

Язык программирования C++

Лекция 3

- Циклы
- Массивы

Циклы: while

В языках программирования C/C++ цикл `while` может быть выполнен ноль или более раз

Общая форма записи:

```
while (expression) {  
    statement;  
    ...  
}
```

`expression` – любое выражение, которое дает численное или логическое значение

сначала вычисляется `expression`, по результату принимается решение выполнять или не выполнять операторы в скобках `{ }`

в скобках `()` применяются те же правила, что и для условного оператора `if`

Циклы: while

Пример

```
#include <iostream>
#include <math.h>
using namespace std;

int main() {
    float x;
    while (cin >> x) {
        cout << x << sqrt(x) << endl;
    }
    return 0;
}
```

`cin` вводит данные с клавиатуры до появления специального символа `end-of-file`

во всех операционных системах семейства `Unix` таким символом является `<ctrl>-d`

Циклы: do-while

В языках программирования C/C++ цикл **do-while** может быть выполнен один или более раз

Общая форма записи:

```
do {  
    statement;  
    ...  
} while (expression);
```

expression – любое выражение, которое дает численное или логическое значение

сначала один раз выполняется оператор(ы) цикла внутри **{ }**, затем вычисляется **expression**, по результату принимается решение выполнять или не выполнять следующую итерацию цикла

в скобках **()** применяются те же правила, что и для условного оператора **if**

Циклы: for

Наиболее часто употребляемая в языках программирования C/C++ конструкция цикла

Общая форма записи:

```
for ( init-statement; test-expr; increment-expr) {  
    statement;  
    ...  
}
```

тестовое выражение `test-expr` – любое выражение, которое дает численное или логическое значение, аналогично как в условном операторе `if`

вызовы функций и ввод/вывод также допускаются

Пример:

```
for ( i = 1; i <= j; i += k) {  
    statement;  
    ...  
}
```

Циклы: for (примеры)

Типичное использование:

```
for (int i = 0; i < count; i++) {  
    // операторы тела цикла  
}
```

переменная цикла `i` объявлена в инициализирующем операторе `init-statement`

Допускаются вложенные циклы

```
for (int i = 0; i < count-1; i++) {  
    // здесь могут быть операторы внешнего цикла  
    for (j=i+1; j < count; j++) {  
        // операторы тела цикла  
    }  
    // здесь тоже могут быть операторы внешнего цикла  
}
```

Можно одновременно использовать две переменных цикла

```
for (i = 0, j=count-1; i < count-1; i++, j--) {  
    // операторы тела цикла  
}
```

выражения для разных переменных отделяются **запятыми**

Циклы: операторы break и continue

Иногда может потребоваться изменить выполнение цикла внутри него, не доходя до конца цикла, а именно прервать выполнение или перейти на следующую итерацию

```
for (int i = 0; i < 100; i ++ ) {  
    if ( i==j ) continue;  
    if ( i > j ) break;  
    // какие-то другие вычисления  
}
```

continue осуществляет переход на следующую итерацию цикла

break прерывает выполнение цикла и осуществляет выход из него

существует еще оператор **goto**, но он очень редко используется в C/C++

Массивы

Набор элементов одного и того же типа

```
float x[100];
```

доступ к первому элементу: `x[0]`

доступ к последнему элементу: `x[99]`

Инициализация элементов массива

```
float x[3] = { 1.1, 2.2, 3.3 };  
float y[ ] = { 1.1, 2.2, 3.3, 4.4 };
```

Многомерные массивы

```
float a[4][4];  
int a[2][3] = { {1,2,3},  
               {4,5,6} };
```

двумерный массив в языке C++ - это фактически одномерный массив, каждый элемент которого в свою очередь является массивом

a [строка] [столбец]

	1	2	...	<i>n</i>
1	a_{11}	a_{12}	...	a_{1n}
2	a_{21}	a_{22}	...	a_{2n}
3	a_{31}	a_{32}	...	a_{3n}
⋮	⋮	⋮	⋮	⋮
<i>m</i>	a_{m1}	a_{m2}	...	a_{mn}

Массивы

Пример умножения матриц

```
float m[3][3], m1[3][3], m2[3][3];
// вычисление элементов матриц m1 и m2
...
// m = m1 * m2
double sum;
for (int i=0; i < 3; i++) {
    for (int j=0; j < 3; j++) {
        sum = 0.0;
        for (int k=0; k < 3; k++) {
            sum += m1[i][k] * m2[k][j];
        }
        m[i][j] = sum;
    }
}
```

Массивы

Примеры инициализации массивов

```
int m[2][3] = { {1,2,3},  
               {4,5,6} };
```

То же самое

```
int m[2][3] = {1,2,3,4,5,6};
```

Массив символов

```
char name[3][9] = { "laura", "michelle", "pohl" }; //конец заполняется '\0'
```

`name[][]` представляет собой три строки, каждая содержит 9 значений типа `char`

```
name[0][0] - это 'l'  
name[0][1]   'a'  
name[0][2]   'u'  
name[0][3]   'r'  
name[0][4]   'a'  
name[0][5]   '\0'  
name[0][6]   '\0'  
name[0][7]   '\0'  
name[0][8]   '\0'
```

Практическая задача (№ С3)

Написать программу на C++, демонстрирующую порядок перебора индексов двумерного массива `int m[2][4]` :

- создать массив
- задать начальные значения всех его элементов
- с помощью двух встроенных циклов напечатать значения индексов и значения каждого элемента массива