

ЛЕКЦИЯ 10 ВЕЛИЧИНИ И ТИПОВЕ ДАННИ

- Понятието величина в ЕПВР
- Видове величини
- Понятието тип данни в ЕПВР
- Основни типове данни в ЕПВР
- Понятие за структура от данни
- Създаване на нови типове данни

ПРОГ_10

1/60

ПОНЯТИЕТО ВЕЛИЧИНА

Определение: Величините са основно средство за изразяване на стойностите, с които се оперира в алгоритмите.

Характеристики на всяка величина са:

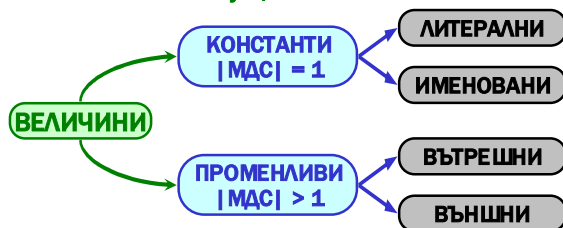
- име (за да ги различаваме);
- множество от допустими стойности (каквито дадена величина може изобщо да получи);
- текуща стойност (във всеки определен момент от времето).

ПРОГ_10

2/60

ВИДОВЕ ВЕЛИЧИНИ

Величините се различават по размера на своето множество от допустими стойности и по начина за определяне на тяхната текуща стойност.



ПРОГ_10

3/60

КОНСТАНТИ

- Тяхното множество от допустими стойности се състои от един единствен елемент, който винаги е и тяхна текуща стойност.
- Имената на литералните константи и тяхната единствена стойност се определят от правилата на езика.
- Имената на именованите константи се избират от програмиста, който чрез възможностите на съответния език посочва и тяхната единствена стойност.

ПРОГ_10

4/60

ЗАБЕЛЕЖКИ

- Обичайни литерални константи: 1, 9, -0.1, 1E-1, 'a', "текст", true и др.
- Числата обикновено се записват в 10-ична ПБС, но често е разрешено да се използват още двоична, осмична и шестнайсетична ПБС.
- Името на именованата константа е идентификатор, който се свързва с литерална или определена по-рано именована константа.

ПРОГ_10

5/60

ПРИМЕР: ИМЕНОВАНИ КОНСТАНТИ

Паскал: в раздела за константи **CONST** чрез записи от следния вид
<име> = <стойност>;

Си: такива константи не са разрешени, но макроапаратът (част от езика) осигурява аналогична възможност.

Вижуъл Бейсик: в отделни редове
[Public | Private] Const <име>
[As <тип>] = <израз>

ПРОГ_10

6/60

ПОЛЗАТА ОТ ИМЕНОВАНИ КОНСТАНТИ

Задачата: Аграрен предприемач ви поръчва да създадете програма за обслужване на неговото стопанство с до пет крави и до пет кокошки.

Анализът: Очевидно е, че максималният брой на кравите и кокошките не може да се променя.

Глупавото решение: Да използвате за своите цели литералните константи 5, където ви потрябват.

Защо е глупаво? Кое 5 какво е крави или кокошки?

Умното решение: Две именовани константи – Макс_Кокошки = 5 и Макс_Крави = 5.

Защо е умно? Утре ще смените само два реда, но ще поискате много ГОЛЯМА благодарност.

ПРОГ_10

7/60

ВЪНШНИ ПРОМЕНЛИВИ

- Тяхната текуща стойност не се определя от създадената програма.
- Техните имена са определени в самия език и не могат да се използват за друго.
- В създадената програма можем да използваме, но не можем да променяме техните текущи стойности.
- Примери:
INPUT в СНОБОЛ;
RND (от random = случаен) в много езици;
DATE и TIME в някои езици.

ПРОГ_10

8/60

ВЪТРЕШНИ ПРОМЕНЛИВИ

- ❶ Тяхното **множество от допустими стойности** се състои от **поне два елемента**.
- ❷ Имената им са **идентификатори**, които се **избират от програмиста**, доколкото му стига **акълът**.
- ❸ За тях **освен** избраното от него **име** програмистът **трябва да посочи и множеството от допустими стойности**.

ПРОГ_10

9/60

ПРОМЕНЛИВИ (прод.)

- ❹ За да бъде **етап** **по-прост** в ЕПВР **съществува** понятието **тип данни**.
- ❺ Тяхната **текуща стойност** е **изцяло под контрола на (определя се от)** създадената **програма**.
- ❻ За такива променливи в езика често се определят **две важни понятия**:
 - ➔ **видимост** на имената (**област на д-е**);
 - ➔ **време за живот** на променливата.

