

Увод в програмирането

Управление на процесите в компютъра (част 1)

2017-2018 г.

ФМИ, специалност „Софтуерно инженерство“

Съдържание

- Основни оператори в C++
- Условни оператори
 - if
 - if-else
- Вложени условни оператори
- Области на видимост на променливите
- Оператор switch

Statement (инструкция)

- Команди в дадена компютърна програма, които се изпълняват в определена последователност
- В C++ има няколко вида инструкции
- Инструкциите включват един или повече оператори

Оператор за присвояване на стойност

- `<променлива> = <израз>;`

където

- `<променлива>` е идентификатор, дефиниран вече като променлива,
- `<израз>` е израз от тип, съвместим с типа на `<променлива>`
- `<lvalue> = <rvalue>`
 - `<lvalue>` — място в паметта със стойност, която може да се променя
 - Например – променлива
 - `<rvalue>` — временна стойност, без специално място в паметта
 - Например: константа, литерал, резултат от пресмятане

Оператор за присвояване на стойност

- Съкратени оператори

- +=
- -=
- *=
- /=

`a += 4 ; // Еквивалентно на a = a + 4 ;`
`a *= 22 ; // Еквивалентно на a = a * 22 ;`
`// и т.н.`

Едноместни операции

- `++a` и `--a` връщат `a`, което е `lvalue`
- `a++` и `a--` връщат `rvalue`
- `--a += 22; //` е валиден израз
- `a-- += 22; //` е невалиден израз

Съкратени оператори за присвояване

- По-общо:

$e1 \text{ op} = e2;$

е еквивалентно на

$e1 = (e1) \text{ op } (e2);$

където op е някой от операторите

$+, -, *, /, \%, \ll, \gg, \&, |, ^$

Асоциативност на операторите

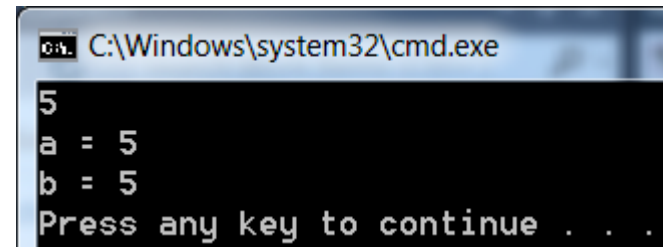
```
int main()
{
    int a, b = 3;

    cout << (a = b = 5) << "\n";

    cout << "a = " << a << endl
         << "b = " << b << endl;

    return 0;
}
```

- Операторът за присвояване е дясноасоциативен
- $a = (b = 5)$
- $\langle lvalue \rangle = \langle rvalue \rangle$



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The window content shows the output of the program: the number '5' on the first line, 'a = 5' on the second line, 'b = 5' on the third line, and 'Press any key to continue . . .' on the fourth line.

Асоциативност на операторите

- Освен дясно-, различаваме и ляво- асоциативни оператори
- Например аритметичните оператори са лявоасоциативни
- Какъв ще бъде резултатът от следния израз:

```
double a = 16 / 2 / 2;
```

Приоритет на операторите

Оператор	Име	Асоциативност
++ --	Postfix Increment Postfix Decrement	Left to Right Left to Right
++ --	Prefix Increment Prefix Decrement	Right to Left Right to Left
* /	Multiplication Division	Left to Right Left to Right
+ - %	Addition Subtraction Modulus	Left to Right Left to Right Left to Right
< >	Less Than Greater Than	Left to Right Left to Right
&&	Logical And	Left to Right
	Logical Or	Left to Right
?:	Conditional	Right to Left
=	Assignment	Right to Left
,	Comma	Left to Right

Приоритет на операторите

- Приоритетът на операторите определя реда, в който те се прилагат в рамките на дадения израз
- Какъв ще бъде резултатът от следния израз:

```
bool isTrue = 30 > 7 + 32/8;
```

Оператор за изброяване

- <израз1>, <израз2>, ..., <изразN>
- Оценяват се всички изрази, отляво на дясно (лявоасоциативен)
- Стойността на оператора се определя от най-десния израз
- Използва се рядко

```
int a = 0;
```

```
1+2, a, a+10; // Стойност на израза: 10
```

```
a++, a++, a++; // Стойност на израза: 2  
                // Стойност на a: 3
```

Празен оператор

- Знакът ;
- Не извършва никакви действия. Използва се когато синтаксисът на някакъв оператор изисква присъствието на поне един оператор, а логиката на програмата не изисква такъв

Съставен оператор (Блок)

