

ЛЕКЦИЯ 11 ОПЕРАЦИИ И ИЗРАЗИ

- ⌚ **Операции и техният запис**
- ⌚ **Правила за определяне на записаната операция**
- ⌚ **Преобразуване на типа**
- ⌚ **Изрази и техният запис**
- ⌚ **Правила за пресмятане на един израз**

ПРОГ_11

1/16

ОПЕРАЦИИ В ЕПВР

- 1 **Наличните операции, правилата за получавания резултат и знакът, с който се отбелязват те, са част от определянето на всеки тип от данни.**
- 2 **Видове операции:**
 - по брой на операндите: с един, с два и много много рядко с три операнда;
 - по тип на резултата: от същия тип като операндите и от друг тип;
 - по категории: аритметични, логически, побитови, сравнения, символни и т. н..
- 3 **С ранг на операции са и всички стандартни функции, определени от езика.**

ПРОГ_11

2/16

ЗАБЕЛЕЖКИ

- 1 **Знак за операция на английски е operator, което неспособните преводачи механично заменят с оператор, чийто английски оригинал е statement.**
- 2 **Заради математиката (почти) няма операции с повече от два операнда.**
- 3 **Операндите са от един и същи тип.**
- 4 **Стойностите на операндите се получават до прилагане на операцията.**
- 5 **Условната операция с три операнда е изключение от правила 3 и 4.**

ПРОГ_11

3/16

ЗАПИС НА ОПЕРАЦИИТЕ

Операциите с един операнд имат **префиксен** запис: **<знак> <операнд>**, а в Си и **постфиксен** запис: **<операнд> <знак>**.

Операциите с два операнда имат **инфиксен** запис: **<ляв операнд> <знак> <десен операнд>**.

Стандартните функции на езика имат **функционален префиксен** запис със скоби: **<име> (<аргумент1> [<аргумент2> , ...])**.

Записът на **условната операция** е **специален**:

Си: <условие> ? <операнд ДА> : <операнд НЕ>

Алгол-60: If <условие> then <оп. ДА> else <оп. НЕ>

ПРОГ_11

4/16

ОПРЕДЕЛЯНЕ НА ОПЕРАЦИЯТА

Операциите в повечето ЕПВР са **твърде много**. Поради това, а и от разбираеми съображения: **едва ли е възможно за всеки знак на операция в езика да има уникален основен символ.**

Пример: знак **плюс (+)** означава както **целочислено събиране**, така и **приближено събиране**, че **дори и конкатенация** на низове.

Двусмислието на знака се премахва чрез анализ на операндите:

1 + 2 => целочислено събиране (резултат = 3);

0.1 + 0.2 => приближено събиране (резултат ≈ 0.3);

"не" + "но" => конкатенация (резултат = "нено");

1 + 0.2 и 0.1 + "но" => некоректни (различни типове).

ПРОГ_11

5/16

ПРЕОБРАЗУВАНЕ НА ТИПА

Анализът на операндите може да бъде **статичен** (проверка на типа по време на превода) или **динамичен** (проверка на текущата стойност по време на изпълнението).

При некоректни операнди има **две решения**:

- 1 **съобщение за намерена грешка:**
 - **при превода** със забрана за изпълнението;
 - **при изпълнението** с аварийно спиране.
- 2 **преобразуване на стойността** на единия операнд към по-висшия тип на другия.

Промяната на типа може да бъде **по премълчаване** (правило на езика) или **явно записана** (чрез операция или стандартна функция).

ПРОГ_11

6/16

ПРИМЕР: ПРОМЯНА НА ТИПА

Паскал: Анализът е **статичен**, **различни типове** операнди **са забранени** (необходима е явна промяна). Някои **диалекти** правят **неявна замяна** на **INTEGER** с **REAL**.

Си: Анализът е **статичен**, **промяната** към най-висшия тип е **неявна и задължителна** дори при един и еднотипни операнди: **винаги char, short и int → long, single → double**; и само **когато трябва long → unsigned, long и unsigned → double, double → long double**; при операция **присвояване** става **загуба на битове** без това да бъде грешка.

Вижуъл Бейсик: Анализът е **динамичен**, а **замяната** на типа е **по премълчаване** при изпълнение:

1 + "2.5" ≈ 3.5; 2.5 & "АБ" = "2.5АБ".

