи накрая

67. 7) {23} Свойството формалност позволява изпълнителят на 1 алгоритъм да бъде лишен от разум автомат.

67. 8) {23} свойство на компютърните алгоритми

67. 9) {24} Свойството формалност е от съществено значение, защото позволява изпълнителят на 1 алгоритъм да бъде лишен от разум почит.

67. 10) {26} Свойства са формалност, крайност, дискретност, определеност, изпълнимост, резултатност

68. 083. Какво представлява свойството „крайност“ на компютърните алгоритми?

68. 1) {до 3} Това е задължителен елемент на всеки алгоритъм. По един или друг начин зададеният алгоритъм трябва да завърши по условие.

68. 2) {до 3} Всеки алгоритъм в компютърната инфорлматика (компютърен алгоритъм е множество от надвусмислени изпълни стъпки

68. 3) {8} дава резултат

68. 4) {8} Свойството крайност ни дава вид пояснение на самия алгоритъм.

68. 5) {9} това с-ство е от съществено значение тъй като позволява изпълнението да бъде лишено от разум. **[Какво ли представлява едно изпълнение, което не е лишено от разум?]**

69. 084. Какво представлява свойството „дискретност“ на компютърните алгоритми?

69. 1) {10} Налага непрекъснатите процеси и обекти да се моделират чрез дискретни

70. 085. Какво представлява свойството „определеност“ на компютърните алгоритми?

70. 1) {12} елементарните дейности могат да се изпалняват и повтаря многократно без допълнителни корекции и указания за всяка стъпка информация за състоянието и протичането на моделирания процес. резултата е определен от началните действия описани от алгоритъма

70. 2) {12} Свойството „определеност“ на комп. алгоритми означава, че всеки алгоритъм съдържа последователност от стъпки, при които **всяка една** от тях **може да определя следващата**. Всеки алгоритъм има входни данни, които определят изходните резултати.

70. 3) {14} Определеност: Създаденият компютърен алгоритъм може да се използва за цял клас от еднотипни задачи

70. 4) {14} Всяка стъпка на алгоритъма трябва да се следи внимателно и да се провери, преди да пристъпим към следващата стъпка.

70. 5) {15} На всяка стъпка за състоянието и протичането на процеса трябва да е достатъчна за да се определи следващото действие. Резултата е напълно определен от входните данни и действия, описани в алгоритъма.

70. 6) {17} Компютърният алгоритъм е естествен език – блок схеми – предимства – анг. език, ясно, точно, ,описание на даните,

70. 7) {17} Определеност означава, че на всяка стъпка информациегъп трябва да е определена в процеса на изпълнение

70. 8) {18} Свойството определеност на компютърните алгоритми пояснява тяхната строго определена структура (Начало – Край)

70. 9) {20} Определеността на компютърните алгоритми означава, че на всяка стъпка информацията за състоянието и протичането на моделирания процес трябва да е достатъчна, за да се определи еднозначно **следващото действие на потребителя**.

70. 10) {23} свойството „определеност“ служи за определяне структурата на алгоритъма и начина му на действие

70. 11) {23} На всяка стъпка информация за състоянието на моделирания процес трябва да е достатъчна, за да определи следващото действие

70. 12) {26} Определеност на алгоритъма означава, че на всяка стъпка информацията за състоянието и протичането **[На какво?]**, трябва да е достатъчна, за да определи следващото действие на изпълнителя

70. 13) {26} Всеки алгр. в комп. инф. е множество от е множество от недвусмислени и изпълними стъпки моделиращи краен и ограничен в пространството и времето инф. процес при който от дадена начална ин-я. се получава като резултат друга информа ца – в по-късен период и евентуално на друго място

70. 14) {26} Решението на задача по зададен алгоритъм може да се извърши многократно по всяко време от различни хора и при това за един еднакъв резултат.

70. 15) {26} Ако процесът е краен, то резултатът е определен от видове данни и действия описани с алгоритъма

71. 086. Какво представлява свойството „масовост“ на компютърните алгоритми?

71. 1) {14} Една програма да бъде използвана няколко пъти.

71. 2) {14} достъпност до голямо количество данни

71. 3) {17} „Масовост“ на алгоритмите е свойството да бъдат прилагани за извършването на различни операции, а не само на една конкретна.

72. 087. Какво представлява свойството „ефективност“ на компютърните алгоритми?

72. 1) {10} Когато завършва в реално време и операциите са нормален брой

72. 2) {10} Свойство – да извършва бързо и точно определени действия и да показва крайния резултат

72. 3) {16} Един компютърен алгоритъм е ефективен когато улеснява решаването на даден проблем и води до крайния желан резултата. Ако се налага може да се използват подалгоритми, за да стане алгоритъма още по-ефективен.

73. 089. Какво представлява свойството „резултативност“ на компютърните алгоритми?

73. 1) {15} Изпълнението на алгоритъма трябва да води до резултат
73. 2) {15} С-вото резултативност е осигурено след краен брой елементарни операции. Резултативността понякога се третира след краен брой стъпки.

74. 090. Кому принадлежат основните идеи за конструиране на автоматично работеща сметачна машина?

74. 1) {8} Чарлз Бабич, Блейс Паскал, Дюрик **[Да бе мирно спряло (след първото име), не би чудо видяло! Даденият отговор показва пълна липса на знания.]**
74. 2) {9} Джон **Винсонт** Атанасов
74. 3) {14} Джордж Булл.
74. 4) {20} Джордж Бул.
74. 5) {21} Джон Атанасов
74. 6) {23} Основните идеи за конструиране на автоматично работеща сметачна машина принадлежат на Джон Атанасов.

75. 091. Какви са основните идеи на Чарлз Бебидж за неговата Аналитична машина?

75. 1) {12} Идеята на Бебидж е да се създаде изчислителна машина 30 × 10 метра, задвижвана от парен двигател, която да има склад (оперативна памет) да приема входни данни чрез перфокарти, да принтира на лист хартия (или да променя перфокарти), действията да се ръководят от мелница (еквивалент на днешния централен процесор) **[Съвременният Централен процесор обединява мелницата и управляващото устройство на Аналитичната машина на Бебидж, а описаното в този отговор няма почти нищо съществено общо с идеите на професора.]**
75. 2) {14} Да извършва всички действия без намесата на човек, ~~сама да подбира алгоритъма за изпълнение на програмата~~ сама да подбира пътя на пресмятанията на базата на получените междинни резултати
75. 3) {14} Идеите на Чарлз Бебидж за неговата аналитична машина са: да може да се изчисляват алгоритми **на компютър** и да може програмата сама да намира пътя за решаването им, като стигне до определен междинен резултат.
75. 4) {14} Да извършва всички действия без намесата на човек. Включително исама да подбира пътя на пресмятанията в зависимост от получени в даден междинен етап резултати
75. 5) {14} Да извършва всички действия без намесата на човек; да подбира сама пътя на пресмятане; на базата на получените междинни резултати
75. 6) {14} идеята за двоична бройна система
75. 7) {17} Щяла е да се състои от „склад“, който е щял да съхранява междинните резултати, „мелница“, която е щяла да извършва пресмятанията, **периферни устройства, които са щели да определят какви ще бъдат входните и изходните данни**.
75. 8) {17} Да извършва всички действия без помощта на човек, включително и **сама да подбира пътя** на пресмятанията.
75. 9) {17} Да извършва всички действия без намеса на човека, включително и **сама да подбира пътя** на пресмятанията, в зависимост от получения междинен резултат.
75. 10) {18} Да изпълнява сама аналитичните и анализиращи действия и без външна намеса да подбира точните логическо и/или математически вярни резултати.
75. 11) {20, 21, 23, 26, 26, 26} Основните му идеи били машината му да извършва всички действия без намеса на човек включително и **сама да подбира** пътя на пресмятанията в зависимост от получените от определен междинен етап резултати.
75. 12) {20} Да създаде автоматично смятаща машина.

75. 13) {20} Идеите са Че ще има място за съхранение на междинните резултати ще извършва изчисленията без човешка намеса, ще използва троичен код. Идеите на Бебич не се реализират по негово време заради недостига на технологии
75. 14) {22} Да извършва всички дейности без намесата на човек, включително и **сама да избира пътя на пресмятания**.
75. 15) {23} Основните идеи на Чарлз Бебидж били машината му да извършва всички действия без намеса на човек, включително и сама да подбира пътя на пресмятанията в зависимост от получените на определен междинен етап резултати.
75. 16) {23} Основните идеи на Чарлз Бебидж за неговата Аналитична машина са тя сама да извършва пресмятанията и дори по време на изчисленията сама да определя пътя на изчисленията и операциите.
75. 17) {25} Чарлз Бебидж и Огъста Едъ въвеждат термини като цикъл, работна клетка и др.
75. 18) {26} Основните му идеи били машината му да извършва всички действия без намеса на човек
75. 19) {26} Основната идея на Чарлз Бебидж за неговата Аналитична машина – машината да извършва всички изчисления без намеса на човека.
75. 20) {27} Основните му идеи били машината му да извършва всички действия без намеса на човек, включително и сама да подбира пътя на пресмятанията.

76. 092. Защо Бебидж не може да реализира идеите си за създаване на своята аналитична машина?

76. 1) {12} Поради финансови причини, защото машината е с размери 10 × 30 метра **[Как ли се определят размерите на нереализирана машина?]** и иска парен двигател.
76. 2) {17} Идеите му са твърде футуристични и чисто хипотетични за времето му. Макар и на теория и с достатъчно желание Бебидж няма необходимите ресурси.
76. 3) {26} Бебидж не успява да реализира идеите си поради рипса на програма за неговата машина, която по-късно се създава от Ада
76. 4) {26} Мисъл, в която твърдим или отричаме нещо, което може да се счита за верно или неверно

77. 095. Какво изобретява Джон Атанасов?

77. 1) {15} Първият компютър „ABC“ е изработил Джон Атанасов
77. 2) {23} Джон Атанасов изобретява компютър ABC
77. 3) {26} Джон Атанасов изобретява **първия компютър**
77. 4) {27} Първият компютър

78. 096. Кой е първият електронен компютър?

78. 1) {14, 20} ENIAC
78. 2) {20} Eniak
78. 3) {20} Компютъра на Джон Атанасов **[Въпросът не е „Кой го е направил?“, а „Как се казва?“]**
78. 4) {23} Първият електронен компютър е ENIAC

79. 097. Кой е първият универсален електронен компютър?

79. 1) {до 3, 10, 10} ABC.
79. 2) {14} Унивак
79. 3) {14} ABC на Джон Атанасов
79. 4) {17} EDIAC
79. 5) {17} Унивак 1.
79. 6) {20} Създаден е от Мокли и Екърт през 1949 г. univac
79. 7) {20} **Първият електронен** компютър е ABC на Джон Атанасов
79. 8) {23} EVIAC е първият универсален компютър
79. 9) {23} **Първият електронен** компютър е ABC

79. 10) {26} ANIAC

80. 098. Кой е първият цивилен компютър?

80. 1) {12} Първият цивилен компютър е този, който е пуснат за първи път в серийно производство. Това е Intel 8008 **[Авторът едва ли подозира, че серийно производство на компютри започва едва в края на миналия век? И естествено той и хал хабер си няма че I8008 е микропроцесор, а не компютър.]**

80. 2) {20} Джон Атанасов и Клифърт създават първия през 40-те. Казва се **Правец**

81. 099. Кои са принципите на фон Нойман за реализиране на електронен компютър?

81. 1) {12} Компютъра трябва да бъде електронно устройство и да използва двоичен код, структурата му да включва аритметико-логическо устройство, устройство за управление, памет и устройство за въвеждане и извеждане на информация и др.

81. 2) {14} Централен процесор – който извършва пресмятанията
Оперативна памет – където се пазят междинните резултати и изпълнявания алгоритъм
Периферни устройства – осигуряват връзката с околния свят

81. 3) {14} аритмомер, тригонометрични и други схеми с функции. Фон Нойман иска да реализира компютърът като средство което да улесни човека и да изпълнява зададените от човек алгоритми

81. 4) {14} Принципите на фон Нойман за реализиране на ел. компютър са той да има: входни устройства, обработващо устройство (процесор), памет и изходни устройства

81. 5) {17} 1) работа с двоична БС
3) достатъчно е само операция събиране
2) програмата се съхранява в паметта

81. 6) {17} Принципите на фон Нойман за електронен компютър за двоична БС и аритметичните операции: събиране и изваждане. **[Типичен отговор в стил: „И аз съм чувал нещо, но нито го знам, нито го разбирам!]**

81. 7) {17} Използва двоичен код

81. 8) {17} **Двуична** бройна система, данните да се съхраняват в паметта, операция събиране.

81. 9) {17, 20, 21, 21} Двоична бройна система. Програмата да се съхранява в паметта. Достатъчна е само операция събиране **[Типичен отговор в стил „И аз съм чувàл нещо по въпроса!“, който твърде отдавна се смята за напълно грешен.]**

81. 10) {20} Обща схема на фон Нойманов компютър. **[Следва рисунка на такъв компютър, само дето въпросът е съвсем друг и няма нищо общо със схемата!]**

81. 11) {20} Двоична бройна система; Програмата да се съхранява в памет; Достатъчна е само операция събиране

81. 12) {20} Нойман създава едни от основните принципи за създаване на електронните компютри, поради неудобството **от използването на използване** на компютри с тежат от 80 тона. Електронните компютри значително намаляват тази големина.

81. 13) {23} Принципите са двуична бройна система, програмата да се съхранява в паметта и достатъчна е само операция събиране

82. 100. Каква е основната разлика между идеите на Чарлз Бебидж и принципите на Джон фон Нойман?

82. 1) {16} Принципите на Джон фон Нойман – двоична бройна с-ма, програмата се съдържа в паметта, необходима е само събиране. Чарлз Бебидж – иска вс да се извършва без намесата на човек, включително и сама да подбира пътя на пресмятанията в зависимост от получените на опр. междшинен етап резултати.

83. 101. Кой е първият компютър, помнещ програмата си в своята памет?

83. 1) {9, 32} ABC

83. 2) {9, 12, 14, 17, 20, 20} EDVAC

83. 3) {9} Първият компютър помнещ програма н паметта си е ABC – Атанасов В. Computer.

83. 4) {14} EDIVAC

83. 5) {14} UNIVAC.

83. 6) {14} VANIAC
83. 7) {17} UNIVAC I
83. 8) {17} EFIAC – първият компютър помнещ програмата в своята памет.
83. 9) {20} Компютърът на фон Нойман.
83. 10) {26} ENIAK
83. 11) {27} ABC (Atanasoff Berry Computer)
83. 12) {28} Първият компютър помнещ програмата си в своята памет е „Правец“ издаден от Атанасов през 1954 г.

84. 102. По какъв критерий компютрите се делят на поколения?

84. 1) {до 3} ① релета (до появата на ABC)
② ел. лампи
③ транзистори
④ ИС с МСРСИ
⑤ ИС с ГСИ.
84. 2) {9} Критерия по който се делят е тяхната функционалност
1во поколение – резистори
2ро поколение – електрически лампи
3то поколение – транзистори
Ние в момента сме от четвърто поколение
84. 3) {14} Елементи на база.
84. 4) {19} Те се делят на поколения по начина им на конструиране, изработка.
84. 5) {20} По това, в коя година са изобретени и кой е техния създател.
84. 6) {20} Компютрите се делят по години, размер, програми
84. 7) {20} Критерия, по който компютрите се делят на поколения е
84. 8) {22} По хардуерни и софтуерни спецификации.

85. 105. Каква е елементната база на компютрите от второ поколение?

85. 1) {12} Елементната база на компютрите от второ поколение са **електрическите лампи**.
85. 2) {15} Електронни вакуум лампи
85. 3) {17} Комп. от второ поколение – електронни вакуумни лампи
85. 4) {17} вакумни лампи
85. 5) {20} Електронни вакумни лампи.
85. 6) {20} Компютрите от второ поколение са с елементна база релета.
85. 7) {26} Основана на двоична ПБС.

86. 106. Каква е елементната база на компютрите от трето поколение?

86. 1) {8, 8, 9} транзистори
86. 2) {9} Компилатор, ОП, периферни устройства, централен процесор, интегрални схеми
86. 3) {14} интегрални схеми **[Без да се посочи какви са тези схеми отговорът става абсолютно грешен и смешен. Да не говорим, че съседчето може би не е имало време да уточни какви са схемите!?!]**
86. 4) {17} Елементната база на компютрите от трето поколение се състои от RAM памет (тя не запазва информацията от дадена програма при изключване на захранването, ROM памет (тя спомага за съхранение на въведената информация, Hard disk (външна памет на компютрите)
86. 5) {17} интегрални схеми с малка и голяма степен на интегразия
86. 6) {20} Компютрите от трето поколение имат интегрална схема **[Каква?!?]** за ел. база, но не са толкова бързи като тези от четвърто поколение.

87. 107. Към кое поколение компютри принадлежат съвременните компютри и каква е тяхната елементна база?

87. 1) {до 3} Съвременните компютри принадлежат към фон Нойманова конфигурация.

87. 2) {10} – днес (транзистори),
87. 3) {14} МС с МСрИ **[За поколението ни дума, ни вопъл, ни стон!]**
87. 4) {14} Съвременните компютри принадлежат **към трето** поколение, тяхната елементна база са **ИС** (интегрални схеми)
87. 5) {14} Шесто поколение.
87. 6) {17} Пето поколение, интегрални схеми със свръх високо ниво на интеграция. Изградени са на базата на архитектурата на фон Нойман

88. 108. Какво е предназначението на ОП?

88. 1) {17} Тези ОП осигуряват връзка с околния свят. В зависимост от посоката те се наричат ОП за вход и изход.
88. 2) {17} Предназначението на ОП е да (~~извършват~~) зареждат правилно програмите и да функционира правилно системата на компютърът.
88. 3) {17} **За съхранение на данните за съхранение** на даден алгоритъм, програма
88. 4) {17} Предназначена за съхраняване на **данните за изпълнение** на даден алгоритъм
88. 5) {23} Предназначението на ОП е да съхранява резултатите от програмата
88. 6) {23} Тези оператори осигуряват връзката с околната среда. В зависимост от посоката те се наричат оператори за вход и изход.

89. 111. Еднакви ли са всички клетки на ОП? Ако не – защо, ако да – как се различават?

89. 1) {10} не защото ОП помни моментни данни с различен размер
89. 2) {14} Преводът от ЕП до МЕ може да бъде извършен от хора е полезен. Защото
89. 3) {17} Не, различават се по размерността и съдържанието им.
89. 4) {20} Не са еднакви. **Всяка** клетка **има име и адрес, в който е изпратена**
89. 5) {26} Всички клетки в ОП са различни, защото съдържат различна информация и са с различни адреси.

90. 118А. Посочете поне две предимства на електрическите памети.

90. 1) {15} Имат по-голяма вместимост, и възможност да се презаписват.

91. 120. Защо е необходима постоянната памет (ROM)?

91. 1) {до 3} Определя изработката на запомнещи елементи. Постоянна ROM памет.

92. 121. Какво е предназначението на ЦП?

92. 1) {9} Изпълнителя на дадена програма
92. 2) {12} **Запомняне** и разбиране на дадената програма
92. 3) {14} Да чете и разбира машинния език. **[Може ли език, пък бил той и машинен, да се чете?!?]**
92. 4) {14} предназначен е за решаване на машинните задачи
92. 5) {14} Предназначението на ЦП е да чете и разбира маш. език.
92. 6) {14} Изпълнява аритметичните действия
92. 7) {14} За решаване на машинните задачи
92. 8) {18} Той прави пресмятанията нужни за реализиране на дадена програма
92. 9) {23} **Да избере** и изпълни зададена програма
92. 10) {23} Предназначението на централния процесо**р е да изчислява алгоритми** и да резултатите обратно в паметта.
92. 11) {23} ЦП декодира и изпълнява инструкциите от програмното осигуряване.
92. 12) {26, 26, 26} Централният процесор изпълнява машинните програми.
92. 13) {26} Централния процесор е „мозъка“ на всеки компютър. Той извършва всички логически операции.
92. 14) {27} Във съвременните устройства ЦП е микрочип Негова задача е изпълнение на двоичния код.

93. 122. От какви подустройства се състои ЦП?

93. 1) {до 3} състои се от ИС.

94. 123. Какви са функциите на Управляващото устройство на ЦП?

94. 1) {до 3, 5–7} Грижи се за правилното и синхронно действие на останалите компоненти.
94. 2) {5–7, 5–7} – извличане
– Декодиране
– изпълняване
– проверяване за прекъсване
94. 3) {9} разчитане, извличане, изпълнение

95. 126. Какво е микропроцесор?

95. 1) {до 3} Микропроцесора е сърцето на компютъра и служи за обработване и изпълнение на програмата която се зарежда в ОП.

96. 129. Какво представят машинните инструкции по отношение на кодирания алгоритъм?

96. 1) {9} 0 и 1

97. 132. Къде трябва да бъде машинната програма за да може ЦП да я изпълни?

97. 1) {9} Трябва да бъде написано на машинен език.
97. 2) {12} За да бъде разбираем от ЦП трябва да се осигури превод на **нашия език може да бъде** и **програма**
97. 3) {14} Програмата трябва да бъде на твърдия диск.
97. 4) {14} В твърдия диск

98. 136. Какви проблеми създава невъзможността един ЦП да изпълнява машинни програми на друг ЦП?

98. 1) {8} Да бъде неразбран (неразличен)

99. 138. Какво е предназначението на периферните устройства?

99. 1) {9} Спомагат за извършване на операции

100. 139. Каква е основната разлика между периферните и главните устройства (ЦП и ОП) на един компютър?

100. 1) {до 3} Разликата е че Периферните са вън от компютъра.

101. 140. Какви видове ПУ познавате?

101. 1) {до 3} Принтер, скенер.
101. 2) {до 3} **Паскал, Си, ВБ**.
101. 3) {5–7} Принтер, скенер, модем.
101. 4) {14} Мишка, клавиатура, монитор.
101. 5) {20} 1) входни – изходни
2) комуникационни
101. 6) {20} 1) входни – изходни
2) комуникационни

102. 144. По какъв начин може да бъде записан един алгоритъм?

102. 1) {до 3} Чрез Логически израз.
102. 2) {до 3} Алгоритъма може да бъде записан като един вход и един изход.
102. 3) {до 3} Алгоритъм се нарича точно и разбираемо предписание определящо изпълнението на последователни елементарни операции чрез които се решава класа от еднотипни задачи.

102. 4) {до 3} Алгоритъма може да бъде записан във форма и вид в който той може да бъде възприет и съответно изпълнен от програмата.
102. 5) {до 3} Алгоритъм се нарича точно и общоразбираемо предписание определяко изпълнението на последователност от елементарни операции, чрез които се решава клас от еднотипни задачи.
102. 6) {14} Алгоритъмът е съвкупност от правила, чрез които последователно се извършват действия за разрешаването на даден проблем.
102. 7) {14} Алгоритмите са последователни, циклични и разклонени.
102. 8) {17} Ч/з естествен знак – алгоритмичен език или блок-схеми
102. 9) {20} Един алгоритъм може да бъде записан чрез използването на естествен човешки език като можем да използваме математическите означения
102. 10) {20} Един алгоритъм може да бъде записан чрез поредица от прости (елементарни действия.

Например алгоритъма за приготвяне на чай ще бъде следният:
① Пригответе се за работа
② Сложете чаят и водата в подходящ съд
③ Поставете върху загрят котлон докато започне да кипи
④ Сервирайте топло
⑤ Прекратете работа.

102. 11) {23} Един алгоритъм може да бъде записан чрез използването на естествен език, като можем да използваме и познатите математическите означения като приемем, че написаното може да бъде разбрано от всеки, който говори използвания език и е учил математика. Възможно е да попаднем на изпълнител, който не може да извърши всички означени действия.
102. 12) {23} Един алгоритъм може да бъде записан чрез използването на естествен (човешки) език, като можем да използваме и познатите математическите означения (+; –; >; ≠), като приемем, че написаното може да бъде разбрано от всеки, който говори използвания език и е учил математика. Възможно е да попаднем на изпълнител, който не може да извърши всички означени действия (като коренуване например)
102. 13) {26} Един алгоритъм може да бъде записан чрез използването на естествен език, като можем да използваме и познатите математическите означения (+; –; >; ≠), като приемем, че написаното може да бъде разбрано от всеки, който говори
102. 14) {26} По два начина по естествен ред на изпълнение (последователен) и разбъркан (непоследователен)

103. 145. Какви заповеди съдържа един алгоритъм, записан на човешки език?

103. 1) {до 3} Изпълни, направи, иди, ако, то.
103. 2) {до 3} Съдържа заповеди за извършването на различни действия за да се направи нещо.
103. 3) {до 3} Изпълняване заповедите за действие на естествен език.
103. 4) {до 3} Един алгоритъм представя дадено сложно действие чрез редица от достатъчно прости (елементарни) действия.
103. 5) {5–7} Изпълни, провери, премини
103. 6) {12, 14, 17} Всяка заповед се номерира с естествено число Изпълнението започва от заповед номер едно **[От този отговор веднага ти става ясно какви видове заповеди има – номерирани!]**
103. 7) {13} Алгоритъма на естествен език не може да съдържа въпросителни или подбудителни изречения. Заповедите трябва да са съобщителни изречения. Изречението: Не сме сами в тази вселена също е съждение въпреки, че не сме сигурни за това.
103. 8) {14} Алгоритъм е последователност от прости действия за решаване на даден проблем, като постигнем желания резултат. **Един алгоритъм трябва да съдържа определеност**, формалност, масовост и ефективност.
103. 9) {14} съдържа последователност от действия, номерирани с цифри, започва от едно

103. 10) {14} Заповедите, които съдържа един алгоритъм на ест. човешки език се номерират с цели числа и се изпълняват като се следва последователно всяка стъпка. Измислят се от човекът, който съставя алгоритъма.
103. 11) {17} Всяка заповед се номерира с естествено число Изпълнението започва от заповед номер 1 Заповеди, които да ги разбере и изпълни **[Кой?]**
103. 12) {17} Заповедите – умножение и деления
103. 13) {17} Всяка заповед се номерира с естествено число Изпълнението започва от заповед 1.
103. 14) {17} Всяка заповед се номерира, започвайки от заповед номер 1.
103. 15) {18} По някои път неясни заповеди; описателни
103. 16) {20, 23, 26} За да можем лесно да указваме коя заповед е следваща, при писането им ги номерираме с естествени числа, като изпълнението на алгоритъма започва от заповед 1. Когато след изпълнение на една заповед трябва да се изпълни тази, чийто номер е с единица по-голям от нейния, това може да не се записва явно в самата заповед. В една заповед може да няма указано друго действие освен посочване на номер на следващата за изпълнение
103. 17) {20} Всяка заповед се номерира с естествено число изпълнението започва от първата заповед
103. 18) {20} На човешки език заповедите в един алгоритъм са под формата на съобщителни или заповедни изречения, а не като стандартни команди, както е при езиците за програмиране
103. 19) {20} Заповедите на човешкия алгоритъм представляват думи, написани на човешкия език, което води до проблем. ЦП не може да превежда човешкия език, а само машинен. Следователно човешкия език не може да се използва в компютърните алгоритми, освен ако не бъде преведен по някакъв начин на машинния език.
103. 20) {21} Последователност от съобщителни и/или заповедни изречения, и/или математически изрази.
103. 21) {21} За да можем лесно да указваме коя заповед е следваща, при писането им ги номерираме с естествени числа, като изпълнението на алгоритъма започва от „Заповед 1“. Когато след изпълнение на една заповед трябва да се изпълни тази, чийто номер е с единица по-голям от нейния, това може да не се записва явно в самата заповед.
103. 22) {23} Всяка заповед се наименува с естествено число. Изпълнението започва от заповед 1
103. 23) {23} За да можем лесно да указваме **в коя** заповед е следващата при писането им ги номерираме с естествени числа, като изпълнението на алгоритъма започва от.
103. 24) {25} За доможе лесно да определим коя заповед коя, ние ги номерираме счисла. След изпълнението на една заповед, то след нея се изпълнява друга, чиито номер е с едно по-голям от нейния. В една заповед може да няма оказана друго изпълнение освен посочване на номер на следващата заповед
103. 25) {26} Алгоритъм е сложно действие, представено от по-прости действия наречени „елементарни действия които са предварително съставени от съставителя след което изпълнени от изпълнителя който следва оказани стъпки и не са му нужни допълнителни обяснения за конструирането на алгоритъма.
103. 26) {26} За да можем да указваме коя заповед е следваща, когато ги записваме ние ги номерираме с числа и изпълнението започва от 1. Програмата процедира с изпълнение на всяко следващо, по-голямо от предишното число.
103. 27) {26} Конюнкция и отрицание
Дизюнкция и отрицание

104. 147. Посочете предимства и недостатъци на човешкия език, като средство за записване на алгоритми.

104. 1) {до 3} Алгоритъмът е множество от правила (инструкции) разпоредба, план и др.
104. 2) {до 3} недостатъците са че в естествения език има звуци и знаци.
104. 3) {до 3} Ясно, точно и еднозначно опесание пригодени за четене от машина не винаги са удобни за хората.

104. 4) {14} Недостатъци:
1) Не може да се чете от машина
2) Трябва да се изучава допълнително

105. 147А. Посочете поне две предимства на човешките езици, като средство за записване на алгоритми.

105. 1) {12} Предимствата на естествения език са: 1) че естественият език е **богат на диалекти и** е **разговорен**
2) естествения език е **икономичен**

105. 2) {14} 1. по-лесно разбираем **[За кого?]**
2. лесен за изпълнение **[Може ли език да се изпълнява?]**
3. съставянето им изисква по-малко време **[?!?]**

105. 3) {17} Ясно определени стъпки за изпълнение

105. 4) {17} 1 Писмена форма
2 Има диалекти

105. 5) {20} – Предимствата, по който, чрез човешките езици можем да покажем записването на алгоритми е, че по този начин се улеснява начина на представяне на дадения алгоритъм.

105. 6) {23} 1. възприемане в краткосрочната памет
2. възприемане в дългосрочната памет
3. запаметяването

105. 7) {25} 1) Имат определение за точен превод; еднакви входни данни довеждат до еднакви резултати.
2) Имат адекватни граматични конструкции, защото трябва да се записват алгоритми.

105. 8) {26} Писмен и говорим, Имат диалекти, Имат подмножества, живите се променят с времето

105. 9) {26} Кодиране и разчитане

105. 10) {26} Пригодени за четене от машина, могат да се разделят на класове

106. 147В. Посочете поне два недостатъка на човешките езици, като средство за записване на алгоритми.

106. 1) {14} Не е четим, което обуславя изпълнение на задачата

106. 2) {15} Трябва да се учат допълнително, провокират допускане на грешка и не са подходящи за хората.

106. 3) {15} не е за машина

106. 4) {20} Недостатъци: не се разчита от компютъра

106. 5) {22} Отнема повече време и по-трудно се достига до резултат.

106. 6) {26} Не всеки ги разбира

107. 148. Какво представляват блок-схемите?

107. 1) {до 3} Оперативна памет от централен процесор.

107. 2) {14} Блок схемата е **част от програмата**, която се пише в квадратни скоби ([]) Това е начин да се опишат **логаритмите** чрез схема.

107. 3) {14} Начин за представяне на алгоритъма чрез схема.

