

Управляващи оператори в C++

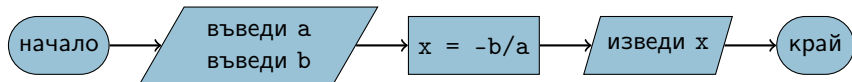
Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток, 2018/19 г.

18 октомври – 8 ноември 2018 г.

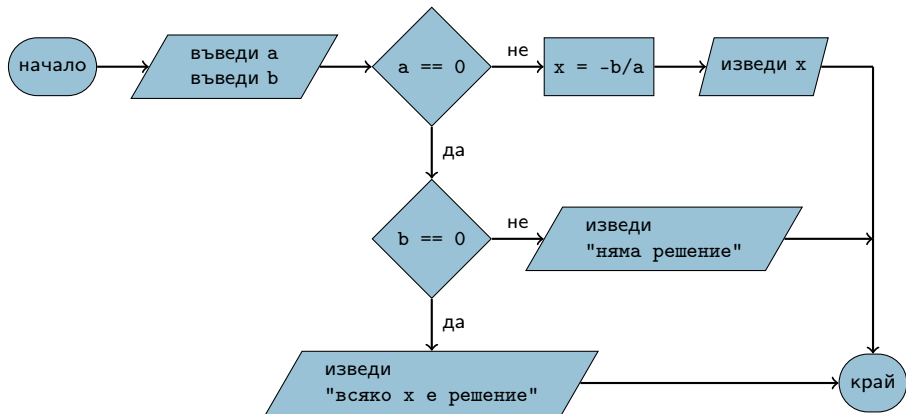
Изчислителни процеси

- Алгоритъм: последователност от стъпки за извършване на пресмятане
- Блок-схема

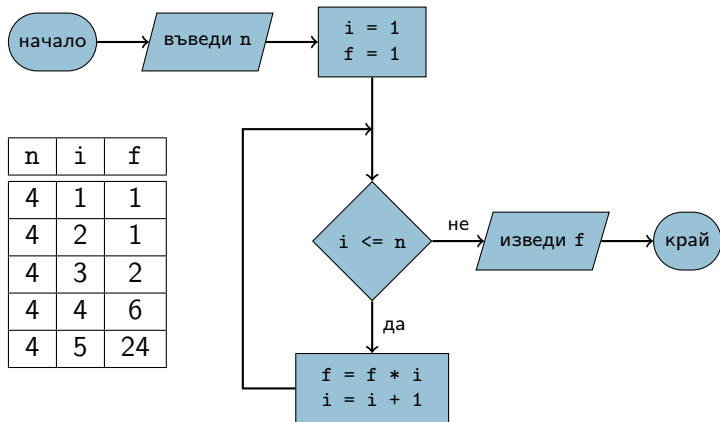


Пример за линеен процес

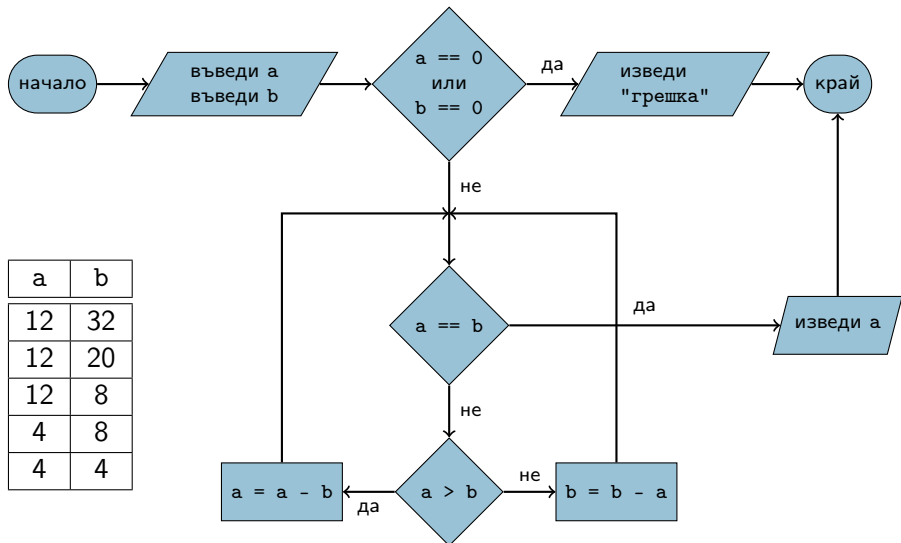
Разклоняващи се процеси



Индуктивни циклични процеси



Итеративни циклични процеси



Структурни езици — разклонение

- 1 Въведи a , b
 - 2 Ако $a == 0$, към 5
 - 3 $x = -b / a$
 - 4 Премини към 9
 - 5 Ако $b == 0$, към 8
 - 6 “Няма решения”
 - 7 Премини към 9
 - 8 “Всяко x е решение”
 - 9 Край
- Въведи a , b
 - Ако $a == 0$
 - Ако $b == 0$
 - “Всяко x е решение”
 - Иначе
 - “Няма решения”
 - Иначе
 - $x = -b / a$