ПРОГ_11

7/16

ПОНЯТИЕТО ИЗРАЗ

- 1 **Изразите в ЕП са механизъм за създаване (чрез изчисляване) на нови стойности.**
- 2 **Елементарни изрази са:**
 - **величините**, които **възпроизвеждат** своята **текуща стойност**;
 - **операциите**, които **изчисляват** определен **резултат** от стойността на своите операнди.
 - **ограден в кръгли скоби (()) подизраз**, който играе роля на **единен операнд**.
- 3 **Така записът на всеки израз съдържа имена на величини, знаци за операции и скоби.**

ПРОГ_11

8/16

ПРИМЕР: ИЗРАЗИ

Паскал: `VAR A, B, C : REAL;`
`(-B + SQRT (B**B - 4.0*A*C) ) / (2.0*A)`
Си: `int i, j, k; char a[5];`
`k = i++ + ++j` (++ увеличаване с 1)
`i < 5 && a[i++]` (&& - логическо И)
Вижуъл Бейсик: `Dim R As Double`
`Const Pi = 3.14159265358979`
`2*Pi*R Pi*R^2`
`if ( B^2 - 4*A*C < 0, "Н. Р. К.",`
`Cstr ( (-B+Sqr (B^2 - 4*A*C)) / 2/A ) )`

ПРОГ_11

9/18

ИЗЧИСЛЯВАНЕ НА ИЗРАЗ

Достатъчно е в един израз да има **знаци за две операции** за да възникне въпросът: „**Коя по-напред?**“ при изпълнение на програмата.

Отговор на този въпрос дават **правилата**, известни като **приоритет на операциите**.

Чрез **тези правила** се определя **както редът за изчисляване на операциите, така и разпознаването на техните операнди**.

Правилата за приоритета на операциите са най-мътната част на всеки език.

ПРОГ_11

10/18

ПРИОРИТЕТ НА ОПЕРАЦИИТЕ

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 вътре във всяка от тях (**БВ**).

ПРОГ_11

11/18

ОБИЧАЕН ПРИОРИТЕТ

Обичайният приоритет на операциите (**по неговото намаляване**) е (**от математиката**):

- ① (под) **изрази**, оградени с кръгли **скоби**;
- ② **функции** (стандартни и потребителски);
- ③ **с един операнд**: смяна на знак, логическо НЕ, увеличаване и намаляване с 1 и др.;
- ④ **мултипликативни**: умножение, делене, намиране на остатък, логическо И и др.;
- ⑤ **адитивни**: събиране, изваждане, ИЛИ и др.;
- ⑥ **сравнения**: равно, различно, по-голямо и др.;
- ⑦ **условна операция** (когато има такава – **Си**);
- ⑧ **присвояване** (когато има такива – **Си**).

ПРОГ_11

12/18

ПРИМЕР: КОШМАРЪТ СИ

- ① Предлага **44 операции**, разпределени по приоритет в **15 групи**!
- ② Освен **европейски** (⇒) **приоритетни групи** има и **арабски** (⇐).
- ③ **Скобите** могат да бъдат **игнорирани**.
- ④ Част от операциите **променят текущата стойност на свой операнд**.
- ⑤ **Десният операнд** на логическите операции **не се изчислява, когато** стойността на **левия е** достатъчна за научаване на **резултата**.
- ⑥ Измамно **приличащи си означения**: **=** ➤ присвояване, **==** ➤ сравняване за равенство.
- ⑦ Наличие на **комбинирани операции**:
`a -= b - c` ≡ `a = a - (b - c)`.

ПРОГ_11

13/18

ПРИМЕР: ИЗРАЗ В СИ

Декларации: `char a[5], b[5], *p, *q;`
Начало: `p = a; q = b; ; p` адр. `a[0]`, `q` адр. `b[0]`.
Израз: `*p++ = *q++;`

Операция	Резултат	Страничен ефект
① <code>p++</code>	адр. <code>a[0]</code>	в <code>p</code> е адр. <code>a[1]</code>
② <code>q++</code>	адр. <code>b[0]</code>	в <code>q</code> е адр. <code>b[1]</code>
③ <code>*p++</code>	адр. <code>a[0]</code> (ляв =)	-
④ <code>*q++</code>	с/т <code>b[0]</code> (десен =)	пр. в <code>long</code>
⑤ <code>=</code>	пр. с/т <code>b[0]</code> в <code>char</code>	<code>a[0] := b[0]</code>

Краен резултат: **пренесената стойност** (като `char`) и когато тя е **нула – ЛЪЖА!**

Краен ефект: поредния елемент на **b** е пренесен в **a** и указателите сочат следващите елементи.

ПРОГ_11

14/18

ЗАКЛЮЧЕНИЕ

Изразите дават възможност за **създаване на нови** (междинни) **стойности**.

Правилата за изчисляване на изразите **са най-мътните места** в един език.

При **неувереност** в приоритетите използвайте **само кратки изрази**.

Много **компилатори** могат да бъдат заставени да **предупреждават за съмнителни места**.

ЕПВР се раждат, когато бива открит **алгоритъм за разбор** на изрази (**Фортран** е съкращение от **преводач на формули**, т. е. **изрази!**).

ПРОГ_11

15/18

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

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