107. 4) {14} Блок-схемите са схеми на една програма, изразяват се с геометрични фигури, всяка от които има определено значение. Само хората могат да разчитат блок-схемите. Те са неразбираеми за машините

107. 5) {14} Блок-схемите представляват геометрични фигури в които се записват данни.

107. 6) {17} Блок-схемите сса методи за изготвяне и записване на алгоритми, които да бъдат изпълнени и съответно да се получи някакъв резултат. Блок-схемите се осъществяват с помощта **на геометрични функции**, като всяка от тях има съответно значение.

107. 7) {17} Схема от блокове, които се използват за една компютърна система.

107. 8) {20} Блок схемите са метод за записване на алгоритмите, който е ясен, разбираем, удобен и сравнително прост. Неудобството на този метод е, че не може да бъде директно разчетен от програма.
107. 9) {22} Блок-схемите за методи и начини за записване на алгоритмите, които да бъдат изпълнени и съответно да се получи някакъв резултат.
107. 10) {24} Блок-схемите за **методи за използване** и записване на алгоритми
107. 11) {26} Метод за изразяване на алгоритъм.

108. 149. Посочете основните блокове на блок-схемите и тяхното предназначение.

108. 1) {до 3} Основните блокове са Централен процесор ОП,}
108. 2) {до 3} Графични изображения се възприемат от хората по лесно структурата може да бъде проследена по леко
В различно оформени блокове се записват словесно или математически . Реда на изпълнение се посочва със стрелки които свързв. последователно блокове.
108. 3) {5–7} Графично изображение.

109. 152А. Посочете поне две предимства на блок-схемите като средство за запис на алгоритми.

109. 1) {17} х и у
109. 2) {20} Бързо и удобно

110. 153. Какви са идеите на алгоритмичните езици като средство за записване на алгоритми (поне 3).

110. 1) {15} Идеите са да можем по-бързо и по-лесна да се справим с поставената задача.може да се разбере и от човека и от компют ъра

111. 154. По какво си приличат човешките и алгоритмичните езици?

111. 1) {до 3} Имат определеност. Необходимо е всички обекти включени в една комуникация да разбират езика който използват, или да има преводач.
111. 2) {15} По символите

112. 155А. Кой определя достъпните конструкции в един алгоритмичен език и техния смисъл?

112. 1) {14} Определя ги програмиста
112. 2) {14} Този, който пише програмата
112. 3) {16} Този, който е съставил алгоритъма.
112. 4) {20} Този който е съставя алгоритмичния език. (**изпълнител**)
112. 5) {20} Машинната програма трябва да бъде записана в оперативната памет, за да може да бъде прочетена от централния процесор (ЦП)
112. 6) {21, 26, 26, 26, 26, 27} Достъпните конструкции в един алгоритмичен език и техния смисъл или са определени предварително. **[Само дето въпросът не е „Кога?“, а „Кой?“]**

113. 155В. Кога и къде се определят достъпните конструкции в един алгоритмичен език и техният смисъл?

113. 1) {14} В началото и в края на алгоритъма като входна и изходна информация (резултат).
113. 2) {14} Достъпните конструкции се определят още в началото и могат да се използват навсякъде в програмата чрез добавяне на файла, чрез това се избягва повторението на едни и същи неща и евентуални грешки, това допринася за прегледността и компактността на програмата.
113. 3) {20} Достъпните конструкции и смисълът им са определени предварително **в програмата**
113. 4) {26} Достъпните конструкции и смисълът им са определени предварително

114. 156. Възможно ли е двусмислено тълкуване на конструкция в алгоритмичен език? Защо?

114. 1) {8} Да, защото може да се получи преплитане на стрелки
114. 2) {8} Да ако са от един и същи клас.

115. 158. Посочете предимства и недостатъци на алгоритмичните езици като средство за записване на алгоритми.

115. 1) {до 3} Алгоритмичният процес е ефективен, ако приключва в „реално “ време и всички присъщи за алгоритъма резултати се получават след приемлив брой стъпки. Ако процесът е краен то резултатът е напълно определен само от началните (входни) данни и действията описани от алгоритъма.
115. 2) {8} Предимства: ясност, последователност
Недостатъци:

116. 159. Какво представляват ЕП?

116. 1) {9} ЕП – език за програмиране

117. 161. Какъв трябва да бъде всеки ЕП (поне 3)?

117. 1) {10} оператор от компилатори
117. 2) {27} 1) Да е съставен от недвусмислени знаци, думи или съждения
2) Да може да се разбира и от машинните езици и от човешките
3)

118. 162. Оценете МЕ на ЦП като ЕП.

118. 1) {5–7} **Трудно**
118. 2) {5–7, 5–7, 5–7} Труден.

119. 163. Посочете поне 3 причини, поради които МЕ е труден за използване от хората.

119. 1) {до 3} Той е машинен език и е най вече компютърен и се избира в памета на компютара или дискета хората нямат толкова памет за да запомнят такава информация или толкова данни.
119. 2) {12} МЕ е труден за хората, защото може да е неразбираем, за тази цел е нужен преводач, който може да бъде и програма.
119. 3) {14} – трудно се разбира от човек, трудно се запомня, изисква много писане и много човешки труд.
119. 4) {17} Машинния език е труден за използване от хората, защото той е език, който трябва да се изучава не е например като (човешки език където не е необходимо изуване).
119. 5) {20} 1 – Трудно разбираем
2 – Бавно **усмисляне**
119. 6) {20} 1) сложен е за хората
2)
3)
119. 7) {26} труден за писане
труден за разчитане

120. 164. Посочете алтернативно на МЕ решение за създаване на програми и неговите предимства и недостатъци.

120. 1) {11} Недостатъкът му **[На кого?]** е че не е разбираем за другите езици **[Доста странно е как един език може да разбира каквото и да е!]**
120. 2) {14} Алтернативно решение притежават компютрите от „0“-во и I^{во} поколение, [Значи то не важи за днешните компютри!] като те са пълни с недостатъци бавно обработване на програмата, голям обем на компютрите, голям разход на енергия.

120. 3) {14} Решение на КЕ. **[Що ли за звяр е то „КЕ“?]**
Предимства: по-лесно съставяне на компютърна програма.
Недостатъци: труден за човек, който не го разбира, допускат се повече грешки.
120. 4) {23} Предимства – лесен за научаване, по разбираем и привичен запис, по-малко грешки в програмата и по-бързо писане на програмата Недостатъци – неразбираем за ЦП
120. 5) {27} Множител с който се изменя стойността на цифра преместена на съседна позиция.

121. 166. Необходимо ли е преводът от ЕП до МЕ да бъде извършен от човек? Защо?

121. 1) {14} Да, само човекът да е добре запознат с езиците, които ще работи
121. 2) {20} Да необходимо е
121. 3) {20} Възможно е преводачът да бъде и програма

122. 167. Възможно ли е преводът от ЕП до МЕ да бъде извършен от човек? Кога?

122. 1) {5–7} Не, че понякога се допускат много грешки.
122. 2) {20, 26, 26, 26, 26} Не, защото ЦП разбира само своя собствен МЕ
122. 3) {26} Преводът от ЕП до МЕ, не е възможно да бъде извършен от човек, Защото ЦП разбира само своя собствен МЕ.

123. 168. Полезно ли е преводът от ЕП до МЕ да бъде извършван от хора? Защо?

123. 1) {19} Да – стига дадения човек да знае и двата езика

124. 169. По какъв критерий се класифицират ЕП?

124. 1) {9} Машинно зависими и машинно не зависими. Машинно зависими са езици за програмиране от ниско равнище, а машинно не зависими са езици за програмиране от високо равнище.
124. 2) {15} Езиците за програмиране се класифицират според тяхната сложност и набор от операции; най-основно са 2 вида:
Пример. ЕПНР – ЕП на ниско равнище
ЕПВР – ЕП на високо равнище

125. 170. Какви класове от ЕП познавате?

125. 1) {до 3} **Асемблери**.

126. 171. Какво е характерно за ЕПНР и как още се наричат те?

126. 1) {20} Структурата и наличните им команди са валидни за конкретна компютърна система. **[Във въпроса се изисква да се упомене и другото име на тези езици!]**
126. 2) {20} **ЕПВП** се наричат езици за централен процесор или асемблерни езици. Те са **логично-зависими** езици, защото тяхната техника зависи от конкретен ЦП.
126. 3) {20} ЕПНР са машинно зависими езици, и още **се наричат машини езици**.
126. 4) {23} ЕПНР се наричат още макроасемблери. Те работят само на компилативен принцип, т. е превеждат цялата програма
126. 5) {23} ЕПНР са **машинно-независими** езици, още **се наричат машини езици**.
126. 6) {26} При тях отделните инструкции се представят чрез мнемонични означения.
126. 7) {26} ЕПНР са машинно-зависими. Още се наричат машинни езици.

127. 171. Какво е характерно за ЕПНР и как още се наричат те?

127. 1) {12} **Те са енергозависими [Голям принос към световната наука е откритието на автора, че един език може да зависи от притока на електроенергия!]** и се наричат асемблерни
127. 2) {14} ЕПНР – асемблерни
127. 3) {14} Наричат се още асемблери.
127. 4) {20} ЕПНР са машинно зависими, още се наричат машинни езици.

128. 172. Защо ЕПНР се наричат и машинно-зависими езици?

128. 1) {до 3} Машинно зависими са, защото са от ниско равнище.
128. 2) {до 3} Защото има езици от ниско равнище.
128. 3) {9} Защото зависят от машинните езици. Защото е единствения разбираем език и е алгоритмичен.
128. 4) {15} **имат** по-високо структорирано ниво на инструкциите и **машинна независимост**
128. 5) {15} Защото зависят от ЦП **[Езиците!?!]**
128. 6) {17} Защото имат машинна зависимост **[И що за звяр е това?]**
128. 7) {17} Защото тяхната структу**ра** ~~от~~ завис**ят** от **компютарна** система.
128. 8) {23} Защото ЕПНР ползва команди които се разчитат от ЦП без нужда от транслация.

129. 173. Какво е характерно за ЕПВР и как още се наричат те?

129. 1) {до 3} ЕПВР – език за програмиране от високо равнище – те са точно разбираеми за тези които не ги познават.
129. 2) {9} ЕПВР се състои от начални (входни) данни, изходни, условни оператори.
129. 3) {9} Наричат се машино зависими. Не зависят от машините езици **[Като не зависят от машинните езици защо се наричат машинно зависими?]**
129. 4) {12} ЕПВР са машиннонезависими. ЕПВР още се наричат **макроасемблери**. Такива езици са Фортран, Кобол и други.
129. 5) {14, 17, 17, 18, 21, 23} Процедурно ориентирани – специализирани и неутрални, проблемно ориентирани
129. 6) {14} Езиците за програмиране от високо равнище са процедурно ориентирани, проблемно ориентирани. и техните подтипове са специализирано и процедурни и неутрално процедурни (идентификатори) ЕПВР са машинно-независими.
129. 7) {14} Проблемно-ориентирани и неосорални
129. 8) {17} Процедурни – специализирани и неутрални. Проблемно ориентирани.
129. 9) {17} ориентирани –специализирани и неитрални проблем но ориентирани
129. 10) {17} Проблемно ориентирани.
129. 11) {20} те се наричат проблемно ориентирани, машинно независими са
129. 12) {20} Характерно за ЕПВР (езици за програмиране на високо равнище) е че при тях се използват алгоритми за решаване , работят със стари програмни езици (ПЕ) чрез компилатори, още се наричат
129. 13) {23} ЕПВР – език за програмиране от високо равнище **Наричат се още машинно-зависими езици**. Характерно за тях е, че са алгоритмични.
129. 14) {23} ЕПВР са машинно-независими, още се наричат алгоретмични
129. 15) {26} По-близки са до човешкия език, Трудно се разбират от процесор транслатор и интерпретатор

130. 174. Защо ЕПВР се наричат и машинно-независими езици?

130. 1) {5–7} защото те работят със всякакви езици
130. 2) {14} Защото създават проблеми поради различието си.
130. 3) {15} Имат по-високо структурно ниво на инструкциите и машинна независимост.
130. 4) {15, 17, 26} Защото тяхната структура и наличните команди зависят от конкретната компютърна система
130. 5) {17} Имат такава **[Каква такава?]** структура и са машинно-независими
130. 6) {17} Имат по–високо структурно ниво на инструкциите и затова са и машинно независими.
130. 7) {20} ЕПВР е машинно-независим език, защото програмата може да бъде записана, но да не се преведе в МЕ и да не бъде компилирана.
130. 8) {21} Имат по-високо структурно ниво на инструкции и машинна независимост
130. 9) {23} Защото са езици от високо равнище.
130. 10) {24} Защото не зависят от конкретна компютърна **програма**.
130. 11) {26} ЕПВР се наричат и машинно- независими езици, защото

130. 12) {26} Защото тяхната структура и наличните команди зависят от конкретната компютърна система.

130. 13) {26} Защото се използват за изграждането на програми и представляват последователност от знаци, групирани и разпознати като последователни елементи. Всеки по-горен ел. се изгражда от елементите от по-долните равнища:

← програмни части	Осн. идея, е че 1 комп. програма на такъв език може да се
← оператори	изпълнява без промени на всеки компютър,
← изрази	за който има транслатор на съотв. език;
← величини	
← осн. символи	

130. 14) {27} Защото всичко зависи от програмиста и неговите способности

131. 175. Какви подкласове имат ЕПНР?

131. 1) {до 3} **Адитивни** и мултипликативни.

132. 177. Какво е характерно за автокодовете (асемблерните езици)?

132. 1) {8} – Че са ЕПНР
– използват 0 и 1 (двойна ПБС)
– машинен език

132. 2) {10} 1:1

132. 3) {10} Всеки знак има определен адрес.

133. 178. Какво е характерно за Макроасемблерите?

133. 1) {8} Те работят като „ЕПВР“

133. 2) {10} Характерно за Макроасемблерите – 1:n

134. 179. Какви подкласове имат ЕПВР и кой от тях съдържа алгоритмичните езици?

134. 1) {9} букви, цифри, разделители, ограничители

134. 2) {9} Алгол–60 е Алгоритмичен език

135. 180. Какво е характерно за процедурно ориентираните езици?

135. 1) {до 3} Проблемите се решават процедурно.

135. 2) {9, 9} как се решават

136. 181. Какво е характерно за проблемно ориентираните езици?

136. 1) {до 3} Тези езици са ориентирани към решаване на отделни проблеми.

136. 2) {до 3} Характерно за тях е това, че са неразбираеми за централния процесор. За да бъдем разбрани от централният процесор трябва да му осигурим превод на нашия език като преводачът може да бъде и програма.

136. 3) {5–7, 5–7, 5–7} каква е

137. 182. Какви могат да бъдат процедурно ориентираните езици?

137. 1) {8} Си, Паскал, C++

137. 2) {8} – от вън на данните
– от вътре на данните

138. 184. Какво е характерно за неутралните процедурно ориентирани езици и какво друго име бихте им дали?

138. 1) {10} Главно за създаване на базов софтоер.

138. 2) {12} Те са машинно независими и решават алгоритмични проблеми **[Как може един език да решава някакъв проблем?]**

138. 3) {12} Неутралните процедурно ориентирани езици **зависят** от съответната компютърна система. Те могат да се наричат още **индеферентни**.

138. 4) {14} Не са създадени с цел да извършват конкретни действия. Можем да ги наречем още и индеферентни.
138. 5) {14} Те не са създадени, за да извършват конкретна дейност и се наричат индеферентни. **[Кой ли език е създаден за извършване на каквато и да е дейност?!?]**
138. 6) {14} Упростени
138. 7) {17} Изискват детайлно **описание на алгоритъма за описание** на задачата, наричат се още алгоритмични езици.
138. 8) {17} **Не са създадени** с цел **да извършват конкретни дейности**. Може да им се даде името – Индиферентни.
138. 9) {17} **Не са създадени** с цел **да извършват конкретни действия**. Може да ги наречем още инисерентни.
138. 10) {17} Действието им **[Как ли си представя авторът на този отговор „Действие на език“?]** не е свързано с точно един определен вид компютър и МЕ. Общоизползваеми.
138. 11) {20} Езици за програмиране от високо равнище
138. 12) {21} Неутрално ориентираните езици са универсални те **[Т. е. езиците !?!]** създават софтуер който решава всякакъв вид задачи

139. 186. Какъв проблем създава използването на собствен ЕП при писане на програма за компютър?

139. 1) {9} води до появата на грешки
139. 2) {20} Собственият език **може да не бъде разбран и разчетен от програмата**.
139. 3) {27} При използването на собствен език за написване на програма, може да не се разчете и изпълни от автомата (компютъра), тъй като е несъвършен и може да съдържа многозначни и двусмислени изрази които не могат да бъдат изпълнени

140. 187. Какво е необходимо за да бъде разбрана от ЦП програма, за чието написване сме използвали собствен ЕП?

140. 1) {8} Предварително определяне на операциите
140. 2) {20} Необходимо е ЦП да поддържа езика, на който програмиран.
140. 3) {23} Компилативен транслатор или интерпретатор също така човек който разбира от машина език за програмиране
140. 4) {26} Трябва да намерим някой, които превод е опит за обединяване на предимствата на двете схеми с цел елиминиране на техните недостатъци.

141. 188. Посочете поне два проблема при превод от един човешки език на друг?

141. 1) {14} Разлика в основите на езика, липса на точност и еднозначност.
141. 2) {17} Липсват кон
141. 3) {20} Човешките езици се различават един от друг по много фактори граматика корен произход и много други. При превод от един език на друг съществува реална опасност от двусмислие и объркване на смисъла на презначеното, което се превежда. Освен това могат да се допуснат грешки и в цеста на преведения текст
141. 4) {20} Липсва еднозначност; става двусмислие
141. 5) {20} Използване на различна знакова система, различни правила при използване на езика.
141. 6) {21} Неточност, иероглифи.
141. 7) {23} загуба на предиози; промяна на смисъла
141. 8) {26} а) При превеждане един човешки език на друг преводач трябва да знае двата езика
б) При преведена дума може да я няма в другия език, или значението и ще бъде неточно
141. 9) {27} Интерпретаторите не могат да осигурят аналогична възможност тъй като при тях преводът и изпълнението се съвместяват.

142. 189. Какви са общите черти на всички алгоритмични езици в сравнение с човешките (поне 2)?

142. 1) {10} При превод се получават по малко синтактични грешки по лесен превод. По разбираем

142. 2) {17} Трудно е да се направи пряко сравнение **[Когато не си научил нищо не е трудно, а безумно трудно!]**, но говоримите писмени човешки езици си приличат с алгоритмичните по това, че имат определени правила и структура. Ако обсъдим въпроса дори по-косвено, може да се приеме, че както човешките ни езици ни помагат да се разбираме помежду си, така и алгоритмичните помагат за връзката човек–машина или дори машина–машина. **[В този отговор е представен изключително богат набор от общи черти!!!]**

142. 3) {18} И двата езика имат писмена част (при алгоритм. езици липсва говор), друга обща черта е тяхната линейност. Алгоритмичните и човешките езици е тяхната линейност. Алгоритмичните и човешките езици имат сходни компоненти (елементи) (букви – осн. символи; секции – модули и др.)

142. 4) {19} дискретност,

142. 5) {27} Знаци и смисъл

142. 6) {28} Че са точни за изразяване. В описанието на един алгоритъм не може да има променливи Те са най-простите алгоритми с който започва изучаването на всеки процедурен ЕП.Действията записани в такива алгоритми се използват само веднъж

143. 191. Как се наричат програмите, които превеждат от ЕП до МЕ?

143. 1) {до 3} Програмите се наричат **изпълними**.

143. 2) {14, 14} Компилатори.

143. 3) {17} Програмите, които превеждат от език за програмиране до машинния език се наричат коммилационни

143. 4) {18} От ЕП до МЕ – ~~програмата не може да бъде извършена от човек~~ програмата която трябва да извършва – не може да бъде извършена от човек.

143. 5) {20, 20} Програмите, които превеждат от ЕП до МЕ се наричат компилтори.

143. 6) {22} Компилатори.

144. 192. Какви схеми на превод познавате при човешките езици?

144. 1) {8} блок-схеми, диаграми, графики

144. 2) {9} транслатори
компилатори

145. 193. Какви схеми на превод може да се използват от транслаторите?

145. 1) {5–7} МЕ към естествен –
естествен език към МЕ.

145. 2) {12} те могат да превеждат дума по дума или цяло изречение

145. 3) {17} Схемите на превод които могат да се използват от транслаторите могат да осъществяват както от човека, така и от компютрите. За по-лесен превод се предпочитат различните програмни продукти. Те превеждат с по-голяма точност, без граматически грешки и по-бързо. Човек, който превежда писмена част първо я прочита и при превода може да допусне доста и значими грешки, които са отзначение за разбираемостта на текста. При говоримия превод се извършва на части – първо се слуша и после превежда. Преводът от компютър е много по-точен и верен и много по-бърз и ефективен.

145. 4) {20} **Всякакви** схеми могат да се използват от транслаторите за превод.

145. 5) {21} ① Прочита се всичко и се появява нов компиративен екземпляр
② Слуша се **говорещия** и се превежда на части това се нарича интерпретатор

145. 6) {26} Цялостна (ползва се от опитни програмисти, когато преводът ще бъде използван многокр.)
Частична (по-сигурен вариант, защото лесно се оправят грешки)

146. 194. Какво е характерно за компилативните транслатори (компилаторите)?

146. 1) {5–7} Компилаторите за разлика от транслаторите превеждат програмата на машинен език на части, а не изцяло при транслаторите. Така се губи време, но са подходящи за учебни цели, защото спират изпълнението на програмата до опасната грешка.
146. 2) {9} Компилаторите превеждат програми **[А какво правят интерпретаторите?]**
146. 3) {10} При тях е гарантирано че няма синтактични грешки
146. 4) {14} Проверяват програмата
146. 5) {15} Привеждат всичко наведнъж
146. 6) {17} Компилативните транслатори – гарантирано е, че програмата няма синтактични грешки
146. 7) {17} Компилативните транслатори служат за превод
146. 8) {20} Характерно за тях, е че при обработка на информацията се трансформират **[Компилаторите !!]** в двоичен код съставен само от 0 и 1
146. 9) {20} **Компилаторите компилират програмата**, задействат написания код **и я привеждат в действие**
146. 10) {23} Превежда целият текст и го записва във файл, за да може пая да го извика в програмата без да се налага повторно компилиране

147. 195. Какво е характерно за интерпретативните транслатори (интерпретаторите)?

147. 1) {до 3} Транслаторите – когато използваме собствен език за програмиране.
147. 2) {до 3} Бързо получаване на желания резултат.
147. 3) {5–7} Превежда програмите на машинен език. Слуша се на даден език и след това се превежда.
147. 4) {14} Интерпретативните транслатори (интерпретаторите) превеждат цялата програма, а не на части. Така те спестяват време; Те са подходящи за използване в работна среда; Те се оптимизират и са подходящи за многократно повторение и използване. **[Дали има смисъл да се създава превеждаща програма за еднократно използване?!?]**
147. 5) {17} Превеждат се само участъци, които ще бъдат изпълнени.
Не е необходимо предварителен **превод на механичен език**.
По време на изпълнението имаме **пълна връзка с текста на прогреса**.
147. 6) {17} Превежда само участъците от програмата, които ще се изпълнят; Не е необходим превод на машинен език; По време на изпълнение има връзка с текста на програмата
147. 7) {17} Определяне на смисловата стойност и своиствата
147. 8) {17} Превеждат се само участъците които ще бъдат изпълнени. Не е необходимо предварителен пр~~ев~~да на МЕ по време на изпълнението имаме пълна връзка с текста на програмата
147. 9) {17} Характерно за тях, че са програми който изследват, чрез въвеждане на база от данни се извеждат определен интерпретатор.
147. 10) {17} Превеждат се само участъците които ще бъдат разглеждани. Не е необходимо предварителен превод на МЕ. По време на операцията имаме пълна връзка с текста
147. 11) {17} Превеждат се само участъците които ще бъдат изпълнени. По време на изпълнението имаме пълна връзка с текста **[Кой текст?]**
147. 12) {18} При тях се превеждат само участъци, които ще бъдат изпълнени. Не е необходим предварителен превод на МЕ. По време на изпълнението имаме пълна връзка с текста на програмата
147. 13) {20} Прилеждат се само участъците, които ще бъдат изпълнени. Не е необходимо преведан на машинен език.
147. 14) {20} Те правят връзката (превеждат) между езикът за програмиране и машинният език **[Що ли правят компилаторите?!?]**. При тях не е нужно да изпълняваме цялата програма, а само частта която ние нужна. Подходящи са за учебни цели.
147. 15) {21} Те са по-бавни и по-трудно използваеми от човека.

147. 16) {23} Интерпретаторите превеждат и изпълняват само малка част от текста, например дадена функция като след това процесът се понтаря отново
147. 17) {23, 24, 25} Интерпретаторите от своя страна превеждат и изпълняват само малка част от текста, например дадена функция, като след това процесът се понтаря отново.
147. 18) {26} Изпълняват (пренасят) данните записани в коя да е променлива извикана с различно име
147. 19) {27} Превеждат част от текста и зписва в паметта, като по-късно може да бъде извикана отново, превежда само най-нужното.

148. 196. Какво е характерно за смесената схема на превод от ЕП до МЕ?

148. 1) {5–7, 5–7} Обединяване предимствата на двете схеми. Елиминиране на техните недостатъци.
148. 2) {5–7} Обединяване на предимствата и на двете схеми. Елиминиране на техните недостатъци
148. 3) {9} Обединяване на предимствата и недостатъците. **[Първо „На какво?“, и второ „За какъв дявол трябва предимства да се обединяват с недостатъци?“]**
148. 4) {9} Характерното при смесената схема от ЕП и МЕ, че грешките са по-малко. **[Защо?]** Преведенит от от еднате не нужно да се привеждат от другата, но бърз превод.
148. 5) {9} Обединяване на предимствата на двете схеми **[Кои две схеми?]** елеминиране на техните недостатъци
148. 6) {9} Обединяване на предимствата и на двете схеми **[Кои две схеми?]** – елиминиране на техника **[Каква?]**
148. 7) {14} опитва да обедини предимствата на компилативната и интерпретативната схема и да елиминира техните недостатъци.
148. 8) {14} ЕП е толкова близо до МЕ колкото и до човешкия ез. и се опитва да съчетава двата **[Какво „двата“? Езика?!?!]**
148. 9) {14} Събира техните **[Кои техни??]** свойства и извлича най-важните, и необходими за извършване на алгоритъма
148. 10) {14} Смесената схема се стреми да премахне недостатъците.
148. 11) {14} Съчетава предимства на интерпретативната и компилативната схема.
148. 12) {17, 17, 17, 18, 20, 23} Опитва да обедени предимствата на двете схеми и да елиминира техните недостатъци
148. 13) {17} Характерното е разликата между ЕП и МЕ
148. 14) {19} Смесената схема прави опит за обединяване на предимствата и елиминиране на недостатъците на компилативната и интерпретативната схема за превод.
148. 15) {20} Характерно
148. 16) {20, 20, 21} Смесената схема на превод е опит за обединяване на предимствата на двете схеми (компилативната и интерпретативната) с цел елиминиране техните недостатъци.
148. 17) {21, 26} Смесената схема от превод е опит за обединяване на предимствата на двете схеми (интерпретативна и компилативна), с цел елиминиране на техните недостатъци.
148. 18) {23} Тя е опит за обединяване на предимствата на двете схеми компилативна и интерпретативна с цел елиминиране на техните недостатъци
148. 19) {23} Смесената схема на превод е опит за обединяване на компилативната и интерпретативната схеми с цел елиминиране на техните недостатъци.
148. 20) {23} Смесената схема на превод съдържа както машинен език, така и език за програмиране от високо ниво.
148. 21) {26} См. схема на ЕП до МЕ е превод да обединяване на предимствата на двете схеми компилативната и интерпретативната с цел на техните недостатъци
148. 22) {26} Смесената схема на превод е опит за обединяване на предимствата на двете схеми с цел премахване на недостатъците им

149. 197. Посочете поне три особености на човешките езици.

149. 1) {10} – произношение
– граматика
– различна знакова азбука

149. 2) {12} Масовост, многозначност, неразбираеми за ЦП

149. 3) {23} Могат да се изказват от човек на човек
Могат да се изписват
Породени са много преди ЕП.
Съществуват диалекти във всеки един език.

149. 4) {26} Човешките езици са:
– писмени
– гонорими

150. 198. Посочете поне три особености на ЕП.

150. 1) {14} ЕП са характерни с това че алгоритмите, които се използват в тях са крайни, не двузначни и резултатни

150. 2) {14} Езиците за програмиране се разделят на езици за програмиране от високо равнище и езици за програмиране ниско равнище;

150. 3) {14} обработка, масовост

150. 4) {17} 1. Те биват ЕПВР (високо равнище) и ЕПНР (ниско равнище)
2. Всеки език за програмиране има свой синтаксис и семантика
3. ЕП е машинен език

150. 5) {17} Алгоритмично ориентирани
Обектно ориентирани
Интернет ориентирани

150. 6) {26} Предимството на **ЕП** е, че **са бързи** а недостатъка им е, че са енергозависими

151. 199. Посочете предимства и недостатъци на компилативната схема за превод.

151. 1) {9} Предимства – нагледна и разбираема.
недостатъци – машината не я чете

151. 2) {9} По бърз, но по трудно разбираем.

152. 199A. Посочете поне две предимства на компилативната схема за превод.

152. 1) {12} Компилативната схема се използва например при промишлена обстановка, защото няма нужда от интерпретиране на текста, а само чрез четене представя бързина.

152. 2) {14} няма синтактични грешки **[Схемата!?!]**

152. 3) {14} Превръща написания код в машинен

152. 4) {20} Три вида

152. 5) {26} **Изпълнява** се много бързо

153. 199B. Посочете поне два недостатъка на компилативната схема за превод.

153. 1) {17} – Компилира целият програмен код (превежда) едновременно (дословно)
– Понякога трудни за откриване грешки
– Изискване на повече ОП за цялостно изпълнение.

154. 200. Посочете предимства и недостатъци на интерпретативната схема за превод.

154. 1) {9} предимства: бърз превод
недостатъци: допускат се грешки **[Какви? Защо?]**

155. 200A. Посочете поне две предимства на интерпретативната схема за превод.

155. 1) {14} При интерпретативната схема за превод информацията се запазва максимално точно.

155. 2) {14} 1. Възможността даден текст данни да бъдат преведени (интерпретирани в друг език? [Въпросителният знак (?) е част от отговора!]

2. Възможността даден алгоритъм, данни да бъдат въведени от всеки език в компютъра и чрез нужните ключови думи те да бъдат разчетени и преведени на компютърният език

155. 3) {27} ENIAC

156. 200B. Посочете поне два недостатъка на интерпретативната схема за превод.

156. 1) {10} Интерпретативната схема за превод може да е непълна и неточна

156. 2) {15} Първият недостатък че превежда буквално, а не смислено

Вторият поради това, че думите имат мн. значения не винаги слага правилното значение

157. 201. Посочете кога трябва да използваме компилатор и кога интерпретатор при превод на програмата стига да разполагаме и с двата? Защо?

157. 1) {10} а) Компилатор се използва, когато искаме да няма синтактични грешки. Интерпретатора превежда само текста, който ни е необходим. Използването и на двата обединява техните предимства и изключва техните недостатъци.