Структурни езици — индуктивен цикъл

- 1 Въведи n
- 2 $i = 1$
- 3 $f = 1$
- 4 Ако $i > n$, към 8
- 5 $f = f * i$
- 6 $i = i + 1$
- 7 Премини към 4
- 8 Изведи f
- 9 Край

- Въведи n
- $i = 1$
- $f = 1$
- Повтаряй n пъти
 - $f = f * i$
 - $i = i + 1$
- Изведи f

Структурни езици — итеративен цикъл

- 1 Въведи a , b
 - 2 Ако $a == b$, към 6.
 - 3 Ако $a > b$, към 5.
 - 4 $b = b - a$; към 2.
 - 5 $a = a - b$; към 2.
 - 6 Изведи a
 - 7 Край
- Въведи a , b
 - Докато $a != b$
 - Ако $a > b$
 - $a = a - b$
 - В противен случай
 - $b = b - a$
 - Изведи a

Основни понятия

- **Операция** (operator)
- **Израз** (expression)
- **Оператор/команда** (statement)
- $\langle \text{израз} \rangle ::= \langle \text{константа} \rangle \mid \langle \text{променлива} \rangle \mid$
 $\langle \text{едноместна_операция} \rangle \langle \text{израз} \rangle \mid$
 $\langle \text{израз} \rangle \langle \text{двуместна_операция} \rangle \langle \text{израз} \rangle$
- $\langle \text{оператор} \rangle ::= \langle \text{израз} \rangle ;$

Оператор за присвояване

- `<променлива> = <израз>;`
- `<lvalue> = <rvalue>;`
- `<lvalue>` — място в паметта със стойност, която може да се променя
 - **Пример:** променлива
- `<rvalue>` — временна стойност, без специално място в паметта
 - **Пример:** константа, литерал, резултат от пресмятане
- стандартно преобразуване на типовете:
`<rvalue>` се преобразува до типа на `<lvalue>`

Присвояването като операция

- **дясноасоциативна** операция
- $a = (b = (c = 2));$
- ~~$((a = b) = c) = 2);$~~
- **Пример:** `cout << x + (b = 2);`
- **Пример:** $(a = b) = a + 3;$

Операция за изброяване

- `<израз1>, <израз2>`
- оценява и двата израза, но крайният резултат е оценката на втория израз
- `a, b, c, d ⇔ (a, (b, (c, d)))`
- **дясноасоциативна**
- използва се рядко
- **Пример:** `a = (cout << x, x);`

Съкратени оператори за присвояване

- $a = a + 2 \Leftrightarrow a += 2$
- $-=$, $*=$, $/=$, $\%=$
- $a = a + 1 \Leftrightarrow ++a$
- $a = a - 1 \Leftrightarrow --a$
- $a++$ увеличава a с 1, но връща предишната стойност на a
 - $a++ \Leftrightarrow (a = (tmp = a) + 1, tmp)$
- $a--$ действа аналогично
- $++a$ връща a , което е **<lvalue>**
 - **Пример:** $++a += 5;$
- $a++$ връща предишната стойност на a , което е **<rvalue>**
 - **Пример:** $x = a++ * b;$ ~~$a++ += 5;$~~

Оператор за блок

- { { <оператор> } }
- { <оператор₁> <оператор₂> ... <оператор_n> }
- Вложени блокове

```
{  
    int x = 2;  
    {  
        x += 2;  
        cout << x;  
    }  
}
```

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
- дефиниция на променлива във вложен блок покрива всички външни дефиниции със същото име

Област на действие (scope) — пример

```

int x = 0;
{
    x++;
    double y = 2.3;
    {
        double x = 1.6;
        y = x * x;
    }
double y = 2.4;
    x += 3;
}
x += 4;
y /= 2.1;

```

	x	y	x	
...	1	2.3	1.6	...

