Программирование

Лекция 4

Операция new для создания динамических массивов

Статическое связывание	•массив встраивается в программу во время компиляции
Динамическое связывание	•массив будет создан во время выполнения программы

Создание динамического массива с помощью операции new

```
int * psome = new int [10]; // получение блока памяти из 10 элементов типа int

delete [] psome;            // освобождение динамического массива
```

Правила использования new и delete:

- Не использовать delete для освобождения той памяти, которая не была выделена new.
- Не использовать delete для освобождения одного и того же блока памяти дважды.
- Использовать delete [], если применялась операция new[] для размещения массива.
- Использовать delete без скобок, если применялась операция new для размещения отдельного элемента.

Использование динамического массива

```
// arraynew.cpp -- использование операции new для массивов
#include <iostream>
int main()
{
    using namespace std;
    double * p3 = new double [3];   // пространство для 3 значений double
    p3[0] = 0.2;                    // трактовать p3 как имя массива
    p3[1] = 0.5;
    p3[2] = 0.8;
    cout << "p3[1] is " << p3[1] << ".\n";          // вывод p3[1]
    p3 = p3 + 1;                                    // увеличение указателя
    cout << "Now p3[0] is " << p3[0] << " and ";    // вывод p3[0]
    cout << "p3[1] is " << p3[1] << ".\n";          // вывод p3[1]
    p3 = p3 - 1;                                    // возврат указателя в начало
    delete [] p3;                                   // освобождение памяти
    return 0;
}
```

```
p3[1] is 0.5.
Now p3[0] is 0.5 and p3[1] is 0.8.
```

Введение в циклы for

```
// forloop.cpp -- представление цикла for
#include <iostream>
int main()
{
    using namespace std;
    int i; // создание счетчика

    // инициализация; проверка; обновление
    for (i = 0; i < 5; i++)
        cout << "C++ knows loops.\n";
    cout << "C++ knows when to stop.\n";
    return 0;
}
```

операция инкремента

тело цикла

```
C++ knows loops.
C++ knows loops.
C++ knows loops.
C++ knows loops.
C++ knows loops.
C++ knows when to stop.
```

Части цикла for

Цикл for представляет собой средство пошагового выполнения повторяющихся действий. Обычно части цикла for выполняют следующие шаги:

1. Установка начального значения.
2. Выполнение проверки условия для продолжения цикла.
3. Выполнение действий цикла.
4. Обновление значения (значений), используемых в проверочном условии.

```
for (инициализация; проверочное выражение; обновляющее выражение)
    тело
```