157. 2) {10} б) При компилативната схема еднократен превод, без синтактични грешки Недостатъка е че тази сист. за превод използва много време, и че се превеждат очастъци, който немат да бъдат използвани.

Интерпретивната схема за превод, бързо полочава резолтат и има, пълна връзка с текста, недостатъците са, че има синтактични грешки и бавно изпълнение

158. 201A. Кога трябва да използваме компилатор при превод на програмата стига да разполагаме с компилатор и с интерпретатор?

158. 1) {12} Когато ни е необходима по-висока точност на превод.

158. 2) {21} При промишлена обстановка (професионална работа) и учебна обстановка (еднократно) **[Скъпи читателю, от този отговор на теб веднага ти стана ясно като в гъста есенна мъгла кога трябва да се използва компилатор!!!]**

158. 3) {23} Когато искаме да преведем цялата програма с цел по-късно да можем да работим по-бързо с нея; когато искаме да има възможно най-малко грешки в програмата ни. Ако използваме за превеждане интерпретатор е възможно грешките да останат скрити за прекалено дълго време.

159. 201B. Кога трябва да използваме интерпретатор при превод на програмата стига да разполагаме с компилатор и с интерпретатор?

159. 1) {17} Използваме интерпретатора, когато искаме задачата да бъде проверявана на части, тъй като компилатора минава през цялата задача.

160. 201C. Защо при учебна и развойна обстановка трябва да използваме интерпретатор при нейния превод?

160. 1) {20} Чрез интерпретатора по-лесно се откриват грешки в текста на програмата. Дава възможност да се изпълни само една част от програмата без да е нужно да се превежда цялата.

160. 2) {20} За да разчете компютъра дадената програма.

160. 3) {20} В учебна и развойна обстановка трябва да използваме интерпретатор, защото е по-бърз и по-неточен

160. 4) {23} При учебна и развойна обстановка трябва да използваме интерпретатор при нейния превод, за да преведе информацията в Те превеждат малка част от текста, например дадена функция

161. 201D. Защо при подготовка за промишлено използване на програмата трябва да използваме компилатор при нейния превод?

161. 1) {14} За да се провери за грешка.
161. 2) {14} Използваме компилаторът за **да преведе написания от нас език**, на машинен език, разбираем за Централния процесор
161. 3) {14} Защото осигурява възможности за едновременно писане на няколко близки помежду си варианта.
161. 4) {14} Защото компилаторът или изчиства **[С метла ли ги чисти или с кофа и парцал?]**, или намера синтактичните грешки в програмата.
161. 5) {17} защото програмата се превежда по-бързо чрез компилация, само че е необходимо предварително да се преведе МЕ **[За какъв дявол трябва да се превежда МЕ? Нима ЦП не си го знае?]** и за това се отделя време в началото.
161. 6) {22} За оптимизация и избягване на допускане на грешки.

162. 204. Посочете поне три съображения при избор на схема за превод от ЕП до МЕ.

162. 1) {5–7} Обединяване на предимствата на двете схеми елиминиране на техните недостатъци
162. 2) {12} Може чрез блок схема,
162. 3) {14} 1) Компилатор
2)
3)
162. 4) {14} 1. Бързина на превода.
2. Еднократен или многократен превод
3. Със синтактични грешки или не.
162. 5) {14} 1) ако се работи в учебна обстановка – използване на компилативна схема
2) ако се работи в професионална обстановка – изп. на интерпретативна схема
3) ако искаме по време на изпълнението да не се губи връзка с текста на програмата – интерпретативна;
4) за съвместяване на предимствата на двете схеми – използваме смесена
5) при многократно изпълнение – използваме оптимизиращ компилатор.
162. 6) {17} Скорост на изпълнение, оптимален алгоритъм, преносимост
162. 7) {18} 1. Лесен за научаване
2. По-разбираем запис
162. 8) {20} 1) ЕП трябва да бъде лесно разбираем
2) ЕП трябва да може да се преобразува до МЕ.
3) ЕП трябва да са чете от програмата и потребителя
162. 9) {20} При избор на схема за превод от език за програмиране (ЕП) **до механичен език** (МЕ) трябва да се вземат под внимание избора на данни, които ще се обработват, да се избегнат двусмислици – да има точност, да се избягват синтактични грешки
162. 10) {20} Смесената схема на превод е опит за обединяване на предимствата на две схеми компилативна и интерпретативна с цел елиминиране на техните недостатъци.

163. 206. Посочете характерни черти на буквите, като елемент на човешките езици (поне 4).

163. 1) {5–7} Познати са на хората. Лесни за използване
163. 2) {5–7} – основен елемент
– формален критерии
определеното от речници не е пълно
163. 3) {8} – A = главна, a = редовна
– A – главна
– a – редовна
– в някои ЕПВР A=a (Паскал)
– в други ЕПВР A≠a

163. 4) {14} Еднозначност. Буквите също са в основата на съставянето на по-големи значими единици – срички и думи. **Буквите са в основата на НПБС**, като пример за това е римското броене, където цифрите се отбелязват с букви. Всяка буква може да се представи с уникален двоичен код.

163. 5) {17} – Буквите изграждат другите елементи на естествените езици
– Буквите могат да бъдат главни (А, Б, В) и малки (а, б, в) …
– Краен брой са

163. 6) {20} диалект, могат да се променят с времето, четими и говорими **[Буквите!?!]**

163. 7) {23} Могат да се изписват и да се изговарят.
С формирането на различни букви се формират и нови езици.
Основен градивен елемент на езика. Съществува формален критерий за отделяне на думите.

163. 8) {23} Писмени и говорими;
Имат подмножества
Също имат диалекти
Почти не се променят с времето

163. 9) {26} 1) Писмени и говорими;
2) Имат подмножества
3) Имат диалекти
4) Живите се променят с времето

164. 207. Посочете характерни черти на думите като елемент на човешките езици (поне 3).

164. 1) {11} – писмени езици, подмножества, има диалекти

164. 2) {14} 1) Думите с времето се променят
2) имат жаргони
3) има два вида на изписване с малка и голяма буква

164. 3) {14} – те са основен градивен елемент на изреченията;
– имат собствено значение
– могат да се записват с главни и редовни букви

164. 4) {15} Характерни черти на думите като елемент на естествените езици: съществително име, прилагателно име, глагол.

164. 5) {17} програмиране, данни, константи

164. 6) {20} те са от линейно естество; уникални са за различните езици; съществуват различни диалекти, невинаги носят значения по подразбиране

164. 7) {20} Букви, Изречения, Глави

164. 8) {20} – Думите са начинът, по-който е възможно изразяването на човешките езици
– Думите помагат за комуникацията
– Думите помагат за разбирането в човешките езици

164. 9) {20} Характерние чертив слов елемент на человеческом езике
– диалект
– читаещие и говоримые
– они могут изменяется со временем;

164. 10) {23} делят се на бяло поле, съставна част на езика

164. 11) {26} многозначност

164. 12) {26} Посочва се действие и условие

164. 13) {27} Знак, действие и смисъл

165. 208. Посочете особеностите на изреченията като елемент на човешките езици.

165. 1) {14} Изреченията са важна част от естествените езици.
Те биват прости и сложни. Може да съдържат логически функции (конюнкция, дизюнкция…). Всяко изречение се отделя за да не се получи двусмислие.

165. 2) {14} Изреченията в естествените езици са с линейна последователност. те нямат характерни особености.
165. 3) {14} Изреченията нямат конкретни особености
165. 4) {14} няма особености
165. 5) {17} Всяко изречение се отделя с определени знаци (точка, запетая и т. н.)
Всяко изречение започва с главна буква
Изреченията биват прости или сложни.
165. 6) {20} Особеностите на изреченията, като елемент на човешките езици са: да са смислени, да имат глаголи за да бъде изречение; то може да бъде просто, сложно, сложно съставно, сложно с разширение,
165. 7) {20} Писмени и говорими, имат диалекти, имат подмножества
165. 8) {23} Няма характерни особености на изреченията като елемент на човешките езици
165. 9) {24} 1. Прочита се всичко и след това се поевява нов екземпляр но вече на другия език:
2. Слуша се говорещият на дадения език и казаното се превежда на части.
165. 10) {26} – Писмени и говорими;
– Имат подмножества;
– Имат диалекти;
– Живите се променят с времето;
165. 11) {26} Изградени са от думи и съюзи. **[Нима съюзите вече не са думи?]** Изграждат смислов текст
165. 12) {26} Езикът може да отчета особеностите на описание с него алгоритми и е линеен.

166. 209. Посочете особености на параграфите като елемент на човешките езици (поне 2).

166. 1) {14} Служат за обособяване на отделни части от текста, като групират определена информация по даден признак – обикновено по признак тема
166. 2) {16} **Параграфът отделя** по-ясно дадена мисъл или **действие**. **Езикът става** по-разбираем и **четлив**.
166. 3) {17} Параграфите са основен елемент на текста и се отделят с малко отстояние един от друг и от рамките на листа.
166. 4) {18} Отделят се един от друг и от размерите на листа с малко разстояние; Градивна част на текста
166. 5) {20} Като особености могат да се разгледат
166. 6) {20} 1. Разделят текста на отделни части
166. 7) {23} 1. Образуват текстове
2. Образуват се от редица изречения
166. 8) {26} Параграфите разделят текста на части и го правят по прегледен.

167. 210. Посочете особености на секциите като елемент на човешките езици.

167. 1) {12} Секциите се състоят от изречения. Всяка секция започва с определен символ (за празно място)
167. 2) {12} Буква → дума → изречение → параграф → глава → текст
167. 3) {14} **Естествените езици**, както и програмите **се състоят от различни части**. Секция е част от естествените езици, която се състои от изречения. Те **[В множествено число са само „естествените езици“. Следователно „те“ замества тези езици!]** се отделят от останалите части на речта чрез специален знак, който е невидим и в текст представлява празно пространство на нов ред.
167. 4) {20} Елементите на човешките езици са:
1.) писмени и говорими
2.) имат подмножества
3.) имат диалекти
4.) живите могат да се променят с времето

167. 5) {23} Секциите се използват само при големи текстове. При тях се използват и съкращения. Секциите изискват голяма по обем памет.
167. 6) {23} текст ← секции ← параграфи ← изречения ← думи ← букви; от дясно на ляво всяко е съставено от елементите на неговата дясна страна.
167. 7) {26} Писмени и говорими, имат подмножества, ичат диалекти живите се променят с времето

168. 212. Посочете елементите, от които се изграждат ЕПВР в техния йерархичен ред.

168. 1) {14} начало (входна информ.) (междина инф с данни, край (изходна инф.)
168. 2) {14} Оператори, функции, типове данни, променливи
168. 3) {20} символи → операнд → операция → структура → програма
168. 4) {21} Текстове
– глави
– параграфи
– изречения
– думи
– Букви
Всеки по-горен ред е поредица от елементите, непосредствено под него.
168. 5) {26} Всеки по-торен елемент се изгражда чрез елементите от по-долните равнища.
168. 6) {26} Имената на начерините константи се определят от програмата.
168. 7) {27} позиционни и непозиционни

169. 213. Какво означава терминът „разделна компилация“ и кога се използва?

169. 1) {14} компилативни транслатори които могат да превеждат само отделна програмна част.
169. 2) {14} Терминът разделна компилация означава, че се изпълняват командите на една определена част от цялата програма
169. 3) {14} Разделната компилация разделя програмата на части, използва се при по-големи програми.
169. 4) {15} Някои компилатори (транслатори) могат да извършват разделна компилация. Използват се при разделяне на програмата на части.
169. 5) {17} (Не е възможно) Често се случва едни и същи програмни части да съдържат еднакъв текст. Вместо да се дублира **[Защо да се дублира като програмните части са едни и същи и текстът е един и същи??]** е пдобре да се запише в отделен файл
169. 6) {17} Разделна компилация е когато програмата бива разделена на части, за да могат няколко човека да работят едновременно върху нея.
169. 7) {26} Някои транслатори могат да превеждат и отделна програмна част
169. 8) {27} Разделна компилация – събиране на дадените системи поотделно

170. 214. Какво представлява една програма от гледна точка на ЕПВР?

170. 1) {до 3} Това е компютърна програма, която е написана на такъв език, така че да може да се изпълнява без промени на всички компютри, за които има транслатор от съответния език.
170. 2) {до 3} Програмата от гледна точка на ЕПВР представлява съвкупност от действие за извършване на дейност.
170. 3) {10, 10} Статично обединяване на програмните части в програма се изпълнява от свързващ редактор
170. 4) {12} Една програма от гледна точка на език за програмиране ВР. Трябва да може да се изпълнявана всеки компютър, защото е независима от машинния код. Тя може да се разделя на части. блок, процедура (функция), модул.
170. 5) {14} Програмата е нещото, което кара компютъра да върши полезна работа
170. 6) {14} Програма, която е подробно описана техната структура и налични команди не зависят от конкретната компютърна с-ма.
170. 7) {14} Алгоритъм да се представи във форма и вид и след определен брой елементарни действия да се изведе резултата

170. 8) {17} Програма на ЕП, което кара компютъра да извършва полезна работа.
170. 9) {17} Основата (базата) на програмите от ЕПВР са интегрални **схеми с висока степен на активност**
170. 10) {18} Процедурно ориентирани – неутрални и специализирани, по проблемно ориентирани
170. 11) {20} Алгоритъм във форма и вид.
170. 12) {20} Програмата е това, което кара компютърът да върши полезна работа.
170. 13) {21} Входова и изходна част с редове от команди, които да бъдат изпълнени
170. 14) {23} Програма предзставрява последователност от операции които след края на изпълнението връщат някакъв резултат.

171. 215. Какви решения за писане на програмата предлагат ЕПВР по отношение на редовете, чрез който се пише тя?

171. 1) {5–7, 5–7} Има изискване определени елементи да се изписват само на един ред.
171. 2) {9} Влиянието на оператори F влошава четенето и разбирането на програмата, и изискване на множество изрази което не е рационално, и може да не е
171. 3) {9} физически редове и логически редове
171. 4) {14} Писането на програма с ЕПВР е улеснено, по отношение на редовете, защото за изпълнението на самата програма няма значение разстояния, табулации и редове.
171. 5) {17} Потребител на един алгоритъм **[Дъхът на читателя замира пред този брилянтен отговор!]**
171. 6) {23} В някои езици за програмиране има специален знак за **продължене** на реда.
171. 7) {26} При някои ЕП има изискване определени елементи да се изписват изцяло в рамките на един ред. Подобно ограничение налага използването на специален знак за продължаване на ред.
171. 8) {26} Може да се пише от ляво надясно, някои програми изискват да бъдат написани на 1 ред и затова т има определен символ за писане на 1 ред. (продължава реда)
171. 9) {26} Изпълняване на програмена последователно (от първи ред) и непоследователна (записване от посочен ред)

172. 216. Какви са особеностите на речниковия запас на ЕПВР?

172. 1) {до 3} ЕПВР са създадени смисълта, че една ковпютърна програма написана на такъв език ще може да се изпълнява без промени на всеки компютър, за което има транслатор от съответния език.
172. 2) {до 3} При ЕПВР езиците осигуряват разглежданата възможност чрез специални конструкции посочващи, че при определени обстоятелства част от текста се третира като коментар. При липса на съответните обстоятелства такива части естествено участват в текста и се превеждат.
172. 3) {9} В ЕПВР за всеки символ има точно определен знак
172. 4) {14} Определя предварително зададена грешка
172. 5) {14} Речниковия запас на всеки ЕПВС е уникален и обикновено не се среща в друг ЕПВР
172. 6) {15} **Изразите** в ЕП **се механизират** за създаване на нови стойност
172. 7) {20} Лесно изпълними.
Имат диалекти както в говоримия език
Написани са от хора

173. 217. Какво представляват коментарите и каква е тяхната роля при писане на програми?

173. 1) {15} Понеио в писането на програма се използват и бележки (коментари). Те не се испълняват от програмата
173. 2) {20} Тъй наречените етикети позволяват на дадена програма да се добавят коментари. Те не се изпълняват от програмата.

174. 218. Има ли съответствие между елементите на човешките езици и тези на ЕП? Ако не – защо, ако да – какво е то?

174. 1) {15} Има съответствие между тях при Vsua-Basic
174. 2) {20} Не, защото при човешките езици се допуска да има двусмислие докъто при ЕП не се допуска.
174. 3) {20} Не, защото езика за програмиране е по труден от човешкия език

175. 219. Какво е предназначението на основните символи на един ЕПВР?

175. 1) {12} Основните символи в ЕПВР **свързват изразите в кода на програмата**
175. 2) {14} Програмата представя алгоритъм в нужния му вид и форма, в които той може да бъде прочетен от компютъра.
175. 3) {14} Основните символи са предназначени **за означаване** на отделните **елементи**.
175. 4) {14} за направата на една програма
175. 5) {15} Основните символи (букви, цифри, ограничители, разделители) служат за обозначаване на елементите (оператори) и немогат да се използват за друго. Те се различават чрез различни графични знаци, а когато липсват от клавиатурата се правят с знаков низ т. е. няколко знака.
175. 6) {17} С тях се **извежда** кода на програмата
175. 7) {17} Служат за означаване.
175. 8) {18} Те не могат да се променят и показват как трябва да се показва дадено нещо
175. 9) {20} Основните символи биват цифра буквени знаци; ограничаващи знаци; знаци за действия; команди
175. 10) {23} Основните символи са предназначени за изписване на алгоритми.
175. 11) {26} Служат за на останалите елементи в езика

176. 221. Какво е желателно да се избягва при подбора на основните символи при създаване на ЕПВР?

176. 1) {17} Желателно е да се избягват повторения и да се допускат грешки
176. 2) {20} Трябва да се внимава с командите за конкретната компютърна система
176. 3) {26} Желателно е да се избягва повторението

177. 222. Каква е най-изгодната стратегия за реализиране на основните символи на един ЕП?

177. 1) {10} **Всеки символ** отговаря на **някоя цифра**
177. 2) {12} Най-изгодната стратегия по отношение на основните символи на ЕПВР е, че **пишещият програмата е улеснен** в създаването ѝ, **благодарение** именно **на основните символи**.
177. 3) {12} Да бъдат общи за всички машини? ASCII
177. 4) {14} Използване на различни символи, **[Символ е нещо което означава друго нещо. Например: Червената светлина на светофара означава забрана за преминаване.]** за лесно различаване на операциите. Тъй като единичните символи са ограничен брой, се използват поредици от символи.
177. 5) {26} Най-изгодната стратегия за реализиране е символите да са с неизвестен ред на изпълнение

178. 223. Възможно ли е да се прилага най-изгодната стратегия при определяне на основните символи на един ЕПВР? Защо?

178. 1) {15} ИзГОДНА стратегия е за всеки основен символ да има отделен графичен знак
178. 2) {15} Да. Защото се използват различни един от друг символи.
178. 3) {16} Изгодна стратегия е за всеки основен символ да има отделен графичен знак
178. 4) {17} Да, но се постига трудно, защото броят е доста голям или липсват необходимите клавиши **[За какво са необходими тези клавиши и какви са те!?!]**
178. 5) {17} Да, защото така алгоритъма става по-лесен за модифициране

178. 6) {17} Да, защото ЕПВР – е език от високо равнище – за **по-бързо разработване на данни**.
178. 7) {20} Разбира се, че е възможно да се прилага най-изгодната стратегия при определяне на исновните символи на един ЕПВР. Защото
178. 8) {26} Да, всеки потребител може да избере най-удобните за него символи.

179. 224. Как се постъпва когато за въвеждане на основните символи на един ЕПВР липсват клавиши на клавиатурата?

179. 1) {8} в този случай на буквите се поставят спец. символи.
179. 2) {14} **Използва се оператор** които има име, което е поредица от символи.
179. 3) {14} Ако липсват клавиши на клавиатурата, можем да въведем основните символи чрез техните ASCII кодове
179. 4) {17} **Използва се таблица** за въвеждане на отделни символи, **интегрирана в програмата**
179. 5) {17} Използват се функции за въвеждане от мишката.
179. 6) {20} Намират се в базата данни на програмата
179. 7) {20} Прескачаме ги или търсим друга клавиатура.

180. 225. По какъв критерий се класифицират основните символи на един ЕПВР?

180. 1) {до 3} ЕПВР са създадени с мисълта, че една компютърна програма, написана на такъв език ще може да се изпълнява без промяна на всеки компютър.
180. 2) {5–7} По множеството на допустимите стойности (МДС)
180. 3) {10} – Букви
– цифри
– знаци
180. 4) {13} основни символи → ↗ Букви (a, б) / → Цифри (1, 2) / ↘ знаци (^, *, _)
180. 5) {14} Символите на един ЕПВР могат да бъдат **константи, променливи, оператори**
180. 6) {15} Чрез прилагане на изгодна стратегия
180. 7) {17} букви → цифри → разделители → ограничители
180. 8) {17} ~~Уникалност, четливост~~. Те трябва да са уникални (да нямат две значения) и да са разбираем за този, който пише програмата
180. 9) {19} Критерийте, по които се класифицират основните символи е да избере една от две идеални точки.
180. 10) {20} Основни символи – букви (A – Z; a – z)
– цифри (0 – 0)
– разделители
– ограничители
180. 11) {20} По типа се класифицират основните символи на ЕПВР.
180. 12) {21, 21} Критерий са символите.

181. 226. Какви класове основни символи съществуват?

181. 1) {14} Букви, думи изречения, параграфи, глави
181. 2) {17} буквени, числови, абстрактни (*)
181. 3) {21} Класовете биват: буквени означения, цифра, препинателни знаци, знаци за действия.
181. 4) {26} букви, цифри и знаци

182. 227. Какво влияние оказва маниерът на изписване на буквите при писане на програма?

182. 1) {14} Може да се създаде проблем с двусмислието
182. 2) {22} Променя синтаксиса

182. 3) {26} При писане на програма, трябва да се внимава с изписването на големи и малки букви защото едни и същи букви с различна капитализация може да се четат различно от компютъра.
182. 4) {26} UNIVAC
182. 5) {26} Програмистът трябва да изписва правилно програмата за да е четима
182. 6) {27} В различните езици функциите се пишат по различен начин, в някои с главня буква в други с малка.

183. 228. Равноправни ли са човешките азбуки по отношение на писането на програми?

183. 1) {5–7} По отношение на програмирането човешките азбуки са необходими за записване за изрази, величини – в това отношение са равноправни. Човешкият език като цяло не е равноправен на езиците за програмиране, защото допуска двусмислие.
183. 2) {14} Има само две възможности **[Брилянтен отговор!?!]**
183. 3) {14} **Някои** човешки **езици** са равноправни.
183. 4) {17} Да равноправни са
183. 5) {20} Не са равноправни, защото загубата на човешки труд се компенсира от работата на програмата **[Обяснението прави отговора забележителен шедьовър!]**
183. 6) {20} Не са равноправни гона има липса на граматични конструкции и съответствие по думи.**[Обяснението прави отговора напълно грешен!]**
183. 7) {20} Първата **азбука, която формулира алгоритми** е математиката
183. 8) {23} Да. За да можеш да програмираш трябва да знаеш само ЕП, ЕП не се влияе от човешките азбуки.
183. 9) {27} Не

184. 229. Как се различават еднаквите елементи, които са необходими при писане на програми?

184. 1) {10} Те са съществени (основен символ – не влияят на записа
184. 2) {11} По техните адреси / тип.
184. 3) {12} За различаване на еднаквите елементи в програмата се използват допълнителни символи (~, _, …)
184. 4) {14} Еднаквите елементи се различават по стойности.
184. 5) {14} В редки случай при създаване на програма се използват еднакви елементи, защото им се добавят допълнителни графични знаци.
184. 6) {14} чрез проверка
184. 7) {23} Защото зависят от конкретния процесор, от конкретната машина

185. 230. Какво представлява понятието „идентификатор“ в ЕПВР?

185. 1) {до 3} разпознава командите умножение, деление и др.
185. 2) {до 3} Идентификатора символите в езиковата програма от високо равнище.
185. 3) {8} – буква или дума, която е от определен тип
185. 4) {9} оператор който проверява и позволява или не позволява да се извърши определена операция в ЕПВР, разпознавателен оператор
185. 5) {12} Идентификаторът: различава се по броя на разрешените знаци и значимите знаци. Служебните думи са забранени идентификатори
185. 6) {14} Съществуват различни видове променливи. При декларирането им те трябва да се именуват. Името, което им даваме се нарича идентификатор.
185. 7) {17} Понятието „идентификатор“ в ЕПВР представлява идентифициране, **разкриване на данни**. Обособяване, разтълкуване и идентифициране на операционната система и дадените данни.
185. 8) {20} Идентификаторът служи за различаване на величини с еднакви имена, но различно значение
185. 9) {20} Използват се за именуване на обекти по уникален начин.
185. 10) {21} Дефиниция на дадена променлива.

185. 11) {26} **Идентификаторът служи** за именуване на операция в програмния код. Когато се използва оператор за безусловен преход чрез идентификаторът се показва към коя операция да се премине.
185. 12) {26} **Използват се** за наименование на променливите и други елементи на програмата
185. 13) {27} В ЕПВР идентификатор представлява набор от букви и цифри, символи Като служат за название на променлива. Идентификатори се формират по правила на езика, и ако ще бъдат нарушени ще приведе към грешка на компилация. В някои езици може да се прави разлика между големи и малки букви (list не е също LIST) Той не какви във програма се зарезервирани идентификатори които ние не можем да използваме като променливи (do, while, for. goto)

186. 232. Какви стратегии за различаване на измислените от програмиста идентификатори прилагат от ЕПВР (поне 3)?

186. 1) {12} Стратегии **за различаване** на идентификаторите измислени от програмиста е използването на **различни имена**
186. 2) {12} Използване на **уникални имена**
186. 3) {14} Обикновено се избягва един и съща символ да се използва с различни цели. Изгодна стратегия за всеки основен символ да има свой графичен знак.
186. 4) {15} Такива които да се разбират да правилни и да могат да се разбет и от човека и от компютъра
186. 5) {17} Обикновено се избягва един и същи символ да се използва с различни цели, но това не винаги е изгодно. Изгодна стратегия е за всеки основен символ на езика да има отделен графичен знак или чрез няколко неделими графични знака. **[И какво общо има и този ужасно дълъг отговор със зададения въпрос?]**
186. 6) {17} 1. За всеки основен символ да има отделен знак
186. 7) {18} Обикновено се избягва един и същи символ да се използва с различни цели, но това не е изгодно. Изгодно е за всеки основен символ на езика да има отделен знак. В някои ЕПВР всеки възможен идентификатор може да се използва за именуване на обекти. В повечето обаче има идентификатори с особен смисъл. **[И какво общо има и този ужасно дълъг отговор със зададения въпрос?]**
186. 8) {25} Обикновено се избягва един и същи символ да си използва с различни цели, но това не всеки път е изгодно. Изгодна стратегия е за всеки основен символ на езика за има отделен графичен знак или (липсват необходимите клавиши) чрез няколко неделими графични знака.
186. 9) {27} Съвкупност от знакове е правила чрез които се записват числата

187. 233. Какво представляват „служебните (ключовите) думи“ на един ЕП?

187. 1) {14} Служебните думи имат определен смисъл, който не се изпълнява от програмата
187. 2) {14} „Служебните“ думи в един език за програмиране са нужните детайли които един програмист трябва да въведе в изграждането на даден алгоритъм
187. 3) {17} „Служебни (ключови) думи“ – уникални за всеки един ЕП, **те** не могат да бъдат имена на променливи, **помагат и улесняват работата на хората** за писането на програмата.
187. 4) {20} **Оператори**, които **спомагат за** разбирането и **изпълнението на програмата**
187. 5) {24} В повечето ЕПВР **има оператори** с особен смисъл на езика Те служат за правилното разчитане на написаното и не могат да се използват за именуване Такива идентификатори се наричат ключове думи.
187. 6) {27} Основните команди в един ЕП, те задават главните действия.

188. 234. Какво представлява понятието „величина“ в ЕП?

188. 1) {12} Величините служат **за работа/обработка** на стойностите в дадена програма
188. 2) {12} Величина – най-малката структурна единица на програмата
188. 3) {26} Величините са основно средство за изразяване **[На какво?]**
188. 4) {26} Тя е скаларна и най-вече се обозначава с Е За въвеждане на понятието в сила.

188. 5) {26} Величините са основно средство за изразяване константи, които чрез възможностите на съответния език посочва и тяхната единствена стойност. Името на именованата константа е идентификатор, който се свързва с литерална и определена по-рано именувана константа.
188. 6) {26} Величина е някаква определена стойност на променлива

189. 235. Как ви са характеристиките на всяка величина?

189. 1) {до 3} Характеристиките на всяка величина са големина обхват; тип и точност.
189. 2) {12} Всяка величина може да бъде променлива или константа
189. 3) {14} име, тип и стойност.
189. 4) {14} Стойност, тип
189. 5) {15} размерност, тип данни
189. 6) {20} Всяка величина трябва да има предварително, зададена стойност, по-която да се извършват конкретните изчисления, за да се постигне дадения резултат. Размера на величината също е нейна характеристика
189. 7) {21} Променлива, тип, обем.
189. 8) {21} Характеристиките на всяка величина са тип, големина
189. 9) {26} вид/тип на величина, дължина
189. 10) {26} Елементната база на компютри от второ поколение е транзисторна

190. 236A. Кой определя имената на величините, използвани в една програма?

190. 1) {14} **Вида на информацията**, която сме въвели и от резултатността – положителна и отрицателна.
190. 2) {14} **Правилата на програмата** определя имената на величините
190. 3) {21} Имената на величините, използвани в една програма определя

191. 236B. Как трябва да определяме имената на величините, които използваме в една програма?

191. 1) {12} **Имената** на величините **ги определя съставителят** на програмата **като им поставя име**.) Поставяме им име.
191. 2) {14} Определя го съставителя на програмата като му дава име Тоест ние сами поставяме име на величините
191. 3) {14} Те се определят от съставителят на програмасти като им поставя имена
191. 4) {15} Съставителят на програмата поставя имена на величините
191. 5) {15, 16} Определя ги съставилия програмата като им поставя име.
191. 6) {17, 26} Определя ги съставителят на програмата **като им поставя имена**
191. 7) {17} съставителя на програмата ги определя и им съставя име.
191. 8) {20} Определяме имената на величините, които използваме ~~посредством~~ в зависимост от това какви променливи означават, посредством символи и знаци разбираеми за ЕП.
191. 9) {20} Трябва да определяме определяме имената чрез знаци, като първият от тях трябва да е буква, а след това може и да е различен знак, и също така не трябва да има повторения в имената на разл. величини.
191. 10) {20} Имената на величините са идентификатори като първия символ трябва да е буква, а останалите могат да бъдат букви и цифри.
191. 11) {21} На специално място в програмата трябва да се запишат вида и типа на дадена величина както и нейното име написано със знаци, позволени от програмата (букви).
191. 12) {23} **Определя ги** съставителя на една програма
191. 13) {23} Като им поставяме име.
191. 14) {26} Трябва да имат различни имена, за да можем да ги различаваме.

192. 237. По какво се различават величините, използвани в една програма?

192. 1) {14} Величините могат да се различават по тип и по стойност.