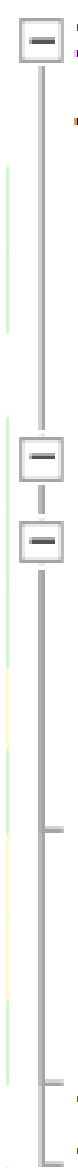
- Синтаксис

```
{<оператор1>;  
  <оператор2>;  
  ...  
  <операторN>;  
}
```

- Семантика

- Обединява нула или повече оператора в един. Може да бъде поставен навсякъде, където по синтаксис стои оператор
- Дефинициите в рамките на блока, се отнасят само за него, т.е. не могат да се използват извън него
- Самият блок не завършва с ;

Вложени блокове



```
int main()
{
    int a = -1;
    int b = 2;

    {
        {
            a += ++b*4;
            double f;
        }

        int s = a + b;
    }
}
```

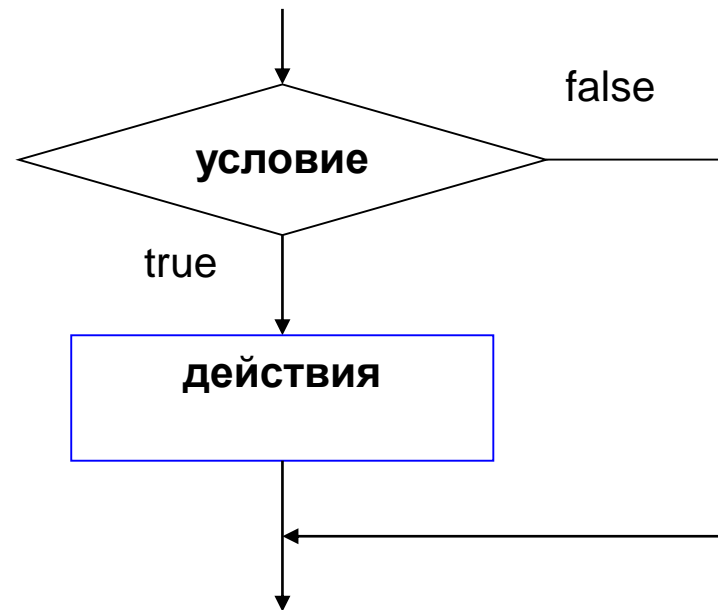
The diagram illustrates nested code blocks. A vertical line on the left side of the code is marked with three small boxes containing minus signs (-). The first box is aligned with the opening curly brace of the main function. The second box is aligned with the opening curly brace of the first nested block. The third box is aligned with the opening curly brace of the second nested block. The code is indented to show the nesting: the first nested block contains a second nested block, which in turn contains the statement `a += ++b*4;` and the declaration `double f;`. After the second nested block, the first nested block contains the statement `int s = a + b;`. The main function ends with a closing curly brace.

Оператори за условен преход

- Чрез тези оператори се реализират разклоняващи се изчислителни процеси.
- Оператор, който дава възможност да се изпълни (или не) един или друг оператор в зависимост от някакво условие, се нарича условен.
- Ще разгледаме следните условни оператори: if, if/else и switch

Условен оператор if

- Чрез този условен оператор се реализира разклоняващ се изчислителен процес



Условен оператор if

- **Синтаксис**

if (<условие>) <оператор>

където

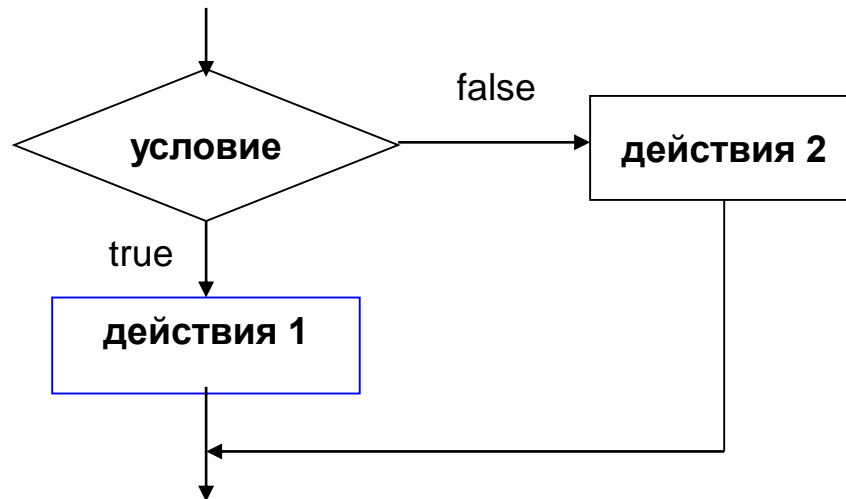
- if е запазена дума;
- <условие> е булев израз;
- <оператор> е произволен оператор.

