ЛЕКЦИЯ 12
ПРОСТИ ОПЕРАТОРИ

- Предназначение и запис на операторите
- Ред за изпълнение
- Именоване на операторите
- Видове оператори
- Основни прости оператори

© доц. д-р Вл. Шкуртов, 2004

Последна редакция 16 Септември 2004 г.

ПРЕДНАЗНАЧЕНИЕ НА ОПЕРАТОРИТЕ

Операторите в ЕП **служат за изразяване на заповедите** в алгоритмите.

Всеки оператор описва определено действие, което може да е съставено **и от отделни по-прости операции**, **но** от гледна точка на програмата **се разглежда като единно действие**.

ПРОГ_12 2/24

© доц. д-р Вл. Шкуртов, 2004

ОПЕРАТОРИ

Включените в даден оператор **изрази** служат за **детайлизиране на** цялостното му **действие**, **но не играят роля извън него**.

Операторите се изграждат на базата на **ограничителите**, **величините** и **изразите**.

ПРОГ_12 **3/24**

© доц. д-р Вл. Шкуртов, 2004

ЗАПИС НА ОПЕРАТОРИТЕ

Операторите в една програма са много и всеки език има свои **правила за определяне на границите им**.

Прилагат се **четири схеми**:

- **във всеки ред** се записва **един** единствен **оператор** (**Фортран, ЕПНР**);
- редовете не влияят и операторите се разделят със **специален знак** (**Паскал**);
- **редовете не влияят** на записа **и няма разделител** (**Си**);
- **операторите в един ред се разделят със** специален **знак** и не се пренасят (**ВБ**).

ПРОГ_12 4/24

© доц. д-р Вл. Шкуртов, 2004

РЕД ЗА ИЗПЪЛНЕНИЕ

Операторите се изпълняват един след друг в реда, по който съставят текста на програмата.

Това се нарича естествен ред на изпълнение на програмата.

Трябва да се отбележи, че за описанието на редица алгоритми естественият ред на изпълнение не е достатъчен.

ПРОГ_12 5/24

© доц. д-р Вл. Шкуртов, 2004

ИМЕНОВАНЕ НА ОПЕРАТОРИТЕ

За да може при необходимост да **бъде нарушаван естественият ред на изпълнение**, в повечето езици съществува специален **оператор за безусловен преход**, който ***явно посочва своя наследник*** в реда на изпълнение.

За използване на подобен оператор трябва да можем да **различаваме операторите на програмата**, като ги снабдим **с имена**.

ПРОГ_12 6/24

© доц. д-р Вл. Шкуртов, 2004

ЕТИКЕТИ

Имената на операторите се наричат **етикети**.

Етикетите се избират от програмиста и могат да бъдат **идентификатори** (**Си**), **цели числа** (**Паскал**, Фортран) или **и двете** (**ВБ**).

Етикет може да се записва **пред всеки** оператор, **но е достатъчно** с етикет **да се отбелязват само желаните** оператори (тези **с неестествен ред на изпълнение**).

ПРОГ_12 7/24

© доц. д-р Вл. Шкуртов, 2004

ЗАПИС НА ЕТИКЕТ

Известни са три начина за записване на етикетите:

1. Използване на специален разделител, най-често двоеточие (:) – Паскал, Си, ВБ, Алгол–60 и др., което позволява един оператор да има и няколко етикета;
2. Специално определено поле в реда – ЕПНР, Фортран и др.;
3. Всички редове от програмата се номерират и тези номера са етикет на първия оператор в реда – Бейсик.

ПРОГ_12 8/24

© доц. д-р Вл. Шкуртов, 2004

ВИДОВЕ ОПЕРАТОРИ

Прието е операторите да се разделят на **две категории в зависимост от правилата за** тяхното **записване**:

❶ **Прости** (**елементарни**), които **не съдържат** като част от своя запис (в себе си) **други оператори** (**действия**);

❷ **Структурни** (**съставни**), които в записа си **съдържат други оператори** и задават **правилата за изпълнение на тези свои** съставни **части** (**поддействия**).

