гл.ас. д-р. Нора Ангелова

Основни конструкции

Декларация на променлива

- <тип> <име_на_променлива>{, <име_на_променлива>}$_{опц}$;

<тип> - тип данни;
<име_на_променлива> - идентификатор;

Пример:

```
int index;
int nextMonthIndex;
```

или

```
int index, nextMonthIndex;
```

Идентификатор

- Последователност от букви, цифри и _.
- Не може да започва с цифра.
- Не може да се използват ключовите думи в езика.

Пример:

```
int index;
int nextMonthIndex;
char firstCharacter;
```

- Езикът различава главните от малките латински букви.

Пример:

```
int index; и int Index;
са различни променливи.
```

- Идентификаторите трябва ясно да показват значението на съответната променлива/функция.

Идентификатор

- Конвенции за именуване:

```
int nextIndex;  // Camel case

int NextIndex;  // Pascal Case

int next_index; // Долни черти
```

Инициализация на променлива

- <тип> <име_на_променлива> = <стойност>
  {, <име_на_променлива> = <стойност>}$_{опц}$;

```
int nextIndex = 1;
```

- <тип> <име_на_променлива>(<стойност>)
  {, <име_на_променлива>(<стойност>)}$_{опц}$;

```
int nextIndex(1);
```

Константи

- const <тип> <име_на_променлива> = <стойност>;

```
// Декларация на константа
const float PI = 3.14;

PI = 3.1415 // Грешка
```

** Имената на константите често се изписват с главни латински букви.*

Аритметични оператори

- + (събиране);
- - (изваждане);
- * (умножение);
- / (целочислено деление);
- % (остатък от целочислено деление);

Пример

```
cout << 11 % 3 << endl;
```

2

```
cout << 11 / 3 << endl;
```

3

Оператори

- Бинарни – има два операнда;

```
a + b
a - b
```

- Унарни – има един операнд;

```
-a
```

Оператори

Характеристики:

- Позиция на оператора спрямо операнда
- Приоритет
- Асоциативност

Оператори

Позиция на оператора спрямо операнда:

- Префиксен – операторът е пред единствения си операнд.
- Инфиксен – операторът е между двата си операнда.
- Постфиксен – операторът е след единствения си операнд.

Примери:

1) Операторът + е както инфиксен (a + b), така и префиксен (+b).

2) Операторът ++ е както постфиксен (a++), така и префиксен (++a).

Оператори

Приоритетът определя реда на изпълнение на операторите в израз (операторен терм). Оператор с по-висок приоритет се изпълнява преди оператор с по-нисък приоритет.

Пример:
Приоритетът на операторите умножение и деление (* и /) е по-висок от този на операторите за събиране и изваждане (+ и -).

Оператори

Асоциативността определя реда на изпълнение на оператори с еднакъв приоритет в израз. В езика C++ има **лявоасоциативни** и **дясноасоциативни** оператори.

Лявоасоциативните оператори се изпълняват отляво надясно, а дясноасоциативните – отдясно наляво.

Примери.
1) Аритметичните оператори +, -, * и / са лявоасоциативни. Затова изразът a-b-c е еквивалентен на (a-b)-c, а изразът a/b/c е еквивалентен на (a/b)/c.

2) Операторът за присвояване = е дясноасоциативен. Затова изразът a = b = c (a, b и c са променливи величини или обекти на клас) е еквивалентен на a = (b = c).

Оператор за присвояване

Възможен е следният запис:

```
int a = 5;
a = a + 5;
```

- `+=, -=, *=, /=`

Пример:

```
a += 5;
```

Оператор за целочислено делене

```
int a = 123 / 10  // Целочислено делене
                  // a == 12


float b = 123 / 10; // Целочислено делене
                    // b == 12

float c = 123.0 / 10; // c == 12.3
```

Оператор ++/--

```
int a = 5;

a++; // a: 6
++a; // a: 7

a--; // a: 6
--a; // a: 5
```

Оператор ++/--

```
int a = 5;

cout << a++; // извежда 5
cout << ++a; // извежда 7

cout << a--; // извежда 7
cout << --a; // извежда 5
```

Логически оператори

- Оператор за логическо умножение (конюнкция)

```
bool A = true;
bool B = true;

A && B; // true;
```

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

Логически оператори

- Оператор за логическо събиране (дизюнкция)

```
bool A = true;
bool B = false;

A || B; // true;
```

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

- Оператор за логическо отрицание

```
bool A = false;
!A; // true;
```

A	!A
false	true
true	false

Логически оператори

- Операндите се оценяват отляво-надясно.
- Оценяването продължава докато се получи стойността на израза.

Оператори за сравнение

- == - сравнение за равно
- != - сравнение за различно
- > - сравнение за по-голямо
- >= - сравнение за по-голямо или равно
- < - сравнение за по-малко
- <= - сравнение за по-малко или равно

Условен оператор

- `if (<условие>) <оператор>`
  - if – запазена дума
  - <условие> – булев израз
  - <оператор> – произволен оператор

Пример:

```
* Операторът може да бъде ограден в {} скоби.

if (a < 3) {
  cout << "a е < 3" << endl;
}
```

Условен оператор

- if (<условие>) {
  <оператор>;
  {<оператор>;}$_{опц}$
}

Пример:

```
* Съвкупността от оператори трябва да бъде оградена в {}
скоби.
bool isSmaller = false;
if (a < 3) {
  cout << "a e < 3" << endl;
  isSmaller = true;
}
```

Оператор if/else

- `if (<условие>) <оператор1>`
  `else <оператор2>`
  - if – запазена дума
  - <условие> - булев израз
  - <оператор1> и <оператор2> - произволни оператори

Пример:

```
if (a < 3) {
  cout << "a е < 3" << endl;
} else {
  cout << "a е >= 3" << endl;
}
```

Вложени условни оператори

- `if (<условие>) <оператор1>`
  `else <оператор2>`
  - <оператор1> и <оператор2> са произволни оператори за управление на изчислителния процес, в това число могат да бъдат условни оператори

Пример:

```
if (a > 4) {
  b = 5;
} else if (a < 4) {
  b = -5;
} else {
  b = 0;
}
```

Тернарен оператор

- (<условие>) ? <оператор1> : <оператор2>

Пример:

```
int a = 5;
int b = 3;
int larger = (a > b) ? a : b;
```

Оператор switch

- ```
  switch(<израз>) {
    case <израз1> : <редица от оператори 1>
    case <израз2> : <редица от оператори 2>
    ...
    [default: <редица от оператори n>]опц
  }
  ```

- `break` – прекратява изпълнението на най-вътрешния, съдържащ го оператор `switch` или оператор за цикъл.
  ```

Пример

```
int x = 1;
int i = 1;
switch(x) {
   case 2: i+=2;
   case 1: i++;
   case 3: i+=5;
   default: i++;
}
cout << i << endl;
```

8

Масиви

- Крайна редица от елементи от един и същ тип.
- T <променлива>[size] = {<редица от константни изрази>}опц
  - T - име или дефиниция на произволен тип, различен от псевдоним, void или функционален.
  - <променлива> - идентификатор.
  - size – **константен** израз от интегрален или изброен тип с положителна стойност.
  - <редица от константни изрази> ::=

<константен израз> |
<константен израз>, <редица от константни изрази>

Константните изрази са от тип T или от тип съвместим с него.

Пример:

```
int a[100];
```

Масиви

- `int a[100]`
  - Индекс – i є [0, 99].
  - Достъп до елементите на масив.
  - Въвеждане и извеждане на елементи на масив.

```
cin >> a[i]; cout << a[i];
```

Край