- **Семантика**

- Пресмята се стойността на булевия израз, представящ <условие>.
- Ако резултатът е true, изпълнява се <оператор>. В противен случай <оператор> не се изпълнява

Оператор if/else

- Операторът се използва за избор на една от две възможни алтернативи в зависимост от стойността на дадено условие.
- Чрез него се реализира разклоняващ се изчислителен процес от вида



Оператор if/else ...

- **Синтаксис**

if (<условие>) <оператор1> else <оператор2>

където

- if и else са запазени думи;
- <условие> е булев израз;
- <оператор1> и <оператор2> са валидни за езика оператори.

- **Семантика**

- Пресмята се стойността на булевия израз, представящ <условие>. Ако резултатът е true, изпълнява се <оператор1>. В противен случай се изпълнява <оператор2>

Какъв ще бъде резултатът от изпълнението?

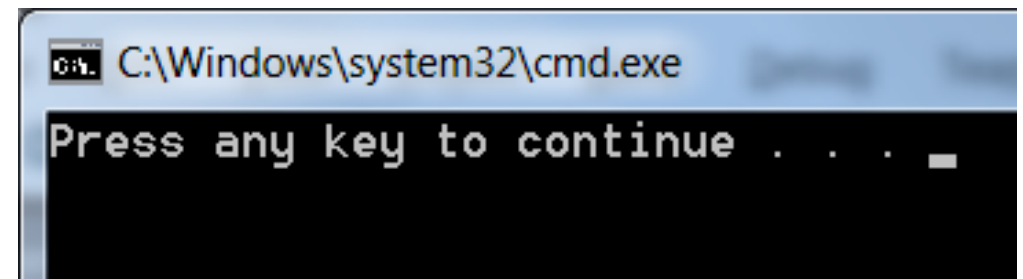
```
int main()
{
    int a = -1;
    int b = 0;

    if (a > 0)
        if (b > 0)
            cout << "First if" << endl;
    else
        cout << "Second if" << endl;

    return 0;
}
```

$(a > 0) \ \&\& \ (b > 0)$

$(a > 0) \ \&\& \ (b \leq 0)$



Какъв ще бъде резултатът от изпълнението?

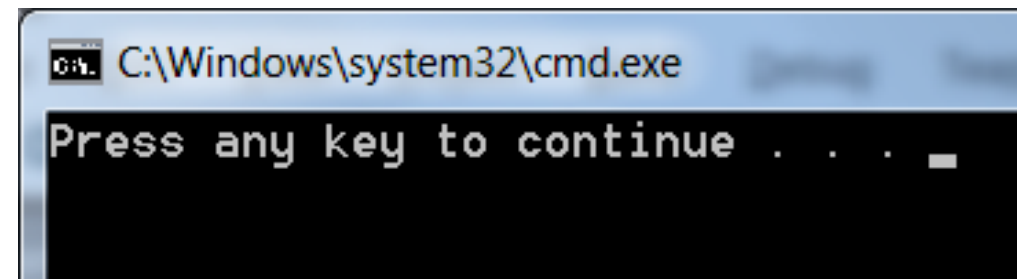
```
int main()
{
    int a = 1;
    int b = 0;

    if (a > 0) {
        if (b > 0)
            cout << "First if" << endl;
    }
    else
        cout << "Second if" << endl;

    return 0;
}
```

$(a > 0) \ \&\& \ (b > 0)$

$a \leq 0$



Използване на логическите операции

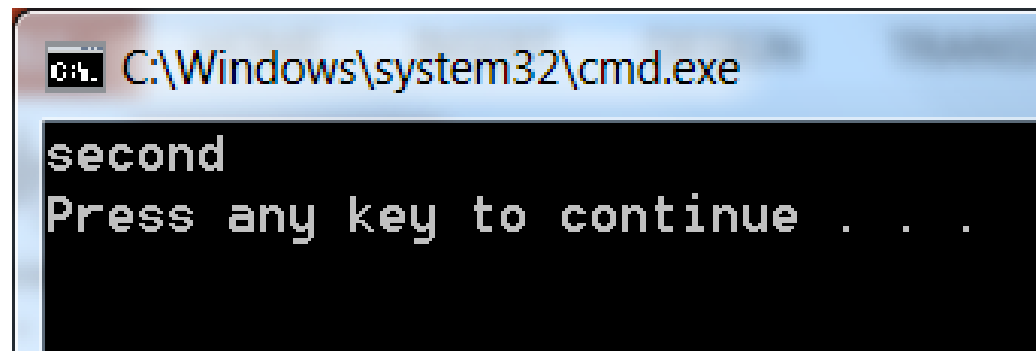
```
int main()
{
    int a = 1;
    int b = 0;

    if (a > 0 && b > 0)
        cout << "first" << endl;
    else
        cout << "second" << endl;

    return 0;
}
```