192. 2) {17, 17, 18, 20, 20, 20, 24, 26, 26} Величините се различават по размера на своето множество от допустими стойности и по начина за определяне на тяхната текуща стойност.
192. 3) {17} величините се различават по размера на своето множество от допустими стойности и **по начина за начина** на тяхната текуща стойност
192. 4) {17} По размера на своето множество от допустими стойности и по начина за определяне на тяхната текуща стойност.
192. 5) {20} Величините в програмата се различават **по** това **какви функции извършват**
192. 6) {20} По размер на своето множество от допустими стойности и **по начина за определяне на тяхната същност**.
192. 7) {21, 23} Величините се различават по размера на своето множество от допустими стойности и по начина за определяне на тяхната текуща стойност.
192. 8) {23} Величините в една програма се различават по своето приложение.
192. 9) {26} Различават се по размера на своето множество от допустими стойности и по начина за определяне на тяхната текуща стойност.

193. 238. Какви класове величини има в ЕП по отношение на тяхното множество от допустими стойности и как се наричат величините от всеки клас?

193. 1) {14} Целочислен тип. (0–9).
Реални, или още приближено числови.
Символни (‘a’–‘z’)
Низове,
Структурни.
193. 2) {17} а) Целочислен – цели числа
б) Текстов – за текст
193. 3) {20} Литерални и именовани, вътрешни и външни.
193. 4) {20} Вътрешни и външни
193. 5) {23} Величини → Константи (МДС = 1) → Литерални и именовани
193. 6) {26, 26, 26, 26} Величини → Константи (МДС = 1) → Литерални и именовани
Величини → Константи (МДС > 1) → Вътрешни и външни

194. 239. Какви видове константи предоставят ЕП и как се определя тяхната единствена текуща стойност?

194. 1) {5–7} – множество от допустими стойности от един единствен елемент е тяхната текуща стойност.
194. 2) {9} – множество от допустими стойности от един единствен елемент в тяхната текуща стойност
194. 3) {15} Литерални и **променливи**. Като **МДС е единичен знак** и именно този знак е итяхната текуща стойност.
194. 4) {23, 23} Множество от допустими стойности на константите се състои от един единствен елемент, който винаги е и тяхна текуща стойност.
194. 5) {26, 26} Множество от допустими стойности на константите се състои от един единствен елемент, който винаги е и тяхна текуща стойност
194. 6) {27} четене и запис

195. 240. Какво представляват литералните константи?

195. 1) {14} Имената на литералните константи и тяхната единствена стойност се определя от правилата на езика. Тяхната стойност е разположена в паметта някъде до достъпа до адреса ѝ не е съществен
195. 2) {14} Това са **константни имена без** зададена **стойност**
195. 3) {14} Константи, чиито имена са запазени в ЕП
195. 4) {14} При литералните константи за добро програмиране се използват 0 и 1.

195. 5) {17} Литералните константи с величини (0, 1,2 … null, …). Към тях се обръщаме със съотв. символ. Навиците на доброто програмиране предполагат използването само на литералните константи 0 и 1.
195. 6) {17} Имената на литералните константи се определят от правилата на езика. Считаме лит. конст. за неадресуеми, въпреки че тяхната стойност е разположена някъде в паметта, достъпът до този адрес не е съществен.
195. 7) {17} Имената на литералните константи и тяхната единствена стойност се определят от правилата на езика
195. 8) {17} Имената на литералните константи и техните стойности се определят от правилата на езика
195. 9) {20} Литералните константи са константи заложени строго определено от езика на за програмиране, които се използва.
195. 10) {20} Не се променят
Имената им се определят с правилата
195. 11) {20} Когато в дадена програма се появи стойност, например, като 1, тя се приема за литерална константа, защото можем да говорим за нея само като за стойност и стойността и не може да бъде променена.
195. 12) {20} Имената на литералните константи и тяхната единствена стойност се определят от правилата на езика. Считаме литералните константи за неадресируеми, въпреки че тяхната стойност е разположена някъде в паметта, достъпът до този адрес не е съществен
195. 13) {23} Когато в дадена програма се появява стойност като 1, например тя се приема за литерална константа.
195. 14) {23} За литералните константи можем да говорим само като стойност, която не може да се променя
195. 15) {23} Когато в дадена програма се появява някаква стойност можем за литерална константа, защото можем да говорим за нея само като за стойност (константа), защото стойноста ѝ не може да бъде променена.
195. 16) {26} Когато в дадена програма се появява можем да говорим за нея като стойност, константа, защото можем да говорим за нея само като за стойност константа, защото стойност ѝ не може да бъде променена.
195. 17) {26} Когато в дадена програма се появяви стойност, тя се приема от литерална константа, литерална защото можем да говорим за нея само като стойност, стойността и не може да бъде променлива
195. 18) {26} Когато в дадена програма се появява стойностка 1 например тя се приема за лит. константа: литерална защото можем да говорим за нея само като стойност, константа, защото ст-ста ѝ не може да бъде променяна.
195. 19) {26} Когато в програма имаме стойност „1“ тя се приема за литерална константа, защото можем да говорим за нея само като стойност, която не може да бъде променена
195. 20) {26} Когато в дад. се появява ст. 1 тя се нар. лит. конст.

196. 241. Кой определя имената на литералните константи в един ЕП и как става това?

196. 1) {15} Администратора определя имената на литералните константи в ЕП
196. 2) {15} Константите си определяме ние потребителите. В още самото начало на програмата.
196. 3) {17} Програмиста
196. 4) {22} Потребителя
196. 5) {26} Имената на литературните константи и тяхната единствена стойност се определят от правилата на езика.
196. 6) {26} Определят се **от езикът** за програмиране.

197. 242. Какво представляват именуваните константи?

197. 1) {8} Константа с <u>етикет</u> = <u>име</u>,
197. 2) {9} константи от определен тип.

197. 3) {14} Тип данни с непроменлива стойност.
197. 4) {23} При използването на константи в една програма е редно да им се присвоят имена, показващи предназначението на съответните константи. Това е важно, когато константите с еднакви стойности се използват за различни цели.
197. 5) {23} При използването на константи в програмата е озчно да им се присвоят имена, илюстриращи предназначението на съответната константа. Това е особено важно, когато константи с еднакви стойности се използват за различни цели.
197. 6) {26} Именуваните константи са константи, които притежават собствено име.
197. 7) {26} При използването на константи в програмата е удачно да и се присфоят имена илюстриращи предназначението на съответната константа. Това е особено важно когато константи с еднаквей стойности се използват за различни цели.

198. 243. Кой определя имената на именованите константи и как става това?

198. 1) {до 3} Задачата,анализът
Примерите са
Паскал, Си, Вижуъл Бейсик.
198. 2) {14} Имената на именованите константи се дефинират още при тяхното въвеждане от програмиста. Не е задължително, но е препоръчително.
198. 3) {18} Определят се от автора на езика
198. 4) {23} Имената на именуваните

199. 244. Има ли полза от именованите константи? Защо?

199. 1) {16} Да има полза да именоваме константите защото по този начин ние и **компютъра ще ги разпознаваме**.

200. 244. Има ли полза от именованите константи? Защо?

200. 1) {до 3} Имената на именованите константи се избира от програмист, който чрез възможностите на съответния език посочва и тяхната единствена стойност.
200. 2) {5–7} – името на именованата константа е идентификатор, който свързва с литералните или определена по рано

201. 245. Длъжен ли е всеки ЕП да осигурява именовани константи?

201. 1) {14, 15} Да
201. 2) {17} Да, длъжен е за да могат те да се различават една от друга.
201. 3) {20} Да, за да могат да се различават
201. 4) {20} Да. Например 3,14 може да го въведем като константа и след това да го използваме само като Pi, а не всеки път да си въвеждаме числото. Може просто да си извикваме Pi
201. 5) {26} Задължително е в ЕП да ила именовани константи. Ние ги именуваме за да бъде по добре четима програмата и за да няма двусмислици.

202. 246. Какво е правилото на доброто програмиране по отношение на литералните константи?

202. 1) {12} Правилото гласи, че литералните константи трябва да се използват така, че ако се наложи да се извършват промени, да се промени минимално количество код. Добра практика е и при използване на литерални константи за оказване на тип данни да се използват главните вместо малките букви – например за типа long да се използва L вместо l, защото l прилича на „1“ и може да доведе до объркания
202. 2) {14} Да се дефинират в началото на програмата
202. 3) {14} Никакво

203. 247. Какви видове променливи има в ЕП по отношение на начините за определяне на тяхната текуща стойност и как се наричат променливите от всеки клас?

203. 1) {5–7} локални и глобални променливи

203. 2) {12} Динамични и статични
203. 3) {12} **Видовете променливи** в ЕП **са константи** и променливи. Константите не променят стойността си по време на изпълнение на програмата, а променливите могат да променят стойността си по време на изпълнение на програмата
203. 4) {17} Литерални и **константни променливи**
203. 5) {20} Видовете променливи в ЕП по отношение на начините за определяне на текущата стойност биват локални, глобални, общи и параметри

204. 248. Какво е характерно за вътрешните променливи?

204. 1) {до 3} Имената им са идентификатори който се избират от програмиста доколкото им стига акъла.
204. 2) {5–7} действието е само във функцията която е декларирана яхната множество от допустими стойности се състои от поне два елемента
204. 3) {9} Имената им са идентификатори и се определят от програмиста
204. 4) {17, 17, 17, 18, 20, 20, 20} Тяхното множество от допустими стойности се състои от поне два елемента

205. 249. Как се определя името на вътрешна променлива и от кого?

205. 1) {14} Името на вътрешната променлива няма значение и се определя от програмиста.

206. 250. Какво е характерно за външните променливи?

206. 1) {5–7} Действието на външните променливи се отнасят за цялата програма
206. 2) {9} Те са променливи, които се отнасят за цялата програма и всички функции.
206. 3) {14} Името, типът и стойността на външните променливи се задават от автора на езика. Те могат да бъдат използвани и променяни от програмиста.
206. 4) {14} начални данни за процедурата и резултат от нейната работа
206. 5) {14} Деленето на програма на части въвежда външни променливи
206. 6) {23} Не, не е задължително
206. 7) {27} Външни променливи ще бъдат видими на една програма, ако ще е включена във файл. Ние можем извушаваме операции със тая променлива, но не можем да променим нейните стойности. В някои езици се изисква да сложим специален знак преди променлива за да се обръщаме към нея.

207. 251. Как се определя името на външна променлива и от кого?

207. 1) {16} Определя се от програмиста
207. 2) {17} Външна променлива се определя от програмиста
207. 3) {17} Определя се от автора на сорс кода [създателя на програмат} обикновено в самото начало, чрез определена дефиниция.
207. 4) {17} Определя се от програмиста, като им поставя имена.
207. 5) {20} Името на външната променлива се въвежда от програмиста съгласно правилата на съответния език за програмиране.
207. 6) {23} Името на външната променлива се определя от човека който пише програмата (програмира) и тя трябва да е с различно име от останалите променливи.
207. 7) {23} Името е определя от програмиста, като именато на тези променливи са идентификатори.

208. 252. Какво представлява понятието „тип данни“ и за какво се използва?

208. 1) {до 3} това е функция в която е записана определен тип данни (информация) и се използва за по лесно намиране (по бързо) на даден тип информация.
208. 2) {11} Определен вид от алгоритми, използват се за съхранение и използване на дадени елементи.

208. 3) {12} Понятието „тип данни“ означава при въвеждане на информация в програмата да се означи и нейният тип, за да може да бъде разбрана от програмата дадената информация, затова се използва понятието „тип данни“
208. 4) {14} използва се от изпълнителя при съставянето на даден алгоритъм Служи за въвеждане и съхранение на данни в алгоритъма като данните въведени от нас имат определен тип (int, char, double, float)
208. 5) {20} Използва се за промени на числовия вид на променливата.
208. 6) {20} Понятието „тип данни“ представлява вида на променливите и константите. Използва се за по-лесна обработка на входната и крайна информация
208. 7) {27} „тип данни“ е видът/формата на знаковете. Използва се при дефиниране на величини, променливи и др.
208. 8) {27} Типът на данните определя какви стойности и в какъв диапазон може да присвои дадената променлива.
208. 9) {27} Данните са цифри и букви който образуват информация за дадено нещо Използва се за писане на програми.
208. 10) {28} Понятието тип данни се използва за да се класифицират видовете данни.

209. 253. Възможно ли е в един ЕП да липсва понятието „тип данни“?

209. 1) {12} Не е възможно. Типът данни посочва множеството от допустими стойности за дадената величина, определя операциите вътре в типа.
209. 2) {15, 15} Не
209. 3) {16} Език за програмиране [ЕП] – не

210. 254. Какво представляват „съвместимите“ типове от данни и защо съществуват“?

210. 1) {до 3} Посочва множеството от допустими стойности за дадена величина.
210. 2) {15} „Съвместимите“ типове от данни обикновено наричаме типове от данни, които може да разглеждаме като:
– подмножества
– надмножества
[Следва диаграма на Вен, обясняващо що е подмножество и що – надмножество.]

211. 255. Какви проблеми създават литералните константи на съвместимите типове от данни и как се решават тези проблеми?

211. 1) {5–7} Литералните константи на един подтип
211. 2) {17} Литералните константи са литерални – техните имена се определят от дадения език за програмиране
211. 3) {20} при външна промяна на константата може да доведе до грешни резултати
211. 4) {26} Тази константа се пише навсякъде в програмата затова е по-лесно да се декларира.
Пример: 3,14 – π
211. 5) {27} Те не подлежат на промяна; решават се чрез заместване с друг тип константи

212. 256. Какви са основните типове данни в повечето ЕП (поне 4)?

212. 1) {9} – само писмени
– имат подмножества
– имат диалект.
212. 2) {9} константи-const,
212. 3) {10} Основните типове данни в ЕП са дискретни, аналогови, указателни и прости (скаларни)
212. 4) {11} символни, нис, реални, естествени, комплексни
212. 5) {12} – прости (**директни**, аналогови, указатели)
– структорни

212. 6) {17} Структурни, блокове
212. 7) {23} а) числови КОП – безлични
б) числови адреси на данните
в) в двоичен код – загадъчен
г) сами разпределяме ОП

213. 257. Какво е характерно за дискретните типове данни?

213. 1) {до 3} дискретното е това, че в тях може да се крие определена информация която не може да се открие от друг освен от човека който оле скрипта може да се открие, чрез парола. Тая парола се знае единствено създателя.
213. 2) {9} съдържат дискретна информация

214. 259. Сравнете типовете цяло и приближено число като посочите техни характерни особености?

214. 1) {до 3} Важни характеристики на типа са броят на верните цифри (точност) и диапазонът на представените числа (обхват на порядъка).
214. 2) {12} – С реалните числа могат да се вършат повече операции **[И кои са тези операции в повече?]**, но заемат повече памет **[Операциите!?!]**
– Целочисленият тип заема по-малко памет **[Защо?]** и операциите се извършват по-бързо. **[Защо по-бързо?]**
– Плаващата запетая на реалните числа **[Каква плаваща запетая има едно реално число, пък и за какво ли му е тя?]** обикновено води до частични грешки в сметките, поради преобразуването в машинен код (пример: 10,4 → 100111,0101) **[Примерът брилянтно илюстрира неясната мисъл на автора.]**

215. 260. Може ли в цифров компютър да се представят реални числа? Защо?

215. 1) {14} Да.
215. 2) {14} Цифров компютър може да представя реални числа, защото може лесно да работи с плаваща запетаи **[Числата с плаваща запетая са рационални!]**
215. 3) {14, 17} Може, защото ЦП разпознават и могат да оперират числа, представени като мантиса и порядък (степен на 2 или 16, а не на 10)
215. 4) {17, 17} Може, защото ЦП разпознават и могат да оперират числа в плаваща запетая представяни като мантиса и порядък.
215. 5) {17, 20} Да
215. 6) {17} Да може.
215. 7) {21} Да. В езиците за програмиране и създадени типове за представяне на реални числа.

216. 261. Защо вместо тип реално число в ЕПВР е по-правилно да се говори за тип приближено число?

216. 1) {5–7} Защото **така е правино** да се нарича
216. 2) {8} защото не се знае коя стойност е била преди и след него
216. 3) {9} За по лесно разчитане от ЕПВР **[Интересно откритие е, че езиците могат да четат!]**
216. 4) {14} Заради преобразуването от десетична в двоична БС и това, че ако едно число е крайно в една система – може да е безкрайно в друга
216. 5) {14} Тип приближено число е по-правилно от тип реално, защото по-правилно дефинира същността на понятието.
216. 6) {14} Защото ако сложим тип реално число а отговора трябва да е с точност. Самият тип закръгля до първото цяло число
216. 7) {17} Защото **реалните числа** могат да **имат безброй много числа** след десетичната запетая, а това би затруднило изпълнението на операциите.
216. 8) {26} Фиктивни променливи в дефиницията на подпрограмата
216. 9) {27} Защото в ЕПВР цифоя/числата са с неточна стойност.

217. 263. Каква е разликата между знаков и текстов тип данни?

217. 1) {9} Текстов тип данни са входните данни.
217. 2) {14} Знаковият тип данни се състои от цифри и знаци, а текстовият тип данни съдържа всички букви, цифри и знаци, тоест текстовият тип данни може да замести знаковия. **[Единственото, което става ясно от този отговор е, че авторът му не смята буквите и цифрите за знаци!]**
217. 3) {15} Знаковия се записва със символи
217. 4) {20} Текстовият тип данни е низ, а **знаковият е от реален тип**
217. 5) {23} Знаковете тип данни са за числа и текстовия тип данни е за текст.
217. 6) {26} Знаковият тип данни е разбираем за **ПЕ**.

218. 264. Задължително ли е един ЕПВР да предоставя едновременно и знаков и текстов тип данни? Защо?

218. 1) {14} Да
218. 2) {17} Да, с цел по-лесното разчитане на изходният код и разграничаване на типа данни при превод на програмата
218. 3) {26} Да, задължително е да се представят едновременно

219. 265. Задължително ли е всеки ЕПВР да предоставя знаков тип данни? Защо?

219. 1) {10} да, защото иначе няма да се извърши самата операция
219. 2) {20} Да задължително е, защото ЕПВР определя редът на изчисляване на операциите и разпознаването на техните операнди
219. 3) {20} Да
219. 4) {21} Да, защото знаковия тип данни се използва в сяка една програма.
219. 5) {26} Да,принципно знаци могат да се кодират с някакво число ЕПВР то е направено за да ни бъде по удобно писане на програмата

220. 266. Какво представлява типът „указател“?

220. 1) {10} Указващ мястото където се намират определени константи.
220. 2) {12} **мястото**, където се посочват данните в оперативната памет
220. 3) {15} Приличат си по първото изчисляване на условието за да покаже липса на повторението

221. 267. Посочете какви нови типове данни предоставят съвременните ЕПВР и защо?

221. 1) {5–7} long integer, long real, long bit, long CHAR
221. 2) {9} ЕПВР – предоставя цифрови типове данни, защото с тях се работи по бързо.
221. 3) {17} **Тип данни** , тип пари и универсални

222. 269B. Посочете поне два недостатъка на явното деклариране на променливите в сравнение с неявното деклариране.

222. 1) {17} Давя яснота и намиране на грешки.
222. 2) {18} – Действията се описват словесно
– Липсват блокове за деклариране на входни данни и резултата.
– Преплитане на линиите влошава структурата и разчитането

223. 269C. Посочете поне две предимства на неявното деклариране на променливите в сравнение с явното им деклариране.

223. 1) {12} При неявното деклариране възможността за **допускане на правописна грешка е по-малка**, освен това **провокира дисциплинираност** у програмиста.
223. 2) {14} Предимства: избягват се правописни грешки, налат дисциплини

224. 269D. Посочете поне два недостатъка на неявното деклариране на променливите в сравнение с явното им деклариране.

224. 1) {17} При неявно деклариране е възможно появата на несъответствие при желания тип данни и този който програмата възприема. Също така при по-сложни програми кода е възможно да бъде по-сложен за разчитане.

224. 2) {20} Неявното деклариране крие много недостатъци пред явното, като пример, липсва прозрачност на явлата декларация и може да се стигне до грешки в числото което се декларира, можем да имаме много грешки и с програмата. За това приенане, че явното деклариране е по-добрият вариант.

224. 3) {20} Когато не е явно е възможно пропускането му; и възникване на грешки

224. 4) {20} По-трудно писане; повече писане

224. 5) {20} **При явното деклариране** на променливите можем да придадем по-голяма точност на самата променлива Също и ни дава по-лесно да правим промени при вече завършената програма

224. 6) {20, 23} повече и по-трудно писане.

224. 7) {26} Недостатъците на неявното деклариране са че писането е повече и става по-трудно.

225. 270. Възможно ли е няколко величини да имат едно и също име? Ако не – защо, ако да – как става различаването им?

225. 1) {16} Да, възможно е. Различаването им става като ясно се обособи къде свършват действията с първата величина и започват с втората. **[Обяснението показва, че студентът не знае отговора на зададения въпрос!]**

225. 2) {18} Не е възможно няколко величини да имат еднакви (едни и същи) имена, тъй като тяхното различаване би било невъзможно. **[Да би мирно седяло, не би чудо видяло. Ако авторът на този отговор би спрял до тук, нещата щяха да бъдат наред. Но всемогъщият автор продължава с пълни глупости, които показват, че хал хабер си няма от материята, по която полага изпит!]** Единствения случай в който отговорът на дадения въпрос би бил „да“ е ако работим с ЕПВР, тъй като променливите с еднакво име биват различавани по техния определен тип. (Пример int abs] double abs в C++, където в C би било представено int abs; float fabs;

225. 3) {26} Не е възможно няколко величини да имат едно и също име, защото няма да могат да се различават

225. 4) {26} Не. Защото няма да могат да се различават

225. 5) {26} Това не е възможно, защото няма да може да се различават.

225. 6) {26} Не, не е възможно, защото няма да могат да се различават

226. 271. Посочете механизми за различаване на величините от съвкупност едноименни величини.

226. 1) {до 3} Няколко величини имат еднакво име което да използваме като ги разглеждаме като единно цяло.

226. 2) {14} По стойност и чрез добавяне на допълнителни символи (индекси)

226. 3) {20} Когато задаваме съвкупност от величини ние трябва да зададем максимален брой на тези величини, докъто при единичната си е самостоятелна.

226. 4) {23} чрез позициониране.

226. 5) {27} Във всяка БС се определят краен брой възлови числа. Всяко възлово число се представя с определен знак наречен цифра.

227. 272. Какво трябва да се посочи при определяне на един масив?

227. 1) {14} Запишите изискват явно деклариране на имената и типовете на своите елементи

227. 2) {14} Името на масива, както и неговата дължина и, и начало. Пр: int a[20];

227. 3) {15} **Стойността** на масива N
Еднаквостта на всички елементи
Граничната двойка Max и miN.

227. 4) {15} променлива
227. 5) {17} **Тип на масива** и големина (брой елементи)
227. 6) {17} Име и размер на масива, (вид)
227. 7) {18} Име, допустими стойности
227. 8) {22} Пример char zaglavie [30]
227. 9) {23} Масивът е структура от данни, която обединява група от елементи с един е същи базов тип.
227. 10) {26} Посочва се действие и условие
227. 11) {26} Теп и големина на масива.

228. 273. Какво трябва да се посочи при определяне на един логически запис?

228. 1) {17} При определяне на един логически запис трябва да се посочи името и типа на елемента
228. 2) {17} Условие или условия и логически връзки (логическо НЕ, И, ИЛИ и др.) между тях. (Различните ЕП ползват различни логически оператори).
228. 3) {23} Като програмна част блокът е най-елементарната програмна част. Блоковете винаги са вложени по-своя маниер на писане
228. 4) {26} Трябва да се знаят 2 неща: 1) Какви стойности може да приема аргументът и на тези ст на аргумента какво стойности на функцията отговарят. 2) Ч/з таблицата за истинност се записва 1 логически запис

229. 273. Какво трябва да се посочи при определяне на един логически запис?

229. 1) {11} Логическо решение
229. 2) {12} Условието и възможните отговори
229. 3) {14} При записване на логически запис трябва да се посочи **име и тип на записа**.
229. 4) {14} Множеството от допустими стойности

230. 274. Защо в много ЕПВР структурите от данни се наричат типове данни?

230. 1) {12} Защото се посочва типа на елементите, съдържащи се в тях.
230. 2) {12} Защото имат литерални константи, операции и релации
230. 3) {14} Защото всички данни от една структура са от един и същи тип
230. 4) {17} Защото в езиците за програмиране данните се структурират така както типовете данни
230. 5) {18} Защото типа данни посочва множеството от допустими стойности за дадена величина и определя операциите вътре и вън от типа, релациите на такива данни и правилата за запис на литералните константи.

231. 275. Защо структурите от данни не са типове данни?

231. 1) {14} Структурите от данни са дадена структура с определено име зададено от програмиста където се въвеждат данни от различен тип и впоследствие се използват във функция и имат свой собствен блок, където се въвеждат данните, докато типовете данни стоят в даден блок.
231. 2) {20} Структурите от данни не са типове от данни, защото
231. 3) {20} Защото имат структура.
231. 4) {20} Защото имат структури, които носят данни **и това ги прави типове данни**.

232. 276. Защо са полезни структурите от данни?

232. 1) {до 3} **Посочва множество от допустими стойности** за дадена величина.
232. 2) {14} Улесняват работата на автора на програмата
232. 3) {14} Структурите от данни са полезни, защото **съдържат няколко оператора в състава си**
232. 4) {17} Структурите от данни са полезни при използването на голям обем променливи със ~~сходни типове~~ сходна роля и/или при групирането им за по-удобно ползване.

232. 5) {17} Структурите от данни са полезни, защото чрез тях е възможно да се извършват множество от действия с програмните продукти и да се стигне до желания краен резултат.
232. 6) {19} Защото има полза при използването им.
232. 7) {20} За по-прегледен вид и удобство на програмиста.
232. 8) {20} Чрез тях от малко човешки труд се получава голямо количество компютърна работа.
232. 9) {20} Структурите от данни са полезни защото са удобни, запазват дадени данните и не е нужно да се пишат на ново.
232. 10) {23} Те се използват за ефективно съхранение на данни. Данните се групират по определен начин. При избор на подходяща структура е възможно по-бързо и икономично обработване на информация.
232. 11) {26} Полезни са защото може да се ползват за надграждането им и използването вдруги програми
232. 12) {26} Структурите от данни могат да се вграждат една в друга, тоест елементите на една структура могат до бъдат структури от данни
232. 13) {27} Чрез тях се намалява сложността улеснява съхранението на данните

233. 277. Кой определя типовете данни, които могат да се използват в един ЕПВР?

233. 1) {9} Програмистът
233. 2) {15} Типовете данни се определят от правилата на съответния език за програмиране високо равнище.
233. 3) {20} Операторите
233. 4) {20} Програмиста

234. 278. Може ли освен предварително определените типове данни пишещият програмата да определя и свои типове данни? Защо?

234. 1) {15} В зависимост от това **дали може да придвидим какъв ще е резултата може да определим** и свои типове данни

235. 279. Какви са основните механизми за конструиране на потребителски типове от данни?

235. 1) {8} Основните механизми за конструиране на потребителски типове могат да бъдат:

236. 281. Има ли полза от използването на тип „поддиапазон“? Защо?

236. 1) {8} Да Защото се дефинират под области на числата
236. 2) {17} Да, защото се дефинират под области
236. 3) {17} Да, защото
236. 4) {20} Има полза. Използват се за опростяване. **[На какво??]**
236. 5) {20} Да има полза, защото се намалява използването на оперативната памет

237. 282. Какво е предназначението на механизма „изброим тип”?

237. 1) {5–7} Изброимия тип е множество от стойност
237. 2) {9} Тип изброими

238. 285. Имат ли константите „тип данни“? Ако да – как се определя той, ако не – защо?

238. 1) {12} Не, защото самите те могат да бъдат подтип на данни
238. 2) {14} Не
238. 3) {20} Константите биват литерални, именувани, **вътрешни и външни**
238. 4) {26} const a = 1

239. 286. Какво представляват операциите в един ЕПВР?

239. 1) {12} Операциите са **един от елементите**, които изграждат ЕПВР. Те са **част от програмата**.

239. 2) {17} С ранг на операциите се отбелязват всички стандартни функции

240. 287. Как се определят операциите в един ЕПВР?

240. 1) {до 3} Видовете оператори се разделят на две категории според правилата на записване.

240. 2) {10} – Управляващо устройство: → извлича; декодира, изпълнява; променява за прекъснатост;
– Аритчетично логическо → разчита и изпълнява програм.
Регистри → работна памет и временно памет

240. 3) {12, 17, 17, 20, 21, 23, 24, 26, 26} Операциите в повечето ЕПВР са твърде много. Поради това едва ли е възможно за всеки знак на операция в езика да има уникален основен символ Двусмислието на знака се премахва чрез анализ на операндите **[И какво общо има този ужасно дълъг отговор със зададения въпрос?]**

240. 4) {12} Чрез изрази, съдържащи знакове, скоби

240. 5) {14} Операциите са твърде много, двусмислието на знаците се премахва чрез анализ на операндите

240. 6) {14, 26} Операциите в повечето ЕПВР са твърде много. Поради това едва ли е възможно за всеки знак на операция и, езика да има уникален основен символ

240. 7) {17} Определят се, чрез базата данни.

240. 8) {17} По приоритета на действие

240. 9) {20} Операциите в един ЕПВР са твърде много. Поради това едва ли е възможно за всеки на операция в езика да има уникален основен символ. Двусмислието на знака се премахва с анализ на операндите.

240. 10) {20} Те са машинно-независими, наричат се още алгоритмични. чрез командите

240. 11) {23} Операциите в един ЕПВР са твърде много. За това е невъзможно за всеки знак да има уникален символ. Като има двусмислие, то се премахва чрез анализ. **[И какво общо има този ужасно дълъг отговор със зададения въпрос?]**

240. 12) {23} Операциите из повечето ЕПВР са твърде много. Поради това едва ли е възможно за всеки знак на операцията в езика да има уникален основен символ. Двусмислието на знака се премахва чрез анализ на операндите.

240. 13) {26} Операциите се определят трудно, защото всеки знак на операция има своя основен символ

240. 14) {26} Операциите в Повечето ЕПВР са твърде много. Поради това едва ли е възможно за всеки знак в езика да има уникален основен символ. Двусмислието на знака се премахва чрез на операциите.

240. 15) {26} Чрез премахване на двусмислието на всеки знак, с помощта на операнди.

240. 16) {26} Те са твърде много, затова едва ли е възможно за всеки знак на операция в езика да има уникален основен символ. Двусмислието на знака се премахва чрез анализ на операндите.

240. 17) {26} При някои ЕП има изискване опр. елементи да се изписват изцяло в рамките на 1 ред: Фортран, Асемблер, Вижуал – Бейсик – частично; Речниковият запас е строго определен предварително. Изгодна стратегия по отношение на символите на 1 ЕПВР е всеки осн. символ на езика да има отделен графичен знак.

241. 288. Кой определя наличните операции в един ЕПВР?

241. 1) {до 3} **Операциите** в 1 ЕПВР **са** твърде **много**.

241. 2) {9} Базовия софтуер

241. 3) {14} Наличните операции в ЕПВР се определят **от съставителят на програмата и от неговите възможности**.

241. 4) {21} Съставителя подготвя алгоритъма като подбира действия и операции и реда на тяхното изпълнение

241. 5) {23} програмиста

242. 289. Какви видове операции има в ЕПВР?