Части цикла for

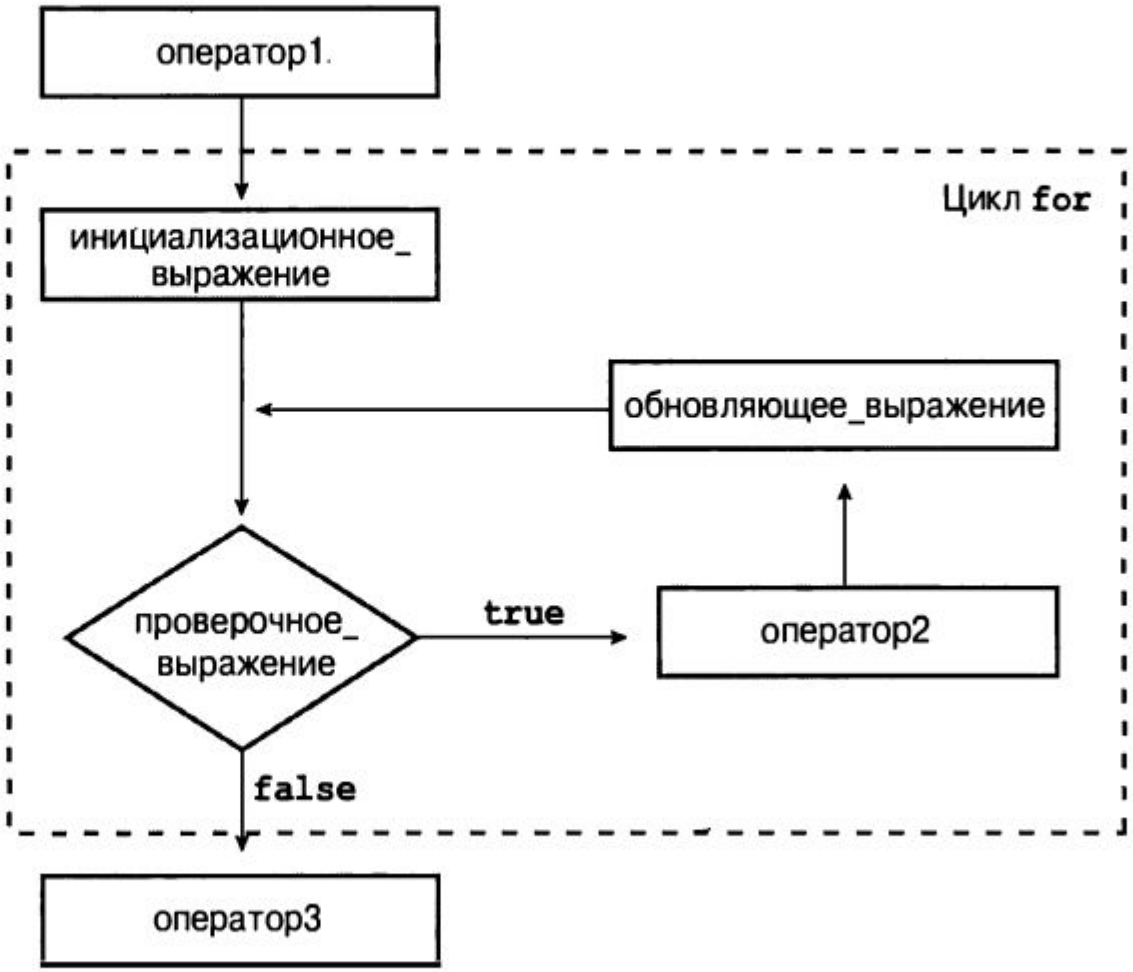
```
#include <iostream>
int main()
{
  using namespace std;
  cout << "Enter the starting countdown value: "; // ввод начального значения счетчика
  int limit;
  cin >> limit;
  int i;
  for (i = limit; i; i--)                          // завершается, когда i равно 0.
     cout << "i = " << i << "\n";
  cout << "Done now that i = " << i << "\n";       // цикл завершен, вывод значения i
  return 0;
}
```

```
Enter the starting countdown value: 4
i = 4
i = 3
i = 2
i = 1
Done now that i = 0
```

Цикл for является циклом с входным условием. Это значит, что проверочное условие выполняется перед каждым шагом цикла.

Структура циклов for

- Обновляющее выражение вычисляется в конце цикла, после того, как выполнено тело цикла.




```
оператор1
for (инициализационное_выражение;
     проверочное_выражение;
     обновляющее_выражение)
  оператор2
оператор3
```

Циклы for

Совет

В C++ принят стиль помещения пробелов между `for` и последующими скобками, а также пропуск пробела между именем функции и следующими за ним скобками:

```
for (i = 6; i < 10; i++)
  smart_function(i);
```

```
for (int i = 0; i < 5; i++)
```

```
for (int i = 0; i < 5; i++)
    cout << "C++ knows loop.\n";
cout << i << endl;          // переменная i больше не определена
```

Циклы for

```
#include <iostream>
const int ArSize = 16;    // пример внешнего объявления
int main()
{
    long long factorials[ArSize];
    factorials[1] = factorials[0] = 1LL;

    for (int i = 2; i < ArSize; i++)
        factorials[i] = i * factorials[i-1];

    for (i = 0; i < ArSize; i++)
        std::cout << i << "! = " << factorials[i] << std::endl;

    return 0;
}
```

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
```

На заметку!

Определение значения `const` для представления размера массива обычно всегда является хорошей идеей. Это значение `const` можно использовать в объявлении массива и во всех других случаях ссылок на его размер, как, например, в циклах `for`.

`i < ArSize` ⟺ `i <= ArSize - 1`

Изменение шага цикла