ПРОГ_10

10/60

ПОНЯТИЕ ЗА ТИП ДАННИ

Просто определение: Посочва **множеството от допустими стойности** за дадена величина.

Пълно определение: Освен **МДС** един тип данни **определя** още и:

- ⌘ **операциите вътре** в типа;
- ⌘ **операциите вън** от типа;
- ⌘ **релациите над** такива данни;
- ⌘ **правилата за запис на литералните константи** (= **МДС** за типа).

ПРОГ_10

11/60

ЗАБЕЛЕЖКИ

- ❶ В повечето езици **типовете данни са подобни (дори еднакви)**, защото се определят **от прагматични съображения**:
 - ➔ Какви данни **разпознават компютрите**?
 - ➔ Кой данни са **удобни за алгоритмите**?
- ❷ Всъщност **под тип данни** можем да разбираме и **начина**, по който ще бъдат **интерпретирани поредиците от 0 и 1**, които в компютрите в действителност се явяват **стойности на величините** от даден тип.

ПРОГ_10

12/60

ЗАБЕЛЕЖКИ (прод.)

- ❸ Съществуват и **безтипови езици**, при които понятието **тип данни липсва (APL)**.
- ❹ Много ЦП могат да **обработват по един и същи начин различни** по своята дължина **поредици от битове**.
- ❺ За да отразят това някои ЕП предлагат **съвместими типове данни**, при които **МДС на единия тип е подмножество на МДС на другия**, а **операциите и релациите са еднакви**.

ПРОГ_10

13/60

СЪВМЕСТИМИ ТИПОВЕ

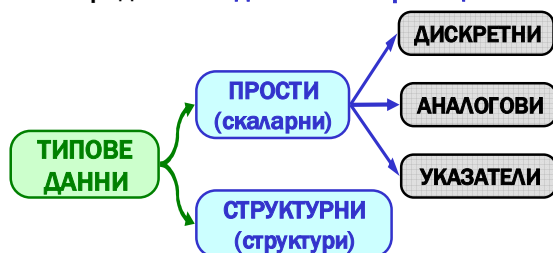
- ❶ Типът с **по-малък брой допустими стойности** обикновено се нарича **подтип** на другия (който често се нарича **надтип**).
- ❷ **Литералните константи на един подтип са законни и за неговия надтип**, което може да породи **двусмислие**.
- ❸ За да се избегне такова **двусмислие** в езиците със съвместими типове **общите литерални константи могат да се пишат по два начина: двусмислено и като стойности за по-общия тип – 1 и 1L**.

ПРОГ_10

14/60

КЛАСИФИКАЦИЯ НА ТИПОВЕТЕ ДАННИ

Никлаус Вирт – съзателят на **Паскал**, предлага **следната класификация**:

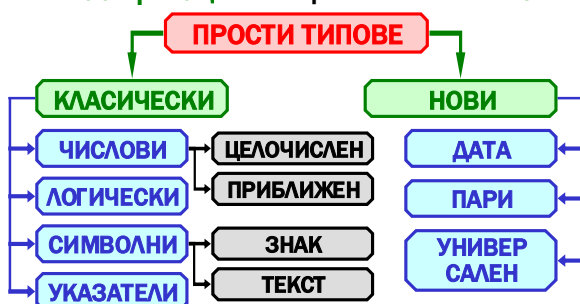


ПРОГ_10

15/60

ДРУГА КЛАСИФИКАЦИЯ

Друга – **компютърно-ориентирана, класификация** на простите типове е:



ПРОГ_10

16/60

ЦЕЛОЧИСЛЕН ТИП

- 1 **Литералните константи** се пишат с **цифри** и **основата** е **десет**: **1**, **+25**, **-37**, ...
- 2 Понякога **основата** може да бъде **две**, **осем** и **шестнайсет**. В тези случаи **преди или след числото** се записва **специален знак** за новата **основа**: **%**, **@**, **\$**; **B**, **O**, **H**.
- 3 **Операциите** са събиране, изваждане и др. и **се извършват точно**! **Литералните константи също** се представят **точно**!
- 4 **Релациите** са на **пълната наредба**.
- 5 **Всяка допустима стойност** има **точно определени** стойности **преди и след** себе си.
- 6 **Точно определени** са още **най-малката и най-голямата** допустими **стойности**.