$(a > 0) \ \&\& \ (b > 0)$

$(a \leq 0) \ || \ (b \leq 0)$



Области на видимост на променливите

- Нарича се още „област на действие“ (scope)
- Областта на действие се простира от дефиницията на променливата до края на блока, в който е дефинирана
- Дефиниция на променлива със същото име в същия блок е забранена
- Дефиницията на променлива във вложен блок припокрива всички дефиниции със същото име във външните блокове

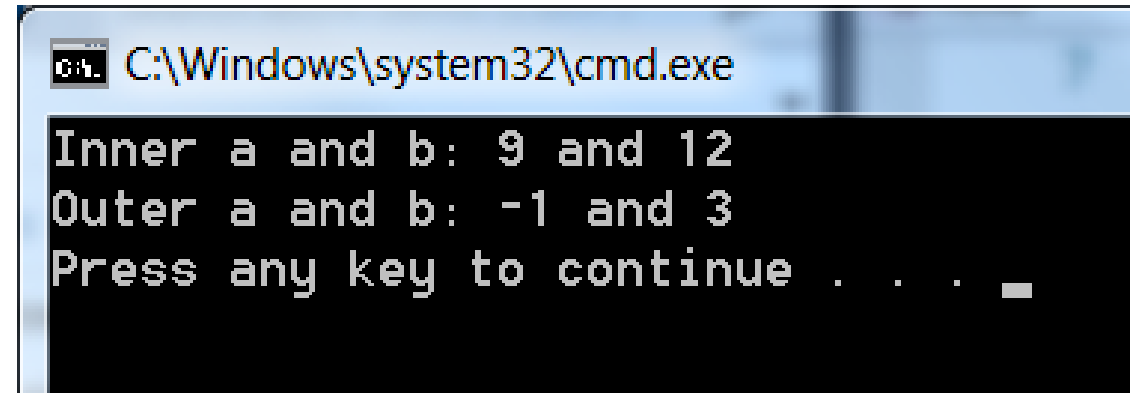
Област на видимост на променливите

```
int main()
{
    int a = -1;
    int b = 2;

    {
        int a = ++b * 3;
        int b = 12;

        cout << "Inner a and b: " << a
              << " and " << b << endl;
    }

    cout << "Outer a and b: " << a
          << " and " << b << endl;
}
```



C:\Windows\system32\cmd.exe

```
Inner a and b: 9 and 12
Outer a and b: -1 and 3
Press any key to continue . . .
```

Област на видимост на променливите

```
int Variable = 100;
```

```
if (Variable > 0)
{
    int Local = 100;
}
```

```
cout << "Variable = " << Variable << endl; // ОК
cout << "Local = " << Local << endl;        // Грешка!
```

Оператор switch

Общ вид:

```
switch ( <израз> )  
{  
  case <константен-израз-1>:  
    <case-израз-1>  
  ...  
  case <константен-израз-N>:  
    <case-израз-N>  
  [ default:  
    <default-израз> ]opt  
}
```

Семантика:

1. Оценява се израз
2. Оценката се сравнява с константните изрази
 - A. Ако е равна на някой от тях, влиза се в първия case-израза за първия такъв;
 - B. Ако не е равна на никой от тях, влиза се в default-израза, ако има такъв
 - C. В противен случай не се прави нищо.