242. 1) {14} операциите биват: по брой на операндите, по тип на резултата и по категории
242. 2) {14} Присвояване, вход/изход, **оператор** за край, безусл. оператор
242. 3) {14} – **Вавеждане**
– Извеждане
– Обработка
– Отпечатване
242. 4) {14} В ЕПВР има операции пресмятане (математически)
242. 5) {15} по брой на операндите, по тип на резултата, по категории
242. 6) {17, 17, 17} Операциите биват:
– по брой на операндите
– по тип на резултата
– по категории **[И какви по точно са по описаните признаци?!?]**
242. 7) {17} Процедурно ориентирани, проблемно ориентирани
242. 8) {17} по брой на операндите,по тип на резултата, по категория
242. 9) {18} Биват: по брой на операндите, по тип на резултата, по категории
242. 10) {20} Прости (елементарни), които не съдържат като част от себе си други оператори
242. 11) {20} условие, подусловие, цикъл, функция, преброяване
242. 12) {21} **Операторите** биват
По брой на операндите, по тип на резултата и по категории
242. 13) {24} ЕПВР има 4 операции 1) блок 2) процедура, 3) функция, 4) модул.
242. 14) {25} По брой на операндите / По тип на резултата / По категорий
242. 15) {26} Образуване от един вид **оперант** в друг по висш
242. 16) {26} Видове операции са за край на изпълнението, за обмен на данни, за безусловен преход, за присвояване, за празен оператор
242. 17) {27} Вътрешни и Външни.

243. 290. Как се отбелязват операциите в ЕПВР?

243. 1) 1) {до 3} наличнвите операции, 2 правилата за получавания резултат и знакът
243. 2) {5–7} Идентификатор
– първият е буква
– останалите са букви или цифри
Те прилагат
по-разширени знаци
по-значими знаци
243. 3) {9} чрез операнди
243. 4) {14} Отбелязват се по различен начин във всеки ЕП като могат да бъдат или да не бъдат разграничавани със символи.
243. 5) {15, 16, 17, 17, 17, 17, 17, 18} С ранг на операции са всички стандартни функции, определени от езика.
243. 6) {15} Знак за операция на английски е operator
243. 7) {17} С ранг на операции са всички вградени функции, определени от езика.
243. 8) {20, 20} с думата <statement>

244. 291. Какво е характерно за операндите на всяка операция?

244. 1) {14} Определят как да се изпълни дадена операция
244. 2) {17} Операндите на всяка операция спомагат за нейното осъществяване.
244. 3) {20} **Оверандите** биват точно определен краен брой 1, 2, 3 (най-често 3) За всеки операн си има действие, което трябва да се извърши
244. 4) {20} Конюнкция и отрицание
Дизюнкция и отрицание
стрелка на Пирс и Щрих на Шефър

244. 5) {26} – почти няма операция с повече от 2 операнда
– стойностите на операндите се получават до прилагане на операцията
244. 6) {26} Характерно за операндите е че операндите биват – с 1, 2 и с 3 много рядко е типът на резултата; по категории които могат да бъдат – аритметични, логически, символни и т. н.

245. 292. Каква форма на запис на операциите използват ЕПВР?

245. 1) {15} **Речников** запис. **[Що ли за звяр е такъв вид запис?]**
245. 2) {15} Поредица от знаци, от които първия е буква, а останалите са букви или цифри.

246. 293. Как се решава проблемът коя операция е означена с даден знак?

246. 1) {до 3} Често символите на клавиатурата могат да са недостатъчни за всяка отделна команда. За разлика от „Паскал“ при който няма значение главен символ „...Aa, Mm ...“, Си прави конкретна разлика и така възможностите се увеличават.
246. 2) {до 3} Означават се с различни знаци.
246. 3) {5–7} Чрез специални символи
246. 4) {12} ??? – Има служебни думи, които не могат да се използват като идентификатори. Те са думи на езика със специална функция. Означават оператори и др. такива.
246. 5) {14} Ако символите от клавиатурата не стигат се налага записване на физически знак написата на 2 символа за да се избелги повторетто.
246. 6) {14} Чрез поставяне на етикет на всяка операция.
246. 7) {17} Като се добавят допълнително символи (Ато друг знак или цифра.
246. 8) {20} Пишещият програмата определя коя операция е означена с даден знак.
246. 9) {26} Проблемът го решава програмиста именувайки коя операция с кой знак се означава.

247. 294. Как може да бъде разрешен проблемът, когато операнд на една операция не е от типа, за който е определена тя?

247. 1) {12} първо се отбелязва като грешка и второ като се даде нова стойност
247. 2) {14} Промяна операнда за да съвпада с операцията.
247. 3) {14} Като се конвертира.
247. 4) {15} Проблемът може да бъде решен при прекъсване на програмата при аварийно спиране; при забрана на изпълнение; при преобразуване на типа от по-ниска стойност към по-висока стойност.
247. 5) {19} Като използва собствен език за програмиране
247. 6) {19} Трябва да се постави етикет.
247. 7) {20} Променяме типа на операнда (ако е възможно) или намираме друг начин да използваме операцията, която искаме
247. 8) {20} Много трудно, защото трябва да се опишат всички действия, набора от допустими стойности и всички възможни релации. Също така и типът.
247. 9) {21} Трябва операндът на операцията да бъде от типа за, който е определена.
247. 10) {21} **Проблема**, когато операндата на една операция не е от типа, за който е определена тя **се решава чрез преобразуване на системата**
247. 11) {23} При превода със забрана
247. 12) {26} Съобщения за открита грешка

248. 295. Как може да бъде променен типът на даден операнд?

248. 1) {20} Като се извика различна функция.
248. 2) {20} Типът на даден операнд може да бъде променен като преобразуваме типа на този операнд **към по-главния тип** на друг операнд.

249. 296. Какво представлява понятието „израз“ в ЕПВР?

249. 1) {до 3} понятието израз означава изречение
249. 2) {11} Обекти от ЕПВР свързани помежду си със табличните функции.

249. 3) {12} Израз – Пресмята и изчислява **моментна стойност на алгоритъма**.
249. 4) {15} „Израз“ – **това математическа част** от програмата
249. 5) {15} език за програмиране на високо равнище
249. 6) {17} Когато се използва макрос – модифицирания код се появява при всяко използване на макроса (загуба на ОП) използването не изисква MI и може да бъде оптимизиран на място (бързина)

250. 297. Какво съдържа записът на един израз в ЕП?

250. 1) {10} цифри, символи и интервали
250. 2) {11} единици и нули
250. 3) {12} Записът на един израз в ЕП съдържа променлива, операнд и стойност, която се присвоява на променливата.
250. 4) {14} Низ от символи
250. 5) {17} Букви, цифри и специални символи
250. 6) {20} Дефиниране на променливи и действия
250. 7) {26} НПБС биват адитивни (когато стойностите на цифрите от записа се събират, за да се получи стойността на представеното алгоритмично число) и мултипликативно (за да се сформира алгоритмичното число се използва операция умножение)

251. 298. Как се определя редът, в който се изчисляват операциите, за да се получи стойността на един израз?

251. 1) {9} Изчислява се по намаляване на математиката
251. 2) {14} **Редът**, в който се изчисляват операциите **се определя от програмиста**, след което програмата изпълнява всички команди последователно.
251. 3) {14} Чрез алгоритъм
251. 4) {14} Обикновено редът на изчисляване е от ляво на дясно
251. 5) {15} Операторите се изпълняват един след друг в реда, по който съставят текста на програмата.
251. 6) {15} Създава се програма, в която чрез алгоритми и подалгоритми се извършват зададените ни операции.
251. 7) {16} Декларираме променливите, въвеждаме ги, извеждаме ги.
251. 8) {17} Редът се определя от програмиста съставил програмата
251. 9) {20} Редът в който се изчисляват операциите, за да се получи стойността на един израз е от вътре на вънка. Извършват се изчисленията в подпрограмите и след това в самата програма
251. 10) {20} Като се съставя определен алгоритъм
251. 11) {21} Редът се определя от поредността, в която са записани операциите, освен ако няка от тях не са специално обозначени да бъдат извършени под друг ред.
251. 12) {23} Редът на изчисление на операциите се извършва чрез правила. които **осве** са определят реда за **изчесленето** на операции разпознават и техните операнди.
251. 13) {26} Редът се определя от така наречения естествен ред на изпълнение
251. 14) {27} **Редът на изпълнение се задава от съставителя** на алгоритъма, като се задава набор от елементарни действия в последователни стъпки
251. 15) {27} Операциите с по-висок приоритет се извършват преди операциите с по-никсък.

252. 299. Какво представлява понятието „приоритет на операциите“ в ЕПВР?

252. 1) {до 3} Приоритета представлява начина на изпълнението.
252. 2) {14} Както в математиката така и в информатиката някои операции имат приоритет пред други. Едни операции ще се изпълняват преди да се изпълнят други.
252. 3) {14} Операцията с по-висок приоритет се изчисляват преди тези с по-ниския При операциите с равен приоритет обикновено е от ляво на дясно

252. 4) {14} В ЕПВР изпълнителя на алгоритъма в зависимост от ЕПВР има възможност да зададе на дадена функция елемент, данни на висок приоритет при изпълнението на програмата (алгоритъма) от другите или предходните (начало)
252. 5) {15, 17} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става оперант на по-ниско приоритетната. При операции с еднакъв приоритет обикновено редът е от ляво на дясно (Но в СИ има и такива от дясно на ляво)
252. 6) {17, 17} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната. При операции с еднакъв приоритет обикновено редът е от ляво на дясно (но в Си има и такива от дясно на ляво
252. 7) {17} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната.
252. 8) {17} В ЕПВР всички оператори се изпълняват последователно и това своиство се нарича Приоритет на операциите.
252. 9) {18} Общ за всички операции и двустепенен. Приоритета е първо от категории, а след това вътре в тях.
252. 10) {20} операциите с по-висок приоритет се решат преди операции с по-нисък приоритет. При операции с еднакъв редът е отляво надясно
252. 11) {21} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната. При операции с еднакъв приоритет обикновено редът е от ляво на дясно
252. 12) {23} Определя се **редът за изчисляване на операторите**, и разпознаването на техните **операнти**
252. 13) {23} Приоритет на операциите определя кое действие след кое ще се извърши в зависимост от текста на програмата. В програма предимство имат операторите от по-високо равнище. Те се изпълняват отдясно на ляво, но в Си има и оператори, които се изпълняват отляво – надясно.
252. 14) {26} При тях при краен брои цифри поради промяната на тяхната стоиност могат да се запишат безкраен брой числа.
252. 15) {27} Алгоритмите, изпълняващи се от зададените операции.

253. 300. Кой и как определя приоритета на операциите в даден ЕПВР?

253. 1) {до 3} Правилата известни като приоритет на операциите. Чрез тези правила се определят както редът за изчисляване на операциите така и разпознаването на техните операнди.
253. 2) {8} Програмистът, чрез определен начин на написването на програма
253. 3) {9} Програмиста, чрез езика за програмиране
253. 4) {14} Първо се определя висока приоритет,
253. 5) {14} Операция с по-висок приоритенет
253. 6) {15} Съставният оператор.
253. 7) {15} Операция с по-голям приоритет се изчислява преди операция с по-малък и нейният резултат става операнд
253. 8) {15} ЦП чрез ЕП
253. 9) {17} Създателя на програмата, чрез правилно структуриране на алгоритъма.
253. 10) {17, 17, 23} Има два начина за определяне на приоритет на операциите: общо за всички операции (Паскал, Си) и двустепенен – първо по категории, а след това е вътре в тях.
253. 11) {18} Приоритета се определя в зависимост от приоритета на използваните операнди
253. 12) {20, 20, 20, 23} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната. При операции с еднакъв приоритет редът на изчисляване обикновено е от ляво на дясно, но в СИ има и равни по приоритет, които се изчисляват от дясно на ляво. Има два начина за определя-

не на приоритета: общо за всички операции (Паскал, СИ!) и двустепенен – първо по категории, а след това вътре във всяка от тях (ВБ)

253. 13) {20} Съставителят и изпълнителят

253. 14) {25} Операциите с по-висок приоритет се изчислява преди операции с по-нисък приоритет, при което нейните резултати стават по-ниско приоритетни. При операции с еднакъв приоритет в С приоритет да се изчисляват от дясно на ляво. Има 2 начина за изчисляване на приоритет: двустепенен (първо от категории и след това вътре във всяка от тях) и общо за всички операции (в Паскал, Си).

253. 15) {26} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната. При операции с еднакъв приоритет редът на изчисляване обикновено е от ляво на дясно, но в СИ има и равни по приоритет операции, които се изчисляват от дясно на ляво. Има два начина за определяне на приоритета: общо за всички операции (Паскал, Си) и двустепенен – първо по категории, а след това във вътре всяка от тях

253. 16) {26} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната.

253. 17) {26} Операция с по-висок приоритет се изчислява преди операция с по-нисък приоритет и нейният резултат става операнд на по-ниско приоритетната. При операции с еднакъв пприоритет редът на изчисляване обикновено е от ляво на дясно, но в Си има и равни по приоритет операции,

253. 18) {26} за съхранение на междинните резултати и изпълнение на програмата

253. 19) {26} Операциите с по-висок приоритет **се изпълняват** преди операциите с по-нисък приоритет Приоритетът е **определен от правилата на самата програма**

253. 20) {26} Операциите с по-висок приоритет **се изпълняват** преди операциите с по-нисък приоритет При операции с еднакъв приоритет обикновено изчислението от ляво надясно. Има и изключения които се изпълняват от дясно наляво

253. 21) {27} **Съставителят на програмата** определя приоритета на операциите в даден ЕПВР, **под формата на алгоритъм**

254. 301. Как трябва да програмираме, когато не сме уверени в познанията си за приоритета на операциите в използвания ЕП?

254. 1) {9} Операциите се изпълняват в реда в който са записани операторите като текст на програмата

254. 2) {9} трябва само да следваме алгоритъма за изпълнение

254. 3) {10} Чрез ЕПВР **[Нима той не е ЕП?]**

254. 4) {14} Трябва да използваме само прости операции.

254. 5) {14} Програмираме бавно стъпка по стъпка

254. 6) {17} Трябва **да изпълняваме** само кратки изрази

254. 7) {17} Когато не сме сигурни – използваме език от ниско равнище

254. 8) {17} Трябва да избягваме сложното и да използваме простото за което сме сигурни

254. 9) {20} Когато не сме уверени в познанията си за приоритета на операциите в използвания ЕП, тогава следваме общоприетия приоритет за операциите, който е в математиката.

254. 10) {26} Като следваме структурното дърво за иконе на програма.

255. 302. За какво служат операторите в един ЕПВР?

255. 1) {до 3} последователно изпълнение е главна характеристика на програмата като единно цяло. В много ЕПВР съществува синтактично ограничение като част от структурен оператор да се задава само един единствен друг оператор.

255. 2) {до 3} Операторите служат за означаване на начало на преход и край на даден алгоритъм.

255. 3) {9} За създаване на базов софтуер

255. 4) {14} Операторите служат **за изчисляване на стойностите**

255. 5) {14} С тях се ограничава процедурата

255. 6) {15} Те служат, **за да бъдат изпълнени заповедите** в алгоритъма
255. 7) {17} За задвижване на дадения алгоритъм. Без тях програмата трудно би съществувала.
255. 8) {18} Имат по-високо структурно ниво на инструкциите и машината зависимост (машинно-независими езици) Структурата и наличието им команди са валидни за конкретната компюр. система
255. 9) {20} Операторите в един ЕПВР служат за изготвянето **и изпълняването** на алгоритъм.
255. 10) {21} За изпълняване на команди
255. 11) {23} служат с мисълта че една програма написана на един език не се променя

256. 303. Какви възможности за запис на оператори предоставят ЕПВР (поне 3)?

256. 1) {12} префиксен
постфиксен
инфиксен
специален
функционално префиксен

257. 304. Как се определя редът на изпълнение на операторите в написаната програма?

257. 1) {до 3} редът на изпълнение се определя по точно определен алгоритъм.
257. 2) {14} Редът се определя от езика на програмиране в зависимост от програмата
257. 3) {14} Степенуват се по **преуритети**
257. 4) {14} Според правилата за приоритет на езика
257. 5) {17} Операторите в един Еп се изпълняват по естествен ред (последователно), освен ако не се използва оператор за безусловен преход. **[Може ли ЕП да се изпълнява и може ли изпълнението да бъде по друг начин, освен един след друг?]**
257. 6) {17} Когато операторът се изпълнява един след друг
257. 7) {23} В зависимост от това как са определени в текста на програмата операторите от високо равнище се изпълняват преди операторите от ниско равнище
257. 8) {26} В написаната програма операторите се изпълняват в реда, **в който са написани** в нея.

258. 305. Какво представлява понятието „естествен ред на изпълнение“ на операторите в една програма?

258. 1) {до 3} Всички стъпки да се извършват последователно без да се прескача нищо.
258. 2) {до 3} Същност на струк. операция – съдържат като част от своя запис други оператори и определят реда за тяхното изпълнение.
258. 3) {14} Естествен ред на изпълнение е **редът, който следва компилаторът** за изпълнението на функциите в дадена програма.
258. 4) {14} Последователно се изпълняват командите
258. 5) {16} При естествения ред на изпълнение **програмата изпълнява** операторите последователно, освен ако програмиста не е задал друг вид изпълнение.
258. 6) {17} „Естествения ред на изпълнение“ на операторите в една програма представлява правилното и последователно и естествено **извършване на реда на действията**.
258. 7) {20} Понятието „естествен ред на изпълнение“ на операторите в една програма представлява последователното изпълнение на операторите
258. 8) {26} Естествен ред за изпълняване на операторите в една програма е последователност от изпълнението на един оператор след друг в зависимост от техните предназначения н програмата.

259. 306. Достатъчен ли е „естественият ред на изпълнение“ за описание на произволен алгоритъм? Защо?

259. 1) {12} Да. По-висока читаемост.
259. 2) {14, 14} Да.

259. 3) {14} Да, защото той се определя от съставителя на алгоритъма, който е преценил, че това е най-бързият и лесен ред на изпълнение
259. 4) {14} Достатъчен е защото в него е изразено всичко необходимо.
259. 5) {20} Алгоритмът представлява множество от данни и затова „естественият ред на изпълнение“ за описание на произволен алгоритъм е достатъчен, защото показва, че по естествен начин могат да бъдат подбрани множество от числа, които са произволни и да покажат, че по този начин могат да създадат описанието на даден произволен алгоритъм.
259. 6) {26} Да, достатъчен е, служи като коментар
259. 7) {28} Да, в него са описани елементарно действията, който трябва да се изпълняват, без да има нужда от допълнителни обяснения.

260. 307. Какво е необходимо за да може да се нарушава „естественият ред на изпълнение“?

260. 1) {15} Необходимо е описване на произволен алгоритъм.

261. 309. За какво е необходимо понятието „етикет“ в ЕП?

261. 1) {5–7} Всяка една програма има източници на информация с околния свят, които биват входни и изходни в зависимост от това дали информацията е насочена: програма → околен свят или околен свят → програма. Тези източници се описват с „етикет“ в дадената програма. За езика Паскал това поле е „LEABLE“. Понякога етикети се слагат на операторите, за да се номерира реда на изпълнение на операторите.
261. 2) {9} Имената на оператора
261. 3) {12} Понятието „етикет“ служи **за наименуване на** дадена **операция**.
261. 4) {14} Понятието „етикет“ служи, за да различаваме еднаквите елементи.
261. 5) {14} Понятието „етикет“ е необходимо в езиците за програмиране, защото идентифицира дадена величина.
261. 6) {18} Етикета служи за име на операция
261. 7) {20} Имената на операторите се наричат „етикети“ (За да знаем/да ни е известно с кой оператор работим)

Това са началните данни за процедура **[?!?]**

262. 310. Колко етикета може да има един оператор?

262. 1) {до 3} По принцип етикета може и да не се слага но подходящо е слагането на етикет само на тези оператори които са с по особено предназначение. Но можем да сложим по 1 етикет на всеки оператор.
262. 2) {до 3} Един оператор може да има **само 1 етикет**.
262. 3) {12} Етикет това е **името на операцията**. Всяка **операция** може да има няколко етикета.
262. 4) {14} Етикетите се избират от програмиста и могат да бъдат идентификатори (Си) и цели числа (Paska Fortrao) или и двете
262. 5) {14} 1 (един)
262. 6) {15} Етикетите се избират от програмиста и могат да бъдат идентификатори, сели числа или и двете. **[И какво общо има между този свръхинтелигентен отговор и зададения въпрос?]**
262. 7) {15} Това са величини чието множество от допустими стойности са адресите на клетките на ОП в които са моделирани величини от определен тип тоест чрез указателя можем косвено да използваме величини
Етикетите се избират от програмиста и могат да бъдат идентификатори, цели числа или и двете
262. 8) {16} **Един** етикет
262. 9) {17, 17} Етикетът може да се записва пред всеки оператор. Те се избират от програмиста и могат да бъдат идентификатори, цели числа или и двете
262. 10) {17} два или три – етикета в оператор

262. 11) {17} Етикетите се избират от програмиста и могат да бъдат идентификатори (Си) и цели числа (Паскал) или и двете
262. 12) {17} Етикет може да се записва пред всеки оператор. Етикетите се избират от програмиста и могат да бъдат идентификатори (Си), цели числа (Паскал, Фортран) или и двете
262. 13) {17} Един оператор може да има само един етикет (иначе се получава двусмислие).
262. 14) {18} Етикет може да се записва пред всеки оператор. Етикетите се избират от програмиста и могат да бъдат идентификатори (Си), цели числа или и двете
262. 15) {20, 20, 20, 23, 23, 23, 26, 26} Етикет може да се записва пред всеки оператор.
262. 16) {21} Те се избират от програмиста и могат да бъдат идентификатори, цели числа или и двете
262. 17) {23} Пред всеки оператор може да се изписва етикет.
262. 18) {26} Един
262. 19) {26} Един етикет може да се записва преди всеки отговор
262. 20) {26} Идентификатори (Си); цели числа (Паскал, Фортран) или и двете

263. 311. Кои оператори задължително трябва да имат етикет? Защо?

263. 1) {8} Които поемат управлението от прехода
263. 2) {9} **Всички**
263. 3) {12} Етикет може да се записва преди всеки пред всеки оператор, но е достатъчно с етикет да се записват само желаните оператори (тези с неестествен ред на изпълнение). **[Объркването на думи като „задължително“ и „достатъчно“, а и употребата на думата „желаните“ прави отговора напълно грешен, защото показва опит за наизустяване (не съвсем успешен!) вместо знания, осъзнаване и разбиране на материала от лекциите.]**
263. 4) {12, 14} Етикет може да се записва преди всеки пред всеки оператор, но е **достатъчно** с етикет да се отбелязват **само желаните** оператори (прости и структурни оператори).
263. 5) {14} Операторите, които ще ни трябват и нататък, които са важни за нас и които ще се променят трябва да имат етикет.
263. 6) {14} **Не е задължително**, но е препоръчително.
263. 7) {14} Тези за безусловен преход.
263. 8) {15} Етикети може да се записват пре6 сички оператори, но е достатъчно с етикети да се отбелязват само желаните оператори
263. 9) {15, 15} Етикет може да се записва преди всеки пред всеки оператор, но е **достатъчно** с етикет да се отбелязват **само желаните** оператори.
263. 10) {17} С етикет се отбелязват само желаните оператори. Това е достатъчно за използването на етикет в програмата
263. 11) {20} Коренните (основните) оператори за един език, за да се ползват по-лесно.
263. 12) {21} **Само операторите с естествен ред** на изпълнение
263. 13) {26} Задължително трябва да имат етикет, тези които са **с неизвестен ред на изпълнение**

264. 312. Как може да бъде записан етикетът на даден оператор?

264. 1) {5–7} чрез 0 и 1
264. 2) {12} Чрез идентификатор и две точки (:) или чрез число и две точки (:)
264. 3) {17} Комбинация от символи и числа
264. 4) {17} На всеки програмен език е различно например на Си с две точки **[Две точки означава записът: „..“!?!]** <етикет>
264. 5) {20} Етикетът на даден оператор се записва като коментар в програмата с цел пояснява-не на смисъла на оператора.
264. 6) {20} Етикет на даден оператор може да бъде записан така:
go to <етикет> (= премини към <етикет>)
264. 7) {20} Етикет на даден оператор може да бъде записан чрез знаков низ ограден в ' ". Етикетът е името на оператора.

264. 8) {20} Етикет на данном операторе может быт отрицательными так как он не познает еще такие слово но наука шагает вперед.
264. 9) {23} Записва се в оператора goto Етикета представлява цяло десетично число без знак. Той задължително трябва да бъде опесан в раздела на етикетите
264. 10) {27} Етищетът може да се **постави/запише пред името на даден оператор**

265. 313. Какви видове оператори има в ЕПВР?

265. 1) {до 3} В повечето ЕПВР представят следните операторите
1. Празен оператор; 2. за край на изпълнението; 3 – за безусловен преход; 4 – за присвояване; 5 – за обем на данни; 6 – за изпълнение на програмна част;
265. 2) {5–7} Видове оператори: , оператор за **уакал**. оператор за присвояване и **операто**
265. 3) {5–7} за безусловен преход, оператор за присвояване, за обмен на днни.
265. 4) {9} Оператори за присвояване, оператор за безусл. преход, празен оператор **[Това видове ли са или конкретни оператори? А и нима ги има във всеки ЕПВР?]**
265. 5) {14} Условни оператори, оператори за присвояване, структурни оператори
265. 6) {14} Условни оператори, оператори за начало и край
265. 7) {14} Условни и Циклични
265. 8) {20} по брой на операндите с един с 2
по тип с еднакъв тип или друг
по категорий логически, побитов
265. 9) {23, 23} Прости (елементарни), които не съдържат като част от своя запис други оператори.
265. 10) {26} Видове оператори – за вход и изход, за присвояване, за избор
265. 11) {26} Операторите биват: по брой на операндите – с 1, 2 и рядко с 3 операции; по тип на резултата, по категории – аритметични, логически, сравнения, символни
265. 12) {26} Прости (елементарни), които не съдържат като част от своя запис другите оператори
265. 13) {26} Интерпретатори и компилатори.

266. 314. По какво структурните оператори се отличават от простите?

266. 1) {15} Простите не съдържат операции в себе си.
266. 2) {17} **структурните** оператори **са процедури**, които включват няколко прости оператори.
266. 3) {17} Структурните съдържат в записа си други оператори. Простите съдържат в записа си други оператори. **[Страхотна разлика!?!]**
266. 4) {20} По-добри са

267. 315. Посочете основните прости оператори в един ЕПВР (поне 4).

267. 1) {14} Процедурно ориентиран и проблемно ориентиран, специализирано процедурни и неутрално процедурни
267. 2) {14} Основни прости оператори в ЕПВР:
– за въвеждане
– за извеждане
– логически
– аритметични
267. 3) {23} И, ИЛИ, АКО, ТО
267. 4) {26} – празен операто р;
– за край на оператора;
– за начало на оператора;
– за междинните резултати;

268. 316. Какво представлява „празният“ оператор?

268. 1) {14} Оператор, който може да приема стойности

268. 2) {14} С понятието алгоритъм са свързани трима: 1. съставител, този който съставя алгоритъма. 2. изпълнител, този който изпълнява последователно стъпките на алгоритъма. 3. потребител, този който задава начало и край на алгоритъма и взема крайния резултат.
268. 3) {20} Оператор при когото липсва зададен библиотек
268. 4) {20} Празният оператор е оператор без точно определено действие Той би могъл да служи за написване на коментар в определено място в програмата.

269. 317. Каква е ползата от празните оператори?

269. 1) {до 3} Приема и забавят действие.
269. 2) {до 3} Всеки оператор описва определено действие, което може да е съставено и от отделни по-прости операции.
269. 3) {9} Празните оператори, неоказват влияние върху записа. Само във Fortran имат имена и се наричат Continue.
269. 4) {14} Чрез тях се преминава от един оператор в друг. Подобряват структурата и четимостта на програмата.
269. 5) {14} Празните оператори могат да приемат стойности
269. 6) {14} Помагат на програмиста в оформлението на програмата, като я правят по ясна и четлива
269. 7) {14} Предават нагледност на написаната програма
269. 8) {17} Липсата на необходимост за изпълнението му
269. 9) {20} Не влияе на изпълнението. Обикновено няма виден запис, но във Фортран има име – Continue.
269. 10) {20} Празните оператори не влияят на изпълнението на програмата.
269. 11) {21} Създава по-приличен и прегледен вид и лесно ползване на задачата.
269. 12) {21} Ползата от празните оператори, е че се заделят в началото на програмат, а **вътре** в нея **се използват като временни стойности**
269. 13) {23} Ползата от празния оператор е, че дава възможност след който и да е оператор **да поставят** произволен брой символи ; ;

270. 319. Какво представлява операторът за безусловен преход и за какво е необходим?

270. 1) {до 3} Оператор, от АНИ. (неправилен превод.) когато е възможна да не изпълтни дад. условие, коректно и легално.
270. 2) {до 3} той представлява „Иди до“ „Go to“, необходим е да накорва програмата да задейства.
270. 3) {10} позволява на самото устройство да изпълни задачата
270. 4) {14} Операторът за безусловен преход служи за прескачане на даден **фрагмен** от програмата
270. 5) {14} Операторът за безусловен преход е оператор, при който не е нужно да има условие за да се изпълни
270. 6) {17} Операторът за безусловен преход **извиква оператор за край** на програмата
270. 7) {26} Това са „скритите оператори и се ползват за преминаване.
270. 8) {27} Нарушава естествения ред и след него се изпълнява явно посоченооператор

271. 320. Защо операторът за безусловен преход е обявен за „престъпен елемент“?

271. 1) {14} Защото пренебрегва закона за последователното изпълнение на отделните компоненти на една програма
271. 2) {21, 26} Наличието на някои структурни оператори го прави излишен и е обявен за „извън закона“ на тенденцията, наречена структурно програмиране **[Веднага става ясно защо е отречен този оператор!]**
271. 3) {23} Наличието на някои структурни оператори го прави излишен и е обявен за „вън от закона“ на тенденцията, наречена структурно програмиране Защото броят на структорните грешки, е правопропорционален

271. 4) {26} Операторът за безусловен преход е премахнат от езиците за програмиране от ВР поради натрупването на грешки в програмата

272. 322. Какво представлява операторът за присвояване?

272. 1) {10} Това е основният оператор на всеки език, защото чрез него става промяна на текущите стойности
272. 2) {14} Оператор за присвояване е „=“ (равно)
272. 3) {14} Основа на оператор за предаване на стойности.
272. 4) {14} чрез него става промяна на текущите стойности.
272. 5) {14} За присвояване на стойност.
272. 6) {17} може да се приеме за основи оператор на всеки език. изпълнението му предвижда изчисляване на посочения в дясно израз и записване чието име е в ляво
272. 7) {17} **Използването му предвижда изчисляване** на посочения в дясно израз и записване на получения резултат като текуща стойност на променливата, чието име е в ляво.
272. 8) {18} Той е основен за всеки ЕП. Чрез него се присвоява стойност
272. 9) {20} Изпълнението на този вид оператор служи за изчисляване на записаният вдясно израз и полученият резултат като текуща стойност, който обикновено стои вляво
272. 10) {20} Операторът за присвояване представлява
272. 11) {20} присвоява дадена стоиност.
272. 12) {23, 23, 23, 26} **Изпълнението му** предвижда изчисляване, на посочения в дясно израз и записване на получения резултат като текуща стойностна променлива, чието име е в ляво.
272. 13) {23} Операторът за присвояване „=“ служи когато ни трябва **дадена променлива да присвои определена стойност** или друга променлива
272. 14) {26} **Изпълнението** му предвижда изчисляване, на израза отдясно и записване на получения резултат като текуща стойностна от ляво.
272. 15) {26} Присвоява дадени стойности в програмните езици който го прави доста полезен оператор имаго (Паскал, Си, ВБ)
272. 16) {26} **Изпълнението** е изчисляване на израз и записване на получения резултат като стойност
272. 17) {26} Операторът за присвояване присвоява определени стойности
272. 18) {27} Оператор за присвояване присвоява някаква стойност или друга променлива на дефинирана променлива. В с++ присвояване се осъществява със знак равно (=) във Pascal (:=), знак за присвояване зависи от език на програмиране

273. 323. Какви са компонентите на оператора за присвояване?