ПРОГ_10

17/80

ПРИМЕР: ИМЕНА НА ЦЕЛОЧИСЛЕНИ ТИПОВЕ

Паскал: единствен тип **INTEGER**.

Си: множество съвместими типове **char**, **signed char**, **unsigned char**, **short [int]**, **unsigned short [int]**, **int**, **unsigned [int]**, **long** и **unsigned long**.

ВБ: три съвместими типа **Byte**, **Integer** и **Long**.

ПРОГ_10

18/80

ПРИБЛИЖЕН ЧИСЛОВ ТИП

- 1 **Неправилно** се нарича **реален тип**.
- 2 **Литералните константи** се пишат само в десетична БС с **цифри**, **знак за дробна част** (.) и **знак за основа десет** (**e**, **d**, **E**, **D**) по два начина: **-1 1 .2** (= 0,2), **3** (= 3,0 и ≠ 3); и **-5.6e+2** (= -5,6×10² = -560).
- 3 **Операциите** са събиране, изваждане и др. и **се извършват приближено** (т. е. **неточно**)! **Литералните константи също** се представят **приближено**!
- 4 **Релациите** са на **пълната наредба**.
- 5 **Не е ясно** нито **коя стойност е точно преди и след дадена допустима**, нито **кои са най-малката и най-голямата стойности**.

ПРОГ_10

19/80

ЗАБЕЛЕЖКИ

- 1 Този тип е продиктуван от **необходимостта при извършване на математически изчисления едновременно да се представят много големи, а също и много малки по своята абсолютна стойност** числа: **ctg x → ∞ при x → 0**.
- 2 **Важни характеристики** на типа са **броят на верните цифри (точност)** и **диапазонът на представяните числа (обхват на порядъка)**.
- 3 ЦП **разпознават и могат да оперират с числа с плаваща запетая**, представяни като **мантиса и порядък (степен на 2 или 16, а не на 10)**.
- 4 Поради **неточността на изчисленията няма смисъл стойностите на две величини от типа да бъдат сравнявани за равенство**. **По-правилно е да приемем, че a=b когато |a-b| < ε**.

ПРОГ_10

20/80

ПРИМЕР: ИМЕНА НА ПРИБЛИЖЕНИ ТИПОВЕ

Паскал: единствен тип **REAL**.

Си: три съвместими типа **float**, **double** и **long double**.

ВБ: два съвместими типа **Single** и **Double**.

При съвместимите типове в **Си** и **ВБ** **увеличеният брой верни цифри се съпътства с разширяване на диапазона** от представяни числа.

ПРОГ_10

21/80

ЛОГИЧЕСКИ ТИП

- 1 **Необходим** е за **оценки по време на изпълнението** на програмата.
- 2 **Литералните константи** са само **две**: **true** (**ИСТИНА**) и **false** (**ЛЪЖА**).
- 3 **Операциите** са отрицание (**НЕ**), конюнкция (**И**), дизюнкция (**ИЛИ**) и др.
- 4 **Основен генератор** на логически стойности **са сравненията, породени от релациите** на другите типове от данни.
- 5 В повечето езици се приема, че **двете допустими стойности са наредени като истината е пряк наследник на лъжата**, т. е. **false < true**.

ПРОГ_10

22/80

ПРИМЕР: ИМЕ НА ЛОГИЧЕСКИЯ ТИП

Паскал: **BOOLEAN**.

Си: **липсва** такъв тип, но има **логически операции**, при които **числова стойност =0** е **лъжа**, а **всички ≠ 0 числови стойности са истина (1)**.

ВБ: **Boolean**, но се използват и правилата на **Си (True = -1)**.

ПРОГ_10

23/80

ЗНАКОВ ТИП

- 1 **Допустима стойност** на типа **е всеки единичен знак**, представян **чрез своя код – цяло число**, в използваната кодова таблица.
- 2 **Литералните константи** се записват като с **апострофи** се оградят **графичен знак** **'Z'**, **'я'**, **'\$'**, ...; **името на непечатим знак** **'\a'**, **'\r'**, ... и дори **кода на произволен знак** **'\ooo'**, **'\xhh'**.
- 3 Типът **няма операции**, но често се позволява **трансформиране на цяло число в знака с такъв код и знак в цялото число**, което е негов код.
- 4 **Релациите** на типа **унаследяват релациите на целите числа**, явяващи се кодове на знаците.
- 5 **При всеки знаков код** кодовете на **цифрите** (от 0 до 9) са **последователни цели числа**, а кодовете на **буквите нарастват по техния азбучен ред**.