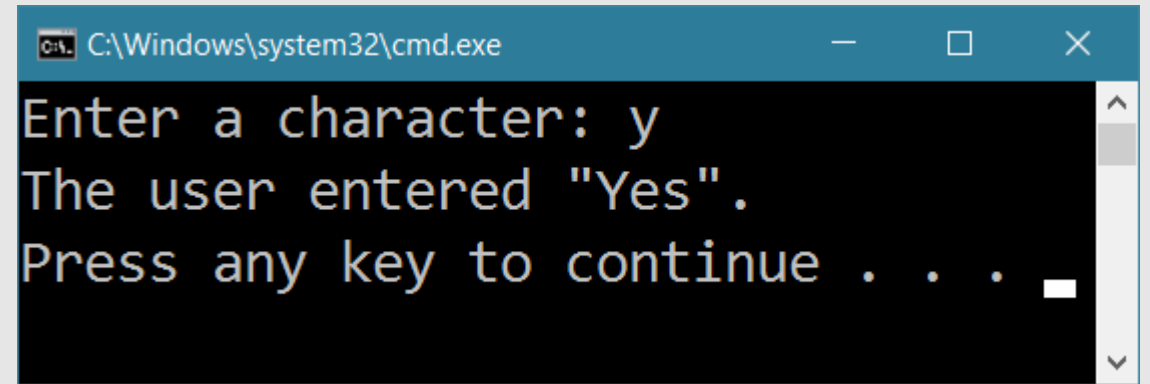
```
cout << "Enter a character: ";  
char Input;  
cin >> Input;
```

```
switch (Input)  
{
```

```
case 'y':  
    cout << "The user entered \"Yes\".\n";  
    break;
```

```
case 'n':  
    cout << "The user entered \"No\".\n";  
    break;
```

```
default:  
    cout << "The user entered something else.\n";  
}
```



```
C:\Windows\system32\cmd.exe  
Enter a character: y  
The user entered "Yes".  
Press any key to continue . . .
```

```
cout << "Enter a character: ";
```

```
char Input;
```

```
cin >> Input;
```

```
switch (Input)
```

```
{
```

```
case 'y':
```

```
    cout << "The user entered \"Yes\".\n";
```

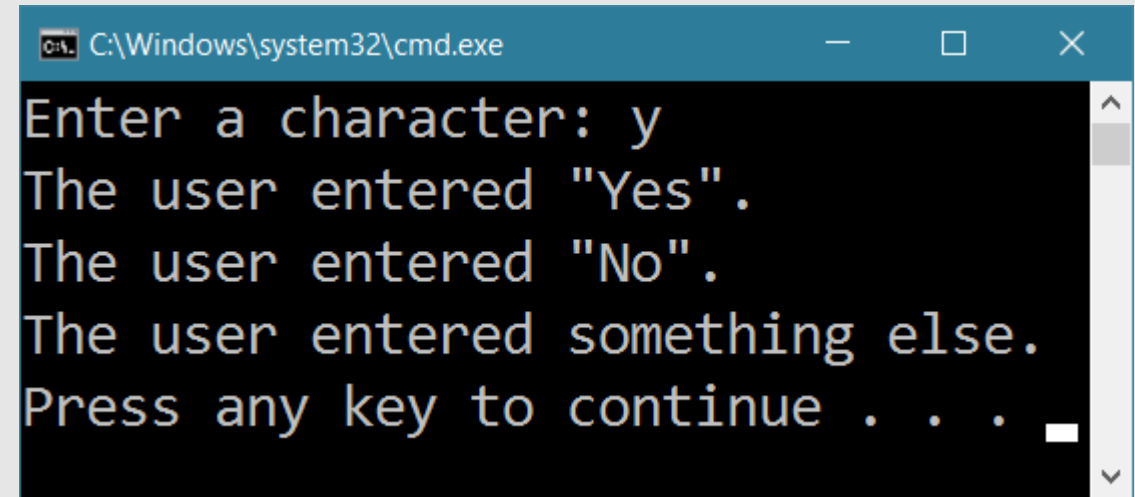
```
case 'n':
```

```
    cout << "The user entered \"No\".\n";
```

```
default:
```

```
    cout << "The user entered something else.\n";
```

```
}
```



```
C:\Windows\system32\cmd.exe
Enter a character: y
The user entered "Yes".
The user entered "No".
The user entered something else.
Press any key to continue . . .
```

```
cout << "Enter a character: ";  
char Input;  
cin >> Input;  
  
switch (Input)  
{  
case 'y':  
case 'Y':  
    cout << "The user entered \"Yes\".\n";  
    break;  
  
case 'n':  
case 'N':  
    cout << "The user entered \"No\".\n";  
    break;  
  
default:  
    cout << "The user entered something else.\n";  
}
```

Условен оператор (?:)

- Общ вид:
 - <условие> ? <израз 1> : <израз 2>
- Ако <условие> е истина (има стойност true), тогава се оценява <израз 1> и тази стойност се използва за стойност на целия израз.
- В противен случай се използва <израз2>

```
int a = (1 < 2) ? 100 : 200;
```

Условен оператор (?:)

```
int a = 20;
```

```
// Какво ли прави това?
```

```
a != 0 ? 100 / a : 0;
```

```
// Еквивалентен, но по-ясен запис.
```

```
(a != 0) ? (100 / a) : 0;
```


- За подготовката на тази презентация са използвани слайдове на:
 - Доц. Александър Григоров
 - Доц. Атанас Семерджиев
 - Доц. Трифон Трифонов