```
#include <iostream>
int main()
{
    using std::cout;                    // объявление using
    using std::cin;
    using std::endl;
    cout << "Enter an integer: ";       // ввод целого числа
    int by;
    cin >> by;
    cout << "Counting by " << by << "s:\n";
    for (int i = 0; i < 100; i = i + by)
        cout << i << endl;
    return 0;
}
```

```
Enter an integer: 17
Counting by 17s:
0
17
34
51
68
85
```

Доступ внутрь строк с помощью цикла for

```
#include <iostream>
#include <string>
int main()
{
    using namespace std;
    cout << "Enter a word: ";
    string word;
    cin >> word;
    // Отображение символов в обратном порядке
    for (int i = word.size() - 1; i >= 0; i--)
    cout << word[i];
    cout << "\nBye.\n";
    return 0;
}
```

```
Enter a word: animal
lamina
Bye.
```

Операции инкремента и декремента

- Операция инкремента (+ +)
- Операция декремента (- -)
- Префиксная версия (++x)
- Постфиксная версия (x++)

```
#include <iostream>
int main()
{
    using std::cout;
    int a = 20;
    int b = 20;
    cout << "a   = " << a << ":   b = " << b << "\n";
    cout << "a++ = " << a++ << ": ++b = " << ++b << "\n";
    cout << "a   = " << a << ":   b = " << b << "\n";
    return 0;
}
```

```
a   = 20:   b = 20
a++ = 20: ++b = 21
a   = 21:   b = 21
```

- нотация ***a++*** означает **«использовать текущее значение a при вычислении выражения, затем увеличить a на единицу»**
- нотация ***++a*** означает **«сначала увеличить значение *a* на единицу, затем использовать новое значение при вычислении выражения»**

```
int x = 5;
int y = ++x;                // изменить x, затем присвоить его y
                            // y равно 6, x равно 6

int z = 5;
int y = z++;                // присвоить y, затем изменить z
                            // y равно 5, z равно 6
```

Сравнение префиксной и постфиксной форм

++x; **или** **x++;**

```
for (n = lim; n > 0; --n)
    ...;
```

```
for (n = lim; n > 0; n--)
    ...;
```

Префиксная и постфиксная формы дают один и тот же результат!

Префиксная функция: увеличивает значение и затем возвращает его.

Постфиксная версия: сначала запоминает копию значения, увеличивает его и возвращает сохраненную копию.

Таким образом, для классов **префиксная версия** немного более эффективна, чем постфиксная.

Комбинация операций присваивания

```
i = i + by   =   i += by
```

Операция	Эффект (L — левый операнд, R — правый операнд)
`+=`	Присваивает `L + R` операнду `L`
`-=`	Присваивает `L - R` операнду `L`
`*=`	Присваивает `L * R` операнду `L`
`/=`	Присваивает `L / R` операнду `L`
`%=`	Присваивает `L % R` операнду `L`

Составные операторы, или блоки

```
#include <iostream>
int main()
{
    cout << "The Amazing Accounto will sum and average ";
    cout << "five numbers for you.\n";
    cout << "Please enter five values:\n";
    double number;
    double sum = 0.0;
    for (int i = 1; i <= 5; i++)
    {                                                    // начало блока
        cout << "Value " << i << ": ";                   // ввод числа
        cin >> number;
        sum += number;
    }                                                    // конец блока
    cout << "Five exquisite choices indeed! ";
    cout << "They sum to " << sum << endl;               // вывод суммы
    cout << "and average to " << sum / 5 << ".\n";       // вывод среднего значения
    cout << "The Amazing Accounto bids you adieu!\n";
    return 0;
}
```

```
The Amazing Accounto will sum and average five numbers for you.
Please enter five values:
Value 1: 1942
Value 2: 1948
Value 3: 1957
Value 4: 1974
Value 5: 1980
Five exquisite choices indeed! They sum to 9801
and average to 1960.2.
The Amazing Accounto bids you adieu!
```

Составные операторы, или блоки