273. 1) {9} На адреса се присвоява стойност.
273. 2) {14, 17} От двете страни на знака за присвояване се използват еднакви имена, но с различен смисъл. Името на променливата или на елемент от структурата в ляво означава текуща стойност на променливата или елемент на структурата от данни.
273. 3) {14} Присвоява посочена от нас стойност на дадена променлива
273. 4) {17} лява част, знак за равенство, дясна част
273. 5) {18} От двете страни на знака за присвояване се използват еднакви имена, но с различен смисъл. Името на променливата или на елемент ни структора в ляво означава текуща стойност ни променливата или елемент на структора от данни, т. е съдържанието на. О. П.
273. 6) {23} <име на променлива><знак><–израз.
273. 7) {27} КомпонеЗнтите за присвояване са

274. 324. Защо операторът за присвояване се смята за основен оператор на всеки език?

274. 1) {до 3} Защото рядко има оператор при който да няма присвояване.
274. 2) {5–7} Защото се смята за **нечестно** използване

275. 325. Как се изпълнява операторът за присвояване?

275. 1) {до 3} Записаната част от дясно се присвоява на ляво и от там цялата програма се присвоява.
275. 2) {5–7} чрез знаците :=
275. 3) {5–7} Присвоява на Адреса на променливата.
275. 4) {5–7} Стойността се присвоява от дясно на ляво
275. 5) {15} За да можем да използваме оператора за присвояване **трябва да има стойност** която да присвоим **щом се изпълнили това** поставяме знак „=“ м/у двете променливи.
275. 6) {15} Изпълнява се **с трета променлива**

276. 326. Задължително ли е всеки ЕПВР да има оператор за присвояване? Защо?

276. 1) {8} Да, защото той е основен за всеки ЕПВР
276. 2) {9} Задължително е, защото така се извършва по-бързо програмата и се избягват грешки **[Що ли значи „извършване на програма“?]**
276. 3) {9} Да. Защото той е основен оператор за всеки език.
276. 4) {15} Да, за да не се усложнява писането на алгоритъм.
276. 5) {15} Не е задължително, защото знаковият тип може да бъде заменен от текст.

277. 327. Какво е предназначението на операторите за обмен на данни?

277. 1) {12} Те са предназначени **да обменят** входните и изходни **данни** (пол. резултати)
277. 2) {12} Основно предназначение е приемането на входни данни и подаването на изходни данни навсякъде в програмата.
277. 3) {14} Предназначението на операторите е да обменят данни и информация.
277. 4) {17} **Изпълнението** на подобни оператори **е в пряка връзка за използването** на ПУ. Монополист в работата ПУОС, изпълнението на тези оператори изисква съдействието на ОС.
277. 5) {17} Чрез тях извършваме присвоявания, обмен на данни от една на друга променлива.
277. 6) {20} Да обменят данни по между си, което спестява време.
277. 7) {20} Предназначението на операторите за обмен на данни е да обменят данни смисъл **да предават определена информация от един към друг**
277. 8) {22} Операторите за обмен на данни позволяват предаване и обменяне на информация. Операторите за обмен на данни позволяват предаването на информацията да става от един на друг носител.
277. 9) {26} Определят условията при обменна на данни
277. 10) {26} Да достатъчен е служи за коментар
277. 11) {28} За да

278. 328. Какво е необходимо за изпълнение на операторите за обмен на данни?

278. 1) {9} Те осигуряват връзка с околния свят
278. 2) {9} локални и глобални променливи
278. 3) {17} За представяне на данни – преобразуване в човешки език (при изход) и обратно (при вход)
278. 4) {17, 17} Често е необходимо компютърно представяне на данни да се преобразува в човешко (при изход) и обратното при вход.
278. 5) {18, 23} Тези оператори осигуряват връзка с околния свят. В зависимост от посоката се наричат оператори за вход и изход
278. 6) {20} Тяхната функция, пътят и посоката на обмяна на данни

279. 329. От какви видове обмен на данни има нужда всяка програма и какво е характерно за всеки от тях?

279. 1) {12} Вход – въвеждане на данни за обработка
Изход – извеждане на обработените данни от входа **[И как по-точно входът ще извежда обработените данни?]**

280. 330. Посочете основните видове алгоритми.

280. 1) {до 3} Логически израз на който може да се отговори с две стойности или с да или не.
280. 2) {12} Видовете алгоритми са сложни и подсъставни
280. 3) {17} Компютърни алгоритми,
280. 4) {20} прости и съставни
280. 5) {20} Основните видове алгоритми са разклонени, **запомнящи**.
280. 6) {26} Алгоритмите се делят на прости и съставни (сложни).
280. 7) {26} Алгоритъм е последователност от действия
280. 8) {27} Алгоритмични знаци с наука и изкуство.

281. 331. {26} Алгоритъм Каква е същността на структурните оператори?

281. 1) {до 3} Описват структората на изпълнение.
281. 2) {26} Създават структура от променливи служещи за въвеждане на данни от различен тип.

282. 332. Какво представлява понятието „съставен оператор“?

282. 1) {11} Оператор който извършва повече от едно действие.
282. 2) {14} Понятието „съставен оператор“ представлява команди за въвеждане и извеждане те се набират така cin << и cout >>
282. 3) {14} „Съставен оператор“ е този който в записа си има други оператори
282. 4) {17} Съставния оператор е най-често използвания оператор след оператора за присвояване. Той може да бъде променлива константа или израз. Съставния оператор (блок) спада към управляващите оператори, но не реализира разклонение, чрез него се оформят телата в C++. Обединява няколко групи оператори в един. Общия вид на блока е: {оператор 1
282. 5) {20} Съставния оператор съдържа в себе си други оператори
282. 6) {23} Оператор, който спазва ограничението **[Кое ограничение?]** в езици, в които има такова ограничение, и в същото време осигурява последователното изпълнение на операторите, които са част от него.
282. 7) {23} Единен оператор, който спазва ограничението **[Кое ограничение?]** и осигурява последователност при изпълнението на операторите които са част от него
282. 8) {23} Един оператор, който спазва ограничението **[Кое ограничение?]** и в същото време осигурява последователно изпълнение на операторите, които са част от него.
282. 9) {26} „Съставният оператор“ е единен оператор, който се използва за преодоляване на дадено ограничение в езици, които имат такова. **[За какво ограничение става въпрос?]** Служи за **последователно изпълнение на съставящите програмата оператори**
282. 10) {26} Оператор който има неявна роля в една програма

283. 333. Защо е необходимо понятието „съставен оператор“?

283. 1) {до 3} Всеки оператор описва определено действие което може да е съставено и от отделни по прости операции
операторите служат за изразяване на заповедите в алгоритмите.
283. 2) {9} Всички езици имат съставни оператори **[Нима?]** Съставния оператор се състои от подоператори които се изпълняват по естествения ред на изпълнение.

284. 334. Възможно ли е в един ЕПВР да липсва „съставен оператор“?Ако не – защо, ако да – кога?

284. 1) {до 3} Не е възможно в един ЕПВР да липсва съставен оператор. Задължително трябва да има съставен оператор защото
284. 2) {10} Не защото осигурява последните изпълнения на операторите които са част от него.
284. 3) {23} Не е възможно в един ЕПВР да липсва „съставен оператор“
284. 4) {23} Не, защото цялата схема ще се объркa

285. 335. Какво е предназначението на операторите за разклонение?

285. 1) {до 3} Те са два вида
1. условен оператор
2. оператор за избор с условие

285. 2) {5–7} – всяка се рисува по два начина
– бяло поле
– препинателни знаци
– цифри

285. 3) {14} използва се двузначна логика

285. 4) {14} Осигуряват приспособимост на **програмат** по време на изпълнението ѝ, **във възникващите обстоятелства**.

285. 5) {14} Служат за логическо разклонение на кода на програмата

285. 6) {17} Разклоняват кода на изпълнение в зависимост от условията на операторът. Примери (C++): if; else; do; while и др.

285. 7) {20} Предназначението на операторите за разклонение е чрез тях се опростява описанието на задачата, има възможност за повече вариянти

285. 8) {26} за проверка на някакво условие.

286. 336. Може ли в един ЕПВР да липсват оператори за разклонение? Защо?

286. 1) {12} Да може, защото те могат да бъдат заменени с други оператори, които извършват същата дейност

286. 2) {14, 14} Да може

286. 3) {14, 17, 20} Да

286. 4) {18} Да, ако алгоритъмът е линеен **[Авторът на този отговор очевидно не прави разлика между език за програмиране и програма, написана на него!]**

286. 5) {20, 26} Да!

286. 6) {26} Може да липсват оператори за разклонение защото те са част от условните оператори.

287. 337. Какви видове оператори за разклонение на алгоритъма може да има в ЕПВР?

287. 1) {12} условен оператор

288. 338. Какво е характерно за условните оператори и от какъв тип е условието в тях?

288. 1) {до 3} Условните оператори осигуряват възможността да създаваме програми към обстоятелствата.

288. 2) {14} При условните оператори програмата може да се изпълни поне веднъж или нито един път взависимост от поставеното условие. **Те биват** два вида: **цикъл** със следусловие и цикъл с предусловие.

288. 3) {15} Условните оператори са от дискретен тип.

288. 4) {17} Условните оператори спомагат за решаването на всякакъв тип задачи с желаната програма. Без тях не може да се постигне крайния резултат.

288. 5) {20} Характерно условно оператору и от такого типа условие в них означает что в них нет ничего общего;

288. 6) {21} Характерно за тях е, че винаги изпълняват някакво условие и независимо какво има резултат.

288. 7) {21} Характерно за условните оператори, е че те задължително имат поне едно условие и трябва да се извърши поне веднъж

288. 8) {23} Те не изпълняват условие а казват на отделните части какво да правят

288. 9) {26} Изпълнението на едно от две зададени алтернативни действия.

289. 339. Какви проблеми създава наличието на две форми на условния оператор?

289. 1) {15} Изпълнението на програмата става по-дълго, тъй като се изискват повече проверки.

290. 341. Какво е характерно за оператора за избор и от какъв тип е условието в него?

290. 1) {5–7} Условието е **логическо**.
290. 2) {5–7} Влианието на оператори IF влошава четенето и разбирането на програмата и изисква изчисляване на множество изрази което не е рационално, а и може да не е желателно
290. 3) {9} Влиянието на оператори IF влошава четенето и изисква изчисляване на множество изрази което не е рационално, а и може да не е желателно
290. 4) {12} от произволен дискретен тип характерно е, че **има две алтернативи**, истина и лъжа,
290. 5) {14} С оператора за избор се изпълняват дадени програмни елементи, които се избират чрез въвеждането на числова променлива или някакъв друг символ.
Оператора за избор се също и от декларативна част, в която се определя стойността чрез която се избира елемент и елементи които се изпълняват.
290. 6) {17} Освен целочислен тип може да бъде от всеки друг дискретен тип
290. 7) {20} Условието му е по избор представлява избора на съставителят.
290. 8) {22} Оператора за избор дава на потребителя, или програмиста право на избор между няколко функции.
290. 9) {26} Чрез негова помощ се изчислява целочислен израз и дава възможност за изпълнение на повече от две алтернативи
290. 10) {26} Изпълнение на само едно от две възможни действия. Условието е от булев тип.
290. 11) {26} Адаптира програма по начин , по който може да има повече от 1 краен избол

291. 342. Абсолютно необходимо ли е в един ЕПВР да има специален оператор за избор? Защо?

291. 1) {15} Да

292. 343. Защо е полезно ЕПВР да предоставят оператори за циклично повторение?

292. 1) {до 3} няма ЕПВР, които да осигурява оператори за повторение на програмен участък.
292. 2) {5–7} Полезно е ЕПВР да предоставят оператори за циклично повторение, защото след като се провери условието за цикъл автоматично се взема решение за спиране или продължаване на цикъла
292. 3) {8} – краткост при писане

293. 344. Абсолютно необходимо ли е в един ЕПВР да има специални оператори за цикъл? Защо?

293. 1) {10} Да. Улеснява се програмата
293. 2) {13} Да, защото понякога се налага определена стъпка да се повтори няколко пъти (for) или да зависи от определено условие (while). А това от своя страна се налага тъй като програмата следва естествения ред на записване (**Алгоритмите се изпълняват един след друг**), а понякога това не е достатъчно

294. 345. Какво представлява понятието „тяло“ на цикъл и от какво може да се състои то?

294. 1) {14} Тяло на цикъл това е фрагмента от дадена програма, който се изпълнява до достигане на определено условие
294. 2) представлява дадено условие за което той **[Кой той?]** да се изпълни
294. 3) {17} При всеки програмен език е различно, но общото е че еденица бива превъртана до с-та зададена от нас.
В тялото се записва дадената стойност (израз), до кога да се превърта и с колко единици на 1 цикъл.
294. 4) {20} Тялото на цикъла е неговата основа
294. 5) {26} При тях част от стойностите повтарят многократно за различни стойности на променливите.
294. 6) {26} Тялото на цикъл се състои от изрази които ще се изпълнят докато условие е вярно

295. 346. Какво означава понятието „итерация“?

295. 1) {17} Променлива

296. 347. Какви видове оператори за циклично повторение на част от алгоритъма има в ЕПВР?

296. 1) {14} if lelse, do/while, for
296. 2) {14} Операторите за циклично повторение са с пред условие и след условие
296. 3) {17} 1. Условен оператор. 2. Оператор за избор с условие от цикличен тип.
296. 4) {17} Процедурно ориентирани и проблемно ориентирани. Алгоритмичните езици съдържат процедурно ориентиран подклас: специализирани и неутрални

297. 349. По какво съществено се различават цикъл с предусловие от цикъл със следусловие?

297. 1) {17} При цикъла с предусловие, преминаваме през тялото на цикъла минимум един път, а при този със средусловие – минимум два пъти ъ
297. 2) {17} Съществото по което се различават цикъл с предусловие от цикъл със следусловие е това че цикълът (~~се~~) при двете се извършва преди и след условието
297. 3) {20} **приличат си** по първото изчисляване на условието за да покаже липса на повторение

298. 348. Какви видове оператори за цикъл по условие предоставят ЕПВР и какво е характерно за всеки от тях?

298. 1) {5–7} Видове оператори: По брой на операндите: с един, с два и много рядко с три операнда:
по тип на резултата: от същия тип като операндите и от друг тип:
по категория: аритметични, логически, побитови, сравнения, символни и т. н.
298. 2) {14} 2 вида – условен оператор с логическо условие и две алтернативи и оператор за условие с повече от 2 алтернативи
298. 3) {14} Прости и структурни.
298. 4) {14} do while; if; while
298. 5) {17, 17} два вида: 1) проверка на условие чрез изчисляване на израз от логически тип – цикли по условие. 2) **преброяване на итерациите** – цикли с преброяване
298. 6) {18} Два вида:
1) проверка на условие чрез изчисляване на израз от логически тип – цикли по условие;
2) **преброяване на итерациите** – цикли с преброяване;
298. 7) {20} ЕПВР предоставя: прости оператори които не съдържат като част от своя запис други оператори; Структурни, които в записа си съдържат други оператори и задават правилата за изпълнение за тези свои съставни части
298. 8) {20, 23} Два вида: 1) проверка на условие чрез изчисляване на израз от логически тип – цикли по условие; 2) преброяване на итерациите – цикли с преброяване
298. 9) {20, 23} Има два вида: 1) Проверка на условие чрез изчисляване на израз от логически тип – цикъл по условие; 2) **Преброяване** на итерациите – **цикъл с преброяване**
298. 10) {20} While – Докато
For – За
If – Ако
Do – Направи
298. 11) {26} – проверка на условие, чрез изчисляване на израз от логически тип – цикли по условие
– преброяване на итерациите – **цикли с преброяване**
298. 12) {26, 26} Два вида:
1) проверка на условие чрез изчисляване на израз от логически тип – цикли по усло-

вие;
2) преброяване на итерациите – **цикли с преброяване**

298. 13) {26} 2 вида
– Първия проверява условието чрез изчисляване на израз от логически тип
– Втория преброява итерациите – **цикли с преброяване**.

299. 354. Кой от операторите за цикъл с условие е „по-необходим“ – с предусловие или със следусловие? Защо?

299. 1) {12} Това не може да бъде определено еднозначно, тъй като програмистът избира кой цикъл да използва според нуждите за реализирането на конкретния алгоритъм или задача

300. 355. Какво е характерно за оператора за цикъл с преброяване?

300. 1) {5–7} прави проверката в началото
300. 2) {9} Цикълът с преброяване или т. нар. цикъл с брояч. при него вместо да извършваме ние преброяването дадения цикъл извършва преброяването.

301. 357. Защо при циклите с преброяване е необходимо да се посочва „управляваща променлива“?

301. 1) {12} Необходимо е посочването на „управляваща променлива“ при циклите с повторение **[Дали бихме могли да говорим за цикъл без евентуално повторение?]** защото чрез нея се преброяват повторението на цикъла

302. 358. От какъв тип трябва да бъде управляващата променлива в циклите с преброяване? Защо?

302. 1) {8} от тип **реален**.
302. 2) {15} Тя трябва да е от тип „int“ защото се отнася за цели числа
302. 3) {15} Променливите трябва да бъдат от цял тип
302. 4) {16} От целочислен тип **[Липсата на обяснение „защо“ прави отговора напълно грешен]**
302. 5) {20} Управляващата променлива трябва да бъде от тип integer защото програмата не може да разчита дробни числа в циклите за преброяване.
302. 6) {20} **От произволен тип**, защото е подходящо за броене
302. 7) {22} Тип цяло число (integer) Защото ако използваме променлива не цяло число, преброяването ще върне грешен или невалиден резултат.
302. 8) {26} int, защото преброява целите числа

303. 362. По какво си приличат цикъл с преброяване и цикъл с предусловие?

303. 1) {5–7} и при двете има цикъл на преброяване
303. 2) {12} И в двата случая цикъла е заложен преди оператора за изпълнение.
303. 3) {14} И двата цикъла трябва да имат променлива, която сменя стойността си за всяко повторение на цикъла.
303. 4) {14} Повторението на дадена част от програмата се нарича цикъл. В този случай **[Кой случай?]** цикълът на преброяване и предусловие си приличат по това, че и двата изискват условие.
303. 5) {17} Цикълът с преброяване и цикълът с предусловие не си приличат, те даже
303. 6) {18} По изчисляване на условието, за да покаже липса на повторението **[Този отговор е неясен и не съвсем пълен!]**
303. 7) {18} Имат допустими стойности
303. 8) {20} Преброяват колко пъти са изпълнени условията
303. 9) {20} Приличат си по това, че поне веднъж трябва да се изпълни ~~(функцията)~~ **цъкъла**
303. 10) {26, 26} Приличат си по първото изчисляване на условието, за да покаже липса на повторение.

304. 366. Кога използването на оператор за безусловен преход не е „престъпление“?

304. 1) {12} Когато се използва go to (етикет) – явен преход
304. 2) {14, 17, 17, 17, 17, 20} Когато имаме go to <етикет>; – явен преход
304. 3) {14} Не е престъпление, когато операторът е със следусловие, така както do{...} while (условие)
304. 4) {14} Почти никога, но се допуска в крайни случаи, когато изпълнението на части от програмата се окаже ненужно
304. 5) {15} Когато използваме оператора go to за напускане на вложения цикъл **и преминаваме във външния**. Това е единствения случай в който трябва да се използва go to.
304. 6) {17} Когато използването на оператора са условен преход е „престъпление“
304. 7) {20} Когато информацията не е важна.
304. 8) {23} Наличието на някои структурни оператори го прави излишен и е обявен „вън от закона“ на тенденцията, наречена структурно програмиране, защото броя на логическите грешки е право пропорционален на неговото изпълнение
304. 9) {26} Когато е необходимо да се наруши естествената последователност на операциите.

305. 368. Защо безкрайните цикли са опасни?

305. 1) {10} Нямат край и постоянно се повтарят **[А нима точно това не се нарича безкраен цикъл?]**
305. 2) {10} Защото повтарянето е безкрайно
305. 3) {14} Защото водят до претоварване на програмата и неправилни резултати.
305. 4) {20} Безкрайните цикли са опасни, закото при тях не е деклариран оператор за прекратяване на действието, което прави работата им безкрайна, освен ако не се декларира такъв оператор. ~~В противен случай е възможно~~

306. 369. Има ли ЕПВР, при който да не може да се програмира безкраен цикъл?

306. 1) {14, 17, 17, 17, 20} Да
306. 2) {17} Зависи от използвания компилатор. Oracle → компилатора на Java предотвратява изпълнението на програма с безкраен цикъл. **[Какво ли общо има между писането и изпълнението на една програма?!?]**
306. 3) {18} Процедурно ориентирани и проблемно ориентирани са подкласовете които имат ЕПВР в алгоритмичните езици. Алгоритмичните езици съдържат процедурно оринтирани подклас: специализирани (Фортран, Кобол) и неутрални (PL–1, Паскал, Си)
306. 4) {26} Да, има.
306. 5) {26} Безкрайните цикли са опасни защото действието няма край

307. 370. Каква е ползата от наличието на оператор за явен безкраен цикъл?

307. 1) {9} Изпълняват се операторите в цикъла
307. 2) {10} Дава възможност безкрайните цикли все пак да завършат своите повторения **[И как по-точно се осигурява това?]**
307. 3) {20} Ползата е безарна Тези оператори намират своето приложение но с тях трябва да се внимава защото все някъде трябва да се сложи край
307. 4) {20} ползото от повторение
307. 5) {21} При неколкократна промяна на стойност, събития или действия, безкрайният цикъл подпомага за непрекъснатото възприемане на промените. Това, че е явен подпомага за по-лесното вегово премахване при нужда от редактиране.
307. 6) {26} В цикли с преброяване и по условие

308. 376. За какво се използват глобалните променливи?

308. 1) {12} Глобалните променливи са такива, че при деклариране им, после чрез извикването им могат да се използват навсякъде в програмата. Те са универсални

309. 371. Кога една програма се дели на части?

309. 1) {до 3} когато се използват функции и процедури които делят информацията на части.
309. 2) {12} Когато е необходима по-голяма прегледност, за да може повече хора да работят и евентуално и компютри по написване на програмата **[Как ли компютър ще пише програма?]**
309. 3) {14} Една програма се дели на части при необходимост към създаването на програмата да се присъединят и други хора и евентуално и други компютри, което дава възможност изготвените и проверени части да бъдат използвани.
309. 4) {15} Дели се на части, за да има по-прегледен вид и когато искаме да използваме предварително съставени части от други програми
309. 5) {17, 17, 21} Една програма се дели на части **когато има по-големи програми**
309. 6) {20} Програма с твърде много алгоритми. За да може да се чете по-лесно.
309. 7) {20} Една програма се дели на части когато
309. 8) {20} Когато за реализиране на програмата са нужни множество от алгоритми и подалго–ритми или изпълнението на определени действия няколко пъти.
309. 9) {26} За да се раздели програмата на части трябва да е създадена от подпрограми
309. 10) {26} Когато се използват по големи програми. Те се сглобяват от отделно оформени програмни части и допълнително написани оператори

310. 372. С какви цели една програма се разделя на отделни части (поне 3)?

310. 1) {16} Програмата придобива по четлив и разбираем вид за програмиста.

311. 373. Какви видове програмни части предлагат ЕПВР?

311. 1) {5–7} Основни символи, оператор, функция и модул.
311. 2) {12} Функции за процедура

312. 374. Какви видове променливи въвежда деленото на програмата на части?

312. 1) {12} Видовете променливи, които въвежда деленето на програмата ~~на части~~ са локалните променливи
312. 2) {13} Локални променливи. Техният живот започва от началото на програмната част и свършва в края на съответния фрагмент.
312. 3) {14} статични и динамични променливи
312. 4) {17} Локалните променливи.
312. 5) {20} За обмен на информация с другите части се използват нелокални променливи биващи: глобални, общи и променливи
312. 6) {20} Намалява зависимостите между различни части на програмата.
312. 7) {23} Видовете променливи, които въвежда деленето на програмата са локалните променливи

313. 375. Какво е характерно за локалните променливи при повечето ЕПВР?

313. 1) {до 3} При нея има най-малко стояния от другите и е одобна за компютри.
313. 2) {до 3} локалната променливи е с по-голям приоритет от глобалните и нейното действие е точно дефинираната ф-я.
313. 3) {5–7} Локалните променливи в ЕПВР имат характеристика, че за разлика от глобалните тяхната стойност се обработва само в отделния блок или подпрограма, а не важи за цялата програма.
313. 4) {14} Характерна особеност на програмните части е възможността във всяка от тях да бъдат определени допълнителни променливи. Тези собствени променливи се наричат локални.
313. 5) {14} Имената на локалните променливи се избират от пишещият програмата. Те съдържат поне 2 допустими стойности. **[Едновременно ли ги съдържат??]**
313. 6) {14} Че се отнасят за определена част от програмата не за цялата задача.

313. 7) {17} Възможността в тях да бъдат определени допълнителните променливи. „ Раждат“ се при започване на изпълнението **[На какво?]** и „Умират“ в края му.
313. 8) {17} Възможността да тях да бъдат определяни допълнителни.
313. 9) {17} да бъдат определяни допълнителни променливи
313. 10) {17, 20} Характерна особеност на програмните части е възможността във всяка от тях да бъдат определени допълнителните променливи.
313. 11) {20} Програмата работи по-бързо, но не ефикасно
313. 12) {23} Могат да се ползват само в съответния блок, в който са **Във всяка от тях** могат да бъдат определени допълнителни променливи.
313. 13) {24} Характерна особеност на програмните части е възможността във всяка от тях да бъдат определени допълнителните
313. 14) {26} Характерна особеност на програмните части е възможността във всяка от тях да бъдат определяни допълнителни променливи. Тези променливи се наричат локални.

314. 377. Каква е разликата между дефиниция и декларация на променлива?

314. 1) {12} В една случаи се задава множеството от допустимите стойности, а в другия видовете операци които могат да се извършва.
314. 2) {16} Дефиниция – начин да се определи променливата.
Декларация – декларираме променливата.

315. 378. Какво представляват общите променливи при някои ЕПВР?

315. 1) {14} Общите променливи в ЕПВР са нужните данни и поредица от прости действия, които трябва да бъдат инициализиране и идентифицирани
315. 2) {15} Изгодна стратегия е за всеки основен символ да има отделен графичен знак
315. 3) {16} Не е възможно да се използват променливи за
315. 4) {17} Това са управляващите променливи
315. 5) {18} Пишат се в началото и се използват и променят в програмата
315. 6) {19} Елементите от които се изграждат ЕПВР – осн. символи, величини, изрази, оператори, програмни части, програми.
315. 7) {23} Общите променливи в ЕПВР съдържат в себе ди по прости променливи. Изграждат сложните от които образуваме съставните.
315. 8) {23} Това са променливи които могат да се виждат във всяка част от програмата.
315. 9) {23} За разлика от глобалните променливи, общите променливи имат различни имена във всяка част, но **не се разполагат** на едно и също място в ОП
315. 10) {26} За разлика от глобалните променливи, общите променливи имат различни имена във всяка част.

316. 379. Какво представляват блоковете?

316. 1) {14} Блоковете представляват геометрични фигури, всяка със собствено значение. Във всеки блок се записва дадена част от алгоритъма
316. 2) {14} Кодове
316. 3) {17} Блоковете са: Представяне по графичен начин даден алгоритъм. Неговото начало → последователност от действия, които трябва да се извършат и край
316. 4) {17} Блоковете са съвкупност от оператори, изпълняващи се един след друг **[Това е определение за „съставен оператор“, а не за „блок“!]**
316. 5) {17} Блоковете – кратко, съкратено – съвкупност от оператори които се изпълняват един след друг
316. 6) {17} Вътре в тези фигури се въвеждат данните със които трябва да работи изпълнителят.
316. 7) {23} Блоковете са метод за изготвяне и записване на алгоритми. Блоковете се записват с помощта на геометрични фигури, като всяка от тях има съответното значение
316. 8) {20} Най-елементарното **действие** в една програма
316. 9) {21} Визуален начин за изписване на даден алгоритъм с помощта на схема.

316. 10) {21} Визуален начин за изписване на даден алгоритъм с помоща също така и на схема Това представляват блоковете
316. 11) {23} **Метод за използване** и записване **на алгоритми**
316. 12) {25} Блоковете са геометрични фигури, с чиято помощ се изразяват блок-схемите, пра което всяка от тях има определено значение.

317. 380. Каква е разликата между блок и съставен оператор?

317. 1) {до 3} Блок опесва начина надействие, а съставния оператор групира няколко оператора в един и се изпълняват в програма.
317. 2) {5–7} *<u>блок</u> → писмено, на лист.
* <u>съставен оператор</u> → в к. програма
317. 3) {12} Разликата между блок и съставен оператор е, че в блока може да се съдържат съставни оператори.
317. 4) {14} Съвкупността от съставни оператори изгражда самият блок т. е. блокът се състои от оператори.
317. 5) {20} Съставния оператор може да се съдържа в блок, който няма край.
317. 6) {20} Блок-схемите са метод за изготвяне и записване на алгоритъм, а съставния оператор изпълнява алгоритъма.
317. 7) {20} Разликата е че при блок оператори се правят блок-схеми а при съставния не.
317. 8) {23} Единствената разлика е че при употребата на блок могат да бъдат **използвани** нови променливи
317. 9) {23} Няма разлика. Блок и съставен оператор са едно и също нещо.

318. 381. Какво е характерно за процедурите като програмни части?

318. 1) {12} Процедурите са основната част, която изгражда програмата.
318. 2) {14} Процедурите (подпрограми) се използват за разделяне на програмата на части с цел компактност и оптимизация на работата.
318. 3) {14} Процедурите са обособени и разделени
318. 4) {15} Процедурата е такава част от програмата в която елементите са еднотипни и там се извършва цялото действие.
318. 5) {17} Характерното за процедурите е, че се (~~извър~~) извършват в реда и последователността на действията.
318. 6) {17} Изискват детайлно описание на задачата **затова се наричат и алгоритмични езици**
318. 7) {20} Процедурите са следствие от последователно свързани слислови действия
318. 8) {23} Процедурата е програмна част от по-високо равнище спрямо блока.
318. 9) {23} По-големите програми се делят. Те се сглобяват от отделно оформени програмни части и допълнително написани оператори
318. 10) {26} Процедурата е програмна част от високо равнище Ползата от процедурите пролича-ва най-ясно когато те могат да се параметризират.
318. 11) {26} се дефинира на едно място но може да се използва във всяка избрана от потребителя точка.

319. 382. Какво представляват формалните параметри на процедурите при ЕПВР?