Празен оператор

- ;
- ; \Leftrightarrow {}
- няма никакъв ефект

Условен оператор

- `if (<израз>) <оператор> [else <оператор>]`
- Съкратената форма \Leftrightarrow пълна форма с празен оператор
 - `if (A) X; \Leftrightarrow if (A) X; else;`
- Пример: `if (x < 2) y = 2;`
- Пример: `if (x > 5) y = 5; else y = 3;`

Вложени условни оператори

Какво имаме предвид, когато пишем:

```
if (a > 0) if (b > 0) cout << 1; else cout << 3;
```

```
if (a > 0) {
    if (b > 0)
        // a > 0 && b > 0
        cout << 1;
    else
        // a > 0 && b ≤ 0
        cout << 3;
}
```

или

```
if (a > 0) {
    if (b > 0)
        // a > 0 && b > 0
        cout << 1;
}
else
    // a ≤ 0
    cout << 3;
```

Съкратено оценяване на логически операции

Представяне на логически операции с вложени условни оператори:

```
if (!A) X;
else    Y;      ⇔      if (A) Y;
                    else    X;
```

```
if (A && B) X;
else      Y;      ⇔      if (A)
                    if (B) X;
                    else  Y;
                    else Y;
```

```
if (A || B) X;
else      Y;      ⇔      if (A) X;
                    else
                    if (B) X;
                    else  Y;
```

Пример: `if (x > 0 && log(x) < 5) ...`

Пример: `if (x == 0 || y / x == 1) ...`

Условна операция

- $\langle \text{булев_израз} \rangle ? \langle \text{израз}_1 \rangle : \langle \text{израз}_2 \rangle$
- триместна (тернарна) операция
- пресмята се $\langle \text{булев_израз} \rangle$
 - При `true` се пресмята $\langle \text{израз}_1 \rangle$ и се връща резултатът
 - При `false` се пресмята $\langle \text{израз}_2 \rangle$ и се връща резултатът
- **Пример:** $x = (y < 2) ? y + 1 : y - 2;$
- $A \Leftrightarrow A ? \text{true} : \text{false}$
- $!A \Leftrightarrow A ? \text{false} : \text{true}$
- $A \ \&\& \ B \Leftrightarrow A ? B : \text{false}$
- $A \ || \ B \Leftrightarrow A ? \text{true} : B$

Задачи за условен оператор

- 1 Да се провери дали три числа образуват растяща редица
- 2 Да се намери най-малкото от три числа
- 3 Да се подредят три числа в растяща редица
- 4 Да се провери дали три числа образуват Питагорова тройка

Оператор за многозначен избор

- `switch (<израз>) {`
 `{ case <константен_израз> : { <оператор> } }`
 `[default : { <оператор> }]`
}

- **Пример:**

```
switch (x) {  
    case 1 : x++;  
    case 2 : x += 2;  
    default : x += 5;  
}
```

Оператор за прекъсване

- `break;`
- **Пример:**

```
switch (x) {  
    case 1 : x++; break;  
    case 2 : x += 2; break;  
    default : x += 5;  
}
```


Задачи за многозначен избор

- 1 Да се пресметне избрана от потребителя целочислена аритметична операция
- 2 Да се провери дали дадена буква е гласна или съгласна

Циклични структури

- Да се пресметне $\sum_{i=1}^5 i^2$
 - `x += 1*1; x += 2*2; x += 3*3; x += 4*4; x += 5*5;`
 - `x += i*i;` за $i = 1, 2, 3, 4, 5$
 - индуктивен цикличен процес
 - `for(int i = 1; i <= 5; i++) x += i * i;`
- Да се намери първата цифра на x
 - `if (x >= 10) x /= 10; if (x >= 10) x /= 10; ...`
 - `x /= 10;` докато е вярно, че `x >= 10`
 - итеративен цикличен процес
 - `while (x >= 10) x /= 10;`

Оператор for

- `for (<израз> ; <израз> ; <израз>) <оператор>`
- `for (<инициализация> ; <условие> ; <корекция>) <тяло>`
- Семантика:
 - `<инициализация>;`
 - `if (<условие>) { <тяло> <корекция>; }`
 - `if (<условие>) { <тяло> <корекция>; }`
 - `if (<условие>) { <тяло> <корекция>; }`
 - ...
- **Изключение:** `<инициализация>` може да е не просто израз, а дефиниция на променлива

Оператор for — примери

```
double sum = 0, x;
int n;
cout << "Въведете брой числа: "; cin >> n;
for(int i = 1; i <= n; i++) {
    cout << "Въведете число: ";
    cin >> x;
    sum += x;
}
cout << "Средно аритметично: " << sum / n << endl;

for(int i = 1, x = 0, y = 1; i < 5; i++) {
    x += i;
    y *= x;
}
```