```
#include <iostream>
int main()
{
    int x = 20;
    {                                   // начало блока
        int y = 100;
        cout << x << endl;              // нормально
        cout << y << endl;              // нормально
    }                                   // конец блока
    cout << x << endl;                  // нормально
    cout << y << endl;                  // ошибка во время компиляции!
    return 0;
}
```

```
#include <iostream>
int main()
{
    using std::cout;
    using std::endl;
    int x = 20;                         // исходная переменная x
    {                                   // начало блока
        cout << x << endl;              // используется исходная переменная x
        int x = 100;                    // новая переменная x
        cout << x << endl;              // используется новая переменная x
    }                                   // конец блока
    cout << x << endl;                  // используется исходная переменная x
    return 0;
}
```

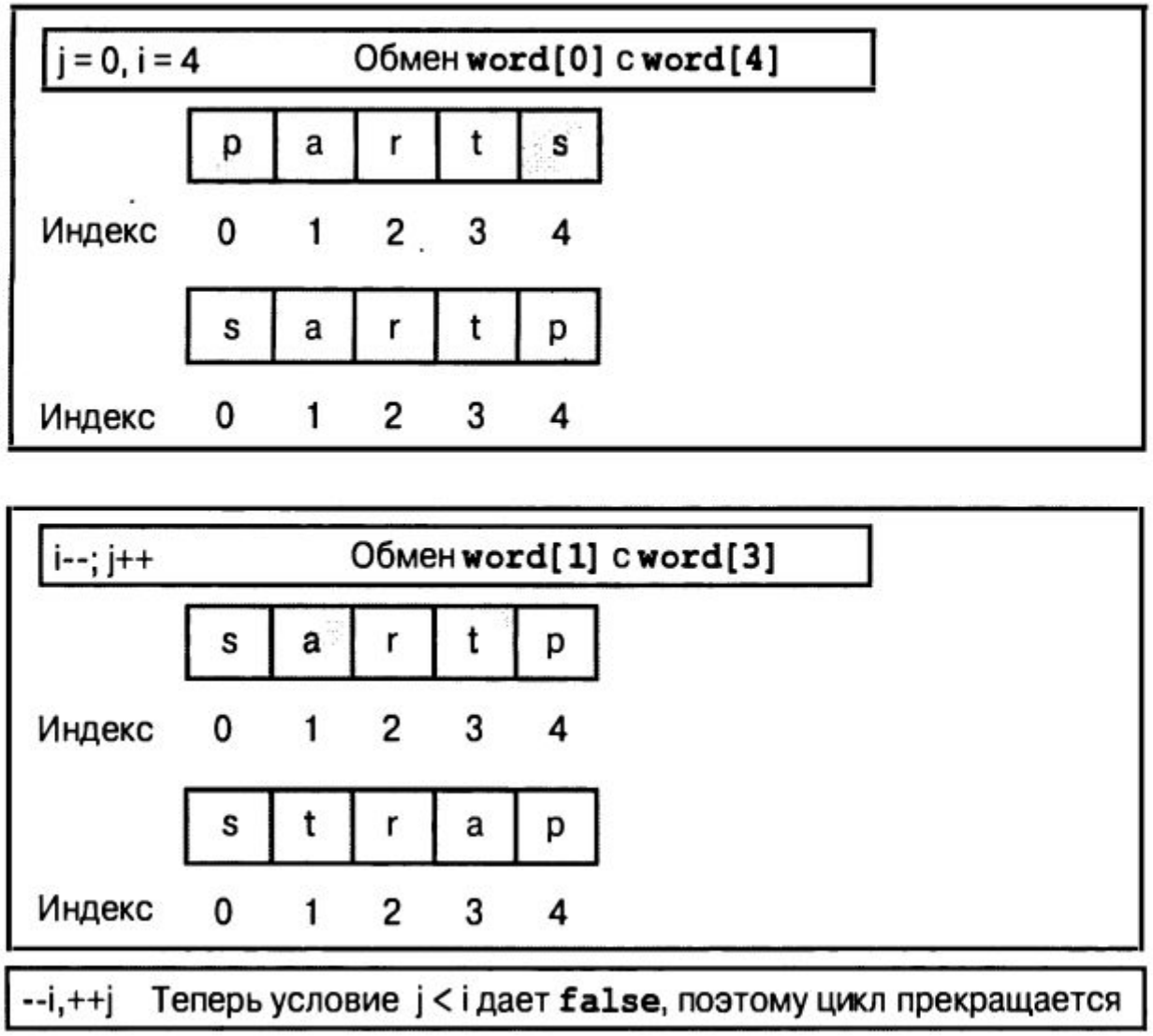
Дополнительные синтаксические трюки