ПРОГ_10

24/80

ПРИМЕР: ИМЕ НА ЗНАКОВИЯ ТИП

Паскал: **CHAR**.

Си: такъв тип липсва – **char** е целочислен тип и това няма защо да се скрива, **но има литерални константи** (те са цели числа!).

ВБ: такъв тип липсва, защото той **може** да бъде заменен от **текст (знаков низ)** с **дължина 1** знак.

ПРОГ_10

25/60

ТЕКСТОВ ТИП

- 1 **Допустима стойност** на типа е **всяка редица от знакове**, поради което е **известен и като тип знаков низ**.
- 2 **Литералните константи** се записват като **редицата от знакове се ограда със знака за инч ("**), неправилно наричан кавичка: **"Зя\п"**.
- 3 Единствената **операция** е **конкатенация (слепване)** – **дописване на втория низ след първия**, но обикновено **езиците предлагат** като функции **редица други** полезни възможности.
- 4 **Релациите** на типа са продиктувани от лексикографската наредба на низове:
 - **с** **с** и **без** отчитане на **рисунька** (**A * a**, **A = a**);
 - **директно по** кодовете на знаците (**ç > z**, **î > z**);
 - **с** **с** **замяна** на букви (**ae = æ**, **ç = c**, **î = i**).

ПРОГ_10

26/60

ПРИМЕР: ИМЕ НА ТЕКСТОВИЯ ТИП

Паскал: такъв тип **липсва** и той се моделира с **масив от CHAR**.

Си: такъв тип **липсва** и той се моделира с **масив от char**. За удобство има подобие на литерални константи.

ВБ: **два** текстови типа **String** и **String * <брой>**.

ПРОГ_10

27/60

ПОНЯТИЕ ЗА УКАЗАТЕЛИ

При изпълнението на една програма от компютър **всяка** нейна **величина** **ще** трябва да **бъде моделирана** в неговата **ОП**.

При това моделиране името на величината **ще** се свърже с **конкретен адрес от ОП**, а **съдържанието** на съответната **клетка в ОП** **ще** **представя текущата стойност** на величината.

Адресите на клетките от ОП **са** **цели числа** и нищо не ни пречи да третираме **записаното в една клетка като адрес на друга клетка** от ОП. Така стигаме до **идеята за величини**, чието **МДС са адресите** на клетки от ОП, в които са моделирани (има) **величини от определен тип**. **Това е** и **понятието указател**, чрез който можем **косвено да използваме друга величина**.

ПРОГ_10

28/60

ЗАБЕЛЕЖКИ

Събирането на указатели е **безсмислено** (какво е **сума на адреси** от ОП?), **но** **изваждането** има **разумна интерпретация** – **брой на клетките** между двата адреса и ако бъде **разделен на размера** на величина от даден тип **ще** се получи **броят** на такива **величини в ОП**.

Добавянето на цяло число към указател също е **смислено**, особено когато числото се **умножава по размера** на величините от даден тип.

Други смислени операции са **получаване на адреса на величина** за да стане той **стойност на указател** и **косвен достъп до стойност на величина** **чийто адрес е в (стойност на) указател**.

ПРОГ_10

29/60

ЗАБЕЛЕЖКИ (прод.)

Единствената **смислена явна стойност** на указател е **адресът**, на който **никога не би могло да бъде моделирана** каквато и да е **величина (празен указател)**.

Неумелото боравене с указатели може да доведе до **големи бели**, поради което **указателите липсват в повечето езици**.

Обратно, **разумното им използване** дава възможност за **елегантно записване на множество интересни алгоритми** (вкл. и **криминални по своята същност**).

ПРОГ_10

30/60

ТИП УКАЗАТЕЛ

- 1 **Допустимите стойности** не се уговарят явно (всъщност **това са адресите от ОП**).
- 2 **Винаги се уговаря към какъв тип** величини **сочи даден тип указател** и **рядко** е разрешен **универсален тип указател** (**void *** в Си).
- 3 **Явна литерална константа** е само **празният указател** (**NI**, **NULL**, **Nothing**), но понякога **неявно** са разрешени **някои константи на целочисления тип** (явяващи се **адреси**).
- 4 **Операциите** (когато има такива) са **изваждане на указатели към еднакъв тип** (**с делене на размера**) и **събиране на указател с цяло число** (**умножено по размера**).
- 5 **Релациите** са за **съвпадение на указатели към еднакъв тип** и с **празен указател** (**=** и **!=**).

