

ЛЕКЦИЯ 16

УСЛОВНА КОМПИЛАЦИЯ

 **Необходимост и същност**

 **Реализиране на условна компиляция**

 **Управление на условната компиляция**

ПРОГ_16

1/15

НЯКОИ ИДЕИ НА ЕПВР

ЕПВР са създадени и с мисълта, че една компютърна **програма**, написана **на такъв език** ще може да **се изпълнява без промени на всеки компютър**, за който има транслятор от съответния език.

Това е едно **добро пожелание**, **което** уви **не може се реализира на практика**.

Различните ОС и различните конфигурации от **ПУ налагат** в създадената програма често да се правят **промени** (**основно** в частите, свързани с реализирането **на В/И**).

ПРОГ_16

2/15

НЕОБХОДИМОСТ

Едно **добро решение** при компилативната **схема** на превод, е осигуряването на **възможности за едновременното писане на няколко** близки помежду си **варианта**.

Разбира се, **всеки** от вариантите ще **се получава** чрез **самостоятелно компилиране**.

Основен **проблем** на компилативната (и смесената) **схема** на превод са **трудностите по локализиране** на програмния **участък**, който съдържа **логическа грешка**.

Горното решение решава и тези **проблеми**.

ПРОГ_16

3/15

СЪЩНОСТ

Езиците (и **компилаторите**) **осигуряват** **разгледаната полезна възможност** чрез специални **конструкции** (за **условна компилация**) посочващи, че **при определени обстоятелства** определена **част** от текста на програмата трябва да **се третира като коментар** (и да **не се превежда**).

При липса на съответните обстоятелства такива части естествено **участват в текста** на програмата (**и се превеждат** като останалото).

ПРОГ_16

4/15

ЗАБЕЛЕЖКИ

- ❶ Важно е да се осъзнае, че става въпрос за **работа с текста** на програмата, от който се **изрязват определени части**.
- ❷ Такова «изрязване» се реализира до получаване на машинната програма и дори преди да започне самия превод, т. е. **условната компилация няма нищо общо с изпълнението** на програмата.
- ❸ Интерпретаторите не могат да осигурят **аналогична възможност**, тъй като **при тях преводът и изпълнението се съвместяват**.

ПРОГ_16

5/15

РЕАЛИЗАЦИЯ

След като **познаваме условния оператор**, който осигурява възможност да създаваме **програми, които се приспособяват към обстоятелствата**, възникнали по време на тяхното изпълнение, **лесно може да се досетим**, че **апаратът за условна компилация** трябва да **предлага директиви, които външно приличат на този условен оператор**.

Съществената разлика с този оператор, е времето когато се проверява предписаното условие: до, а не по време на, изпълнението.

ПРОГ_16

6/15

ОСОБЕНОСТ НА УСЛОВИЯТА

За формиране на **условия**, проверявани в директивите за **условна компиляция**, не е възможно да се използват променливите на създаваната програма.

За целта е **необходимо** да има възможност за поименно **създаване на други елементи**.

Тъй като **тези елементи не могат да променят стойността си**, те са наречени **константи на условната компиляция**.

Определянето им е изгодно да се реализира вън и независимо от текста на програмата.

ПРОГ_16

7/15

АПАРАТ ЗА УСЛОВНА КОМПИЛАЦИЯ

Ползата от наличие на **апарат за условна компиляция** в дефиницията на **един ЕП** е **безспорна**.

Въпреки това, подобен апарат рядко присъства в езиците от високо равнище – той е по-характерен за асемблерните езици.

Апарат за **условна компиляция** липсва в езика **Паскал**, но съществува в **Си** и **Вижуъл Бейсик**. При това **Си**, за разлика от **Вижуъл Бейсик**, **предлага и макроапарат**.

ПРОГ_16

8/15

ПРИМЕР: АПАРАТЪТ НА СИ

```
{ #if <константен израз 1> |  
  #ifdef <идентификатор> |  
  #ifndef <идентификатор> }  
  <текст 1>  
[ #elif <константен израз 2>  
  <текст 2>  
  ... ]  
[ #else  
  <резервен текст> ]  
#endif
```

ПРОГ_16

9/15

ПРИМЕР: СИ (продължение)

В изразите след **#if** и **#elif** участват само целочислени константи. **Нула** означава **ЛЪЖА**, а **различно от 0** – **ИСТИНА**.

Наличието на **макроапарат** в **Си** осигурява възможност за **определяне на константи** за условната компилация **чрез #define**.

В изразите може да се използва и **функция defined(<име>)** със **стойност 1 (истина)**, когато **<име>** е **дефинирано** като макрос.

```
#ifdef <име> = #if defined(<име>)  
#ifndef <име> = #if !defined(<име>)
```

ПРОГ_16

10/15

ПРИМЕР: АПАРАТЪТ НА ВБ

```
#If <израз 1> Then ◀  
    <текст 1>  
[ #ElseIf <израз 1> Then ◀  
    <текст 2>  
    . . . ]  
[ #Else ◀  
    <резервен текст>  
#End If ◀  
  
#Const <име> = <константен израз> ◀
```

ПРОГ_16 11/15

УПРАВЛЕНИЕ НА УСЛОВНА КОМПИЛАЦИЯ

Управлението на условната компиляция се реализира **чрез подбор на стойностите на константите** за условна компиляция.

Повечето компилатори позволяват **името и стойността на константа за условна компиляция** да бъдат **посочени и чрез ръчно указание** пряко **към компилатора**.

Много компилатори предоставят **наготово определен брой константи**.

МАКРОАПАРАТ И УСЛОВНА КОМПИЛАЦИЯ

Макроапаратът и условната компилация преследват **различни цели**, реализирани **на равнището на текста на програмата**.

От това гледище **те не са свързани със създаването на компютърни програми**, но **наличието им облекчава програмирането**.

Макроапаратът може да реши и проблемите на условната компилация, но по-тромаво.

Параметризирането на макродефинициите става по-гъвкаво чрез условна компилация.

ПРОГ_16

13/15

**ЕДИН ЛЕКЦИОНЕН
КУРС ДОРИ И ДОБРЕ
ДА Е НАПРАВЕН
☺ ЗАВЪРШВА
И ВРЕМЕ ЗА ИЗПИТ
☹ НАСТАВА**

**НА ВСИЧКИ
ЧИТАТЕЛИ
НАЙ-ИСКРЕНО
ПОЖЕЛАВАМ ОТЛИЧНО
ПРЕДСТАВЯНЕ
НА НАШАТА СЛЕДВАЩА
❖ ПОСЛЕДНА СРЕЩА 😊!**