```
#include <iostream>
#include <string>
int main()
{
    using namespace std;
    cout << "Enter a word: ";
    string word;
    cin >> word;

    // Физическая модификация объекта string
    char temp;
    int i, j;
    for (j = 0, i = word.size() - 1; j < i; --i, ++j)
    {                                   // начало блока
        temp = word[i];
        word[i] = word[j];
        word[j] = temp;
    }                                   // конец блока
    cout << word << "\nDone\n";
    return 0;
}
```

```
Enter a word: stressed
desserts
Done
```

Дополнительные синтаксические трюки



Выражения отношений

Операция	Описание
<	Меньше чем
<=	Меньше или равно
==	Равно
>	Больше чем
>=	Больше или равно
!=	Не равно

Примеры сравнений:

```
for (x = 20; x > 5; x--)
for (x = 1; y != x; ++x)
```

Операции отношений обладают более низким приоритетом, нежели арифметические операции:

```
x + 3 > y - 2          // выражение 1

(x + 3) > (y - 2)      // выражение 2

x + (3 > y) - 2        // выражение 3
```

Присваивание, сравнение и вероятные ошибки

```
musicians == 4        // сравнение
musicians = 4         // присваивание
```

```
#include <iostream>
int main()
{
    using namespace std;
    int quizscores[10] =
        { 20, 20, 20, 20, 20, 19, 20, 18, 20, 20};
    cout << "Doing it right:\n";                    // правильно
    int i;
    for (i = 0; quizscores[i] == 20; i++)
        cout << "quiz " << i << " is a 20\n";
    // Предупреждение: возможно, лучше почитать об этой про
    // чем в действительности запускать ее.
    cout << "Doing it dangerously wrong:\n";  // неправильн
    for (i = 0; quizscores[i] = 20; i++)
        cout << "quiz " << i << " is a 20\n";
    return 0;
}
```

```
Doing it right:
quiz 0 is a 20
quiz 1 is a 20
quiz 2 is a 20
quiz 3 is a 20
quiz 4 is a 20
Doing it dangerously wrong:
quiz 0 is a 20
quiz 1 is a 20
quiz 2 is a 20
quiz 3 is a 20
quiz 4 is a 20
quiz 5 is a 20
quiz 6 is a 20
quiz 7 is a 20
quiz 8 is a 20
quiz 9 is a 20
quiz 10 is a 20
quiz 11 is a 20
quiz 12 is a 20
quiz 13 is a 20
...
```

Сравнение строк в стиле C

```
// compstr1.cpp -- сравнение строк с использованием массивов
#include <iostream>
#include <cstring>  // прототип для strcmp()
int main()
{
   using namespace std;
   char word[5] = "?ate";
   for (char ch = 'a'; strcmp(word, "mate"); ch++)
   {
      cout << word << endl;
      word[0] = ch;
   }
   cout << "After loop ends, word is " << word << endl; // вывод word по завершении цикла
   return 0;
}
```