ПРОГ_10

31/60

ПРИМЕР: ИМЕ НА ТИП УКАЗАТЕЛ КЪМ

Паскал: явен тип **липсва**, но т. нар. **динамични променливи** го заместват напълно.

Си: **<тип> ***, където **<тип>** е **void** или произволен тип на езика, **включително и указател към**.

ВБ: **<клас от обекти>**, т. нар. **обектови променливи**.

ПРОГ_10

32/60

НОВИ ТИПОВЕ ДАННИ

Появяват се поради нарастващото приложение на компютрите в живота на хората, което изисква създаването на все по-разнообразни програми.

Обикновено присъстват само в новите езици за програмиране (**VB**, **SQL** и др.).

В повечето езици са свързани с особено тълкуване на стойностите на традиционните типове от данни и собствени литерални константи.

ПРОГ_10

33/60

ТИП ДАТА

Стойности на типа са **календарните дати**. Често датата се комбинира с времето от денонощието, макар че в някои езици има отделни типове за **дати** (**Date**), **времена** (**Time**), и комбинация от двете (**DateTimeStamp**).

Във **VB** типът **Date** е комбиниран и **литералните константи** представляват правилно записани дати и времена, заградени със знака **диез (#)**:

#1 януари 2002 10:23:55#. Във **VB** допустими са датите от 1.I.100 до 31.XII.9999 г.

Също във **VB** **стойностите** се представят чрез **приближения тип** като цялата част определя броя на дните спрямо 30.XII.1899 г., а **дробната част** – часа от денонощието (**0** = **полунощ**, **½** = **ладне**).

ПРОГ_10

34/60

ТИП ПАРИ (ВАЛУТА)

Парите са важен елемент в живота на хората. В повечето държави **паричните единици имат и подразделения**, което означава, че **паричните стойности** трябва да бъдат **дробни числа** с **вярно представяне и верни операции**.

Следователно, за представяне на парични стойности **не са подходящи** **обичайните числови типове** (**целочислен** и **приближен**).

Много езици правят **опит за разрешаване** на въпроса. В **Кобол** и **ПЛ-1** за целта се използват двоично-кодирани десетични числа. **VB** има тип **Currency** – **цели числа**, представящи **паричната стойност умножена по 10 000**. Така имаме **15 десетични цифри цяла част** и **4 – дробна**.

ПРОГ_10

35/60

УНИВЕРСАЛЕН ТИП

VB предоставя **интересен универсален тип** данни, наречен **Variant**. Неговото **МДС** е **обединение от МДС** на всички останали типове и още **две специални стойности**: **Липсваща (Empty)** и **Неизвестна (Null)**.

Променливите от този тип могат да получат **произволна текуща стойност**, която **временно** ги причислява към типа, на който принадлежи.

В последните версии на **VB** променливите от универсален тип **могат да получат** като своя текуща стойност **дори структура от данни**.

Неумелото използване на универсалния тип може да пород **много грешки**. Обратно, чрез **разумното му използване** някои интересни **задачи** могат да бъдат решени **твърде леко и елегантно**.

ПРОГ_10

36/60

ОПРЕДЕЛЯНЕ НА ТИПА

За да използваме нова **променлива** в една програма **трябва да определим** нейните **име и тип**.

Измислянето на име е въпрос на **творческо виждане**. **Определянето на типа** може да се осъществи **по два начина**:

① **неявно** чрез **името** на променливата:

♣ от **неговата първа буква** (**Фортран**: **I, J, K, L, M, N** – **целочислен**, друга – **приближен**);

♣ от **специален знак след него** (**VB**: **%** – **Integer**, **&** – **Long**, **\$** – **String**, **@** – **Currency** и др.).

② **явно** чрез **специална декларация** за създаване на нова променлива **преди използването** на тази променлива.

ПРОГ_10

37/60

СРАВНЯВАНЕ НА НАЧИНИТЕ

① **Неявно определяне на типа**:

- ☺ по-малко писане;
- ☺ разрешено е **на всяко място** в текста.
- ☒ **правописните грешки** са **твърде скъпи**;
- ☒ **провокира недисциплинираност**;
- ☒ **липсва яснота** за ролята на променливата.

② **Явно деклариране на типа**:

- ☺ избягват се **глупави правописни грешки**;
- ☺ налага **дисциплина** и **дава яснота**.
- ☒ **повече и по-трудно писане**;
- ☒ **липсата на грешки е измамна**.