Компонент на структурен оператор може да бъде **произволен оператор** (**вкл. структурен**).

ПРОГ_12 9/24

© доц. д-р Вл. Шкуртов, 2004

ОСНОВНИ ПРОСТИ ОПЕРАТОРИ

Повечето ЕПВР предоставят следните **прости оператори**:

1. **Празен** оператор;
2. За **край на изпълнението**;
3. За **безусловен преход**;
4. За **присвояване**;
5. За **обмен на данни** (вход и изход);
6. За **изпълнение на програмна част** (активиране на процедура).

ПРОГ_12 10/24

© доц. д-р Вл. Шкуртов, 2004

ПРАЗЕН ОПЕРАТОР

Не влияе на изпълнението.

Обикновено **няма видим запис**,
но във Фортран има име – **CONTINUE**.

Ползата от такъв оператор **е, че той**:

1. може да има **етикет**;
2. може да има **коментар**;
3. позволява **многократното записване на разделител** между операторите да **не** бъде третирано като **грешка**.

ПРОГ_12 11/24

© доц. д-р Вл. Шкуртов, 2004

ОПЕРАТОР ЗА КРАЙ

Достигането до **края на текста** на програмата (и на програмна част) **прекратява** нейното **изпълнение**.

За **да се избегне** излишното записване на оператори за **безусловен преход** повечето езици предоставят **специални оператори**, чието изпълнение **спира** изпълнението на **програмата** (програмната част).

ПРОГ_12 12/24

© доц. д-р Вл. Шкуртов, 2004

ПРИМЕР: ОПЕРАТОРИ ЗА КРАЙ

Паскал: **няма** такъв оператор.

Си: `return` [<израз>]; край на програмна част. Програмата завършва с края на програмна част, наречена `main`;

`exit (`<израз>`);` – вградена функция за край на програмата.

ВБ: `End` – край на програмата;

`Stop` – временно спиране;

`Exit` … – край на програмна част.

ПРОГ_12 13/24

© доц. д-р Вл. Шкуртов, 2004

БЕЗУСЛОВЕН ПРЕХОД

Почти винаги има вида
`goto` <етикет> (= премини към <етикет>).

Нарушава естествения ред и след него се изпълнява **явно посочен оператор**.

Наличието на някои **структурни оператори го прави излишен** и е обявен „**вън от закона**“ на тенденцията, наречена **структурното програмиране**, защото броят на **логическите грешки е право пропорционален** на неговото използване.

ПРОГ_12 14/24

© доц. д-р Вл. Шкуртов, 2004

РЕАБИЛИТАЦИЯТА НА GO TO

Вината на go to е, че замъглява четенето и разбирането на програмата.

Теоретиците на структурното програмиране са доказали, че всяка програма с go to може да бъде заменена с еквивалентна, в която той отсъства.

Уви! Еквивалентната програма е много по-мъглява от тази с go to, но използван само там, където му е мястото.

ПРОГ_12 15/24

© доц. д-р Вл. Шкуртов, 2004

ПРИМЕР: ОПЕРАТОРИ ЗА БЕЗУСЛОВЕН ПРЕХОД

Паскал: `GOTO` <етикет> `;` но всеки етикет трябва да бъде деклариран предварително в частта `LABEL` … .

Си: `goto` <етикет> `;` – явен преход;

`break;` – неявен преход;

`continue;` – друг неявен.

ВБ: `Go To` <етикет> – явен преход;

`Exit` … – неявен преход.

ПРОГ_12 16/24

© доц. д-р Вл. Шкуртов, 2004

ОПЕРАТОР ЗА ПРИСВОЯВАНЕ

Може да се приеме, че **това е *основният* оператор** на всеки език, защото **чрез него** става **промяна на текущите стойности**.