319. 1) {до 3} Заедно с името на макроса се опр. и имената, с които ще се отбелязват параметризираните участъци.
319. 2) {до 3} Не е необходимо изпълнителят да има представа за решаваната задача и естеството на получаваните резултати, достатъчно е той да изпълни една след друга заповедите.
319. 3) {5–7} Параметри на, които в програмата се слага фактическа стойност
319. 4) {12} Параметрите биват фактически и формални.
319. 5) {17} Променли**ви, които се използва** вътре в процедурата
319. 6) {17} **Фиктивните параметри** в дефиницията на подпрограмата се наричат формални параметри.

319. 7) {20} Параметърът формалност означава, че програмистът няма нужда да знае целия код на програмата на готово, а да знаекретка, последователно как да напише предписаните му елементарни операции

320. 383. Какво представляват фактическите параметри на процедурите при ЕПВР?

320. 1) {5–7} Носят истинската стойност.
320. 2) {9} Посочва името на основния параметър
320. 3) {14} Това са параметрите, които се поставят при извикването на дадена процедура.
320. 4) {14} Това са променливите, които се дават при деклариране на функцията.
320. 5) {20} Фактически параметри те може да се използват в програмата
320. 6) {20} Фактически параметри това са величините, **които участват в извикването** на конкретна подпрограма
320. 7) {20} Физическите реализации на формалните параметри се наричат фактически параметри.

321. 384. Какво представляват входните параметри на процедурите при ЕПВР?

321. 1) {14} Входните параметри са първоначално въведените данни в програмата
321. 2) {17} Параметри за вход които по нататък ще се използват в програмата
321. 3) {20} Това са **основните параметри**, които **се въвеждат в началото**
321. 4) {21} **Това е резултата** от работата на процедурата
321. 5) {23} Фиктивните променливи в дефиницията на подпрограмата се наричет формални параметри.

322. 385. Какво представляват изходните параметри на процедурите при ЕПВР?

322. 1) {14} Изходните параметри могат да бъдат от различни (видове) типове, но трябва да бъдат от същия тип като функцията.
322. 2) {17} Изходните параметри са променливи които оказват влияние върху работата на експертната система

323. 386. Какво представляват функциите при ЕПВР?

323. 1) {до 3} Конкретно действие с определена обективна цел.
323. 2) {до 3} Функциите представляват възможност на дадена програма или машина.
323. 3) {до 3} Стандартните функции определени от езика са с ранг на операции.
323. 4) {9} Дефинират се в специален раздел на в началото на всяка програмна част и могат да се ползват в нея и в нейните вложени части [При всеки ЕПВР ли? А и това дали обяснява що е функция?]
323. 5) {14} те предоставят на програмиста начин да раздели програмата си така че да не трябва постоянно да пише един и същ код, а просто да извика функцията. Чрез тях кода е по-малък и функционален **[Изключително дълъг и неправилен отговор, от който става ясно единствено, че авторът му пак не е учил за изпита!]**
323. 6) {14} Представляват подалгоритми в програмния код, които се извикват от главната функция
323. 7) {17, 21} Те представят на програмиста начин да раздели програмата си така, че да не трябва непрекъснато да се пише един и същи код, а просто да извиква функцията
323. 8) {17} Те представят начин на програмиста да раздели програмата си така, че да не трябва непрекъснато да се пише един и същи код, а просто да извика функцията. Чрез функциите кодът се поддържа малък, чист и функционален.
323. 9) {17} Те представят начин на програмиста да раздели програмата си така, че да не трябва непрекъснато **да се пише на един и същи код**, а просто да извика функцията. Чрез функциите кодът се поддържа малък, чист и функционален.
323. 10) {17} В тялото на функцията написваме стъпка от алгоритъма, която ще се повтаря повече от веднъж. И след това извикваме функцията на определените места.

323. 11) {20} Те са от съществено значение. Използват се за улеснение, тъй като могат да бъдат извиквани многократно и по всяко време в процеса на програмирането.
323. 12) {26} Отделна част от код на програмата, който може да се счита като обособена подпрограма, която може да се извиква периодично в по-нататъчните редове на кода с цел съкращаване на писания код.
323. 13) {26} Операциите в повечето ЕПВР ся твърде много Поради това едва ли е възможно за всеки знак на операцията в езика да има уникален основен символ. Двусмислието на знака се премахва чрез анализ на операндите
323. 14) {27} Сбор от команди и алгоритми

324. 388. Необходимо ли е броят на фактическите параметри да съвпада с броя на формалните?

324. 1) {9} да, всеки параметър трябва да се изрази
324. 2) {14} Желателно е. Обикновено езиците за програмиране високо равнище изискват броя на факт. параметри да бъде фиксиран.
324. 3) {14, 17} Да
324. 4) {17} ЕПВР изискват броя на формалните параметри да бъде фиксиран и всяко извикване да бъде винаги със същия брой фактически параметри
324. 5) {21, 26} **Обикновено** ЕПВР изискват броят на формалните параметри да бъде фиксиран и всяко използване да бъде винаги със същия брой фактически параметри.
324. 6) {23} Не е необходимо броя на формалните и фактически параметри да съвпада.
324. 7) {26} Не
324. 8) {26} Необходимо е броя на фактическите параметри да съвпада с формалните
324. 9) {26} Задължиче са равни

325. 389. Какво представляват ключовите параметри при ЕП?

325. 1) {9} забранени идентификатори

326. 392. Каква е разликата между обикновена и динамична променлива?

326. 1) {до 3} Динамичната се дефинира в момента докато обикновената предварително.

327. 393. Кой е отговорен за заделянето и освобождаването на ОП, необходима за реализиране на обикновена променлива?

327. 1) {12} авторът на програмата
327. 2) {14} Програмата.
327. 3) {14} съставителят на програмата, при въвеждането на типа на променливата
327. 4) {17, 18, 20} Памет за обикновенните променливи се отделя при започване на изпълнението на съответната програмна част и се освобождава при завършване на изпълнението **[Чие изпълнение?]**
327. 5) {17} Управляващият блок на ЦП
327. 6) {20} самата програма
327. 7) {21} Централния процесор.
327. 8) {21} Отговорен за заделянето и освобождаването на ОП, необходима за реализиране на обикновена променлива е
327. 9) {23, 23, 26} **Памет за динамични променливи** се заделят и освобождава **в паметта** по заявка на програмата
327. 10) {26} Паметта за обикновенна променлива се определя при започване на изпълнението на съответната програмна част, и се освобождава при завършване на изпълнението.
327. 11) {26} Програмиста отговорен за заделянето и освобождаването на ОП, необходима за реализиране на обикновената променлива.

328. 394. Кой е отговорен за заделянето и освобождаването на ОП, необходима за реализиране на динамична променлива?

328. 1) {14, 14} Централният процесор.
328. 2) {17} Централният процесор. (след разпознаване на програмния код, който изисква това.)
328. 3) {20} Компютърът!
328. 4) {23} **Памет за обикновените променливи** се отделя при започването на изпълнението на съответната програмна част и се освобождава след като свърши изпълнението ѝ.
328. 5) {23} **Памет за обикновените променливи** се отделя при започване на изпълнението на съответната програмна част и се освобождава при завършването на изпълнението

329. 395. Какво представляват статичните променливи?

329. 1) {9} Статичните променливи се въвеждат от самия език на който пишем
329. 2) {12} Статична променлива – Всички променливи с изключение на тези които са локални – функциите и процедурите.
329. 3) {14} Тяхната текуща стойност не се променя в хода на програмата **[Защо тогава се наричат „променливи“?]**
329. 4) {15} Защото ИПВР имат структура, която е точно определена и зависи от (УУ) компютъра.
329. 5) {17} Статична променлива се определя от класа на паметта static. Модификаторът static се поставя преди цифровите данни.
329. 6) {17} Статичните променливи се определят от класа на паметта static. Идентификатора static се поставя преди типовете данни.
329. 7) {18} Статична променлива се определя от класа на паметта „static“ (поставя се преди типовете данни >static)
329. 8) {20,26} Статичните променливи се определят от класа на паметта static. Записването на static пред името на функцията е прави локална. Записването на static пред името на процедурата означава, че всички нейни локални променливи са статични (Вижуал Бейсик).
329. 9) {21, 27} Статичните променливи се определят от класа на паметта static. Записването на static пред името на ф-ята я прави локална (Си); Записването на static пред името на процедура означава, че всички нейни локални променливи са статични.
329. 10) {23} Статична променлива се определя от класа на паметта static, записването на static пред името на процедура означава, че всички нейни локални променливи са статични.
329. 11) {23} Написването пред функция static я прави локална, а пред процедура означава че всички нейни локални променливи са статични
329. 12) {26} Статична променлива се определя от класа паметта static, записването на static пред името на функцията я прави локална
329. 13) {26} За разлика от глобалните променливи, статичните променливи действат на локално ниво т. е. във функцията която са използвани.
329. 14) {26} Статични променливи се определят от класа на паметта static. Записването на static пред името на функцията я прави локална
329. 15) {27} Декларират се със static в началото. определя се от класа на паметта static
329. 16) {27} Статични променливи са от клас static.

330. 399. Какво представляват рекурсивните процедури?

330. 1) {5–7} Процедура която предоставя управлението една на друга.
330. 2) {20} Рекурсивните процедури променят реда на изпълнение на командите и операторите в програмата
330. 3) {26} Процедури, при които когато се изпълняват, Изпълняват само себе си.
330. 4) {26} Казва се, че един обект е рекурсивен, ако той частично се съдържа в себе си или е дефиниран чрез себе си

331. 400. Какви видове рекурсия съществуват?

331. 1) {17} Рекурсията бива два вида – права и **циклична**.

331. 2) {26} 2 вида – права и обратна

332. 401. Какво представлява пряката рекурсия?

332. 1) {9} **Рекурсия** която използва себе си

333. 402. Какво представлява косвената рекурсия?

333. 1) {до 3} една функция която върви сама по-себе си.
333. 2) {5–7} Косвената рекурсия използва пряката рекурсия

334. 406. Какви проблеми създават големите текстове?

334. 1) {до 3} Лесно се възприемат, но трудно се разбират.
334. 2) {12} Тъй като в повечето ЕП **[Да има „ЕП“ във въпроса?]** текстовият тип е ограничен до 255 символа **[О, минало незабравимо!]**, **проблем** при използването **[Във въпроса няма и намек за използване!]** на голям текст би било **разделянето му на части в няколко променливи** за да се съхрани и обработката му след това. За целта ЕП предлагат четене и записване във файл
334. 3) {14} големите текстове създават предпоставки за повече синтактични грешки
334. 4) {14, 17} Единствената реакция при писане на твърде дълъг текст и при ограничение във времето за писане е да се използват съкращения.
334. 5) {17} Единственото **решение** е да се използват съкращения за да не бъдат толкова обемни.
334. 6) {17} писане на твърде дълъг текст, и ограничение при времето
334. 7) {20} загуба на ценно време
334. 8) {20} Има по-голяма вероятност за грешки.
334. 9) {22} Натоварват излишно програмата
334. 10) {28} Трудни са за обработка и проверка

335. 407. Какво позволява да се ускори създаването на голям текст?

335. 1) {до 3} Ускоряване за създаване на голям текст извършва макроапарата.
335. 2) {26} Транслаторите и преводите и разбирането му от повече хора

336. 408. Какъв проблем създава използването на съкращения при писане на текст?

336. 1) {до 3} Използване на съкращения могат да предизвикат загуба на смисъла на текста и лесно може да се допусне грешка.
336. 2) {14} Може да възникне двусмислие и неточности.
336. 3) {20} Неразбираемост при хората, които не са запознати с текста.
336. 4) {20} Използването на съкращения при писане на текст довежда до много грешки.
336. 5) {26} а) Абривиатура на съкращения трябва да знае и изпълнителя
б) съкращения трябва да бъдат различни, за да не се получи двусмислица

337. 409. Как може да бъде решен проблемът за разчитане на текст, който съдържа съкращения?

337. 1) {до 3} Проверка на думата която е с точка, проверяване в речника за грешка или дума.
337. 2) {9} Чрез коментар
337. 3) {14} Чрез макроапарат
337. 4) {14} Проблема може да бъде решен ако има коментар върху самото съкращение.
337. 5) {21} Като се набави програма или компютър с възможност да разчита дадените съкращение Може и да се използва макроапарат

338. 410. Какво представлява понятието „макроапарат“?

338. 1) {до 3} макроапарат
макро – голям следва, че е **голям апарат**.

338. 2) {12} При писането на текстове в които използваме съкращения макроапарата представя съкратената дума в нейната пълна форма.
338. 3) {12} **Устройство**, което извиква и превежда съкращения.
338. 4) {14} Макроапарата се използва за по лесно и бързо писане на текстове със съкращения – макроси.
338. 5) {20} Понятието „макроапарат“ служи за съкращение на текст. Идва от гръцката дума „макро„.
338. 6) {20} Макроапарата се е цялостен апарат който се състои от други по-малки Съставни части.
338. 7) {26} Под понятието „макроапарат“ се разбира как ще става дефинирането и съкращенията в текста.

339. 411. Как може да бъде реализиран макроапарат?

339. 1) {14} избрлояване и ограничение
339. 2) {14} Чрез използване на съкращения на думите, които са дефинирани предварително в списък.
339. 3) {18} Макроапарата е съвкупност от съкращения и техните дефиниции
339. 4) {20} Реализация на макроапарата е чрез него се опростява алгоритмичните програми
339. 5) {21} Макроапарат може да бъде реализиран по начин Макроапарата може да бъде реализиран като **[Изключително подробен и изчерпателен отговор!?!]**
339. 6) {21} Макроапарат може да бъде реализиран
339. 7) {26} правилата, които опр. как става дефинирането и използв. на съкращенияв текста

340. 412A. Посочете поне две предимства на диалоговия (оперативен) в сравнение с пакетния (опосредствен) макроапарат.

340. 1) {26} По-удобен е за работа и предлага повече възможности на потребителя.

341. 412B. Посочете поне два недостатъка на диалоговия (оперативен) в сравнение с пакетния (опосредствен) макроапарат.

341. 1) {17} ① при **диагоналния** се налага да използваме компютър
②
341. 2) {17} Реализира се само при използването на компютър, защото е необходима специална програма

342. 414. Защо е полезно един текст да бъде копиран в различни програмни части?

342. 1) {до 3} За по лесна транслация.
342. 2) {5–7} Полезно е един текст да бъда записан в различни програмни части, защото така информацията може да бъде запазена при евентуална грешка.
342. 3) {8} За да бъде разбран и достъпен за потребителя
342. 4) {8} Един текст може да бъде дублиран или да бъде изкаран на отделен файл. Това позволява по лесно да се подобрява.
342. 5) {9} Вместо този текст да се дублира във всяка програмна част по-удобно е той да бъде записан в отделен файл с свяка програмна част да посочва включването на този файл **[Въпросът е за ползата от подобно решение!]**
342. 6) {14} Полезно е един текст да бъде копиран в различни програмни части, защото така върху програмата могат да работят повече хора **[Как?]**, по лесно се откриват грешки **[Защо?]**. Всяка отделна програмна част може да се използва и в друга програма.
342. 7) {14} Можем да използваме величини вече готови и проверени текстове
342. 8) {14} За уленение
342. 9) {17} Защото понякога да използваме даден алгоритъм (или дефиниция) от една програмна част в друга. Копирането на текста (вместо използването на функция например) би довело до по-бързо изпълнение на програмата за сметка на повече използвана ОП.
342. 10) {17} Като резерва, в случай ме в някоя част бъде изтрита

342. 11) {17} Един текст копиран в различни програмни части може да бъде представен под различен стил и вид, различна структура и с новия си вид да бъде запаметен със съответната програма.
342. 12) {20} За спестяване на време и по-голямо удобство на програмиста
342. 13) {23} Полезно е защото така у възможно да се включат повече хора в съставянето на програмата като това би спомогнало за допускането на по-малко грешки
342. 14) {26} По-компактна. Позволява много хора да работят по една програма
342. 15) {27} **За да бъдат** избегнати (**коригирани**) **грешки**.
342. 16) {28} Оптимизация

343. 415. Какво означава понятието „макродефиниция“?

343. 1) {12} Макродефиниция е определяне на множество от допустими стойности
343. 2) {14} Малка дефиниция, определяне за нещо

344. 416. Кога се повишава ползата от използване на макродефиниции?

344. 1) {12} Макродефиниции – заместват съкращения м програмата макродефиниция. дават точните думи (изрази), които трябва да са на мястото на съкр. за да включим макроапарата използваме # include <име на файла, който обичайно е с разширение .h>. Повишава се четимостта и се пише бързо програмата, защото има много части от програмата, които се повтарят.

345. 418. Какво означава понятието „макроизвикване“?

345. 1) {14} Извикване на макродефиниция
345. 2) {14} Използва се при макроапарата
345. 3) {17} Понятието „макроизвикване“ е името на вече дефинирана макрос
345. 4) {19} макроизвивка –

346. 419. Какво означава понятието „макроразширение“?

346. 1) {до 3} макроразширение – **голямо разширение**.
346. 2) {8} Пишем текст със съкращения, а ни се извежда без съкращения.
346. 3) {9} Макроразширението представлява използването на макроиз. чрез дефинирането на Макрос **[Как нещо може да се използва чрез, а не след, дефинирането си?]**
346. 4) {20} Свързан е с Цп.
346. 5) {23} „Макроразширението“ еначин на изпращане на информация, като се търси информация по списъци.
346. 6) {26} При неувереност в програмирането използваме съкратени изрази

347. 420. По какво си приличат процедурите и макроизвикванията?

347. 1) {14} Приликите са само външни
347. 2) {17} И двете извикват стойността на съкратения код
347. 3) {18} Те следват различни цели [Страховита прилика!]
347. 4) {20} И двете могат да бъдат използвани повече от един път
347. 5) {20} Макроизвикване – използването на име извикано от друга програма

348. 421. По какво се различават процедурите и макроизвикванията?

348. 1) {8} Процедурите са част от програмирането, а макроизвикванията обработват текста
348. 2) {9} Служат за извикване на данни – макроизвикванията процедурите са свързани с определени преобразования , (умножене, делене)
348. 3) {14} Процедурите се състоят от оператори, изрази и т. н., докато макроизвикването – това е когато на мястото на съкращението излезне ~~токста~~ пълния текст.
348. 4) {19} **Те си приличат**, но се различават по това, че макроизвикванията **използват вече използван** макрос.
348. 5) {23} Условната операция е с 3 операнда и е изключение от т. 2 и т. 3.

349. 422. Какво печелим и какво губим при използване на процедури при програмиране?

349. 1) {14} При използване на процедури, можем да ги извикаме, когато си поискаме по време на програмата и така самата програма става по подредена и събрана, губим време при писането ѝ.

350. 422A. Какво печелим при използване на процедури вместо макроси при програмиране?

350. 1) {17} Печелим време
350. 2) {17} Печелим бързина, но губим оперативна памет.
350. 3) {20} процедурите са по лесни за употреба и манипулиране
350. 4) {20} Печелим време
350. 5) {23, 26} Печелим време при използването на процедури вместо макроси при програмирането
350. 6) {26} Когато използваме процедури разбираме по-добре и печелим време

351. 422B. Какво губим при използване на процедури вместо макроси при програмиране?

351. 1) {12} Губим оперативна памет, **печелим** време
351. 2) {14} губим ОП
351. 3) {14} При използване на процедури губим средства
351. 4) {20} **Губим** ОП, но печелим бързина.
351. 5) {20} Губим **ефективност**

352. 423. Какво печелим и какво губим при използване на макроси при програмиране?

352. 1) {4} Макроапарата може да бъде реализиран по два начина оперативен диалогов щом завърши писането на съкращение то незабавно се замена е пакетен (опосредствен) път: текст който съдържа дефиниции и съкращения се обработва.

352. 2) {8} – лесно се чете ⎫
– малък обем ⎬ печели
– ОП печели ⎭
– губи се време

352. 3) {9}

Печелим:	Губим:
– време **[Чие и защо?]**	– трудно за разчитане **[Защо?]**
– място **[Как и защо?]**	

353. 423A. Какво печелим при използване на макроси вместо процедури при програмиране?

353. 1) {14} **Губим** време, печелим ОП
353. 2) {14} Използването на макроси ни дава възможност да спечелим време и труд.

354. 423B. Какво губим при използване на макроси вместо процедури при програмиране?

354. 1) {17} При използването на макроси вместо процедури, макар и да си спестяваме труд губим много голяма част от гъвкавостта на даден алгоритъм

355. 424. Може ли да използваме макроси при интерпретативен транслатор? Защо?

355. 1) {12} Да, за улеснение на писането на програмата

356. 425. Каква е основната идея, с която са създадени ЕПВР?

356. 1) {14} За улеснение на програмиста понеже езиците от ниско равнище (машинно зависими) са далеч по-сложни
356. 2) {16} Те са **системно независими**, докато например **един ЕПНР** може да **работи** само с определен Ц.П.

356. 3) {17} Създадени са с мисъл, че **на всяка компютърна програма**, написана на такъв език **ще може да се изпълни** без проблеми **на** всеки **компютър**, **който има транслатор на своя език**
356. 4) {17} ЕПВР – език за програмиране от високо равнище – за по-бърза обработка на данни – за най-бърза и най-изгодно **стратегия за решаване на данни**
356. 5) {17} Основната идея, с която са създадени ЕПВР е за разпознаване, структуриране и т. н. типовете данни в ОП и в целия компютър
356. 6) {20} За създаване на различни и богати по своята структура и значение програми
356. 7) {20} ЕПВР са с по-голяма гъвкавост и въпреки че отнемат повече време са предпочитани за използване в професионалните сфери.
356. 8) {26} Улеснение на пишещия програмата.
356. 9) {26} За да се опрости и ускори програмирането без проблеми и различия на всеки компютър.
356. 10) {26} цифрите променят стойно-та формиране стойноста на алгоритмичното число, зависимост от позицията която заемат в нов запис
356. 11) {26} Използването на алгоритми **за решаването на дадена програма**

357. 426. Реализирана ли е на практика идеята една и съща програма да се изпълнява на различни компютри? Защо?

357. 1) {14} Нереализирана на практика същото, всеки компютър има различен централен процесор.
357. 2) {17} Да, с езиците за програмиране от високо равнище
357. 3) {22} Да, реализирана е, **за да улеснява потребителя**.
357. 4) {26} Да

358. 427. Посочете основните причини за нереалност на идеята за изпълнение на една програма, написана на ЕПВР, на много компютри.

358. 1) {10} Различни ЦП
358. 2) {14} Различните машинни езици правят така, че една програма не може да се изпълни от всеки централен процесор. (тъй като всеки централен процесор разпознава само своя машинен език); **за разчитането на програма са нужни транслатори**.
358. 3) {14} Основната причина за нереалност на идеята за изп на една програма на много компютри, е че различните процесори работят с различен машинен код или така наречената несъвместимост.
358. 4) {17} Неразбираеми за машите, но от човека език, спецификата на изпълнител
358. 5) {17} Тъй като централния процесор на дадения компютър може да не чете дадената програма **[Като че ли има ЦП, който може да чете програма, написана на ЕПВР!?!]**
358. 6) {18} Различните ОС и **операции от ПУ** налагат в програмата често да се правят промени.
358. 7) {20} Разликата в ЦП на всеки компютър (разлика в машинните езици)

359. 430. Каква е същността на „условната“ компилация?

359. 1) {15} Въвежда данните без много излишна информация.
359. 2) {17} „Условната“ компилация компилира главната програма **[А що ли прави „безусловната“ и как ли се компилира второстепенна програма?!?]**
359. 3) {17} Условните компилаци са тези които имат в себе си някакво условие, за разклонение.
359. 4) {20} Управлението на условната компилация се реализира, чрез подбор на стойностите на константите за условната компилация
359. 5) {20} Същността на условната компилация е да видим къде са ни грешките в дадената програма

360. 431. Кой и кога определя коя част на текста съставя превежданата програма при наличие на директиви за условна компилация.

360. 1) {15} Езиците осигуряват възможността чрез специални конструкции

360. 2) {23} Езиците, осигуряват полезната възможност чрез специални конструкции посочващи, че при определени обстоятелства, определена част от текста може да се третира като коментар
360. 3) {23, 26} Езиците, осигуряват полезната възможност, чрез специални конструкции, посочващи, че при определени обстоятелства, определена част от текста на програмата трябва да се третира като коментар
360. 4) {26} Езиците (и компилаторите) осигуряват полезната възможност, чрез специални конструкции (за условна компилация), посочващи, че при определени обстоятелства, определена част от текста на програмата трябва да се третира като коментар. При липса на съответните обстоятелства такива части участват в текста на програмата (и се превеждат)
360. 5) {26} Езиците, осигуряват полезната възможност чрез специални конструкции посочващи, че при определени обстоятелства, определена част от текста на програмата трябва да се третира като коментар. При рипса на съответните обстоятелства такива части естествено участват в текста на програмата.
360. 6) {26} Езиците, осигуряват възможност чрез специални конструкции, посочващи, че при определени обстоятелства определени част от текста на програмата, трябва да се третира като коментар. При липса на съответните обстоятелства такива части участват в текста на програмата (и се превеждат
360. 7) {27} В зависимост дали е интерпретатор или не, когато бъде повикан превода, се извежда от превеждащата програма

361. 433. Какво е необходимо за да се реализира условна компилация?

361. 1) {15, 15} Апаратът за условна компилация, трябва да предлага директиви, които външно приличат на този условен оператор **[Кой този?]**

362. 436. Какви са особеностите на условията в директивите за условна компилация?

362. 1) {15} Могат да не бъдат изпълнени нито веднъж.
362. 2) {16} Знак Формиране на условия проверявани в директивите за условна компилация

363. 437. В кой вид езици – тези от ниско или тези от високо равнище, по-често присъства апарат за условна компилация? Защо?

363. 1) {15} Апаратът за условна транслация присъства по-често в ЕПВР.

364. 438. Как програмистът управлява условната компилация?

364. 1) {10} със команди
364. 2) {16} Чрез ЕП

365. 439. Как се получават различните варианти на машинната програма при наличие на апарат за условна компилация?

365. 1) {10} Само при определени условия които са правилни условия компилатор може да се задейства

НЕПЪЛНИ И ЧАСТИЧНО СМЕШНИ ОТГОВОРИ

366. 004. Какви могат да бъдат НПБС и по какво се различават те?

366. 1) {14} Адитивни: при които е разрешено събирането; Мултипликативни: при които е разрешено умножението;

367. 013. Еднозначен ли е записът на произволно цяло число в ПБС с дадена основа? Ако да – кога, ако не – защо?

367. 1) {12} Да, тъй като позицията на цифрата има значение
367. 2) {20} Да, защото множителят с който се променя стойността на цифра преместена на съседна позиция е един и същ

367. 3) {25} Възможене, като използваме основната теорема на ПБС.

368. 014. Еднозначен ли е записът на произволно цяло число в НПБС? Защо?

368. 1) Не, защото е невъзможно да се запише произволно число, поради липса на еднозначност.
368. 2) {20} Не, защото липсва еднозначност **[Дървените неща се наричат дървени, защото са направени от дърво!?!]** при записа на едно цяло число в НПБС
368. 3) {26, 26, 26, 26} Да, защото при НПБС липсва единственост.
368. 4) {26} Не, всеки запис представлява различно естествено число.
368. 5) {26} Да

369. 016. Колко ПБС могат да съществуват и по какво се различават те?

369. 1) {14} Могат да съществуват **много ПБС** в зависимост от това колко цифри (знаци) ще създадем. В зависимост от този брой всеки знак ще променя своята стойност при преместване с една позиция назад.
369. 2) {15} ПБС могат да са: **адитивна и мултипликативна**, различават се по своите основи.
369. 3) {17} Могат да съществуват n на брой ПБС Те се различават по своята основа. Пр. ПБС с основа 2, с основа 51 и т. н.

370. 020. Коя операция – умножение или деление, ще използва човек при преобразуване на десетичен запис на естествено число в двоичен? Защо?

370. 1) {14} Деление, защото човек по-лесно извършва преобразуването от десетичен в двоичен запис.
370. 2) {14} деление на 2. От остатъка се получава двоичния запис
370. 3) {17} Деление, защото **човек умее добре да дели с десетични записи**
370. 4) {17} Деление, защото това е по-лесния начин за преобразуване на дадено естествено число от десетична в двуична бройна система
370. 5) {20} Операция делене. Защото от десетично число преминаването в друга бройна система става по-лесно
370. 6) {20} Деление. За да използва остатъка за да запише 2-ичното число.
370. 7) {20} Деление. Защото методът е като **разделим десетичното число на 2 многократно** и записваме остатъците като ново число в **двуичен** запис.
370. 8) {20} Човек би използвал операцията делене, за да може чрез остатъците ат делението да сформира двоичния запис на числото
370. 9) {21} Ще използва деление. Такъв е алгоритмът за преобразуване.
370. 10) {21} Човек би използвал операцията деление, за да може чрез остатъците от делението да сформира двоичния запис на естественото число.

371. 021. Коя операция – умножение или деление, ще използва компютър при преобразуване на десетичен запис на естествено число в двоичен? Защо?

371. 1) {20} Компютъра ще използва операция умножение. Умножението е по ½.
371. 2) {24} За преобразуване на десетично число в двоично число компютъра ще използва операция деление

372. 026. Има ли някакви опасности при преобразуване на записа на правилна дроб от ПБС с една основа в ПБС с друга основа? Ако да – какви са те и как се решават, ако не – защо?

372. 1) {5–7} Да – M е единствено стига в него да няма период 0 или p–1
372. 2) {14} Да има, защото дробта на другата ПБС може да съдържа 0 или 1. **[И какво от това?]**
372. 3) {14} Има опасности при преобразуване на записа на правилна дроб от ПБС с една основа в ПБС с друга основа, понеже хората могат да правят грешки. Проблема се решава с повече внимание и знания.
372. 4) {14} Да има, ако дробта в новата ПБС може да има период 0 или ≠ от 1.

372. 5) {17, 17, 17, 20, 21, 21, 23, 23, 23, 26} Да има, защото дробта в новата ПБС може да има период 0 или p–1
372. 6) {17} Главната опасност при преобразуването на правилна дроб от една ПБС в друга е допускането на човешка грешка, но и също така неточно съответствие в дадените съотношения. За по-точни резултати е най-добре да се използват десетични дроби и преминаване през ПБС с основа 10 за улесняване на човешката дейност.
372. 7) {20} Да има! Получената дроб може да има период 0 или –1
372. 8) {26} Чрез умножение с едната основа и деление с другата.
372. 9) {26} Да има, защото дробта в новата ПБС може да има период 0 или n–1
372. 10) {27} Да има – при пресмятане; решават се чрез повече съсредоточение и евентуално с коригиране на вече открита такава

373. 030. Може ли да се опростят алгоритмите за намиране на записа на число в ПБС с основа q, когато знаем неговия запис в ПБС с основа p и q е точна степен на p ($q = p^n$) или обратно? Ако да – как, ако не – защо?