ПРОГ_10

38/60

ПОМНЕТЕ ВЗРИВЕНАТА АМЕРИКАНСКА РАКЕТА!

Полетите на космически ракети винаги са били управлявани от компютри.

В 60-те години на XX век в САЩ програмите се пишат на **Фортран** – първият ЕПВР, в който **интервалите се игнорират!**

В програма за управление на първата ракета към Марс е допусната **правописна грешка**: една **запетая е заменена с точка**:

DO 100 I = 1, 10 – оператор за **цикъл**

DO100I = 1, 10 – оператор за **присвояване**

ПРОГ_10

39/60

ПРИМЕР: ОПРЕДЕЛЯНЕ НА ТИП

Паскал: **само явно** в раздела за променливи **VAR** чрез записи от вида

<име> : <тип>;

Си: **само явно** чрез записи от вида

[<вид памет>] <тип> <име> [=<нач. ст-т>];

VB: има **и двата варианта** като **неявното** определяне може да бъде **забранено** чрез **Option Explicit**

[Dim | Public | Private] <име>

[As <тип>] [, <име> [As <тип>] ...]

ПРОГ_10

40/60

ПОНЯТИЕ ЗА СТРУКТУРА

? Може ли две величини да имат **еднакви имена**?

⚡ **Не, разбира се!** Как ще ги различаваме?

? А **полезно ли е** да има **еднакви имена**?

⚡ Отговорът се вижда от следните **задачи**:

- ① Еднаква обработка на **3 числа**?
➤ **3 имена**, например **a, b и c**.
- ② Еднаква обработка на **300 числа**?
➤ **300 имена**, например **a001, a002, ..., a300**.
- ③ Еднаква обработка на **30 000 числа**?
➤ **30 000 имена**? Трудно, но **възможно!**
- ④ Еднаква обработка на **неизвестен брой**?
➤ Сега вече **мисията е невъзможна!**

ПРОГ_10

41/80

СТРУКТУРИ ОТ ДАННИ

Вижда се, че **е полезно** няколко **величини** да имат **еднакво име**, което да използваме когато ги разглеждаме **като единно цяло**, какъвто е и смисълът на понятието **структура от данни**.

Разбира се, ще трябва да уточним **механизъм за идентифициране на всяка отделна величина**, явяваща се **елемент на структурата**:

① чрез **наредена n -орка цели числа**, когато **смисълът** на елементите **е еднакъв** и следователно **те са от един и същи тип**;

② чрез **индивидуални (под) имена**, когато **смисълът** на елементите **е различен**, което **не е гаранция**, че и **типът им е различен**.

ПРОГ_10

42/80

МАСИВИ

Структурите с **идентификация ①** се наричат **масиви** или **променливи с индекси**.

Числото **n** се нарича **размерност на масива**.

Идентифициращото цяло число по **i -тото измерение** се нарича **i -ти индекс**.

Минималната и максималната стойности на **i -тия индекс** се наричат **i -та гранична двойка**.

Масивите **изискват явно деклариране** на:

- ① размерността **n** ;
- ② граничните двойки **$m_i n_i$** и **$max_i, i=1, 2, \dots, n$** ;
- ③ **еднакия тип на всички елементи**.

ПРОГ_10

43/80

ЗАБЕЛЕЖКИ

① Целите числа са със **строго определено подреждане**. В някои езици **за по-голямо удобство** като **индекс** по дадено измерение може да се използва **по-общо – произволен дискретен тип** (**знак, логически** и др.).

② **Параметрите** на един масив трябва да бъдат **обявени до неговото използване**.

③ За **по-бързо деклариране** в много езици **долната граница на индексите е фиксирана** и не се посочва: **0 (Си)** или **1 (Фортран)**.

④ **Масивите** биват **статични (Паскал, Си)** и **по-рядко динамични (ВБ)** – доста по-удобни.

ПРОГ_10

44/80

(ЛОГИЧЕСКИ) ЗАПИСИ

Структурите с **идентификация ②** се наричат (логически) **записи (Паскал)**, само **структури (Си)** или **променливи с полета**.

Записите изискват **явно деклариране на имената и типовете** на своите елементи. **За удобство** това може да се обяви **и отделно**.

Освен **записи, съдържащи винаги всички свои именувани елементи**, в **Паскал** има и **вариантни записи**, в които **част от елементите присъстват само когато стойността на посочен елемент отговаря на определено условие**.