```
?ate
aate
bate
cate
date
eate
fate
gate
hate
iate
jate
kate
late
After loop ends, word is mate
```

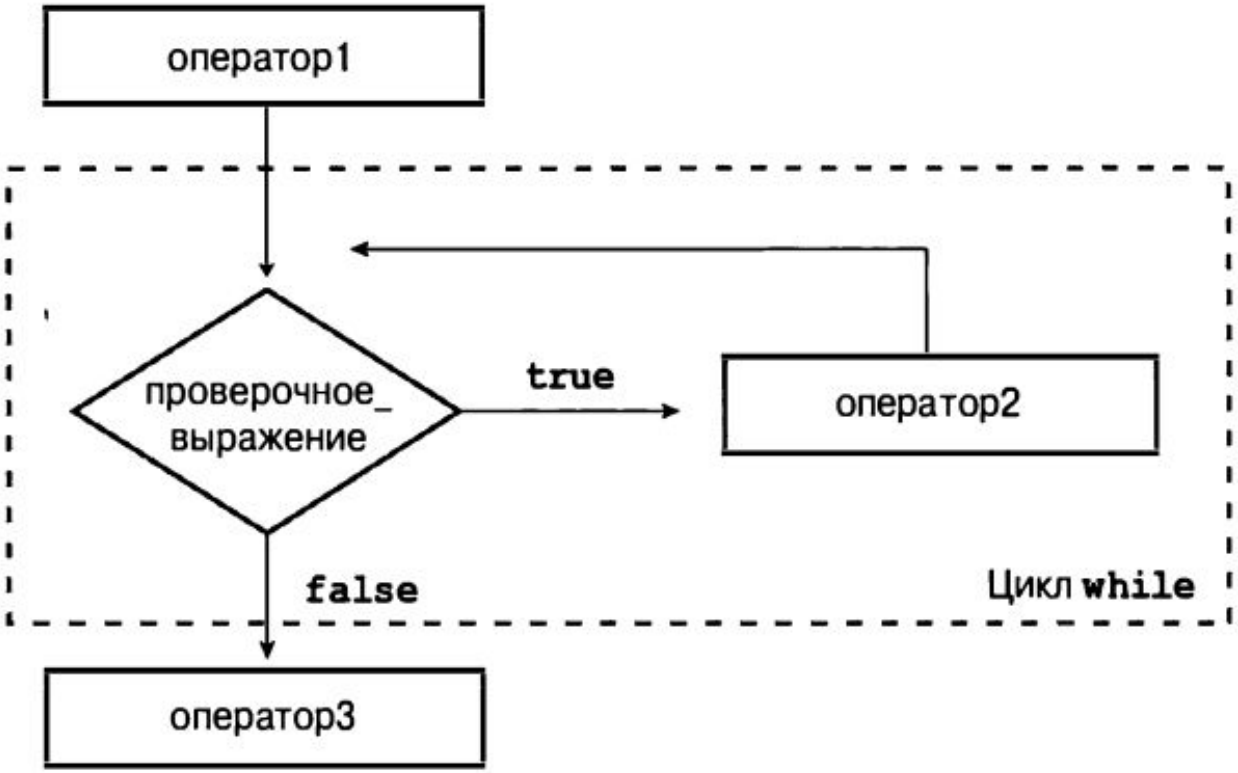
Сравнение строк класса string

```
#include <iostream>
#include <string>  // класс string
int main()
{
    using namespace std;
    string word = "?ate";
    for (char ch = 'a'; word != "mate"; ch++)
    {
        cout << word << endl;
        word[0] = ch;
    }
    cout << "After loop ends, word is " << word << endl;
    return 0;
}
```

Цикл while

```
while (проверочное_условие)
    тело
```

```
оператор1
while (проверочное_выражение)
    оператор2
оператор3
```



Цикл while

```
#include <iostream>
const int ArSize = 20;
int main()
{
    using namespace std;
    char name[ArSize];

    cout << "Your first name, please: ";  // ввод имени
    cin >> name;

    // Вывод имени посимвольно и в кодах ASCII
    cout << "Here is your name, verticalized and ASCIIized:\n";
    int i = 0;                           // начать с начала строки
    while (name[i] != '\0')   while (name[i])  до конца строки
    {
        cout << name[i] << ": " << int(name[i]) << endl;
        i++;                             // не забудьте этот шаг
    }
    return 0;
}
```

```
Your first name, please: Muffy
Here is your name, verticalized and ASCIIized:
M: 77
u: 117
f: 102
f: 102
y: 121
```

Сравнение циклов for и while