373. 1) {14} Може чрез предварително изготвяне на подобни **[На какво са подобни?]** преобразования.
373. 2) {14} Може, използвайки основната теорема!
373. 3) {14} ПБС най-често се използва в 10чната система. Събирането и умножението на числа от ПБС се изчислява точно, затова според мен е възможно опростяването на запис
373. 4) {14, 17, 17, 21} Да, трябва само последователно използване на таблица за подобни **[На какво са подобни?]** замени
373. 5) {17} Да, като използваме за основа r.
373. 6) {18} Да! $N = \frac{q}{p}$
373. 7) {23} Да като използваме изготвена предварително таблица за такива **[Какви такива?]** замествания
373. 8) {23, 26} Да,трябва само предварително изготвяне на таблица на подобни **[Подобни на какво?]** замени
373. 9) {26} Два вида пряка (явна) и косвена (неявна)

374. 031. Може ли да се опростят алгоритмите за намиране на записа на число в ПБС с основа q, когато знаем неговия запис в ПБС с основа p и p и q са точни степени на трето число r ($p = r^s$, $q = r^t$)? Ако да – как, ако не – защо?

374. 1) {12} Да трябва само предварително изготвяне на таблица на подобни замени. **[На какво да са подобни замените?!?]**
374. 2) {17} Да трябва само предварително изготвяне на таблица. **[Каква е тази таблица и как с изготвянето ѝ се опростяват алгоритмите?!?]**
374. 3) {20} Да трябва само предварително изготвяне на таблица на подобни знаци. **[На какво да са подобни знаците и кои са те?!?]**

375. 033. Що е „двузначна логика“?

375. 1) {12} Логика, в която въпросът за вярност има два възможни отговора – Да или Не.
375. 2) {15} Логика, в която този въпрос **[Кой е „този“?]** за верността има само два възможни отговора
375. 3) {26} Логика която има две значения.

376. 064. Посочете понятия, най-близки до интуитивната представа за алгоритъм.

376. 1) {до 3} Алгоритъм е средство да се възложи определена дейност на изпълнява елементарни действия и да ги изпълнява в посочения ред.
376. 2) {14, 23} Алгоритъмът представя сложно действие чрез редица от достатъчно прости действия, които изпълняващият ги може да извърши в последователни стъпки и без допълнителни обяснения

376. 3) {14} сложно действие образувано от множество по прости действшия при които от една начална информация се получава друга изходна информация.

377. 068. Какво извършва изпълнителят на един алгоритъм?

377. 1) {13} Допълнителните лица, които са свързани с изработването на алгоритми са съставител, изпълнител и потребител. Изпълнителят изпълнява програмата и проверява за евентуални грешки.

377. 2) {14} изпълнител: изпълнява предписаните елементарни действия при конкретни начини за условия

378. 073. Дайте пример за име на популярен алгоритъм и задачата, която се решава чрез него.

378. 1) {12} Алгоритъм на Евклид за **намиране на общ знаменател**

378. 2) {14} На пример метода за намиране на най-малко общо кратно. Пр.: НОК на знаменателите 6 и 9 е 18 **[А пък аз глупака си мислех, че НОК на 6 и 9 е 3! А и като изпълних бързичко алгоритъма, чието име така и не е написано, пак получих (за кой ли път?) още на третата стъпка 3!]**

379. 076. Каква е ползата от подалгоритмите?

379. 1) {17} Чрез тях се намалява сложността на решаваната задача и да се освободим от ограничения парк от елементарни действия на изпълнителите.

379. 2) {28} За да обосновават алгоритмите.

380. 080. Какви са параметрите на един алгоритъм (поне 5)?

380. 1) {23} 1. множество на невъзможните данни
2. множество на невъзможните изходни данни
3. множество на междинните резултати
4. правило за започване
5. правила за непосредствена обработка

381. 093. Кой е първият програмист и защо?

381. 1) {14} Августа Ада е първият програмист, защото решава първата програма в двоичен код.

381. 2) {17} Аугуста Ада, създала първия език за програмиране, наречен Ада.

381. 3) {20} Ада Лъвлейс, защото тя създава първият език за програмиране, който е наречен на нейно име – АДА

382. 107. Към кое поколение компютри принадлежат съвременните компютри и каква е тяхната елементна база?

382. 1) {14} Съвременните компютри са от четвърто поколение. Съставени са от **платки** с големини: средна, голяма и много голяма.

382. 2) {21} Съвременните компютри принадлежат към четвърто поколение процесори от високо равнище. **Елементарната** им база е микропроцесор.

382. 3) {27} Съвременните компютри принадлежат към трето поколение, **тяхната елементна база е двоична**.

383. 108. Какво е предназначението на ОП?

383. 1) {до 3} Предназначението на ОП служи за съхранение на програми, ОП съхранява или **обработва** дадена информация.

383. 2) {14} Тя запаметява, съхранява стойностите на операциите в клетки в 2 състояния с уникални адреси за достъп

383. 3) {14} Тя е предназначена за съхранение на данните, необходими за изпълнение на даден алгоритъм.

384. 111. Еднакви ли са всички клетки на ОП? Ако не – защо, ако да – как се различават?

384. 1) {17} Да. Различават се по начина на записването им

385. 118. Посочете предимства и недостатъци на електрическите памети.

385. 1) {5–7} – предимства – съхраняване на по голям обем данни.
– недостатъци – енергозависимиса

386. 121. Какво е предназначението на ЦП?

386. 1) {17} **Да избере** и изпълни програма
386. 2) {23} за изпълнение на машинните програми.
386. 3) {24, 25, 26} ЦП изпълнява машинните програми

387. 124. Какви са функциите на Аритметико-логическото устройство на ЦП?

387. 1) {до 3} функциите му са логическо пресмятане на задачи.

388. 144. По какъв начин може да бъде записан един алгоритъм?

388. 1) {14} Един алгоритъм може да се запише на естествен език, чрез блок-схеми, на машинен език.

389. 157. Възможно ли е един алгоритмичен език да отчита особеностите на алгоритмите, за чието описание е предназначен? Ако не – защо, ако да – как?

389. 1) {15} Да, възможно е; отчита ги като справки.

390. 166. Необходимо ли е преводът от ЕП до МЕ да бъде извършен от човек? Защо?

390. 1) {12} Не е необходимо, дори не е желателно, тъй като човек би изгубил огромно време, а това би попречило компютрите да навлязат така в бита, както в съвремието. ЕП е линеен **[И какво от това?]**, с ясни правила и разбираем, поради което е най-подходящо преводът да се извършва от транслатор
390. 2) {17} не защото човекът който извършва превода е възможно да допусне грешки
390. 3) {26} Не, преводачът може да бъде и програма, защото хората разбират свиете човешки езици, езика на математиката и т. н., а ЕП са някъде между средата на МЕ и човешките езици.
390. 4) {26} Не е необходимо, не е и желателно поради факта че машинните езици са трудни за хората.

391. 167. Възможно ли е преводът от ЕП до МЕ да бъде извършен от човек? Кога?

391. 1) {12} Възможно е, но би отнело време. **[Във въпроса не се споменава за времето на превода, а само се пита възможен ли е и кога е възможен?]** Превода вече е заложен в ЕП, от програмистите създали даден ЕП **[Авторът обогатява световната наука с откритието, че в един език може да бъде заложен и неговият превод и че ЕП се създават само от програмисти!]**
391. 2) {21} Не неможе, защото машинния език е неразбираем за човека.
391. 3) {23} Не, защото ЦП разбира само своя собствен МЕ
391. 4) {23} Не е възможно преводът от ЕП до МЕ да бъде извършен от човек, защото процесора познава своя собствен език.
391. 5) {24} Не, защото ЦП само свое собствено МЕ.

392. 168. Полезно ли е преводът от ЕП до МЕ да бъде извършван от хора? Защо?

392. 1) {5–7} Не – допускат се грешки.
392. 2) {12} Не, не е полезно преводачът може да бъде и програма.
392. 3) {14} Не, не е полезно, защото човек може да допусне грешки.
392. 4) {14} Не, защото може да бъде допусната грешка
392. 5) {17} Не, защото човекът може да допусне правописни грешки

393. 190. Какво е следствието от факта, че алгоритмичните езици са създадени за записване на алгоритми?

393. 1) {15} Алгоритмичните езици имат адекватни конструкции, защото описват алгоритми, използва се двоичен загадъчен код **[И при ЕПВР ли?]**, няма двусмислие, карат компютрите да вършат полезна работа **[Езиците ли?]**

394. 191. Как се наричат програмите, които превеждат от ЕП до МЕ?

394. 1) {14} Интерпретатори и езикови процесори

395. 194. Какво е характерно за компилативните транслатори (компилаторите)?

395. 1) {15, 15} Еднократен превод; Гарантирано е, че в програмата няма синтактични грешки; **изпълнението е много бързо**.

395. 2) {15} Компилаторите се използват в служебните среди, те служат за откриването на грешки в една програма.

396. 202. Какво представляват оптимизиращите компилатори?

396. 1) {до 3} Оптимизиращите компилатори дават възможност по време на превод на програмата да се извършват промени свързани с грешки така и по-подходящ начин за превод. При превод оптимизиращия компилатор изисква повече време и малко бави машината.

396. 2) {до 3} Езикови процесори при които се прочита всичко и след това се появява нов екземпляр, но вече на другия език се наричат компилатори. При промишлена обстановка си струва еднократно да жертваме време за настройка (оптимизиране) на машината, програма за да печелим време при всяко нейно многократно изпълнение – оптимизиращ компилатор.

396. 3) {5–7} Оптимизиращите компилатори се използват в промишлената сфера, където се отделя повече време за оптимизиране на машинната програма и по-малко време за проверка.

396. 4) {14} Оптимизиращите компилатори дават възможност на програмиращия да осъвършенства написаното от него, като му дава съвети **[По законите на българския език от написаното се разбира, че се дават съвети на написаното!?!]**

396. 5) {14} Оптимизиращите компилатори служат за **улеснение** и бързина **на програмата**

396. 6) {17} Компилатори който проверяват за грешки даден програмен код.

396. 7) {17} **Оптимизация** на машинната програма с цел да се печели време.

396. 8) {18} **Оптимизация** на машинната програма с цел да се печели време при всяко нейно многократно изпълнение

396. 9) {20} Оптимизиращите компилатори използват предимствата на другите два компилатора

396. 10) {20} Оптимизиращите компилатори служат за обновяване и развитие на ЕП.

396. 11) {20} Програма което превежда програмен код в машинен код. Изменят се с времето.

396. 12) {26} Компилатори които са оптимизирани –за даден вид дейност – математическа, финансова и др.

397. 205. Посочете елементите, от които се изграждат човешките езици в техния йерархичен ред.

397. 1) {21} Елементите, от които се изграждат човешките езици са : буква → **сричка** → дума → изречение → текст

397. 2) {26} буква – дума – изречение – текст **[Непълно изброяване]**?

398. 209. Посочете особености на параграфите като елемент на човешките езици (поне 2).

398. 1) {20} ① За определяне на параграфите няма формален признак
② Разделянето на параграфите става чрез лист хартия

399. 210. Посочете особености на секциите като елемент на човешките езици.

399. 1) {23} 1) Съществуват само при големите текстове
2) Параграфите изискват специален знак

400. 218. Има ли съответствие между елементите на човешките езици и тези на ЕП? Ако не – защо, ако да – какво е то?

400. 1) {15} Да защото **има азбука** (букви) **и знаци** (+, –, * …)
400. 2) {22} Да има съответствие, **защото ЕП е измислен от човек**.
400. 3) {26} Има. Езикът може да отчита особеностите на описаните с него алгоритми и е линеен (като човешките езици)

401. 231. За какво служат идентификаторите в ЕП?

401. 1) {до 3} Поради наличие на много символи в езика за програмиране не може всеки елемент да бъде със свой собствен а и на клавиатурата няма достатъчно символи затова се използват идентификатори – поредица от буква и цифра от които първият символ е буква.
401. 2) {до 3} Идентификаторите служат за идентифициране на определен тип данни.
401. 3) {5–7} За разозначаване на програмите.
401. 4) {9} служат за различаване на имена **[А не са ли самите идентификатори имена?]**
401. 5) {14} Те дефинират име и тип на величините.
401. 6) {14} идентифицира даден тип данни, за да може да бъде изпълнен в компютъра.
401. 7) {14} идентификаторите поредица от знаци първия от който е буква, останалите остава цифри или букви. Що е БС – съвкупност от знакове и правила, чрез, които се представят цифрите
401. 8) {17} Дефиниция и декларация
401. 9) {17} Идентификаторите – това са имената на променливите
401. 10) {20} идентификаторите в ЕП служат **за разпознаване на знаци и символи в един ЕП**
401. 11) {20} Идентификаторите служат за
401. 12) {20} Идентификаторите в езика за програмиране служат за да не се получава двусмисленица, а да има ясни действия за изпълнение на програмата
401. 13) {23} Идентификаторите служат за ~~показване на функциите на оператора~~, поясняване на самата функция.

402. 233. Какво представляват „служебните (ключовите) думи“ на един ЕП?

402. 1) {5–7} забранени идентификатори.
402. 2) {36} Оградени в програмата

403. 236А. Кой определя имената на величините, използвани в една програма?

403. 1) {17} Този, който пише програмата **и езикът за програмиране**
403. 2) {20} Зависят от програмата, чрез която са въведени

404. 239. Какви видове константи предоставят ЕП и как се определя тяхната единствена текуща стойност?

404. 1) {12} Литерални и именувани константи. Стойностите на литералните се определят от програмистта
404. 2) {17} Литерални и именувани константи. Техните допустими стойности са 1 елемен (константен.)
404. 3) {17} Декларирането им се извършва в началото на програмата, задава се стойността им, тя не се променя и може да бъде използвана„на готово.“
404. 4) {18} Тяхната текуща стойност е една и съща с допустимите стойности и се определя от автора на езика
404. 5) {20, 23} Множеството от допустими стойности на константите се състои от един единствен елемент, който винаги е и тяхна текуща стойност

404. 6) {23} ЕП са множество от допустимите стойности на константите, който се състои от един единствен елемент, който винаги е и тяхна текуща стойност.
404. 7) {26} Един единствен елемент изгражда множеството от допустими стойности на константите. Той е винаги тяхна текуща стойност.
404. 8) {26} Множества от допустими стойности на константа се състои от един единствен елемент.

405. 241. Кой определя имената на литералните константи в един ЕП и как става това?

405. 1) {12} Определят се от правилата на ЕП (езика за програмиране)
405. 2) {14} определят се в зависимост **от правилата на програмата**

406. 245. Длъжен ли е всеки ЕП да осигурява именовани константи? Защо?

406. 1) {20} Не. Трябва да се използват само двете литерални константи – нула и единица.
406. 2) {21} Не. Имена им се дават от пишещият програмата за улеснение на дейността. **[Брилянтно обяснение!]**
406. 3) {26} Не

407. 250. Какво е характерно за външните променливи?

407. 1) {23} – не се определя от съставената програма **[Какво не се определя?]**
– техните имена са точно определени **[Къде?]** и не се използват за друго
– не можем да променяме техните стойности

408. 254. Какво представляват „съвместимите“ типове от данни и защо съществуват“?

408. 1) {5–7} МДС на единия тип е подмножество на другия а операциите релациите са еднакви.

409. 256. Какви са основните типове данни в повечето ЕП (поне 4)?

409. 1) {23} **Реален**, Цял, Булев, символен низ, **Текстов низ**

410. 255. Какви проблеми създават литералните константи на съвместимите типове от данни и как се решават тези проблеми?

410. 1) {12} Получава се двусмислие и се използват само за общия тип.

411. 260. Може ли в цифров компютър да се представят реални числа? Защо?

411. 1) {20} Не – защото ги смята приближено

412. 265. Задължително ли е всеки ЕПВР да предоставя знаков тип данни? Защо?

412. 1) {12} Знаков тип – за C – char. Този тип се използва за означаване на знаци (букви) в програмите. Не е задължително всеки ЕПВР да предоставя знаков тип
412. 2) {17} Не, не е задължително. **Той може да представя и текстов тип**
412. 3) {17} Не. Защото може да се замени с другия тип данни. **[Кой е този друг тип за да бъде члениван, т. е. определен в отговора?!?]**
412. 4) {21} Не е задължително всеки ЕПВР да предоставя знаков тип данни, защото може да се използва и знаков тип данни на друг ЕПВР
412. 5) {21} Не е задължително, защото единият тип може да бъде заменен от другия **[Върви, че гадай кой е единия тип и кой – другия!]**
412. 6) {25} Не, защото е възможно да съществуват такива ЕП. Те се наричат безтипови (APL).

413. 268. Какви възможности за свързване на името на една променлива с множеството от допустими стойности на означената с него величина, представено чрез понятието „тип данни“, осигуряват ЕПВР?

413. 1) {13} Свързване на дадена величина с конкретен тип данни става чрез деклариране Например: int i;

414. 269В. Посочете поне два недостатъка на явното деклариране на променливите в сравнение с неявното деклариране.

414. 1) {23} Повече и **по-трудно** писане.

415. 271. Посочете механизми за различаване на величините от съвкупност едноименни величини.

415. 1) {12} Механизмите за различаване на величините от съвкупност едноименни величини са чрез наредени n-торки цели числа и чрез използването на структури

416. 272. Какво трябва да се посочи при определяне на един масив?

416. 1) {23} 1. **размерите на n**;
2. граничните двойки min и max
3. еднаквия тип на всички елементи.

417. 276. Защо са полезни структурите от данни?

417. 1) {12} Защото могат да се влагат една в друга

417. 2) {17} Защото чрез тях можем да запишем различни типове от дани в структура и да ги използваме многократно

418. 283. Има ли смисъл да използваме „изброими типове“, след като можем да използваме целите числа? Защо?

418. 1) {до 3} Да, защото могат да заемат целите числа.

419. 285. Имат ли константите „тип данни“? Ако да – как се определя той, ако не – защо?

419. 1) {23} Да, константите имат тип данни Те се определят в зависимост от множеството на допустимите стойности. Понятието се използва най-често за определяне на дадена величина

419. 2) {28} Памет, защото са постоянни величини.

420. 285. Имат ли константите „тип данни“? Ако да – как се определя той, ако не – защо?

420. 1) {14} Константите имат тип данни. Той се определя от ~~човека~~ въвел константите и той не се променя.

420. 2) {21} Да имат, типът се определя от това с каква точност да бъде стойността на константата и дали ще бъде стойност или символ

420. 3) {23} Да имат **спорет** това дали са числови или текстови. Той се определя от съставителя на програмата.

421. 296. Какво представлява понятието „израз“ в ЕПВР?

421. 1) {до 3} Израза е съвкупност от променливи свързани със знаци за опр. действия.

421. 2) {12} Изразите служат за изчисляване, величината която стои в лявата част получава като текуща стойност; стойността получена пресмятането на дясната част. **[Имат ли изразите лява и дясна част?]**

421. 3) {14} понятието израз в ЕПВР представлява **написаното** от програмиста **на един ред**. Например една функция се получава при съвкупност от изрази между скобите на ф-та в C++.

422. 313. Какви видове оператори има в ЕПВР?

422. 1) {20} Операторите биват структурни, логически, управляващи.

423. 334. Възможно ли е в един ЕПВР да липсва „съставен оператор“?Ако не – защо, ако да – кога?

423. 1) {17} Да, защото програмиста сам определя дали да използва съставен оператор или не.

423. 2) {23} да възможно е Защото в много ЕПВР съществува синтактично ограничение като част от структурния оператор.
423. 3) {24} Да възможно е Защото в много ЕПВР съществува синтактично ограничение като част от структурен оператор да се задава само един единствен оператор.
423. 4) {26} Да възможно е, защото в много ЕПВР синтактично ограничение като част от структурен оператор да се задава само един единствен друг оператор.
423. 5) {26} Да възможно е. В много ЕПВР съществува синтактично определение като част от структурен оператор, да задава само един единствен – друг оператор.

424. 334. Възможно ли е в един ЕПВР да липсва „съставен оператор“?Ако не – защо, ако да – кога?

424. 1) {26} Да възможно е, защото в много ЕПВР съществува синтактично.

425. 336. Може ли в един ЕПВР да липсват оператори за разклонение? Защо?

425. 1) {12} Не може да липсват. Защото те дават възможност за избор на вариант.
425. 2) {20} Не! защото такъв език няма да предоставя никаква **фукуионалност**.

426. 345. Какво представлява понятието „тяло“ на цикъл и от какво може да се състои то?

426. 1) {17} Тяло на цикъл е тази част, която се повтаря по време на изпълнение. Може да се състои от условия и оператори.

427. 347. Какви видове оператори за циклично повторение на част от алгоритъма има в ЕПВР?

427. 1) {19} 1) цикъл с предусловие
2) цикъл със следусловие
427. 2) {19, 19} цикъл с предусловие и цикъл със следусловие

428. 355. Какво е характерно за оператора за цикъл с преброяване?

428. 1) {до 3} Изпълнява се точно определен брой пъти.

429. 379. Какво представляват блоковете?

429. 1) {14} Блоковете представляват програмни части.

430. 383. Какво представляват фактическите параметри на процедурите при ЕПВР?

430. 1) {26} Фактическите параметри – величини, използване при конкретно задаване от програмиста

431. 395. Какво представляват статичните променливи?

431. 1) {12} При статичните променливи заделянето на памет за тях става при започване на изпълнението **[На какво?]** и ст-та, която им се запазва до края

432. 399. Какво представляват рекурсивните процедури?

432. 1) {14} **Рекурсията е функция**, която извиква сама себе си

433. 406. Какви проблеми създават големите текстове?

433. 1) {12} Трудно и бавно четене.

434. 426. Реализирана ли е на практика идеята една и съща програма да се изпълнява на различни компютри? Защо?

434. 1) {17} Не може да се реализира на практика. Различните ОС и различните операции от Пу налагат в създадената програма често да се правят промени
434. 2) {20} Не, всеки компютър има свой уникален

СТАТИСТИЧЕСКИ ДАННИ

Общ брой на студентите, участващи в тази рубрика: **565**.

Нови участници: 2.

Най-добре представил се нов участник: **12 т**. **(25,53% от постижението на водача до момента).**

Десетте върхови (**top 10**) постижения принадлежат на:

1. **47 точки: Венелин Милков Бабанов=**, ф№1101561108, (23+1) [от 1];
2. **43 точки: Димитър Благомиров Захариев=**, ф№1101561092, (21+1) [от 2];
3. **42 точки: Ина Станкова Димитрова=**, ф№20063005, (21+0) [от 3 м.];
4. **40 точки: Сейди Сали Къртъл=**, ф№20043005, (20+0) [от 4 място];
5. **40 точки: Благой Атанасов Панайотов=**, ф№1201561047, (19+2) [от 5];
6. **37 точки: Иван Добринов Иванов=**, ф№1101561045, (18+1) [от 6 м.];
7. **34 точки: Александър Иванов Александров=**, ф№1001561076, (17+0) [от 7–10];
   34 точки: Николай Красимиров Тончев=, ф№1101561047, (17+0) [от 7–10];
   34 точки: Владимир Руменов Узунов=, ф№1001561002, (17+0) [от 7–10];
   34 точки: Моника Михайлова Николова=, ф№1201561092, (17+0) [от 7–10];

Водачи в отделните възрастови категории са:

1. **Електроенергетика и електрообзавеждане: (42 т., 3 м.=) Ина Станкова Димитрова**, ф.№20063005, **(40 т., 4 м.=) Сейди Сали Къртъл**, ф.№20043005, **(24 т., 29–31 м.=) Боян Маринов Илиев**, ф.№20043005;
2. **Комуникационна техника и технологии: (26 т., 21–27 м.=) Станка Георгиева Колева**, ф№20064009, **(22 т., 37–43 м.=) Ангел Христов Чекичев**, ф№20064008, **(20 т., 46–53 м.⇓) Стоян Шопов**, ф№20044008;
3. **Компютърни и комуникационни системи: (28 т., 19 м.=) Стилиян Филипов Узински**, ф№20065003, **(22 т., 37–43 м.=) Никола Александров Раев**, ф№20065013, **(18 т., 57–62 м.⇓) Ивайло Невенов Мишев**, ф№20065008;
4. **Бизнес информационни технологии, редовно обучение: (47 т., 1 м.=) Венелин Милков Бабанов**, ф№1101561108, **(43 т., 2 м.=) Димитър Благомиров Захариев**, ф№1101561092, ф№1101561108, **(40 т., 5 м.=) Благой Атанасов Панайотов**, ф№1201561047.
5. **Приложна математика, редовно обучение: [поради единствено участие, постигнат малък брой шедьоври и липса на възможност за рецидив имената не се споменават] (9 т., 201–210 м.⇓)**, **(6 т., 299–374 м.⇓)**, **(5 т., 375–389 м.⇓)**.

Най-добре представил се нов участник (без името му):

(за първите 7 издания на рубриката няма запазени статистически данни)

8 издание, 30 декември 2006 г., 14 точки (7+0), 6–11 място от 59 [3 броя].
9 издание, 26 януари 2007 г., 18 точки (9+0), 9 място от 82.
10 издание, 3 февруари 2007 г., 12 точки (6+0), 23–27 място от 85.
11 издание, 5 септември 2007 г., 6 точки (3+0), 54–66 място от 86.
12 издание, 24 март 2010 г., 17 точки (8+1), 14 място от 113.
13 издание, 29 март 2010 г., няма нови участници.
14 издание, 21 март 2011 г., 26 точки (13+0), 4–5 място от 203;
15 издание, 7 април 2011 г., 26 точки (13+0), 11 място от 206;
16 издание, 11 септември 2011 г., няма нови участници;
17 издание, 26 март 2012 г., 25 точки (12+1), 14 място от 294;
18 издание, 4 април 2012 г., 6 точки (3+0), 176–213 място от 297;

19 издание, 13 декември 2012 г., 9 точки (4+1), 130–133 място от 303;
20 издание, 21 март 2013 г., 24 точки (12+0), 19–20 място от 383;
21 издание, 5 април 2013 г., 10 точки (5+0), 133–166 място от 389;
22 издание, 10 септември 2013 г., 6 точки (3+0), 230–279 място от 390;
23 издание, 27 март 2014 г., 14 точки (7+0), 74–96 място от 464 [2 броя];
24 издание, 3 април 2014 г., 16 точки (3+0), 74–96 място от 465.
25 издание, 16 септември 2014 г., 8 точки (4+0), 187–239 място от 467;
26 издание, 18 март 2015 г., 24 точки (12+0), 24–26 място от 560;
27 издание, 2 април 2015 г., 21 точки (10+1), 44 място от 563;
28 издание, 10 септември 2015 г., 12 точки (6+0), 116-144 място от 565.

Брой на представените шедьоври:

29 броя – 1 (1) участник, с 3 участия, на 1 място в класацията.
22 броя – 1 (1) участник, с 2 участия, на 2 място в класацията.
21 броя – 2 (2) участника, на 3 и 5 място в класацията:
  с 2 участия – 1 (1) участник, на 5 място в класацията;
  с 3 участия – 1 (1) участник, на 3 място в класацията.
20 броя – 1 (1) участник, с 1 участие, на 4 място в класацията.
19 броя – 1 (1) участник, с 2 участия, на 6 място в класацията.
18 броя – 2 (2) участника, с по 2 участия, на 11–12 място в класацията.
17 броя – 5 (5) участника, с по 2 участия, на 7–10 и 15 място в класацията.
16 броя – 3 (3) участника, на 13–14 и на 16 място в класацията.
  с 2 участия – 2 (2) участника, на 13–14 място в класацията;
  с 5 участия – 1 (1) участник, на 16 място в класацията.
15 броя – 2 (2) участника, от 17 до 18 място в класацията.
  с 2 участия – 1 (1) участник, на 18 място в класацията;
  с 3 участия – 1 (1) участник, на 17 място в класацията.
14 броя – 3 (3) участника, с по 2 участия, от 19 до 20 и на 21–27 място.
  с 2 участия – 2 (2) участник, от 19 до 20 място в класацията;
  с 3 участия – 1 (1) участник, на 21–27 място в класацията.
13 броя – 9 (9) участника, на 21–27 и на 32–33 място в класацията:
  с 2 участия – 7 (7) участника, на 21–27 и на 32–33 място в класацията;
  с 3 участия – 2 (2) участника, на 21–27 и на 32–33 място в класацията.
12 броя – 7 (7) участника, на 29–31, 34–36 и на 45 място в класацията:
  с 1 участие – 3 (3) участника, на 29–31, 34–36 и на 45 място в класацията;
  с 2 участия – 2 (3) участник, на 29–31 място в класацията;
  с 3 участия – 2 (2) участника, на 34–36 място в класацията.
11 броя – 10 (10) участника, от 37–43 до 44 и на 54–55 място.
  с 1 участие – 1 (4) участник на 44 място в класацията;
  с 2 участия – 9 (9) участника, на 37–43 и на 54–55 място в класацията;
10 броя – 9 (8) участника, на 46–53 и на 56 място в класацията.
  с 1 участие – 4 (4) участника, на 46–53 място в класацията;
  с 2 участия – 3 (3) участника, на 46–53 и на 56 място в класацията.
  с 3 участия – 1 (0) участника, на 46–53 и на 55 място в класацията.
  с 4 участия – 1 (1) участник, на 46–53 място в класацията.
 9 броя – 9 (10) участника, от 57–62 до 63–64 и на 80 място в класацията:
  с 1 участие – 6 (6) участника, от 57–62 до 63–64 и на 80 място в класацията;
  с 2 участия – 2 (3) участника, на 57–62 място в класацията;
  с 3 участия – 1 (1) участник, на 57–62 място в класацията.
 8 броя – 20 (20) участника, на 65–79, 81–84 и 115 място в класацията:
  с 1 участие – 10 (10) участника, на 65–79, 81–84 и 115 място в класацията;
  с 2 участия – 9 (9) участника, на 65–79 и на 81–84 място в класацията;
  с 4 участия – 1 (1) участник, на 65–79 място в класацията.

7 броя – 32 (32) участника, от 85–108 до 109–114 и на 145–146 място:
  с 1 участие – 21 (21) участника, от 85–108 до 109–114 и на 145–146 място;
  с 2 участия – 10 (10) участника, от 85–108 до 109–114 място в класацията;
  с 3 участия – 1 (1) участник, на 85–108 място в класацията.
6 броя – 41 (40) участника, на 116–144, 147–157 и на 200 място в класацията:
  с 1 участие – 34 (33) участника, на 116–144, 147–157 и на 200 място в класацията;
  с 2 участия – 5 (5) участника, на 116–144 и 147–157 място в класацията;
  с 3 участия – 2 (2) участника, на 116–144 място в класацията.
5 броя – 55 (52) участника, на 158–199, 201–210 и 285–287 място:
  с 1 участие – 43 (43) участника, на 158–199, 201–210 и 285–287 място;
  с 2 участия – 12 (9) участника, на 158–199 и 201–210 място в класацията.
4 броя – 86 (86) участника, на 211–284, 288–298 и 478 място в класацията:
  с 1 участие – 79 (79) участника, на 211–284, 288–298 и 478 място в класацията;
  с 2 участия – 6 (6) участника, на 211–284 място в класацията;
  с 3 участия – 1 (1) участника, на 211–284 място в класацията.
3 броя – 95 (95) участника, от 299–374 до 375–389 и на 474–477 място:
  с 1 участие – 91 (91) участника, от 299–374 до 375–389 и на 474–477 място;
  с 2 участия – 4 (4) участника, от 299–374 до 375–389 място.
2 броя – 101 (100) участника, на 390–473 и 479–495 място в класацията:
  с 1 участие – 99 (98) участника, на 390–473 и 479–495 място в класацията;
  с 2 участия – 2 (2) участник, на 390–473 място в класацията.
1 брой – 70 (70) участника, от 496–561 до 562–565 място в класацията.