ПРОГ_10

45/80

ЗАЩО СТРУКТУРИТЕ ОТ ДАННИ СЕ НАРИЧАТ ТИПОВЕ ДАННИ

① **Синтаксисът (граматиката) на езика** за деклариране на множество едноименни променливи (структура от данни) **изисква характеристиките на структурата да бъдат записани там, където се записва типът на простите променливи**.

② **Характеристиките често се изнасят пред скоби и именуват**.

ПРОГ_10

46/80

ЗАЩО СТРУКТУРИТЕ ОТ ДАННИ НЕ СА ТИП ДАННИ

① За тях липсват:

- ⚡ **литерални константи** (но **ВБ** има за масив);
- ⚡ **операции** (но в **ПЛ-1** и **ВБ** частично има);
- ⚡ **релации**.

② Кое е **типът данни**:

- ⚡ **масив, едномерен масив, двумерен масив?**
- ⚡ **едномерен масив с 5, с 10, с 27 елемента?**
- ⚡ **едномерен масив с 5 елемента и гранични двойки 1 и 5, 3 и 7, 0 и 4, -2 и 2, 'В' и 'Г'?**

ПРОГ_10

47/80

ЗАЩО СТРУКТУРИТЕ ОТ ДАННИ СА ПОЛЕЗНИ?

① **Моделират често срещани житейски ситуации**:

- ⚡ **Масив** → важни **математически понятия** като **вектор, матрица** и др.
- ⚡ **Запис** → **характеристики на обект, ред от таблица** и т. п.

② **Могат да се вграждат една в друга, т. е. елементите на една структура също могат да бъдат структури от данни**.

ПРОГ_10

48/80

ПРИМЕР: ДЕКЛАРИРАНЕ НА МАСИВ

Паскал: <име> : ARRAY [<гр. двойка1> [, <гр. двойка2> ...] OF <тип ел. >;

Си: разрешени са само едномерни масиви с долна граница на индекса 0: <тип ел. > <име> [<брой ел-ти>]; и **extern** <тип ел. > <име> [];

VB: [Dim | Public | Private] <име> (<гр. двойки>) [As <тип>] <гр. двойки> е [<мин. > To] <макс. > [, [<мин. > To] <макс. > ...]

И Dim <име> () [As <тип>]

ПРОГ_10

49/60

ПРИМЕР: ЦИТИРАНЕ НА ЕЛЕМЕНТ НА МАСИВ

Паскал: <име> [<инд.1> [, <инд.2> ...]

Масив ['a' , -5, true]

Си: <име> [<индекс>] , но и <име> [<индекс>] [<индекс>] ...

Масив [7] и Масив_От_Масиви [7] [16]

VB: <име> (<инд.1> [, <инд.2> ...]) и <име> (<инд.1>...) (<инд.2>...) , когато елемент на масив от тип Variant има за своя текуща стойност друг масив.

Масив (7, -5) и Вариант (6) (1, 3)

ПРОГ_10

60/60

ПРИМЕР: ДЕКЛАРИРАНЕ НА ЗАПИС

Паскал: <име> : RECORD <подиме1> : <тип1>; <подиме2> : <тип2>; ... END;

Си: struct [<Мак. >] {<тип1> <подиме1>; <тип2> <подиме2>; ... } <име>

VB: [Public | Private] Type <Макет> <подиме1> [(<гр. дв. >)] As <тип1> <подиме2> [(<гр. дв. >)] As <тип2> ...

End Type – предварително описание

И Dim <име> [(<гр. дв. >)] As <Макет>

ПРОГ_10

61/60

ПРИМЕР: ЦИТИРАНЕ НА ЕЛЕМЕНТ НА ЗАПИС

Паскал: <име> . <подиме>

C : RECORD Re: REAL; Im: REAL; END;

C.Re C.Im и WITH C DO Re Im

Си: <име> . <подиме> <ук. > -> <подиме>

struct Cmpl {float Re; float Im} C;

struct Cmpl *CPtr;

C.Re C.Im CPtr -> Re CPtr -> Im

VB: <име> . <подиме>

Type Complex ◀ Re As Single

Im As Single ◀ End Type

Dim C As Complex ◀ C.Re C.Im

ПРОГ_10

62/60

ЗА ТИПОВЕТЕ ОТ ДАННИ

? Кой определя типове от данни?

♣ Авторът на езика.

? Може ли програмистът да определя и **свои** типове от данни?

♣ Уви, почти **не**, защото трябва да се посочат:

- всички литерални константи (=МДС).
- разрешените **операции**;
- съществуващите **релации**.

? Какво се прави, когато типове не стигат?