Задачи за for

- 1 Да се пресметне $n!$
- 2 Да се пресметне сумата $\sum_{i=0}^n \frac{x^i}{i!}$
- 3 Да се намери броят на тези от числата $x_i = n^3 + 5i^2n - 8i$, които са кратни на 3 за $i = 1, \dots, n$
- 4 Да се намери най-голямото число от вида $x_i = n^3 + 5i^2n - 8i$ за $i = 1, \dots, n$

Оператор while

- **while** (<израз>) <оператор>
- **while** (<условие>) <тяло>
- Семантика:
 - **if** (<условие>) <тяло>
 - **if** (<условие>) <тяло>
 - **if** (<условие>) <тяло>
 - ...
- **while** чрез **for**
 - **while** (<условие>) <тяло> ⇔ **for**(;<условие>;)<тяло>
- **for** чрез **while**
 - **for**(<инициализация>;<условие>;<корекция>)<тяло>
 - ⇔
 - <инициализация>; **while**(<условие>) { <тяло> <корекция>; }

Оператор while — примери

```
cout << "НОД(" << a << ', ' << b << ") = ";
while (a != b)
    if (a > b) a %= b;
    else      b %= a;
cout << a << endl;

int n;
cout << "Въведете n: "; cin >> n;
int i = 0;
while (n > 1) {
    if (n % 2 == 0) n /= 2;
    else          (n *= 3)++;
    cout << "n = " << n << endl;
    i++;
}
cout << "Направени " << i << " стъпки" << endl;
```

Задачи за while

- 1 Да се пресметне $n!$
- 2 Да се намери средното аритметично на поредица от числа
- 3 Да се пресметне сумата $\sum_{i=0}^n \frac{x^i}{i!}$ с точност ε
- 4 Да се намери сумата на цифрите на n
- 5 Да се провери дали n съдържа цифрата 5

Оператор do/while

- `do` <оператор> `while` (<израз>);
- `do` <тяло> `while` (<условие>);
- Семантика:
 - <тяло>
 - `while` (<условие>) <тяло>

Оператор do/while — пример

```
char c;  
do {  
    char c;  
    cout << "Въведете символ: ";  
    cin >> c;  
    cout << "ASCII код: " << (int)c;  
} while (c != 'q');
```

while или do/while?

Как да изберем кой от циклите да използваме?

- Ако допускаме тялото да не се изпълни нито веднъж — `while`
- Ако искаме тялото да се изпълни поне веднъж — `do/while`

- `do` <тяло> `while` (<условие>);

⇔

<тяло> `while` (<условие>) <тяло>

- `while` (<условие>) <тяло>

⇔

`do if` (<условие>) <тяло> `while` (<условие>);

- стига <условие> да няма странични ефекти...
- **Пример:** `while` (--i > 0) cout << i << endl;

Задачи за do/while

- 1 Да се провери дали n е просто число
- 2 Да се изчисли приблизително \sqrt{x} по метода на Нютон:

$$y_0 = x$$
$$y_{n+1} = \frac{1}{2} \left(y_n + \frac{x}{y_n} \right)$$
$$\lim_{n \rightarrow \infty} y_n = \sqrt{x}$$

Вложени цикли — примери

- $a = 1$
 - $b = 1$
 - $b = 2$
 - $b = 3$
 - $a = 2$
 - $b = 1$
 - $b = 2$
 - $b = 3$
 - $a = 3$
 - $b = 1$
 - $b = 2$
 - $b = 3$
 - $a = 4$
 - $b = 1$
 - $b = 2$
 - $b = 3$
- $i = 1$
 - $j = 1$
 - $k = 1$
 - $k = 2$
 - $j = 2$
 - $k = 1$
 - $k = 2$
 - $j = 3$
 - $k = 1$
 - $k = 2$
 - $i = 2$
 - $j = 1$
 - $k = 1$
 - $k = 2$
 - ...
 - ...

Вложени цикли — примери

	Брояч
Пирамида	2:03
	2:02
	2:01
1	2:00
1 2	1:59
1 2 3	1:58
1 2 3 4	...
1 2 3 4 5	1:01
1 2 3 4 5 6	1:00
...	0:59
	...
	0:01
	0:00

Задачи за вложени цикли

- 1 Да се изведат всички плочки за играта домино
- 2 Да се провери дали в едно число има две еднакви цифри
- 3 Да се изведат всички цифри, които се срещат едновременно в числата m и n