Обичайният му **вид** на този оператор **е**:

<име на променлива> <знак> <израз>

Изпълнението му **предвижда изчисляване на** посочения **в дясно израз и записване на** получения **резултат** като **текуща стойност** на променливата, чието име е **в ляво**.

ПРОГ_12 17/24

© доц. д-р Вл. Шкуртов, 2004

ЗАБЕЛЕЖКИ

От двете страни на знака за присвояване се използват **еднакви имена**, но с **различен смисъл**.

Името на променлива или на елемент на структура **в ляво** означава **адрес от ОП**.

Името в дясната част означава **текуща стойност** на променлива или елемент на структура от данни, т. е. ***съдържание на ОП*** (запомнената **поредица от 0 и 1**).

Типичен пример: **X = X + 1**.

ПРОГ_12 18/24

© доц. д-р Вл. Шкуртов, 2004

ПРИМЕР: ОПЕРАТОР ЗА ПРИСВОЯВАНЕ

Паскал: <име> := <израз> ;

Си: **Няма** такъв **оператор**, защото **присвояването е операция**. Но има оператор-израз: <израз> ;

ВБ: [`Let`] <име> = <израз> и

`Set` <име> = <израз> обектови променливи (т. е. **указатели**).

ПРОГ_12 19/24

© доц. д-р Вл. Шкуртов, 2004

ОПЕРАТОРИ ЗА ОБМЕН

Тези оператори осигуряват връзка с околния свят. В зависимост от посоката те се наричат оператори за вход и изход.

Без подобни оператори *няма* реално действаща програма.

Изпълнението на подобни оператори е в пряка връзка с използването на ПУ.

Монополист в работата с ПУ е ОС, т. е. изпълнението на тези оператори изисква *съдействието* на ОС.

ПРОГ_12 20/24

© доц. д-р Вл. Шкуртов, 2004

ОБМЕН НА ДАННИ

Връзката с околния свят е от два типа:

- **програма (компютър) ↔ човек;**
- **програма (компютър) ↔ ЗУ (компютър).**

Хората **обичат 10-ичната БС и знаците, а** ***компютрите*** **– двоичната БС и ОП.**

Затова при изпълнение на операторите за обмен често (но не винаги) е необходимо компютърното представяне на данни да се преобразува в човешко (при изход) и обратно (при вход).

ПРОГ_12 21/24

© доц. д-р Вл. Шкуртов, 2004

ПРИМЕР: ОПЕРАТОРИ ЗА ВХОД НА ДАННИ

Паскал: `READ(`<променливи>`);` и

`READLN(`<променливи>`);`

Си: **Няма** такива **оператори**.

ВБ: твърде много и то **само при** обмен с **дисков файл**:

`Input #n,` <променливи>

`Line Input #n,` <текст. пр.>

ПРОГ_12 **22/24**

© доц. д-р Вл. Шкуртов, 2004

ПРИМЕР: ОПЕРАТОРИ ЗА ИЗХОД НА ДАННИ

Паскал: `WRITE (`<изрази[: n [: m]]>`) ;`

`WRITELN (`<изрази[: n [: m]]>`) ;`

Си: **Няма** такива **оператори**.

ВБ: твърде много и то **само при** обмен с **дисков файл**:

`Print #`*n*`[ ,` <пром. 1> [{ `,` | `;`} <пром. 2> ...] [{ `,` | `;`}]

© доц. д-р Вл. Шкуртов, 2004

БЛАГОДАРЯ ВИ ЗА ВНИМАНИЕТО!

БЪДЕТЕ С МЕН И В СЛЕДВАЩАТА ЛЕКЦИЯ, КОЯТО ЩЕ НИ ОТВЕДЕ В НЕВЕРОЯТНИЯ СВЯТ НА СТРУКТУРНИТЕ ОПЕРАТОРИ

© доц. д-р Вл. Шкуртов, 2004