♣ Предполага се, че **комбинирането** на целочисления тип, структурите и деленето на програмни части **ще бъде достатъчно**.

ПРОГ_10

63/60

ЗА ТИПОВЕТЕ (прод.)

? Защо в ЕПВР се говори за нови типове?

♣ Декларацията на структура от данни:

- синтактично **прилича** на тип данни;
- може да бъде доста **объркваща (Си)**;
- за по-бързо писане и по-лесно възприемане е удобно да бъде **изнесена пред скоби и именована**.

? И все пак прави ли се нещо по въпроса?

♣ Да, и то **доста**:

- **Алгол-68** разрешава **пълно създаване на нов тип**;
- **клас от обекти в обектното програмиране (VB) и ООП (Си++ и др.)** **донякъде е обобщение на тип**, а обект (екземпляр) – на променлива;
- много езици (**Паскал, Си, VB** и др.) предоставят два **механизма за частично създаване на нови (непълни) типове** от данни: **изброяване и ограничаване**.

ПРОГ_10

64/60

МЕХАНИЗЪМ ИЗБРОЯВАНЕ

❶ Новият тип се нарича **изброим**.

❷ **Името на новия тип** се определя от програмиста чрез идентификатор.

❸ **Заедно с името** също чрез идентификатори се посочват **литералните константи** и техният **ред на пряко следване**.

❹ **Операции не са разрешени**. Има стандартни функции за трансформиране на **стойност в пореден номер и обратно**.

❺ **Релациите** са на **пълната наредба**, породена от прякото следване.

❻ **След определянето на новия дискретен тип** могат да се използват **неговото име и имената** на неговите литерални **константи**.

ПРОГ_10

65/60

ЗАБЕЛЕЖКИ

❶ **На пръв поглед механизмът за изброяване изглежда излишен: целите числа**, като универсална азбука за кодиране, **вършат същата работа, особено** когато е разрешено използването на **именовани константи**.

❷ Всъщност нещата са доста **по-сложни**:

♣ **целочисленият тип има операции**, които може да са нежелани – какво е **понеделник + вторник?**

♣ **чрез целите числа** може да се постигне **циклично следване**, което не е възможно при изброимите типове – последният ден **неделя е и преди първия понеделник**.

ПРОГ_10

66/60

ПРИМЕР: ОПРЕДЕЛЯНЕ НА НОВ ИЗБРОЙМ ТИП

Паскал: в раздела за типове **TYPE** чрез
 <Име Тип> = (<конст1> [, <конст2> ...]);
 ДЕН = (ПОН, ВТО, СРЯ, ЧЕТ, ПЕТ, СЪБ, НЕД);

Си: enum [<Име Тип>] {<конст1> [= <цяло1>] [,
 <конст2> [= <цяло2>] ...] [<Пром.>];
 Enum Ден = {Пон=1, Вто, Сря, Чет, Пет, Съб,
 Нед} Днес; и Enum Ден Утре;

VB: [Public | Private] Enum <Име Тип>
 <Име Конст1> [= <израз1>]
 ...
 End Enum
 Enum Ден ◀ Пон=0 ◀ ... ◀ End Enum

ПРОГ_10

67/60

МЕХАНИЗЪМ ЗА ОГРАНИЧАВАНЕ НА ТИП

- ❶ **Новият тип съдържа** краен **интервал от стойностите** на съществуващ дискретен тип и е **съвместим** с него.
- ❷ **Името** на новия тип **и краищата на интервала** се определят от програмиста.
- ❸ **Операциите и релациите** се наследяват от ограничавания базов тип.
- ❹ **Не е необходим** в езиците, които предлагат **съвместими типове**.
- ❺ **Повишава яснотата** на програмата и има **теоретично значение** без почти никаква **практическа полза**.

ПРОГ_10

68/60

ПРИМЕР: ОПРЕДЕЛЯНЕ НА ОГРАНИЧЕН ТИП

Паскал: в раздела за типове **TYPE** чрез
 <Име Тип> = <От> .. <До>;
 ПЪРВА_ИНДЕКСНА_ДВОЙКА = -5 .. +7;

Единствената практическа **полза** от този механизъм **е** за **определяне на индексните двойки** на масиви.

Си: Поради наличие на съвместими типове такъв механизъм **липсва**.

VB: Поради наличие на съвместими типове такъв механизъм **липсва**.

ПРОГ_10

69/60

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И
В СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
ОПЕРАЦИИТЕ И
ИЗРАЗИТЕ**