```
for (инициализирующее-выражение; проверочное-выражение; обновляющее-выражение)
{
    оператор(ы)
}
```

```
инициализирующее-выражение;
while (проверочное-выражение)
{
    оператор(ы)
    обновляющее-выражение;
}
```

```
while (проверочное-выражение)
    тело
```

```
for ( ; проверочное-выражение;)
    тело
```

```
for( ; ; )      // бесконечный цикл
    тело
```

- Обычно программисты применяют циклы for для циклов со счетчиками.
- Цикл while используется, когда заранее не известно, сколько раз будет выполняться цикл.

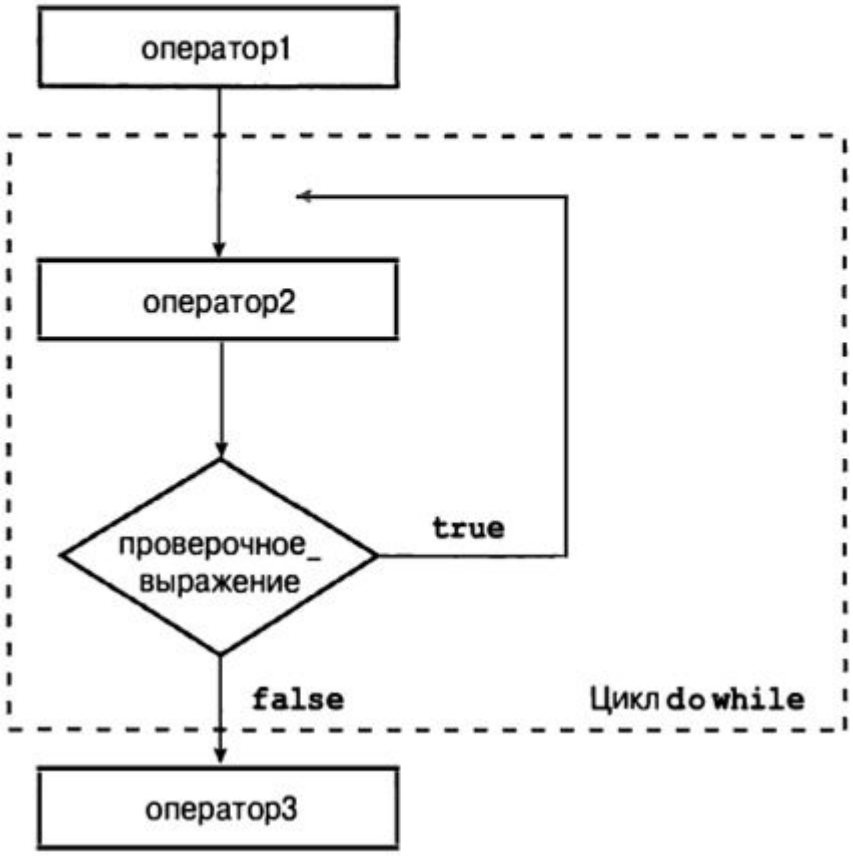
Плохая пунктуация

```
i = 0;
while (name[i] != '\0')
    cout << name[i] << endl;
    i++;
cout << "Done\n";
```

```
i = 0;
while (name[i] != '\0');  // проблема кроется в точке с запятой
{
    cout << name[i] << endl;
    i++;
}
cout << "Done\n";
```

Цикл do while

оператор1
do
оператор2
while (проверочное_выражение);
оператор3




```
do
    тело
while (проверочное-выражение);
```

Цикл do while

```
#include <iostream>
int main()
{
    using namespace std;
    int n;
    cout << "Enter numbers in the range 1-10 to find ";
    cout << "my favorite number\n"; // запрос на ввод любимого числа из диапазона 1-10
    do
    {
        cin >> n;                          // выполнить тело
    } while (n != 7);                      // затем проверить
    cout << "Yes, 7 is my favorite.\n" ;  // любимое число - 7
    return 0;
}
```

```
Enter numbers in the range 1-10 to find my favorite number
9
4
7
Yes, 7 is my favorite.
```