Учебная практика 2016-2017

Основы C++

День 1

Часть 0

- Создание проекта
- Переключение на русский язык

Часть 1

- **Стандарты C++**
- **Общие сведения**
- **Структура простой программы**
- **Простые типы данных**
- **Основные операции и операторы**

Часть 2

- Массивы, указатели, ссылки

СТАНДАРТЫ C++

Стандарты C++

Стандартизация C++ с 1989 г.

ISO (International Organization for Standartization) – группа национальных организаций по стандартам

- **C++98** первый стандарт (1998 г.) ISO/IEC 14882:1998
- **C++03** технические поправки стандарта C++98 ISO/IEC 14882:2003

C++98 & C++03 = первый стандарт C++

- **TR1** – расширения библиотеки первого стандарта ISO/IEC 19768:2007. Все изменения в пространстве имен `std::tr1`

Род. 30 декабря 1950
Орхус, Дания
Орхусский университет, Кембриджский университет (дисс.)
Доктор наук
Работал в AT&T Bell Laboratories, AT&T, сейчас – в университете Техаса
www.stroustrup.com parasol.tamu.edu/people/bs

Стандарты C++

C++0x - условное наименование нового (второго) стандарта

Получилось **C++11** **ISO/IEC 14882:2011**

Улучшение языка и библиотеки. TR1 – в пространство имен `std`

MS VS начиная с 2013 поддерживает C++11

Изменения - https://ru.wikipedia.org/wiki/C%2B%2B11

Описания, документация, примеры и т.д. www.cplusplus.com

C++14 - небольшое расширение над C++11, содержит в основном исправления ошибок и небольшие улучшения. 18.08.2014

Совместимость стандартов

Стандарт C++11 должен сохранить обратную совместимость с C++98.

Все программы, составленные по стандартам C++98 или C++03 должны компилироваться по стандарту C++11.

Но могут быть проблемы. Например, имена переменных могут совпадать с новыми ключевыми словами.

Если код должен работать в разных версиях C++, используя при этом преимущества C++11, можно использовать заранее определенный макрос **`__cplusplus`**.

Директива, определяющая использование стандарта C++ при компиляции единицы трансляции на языке C++

```
#define __cplusplus 201103L
```

И обратно

```
#define __cplusplus 199711L
```

Как и многое, макрос зависит от компилятора

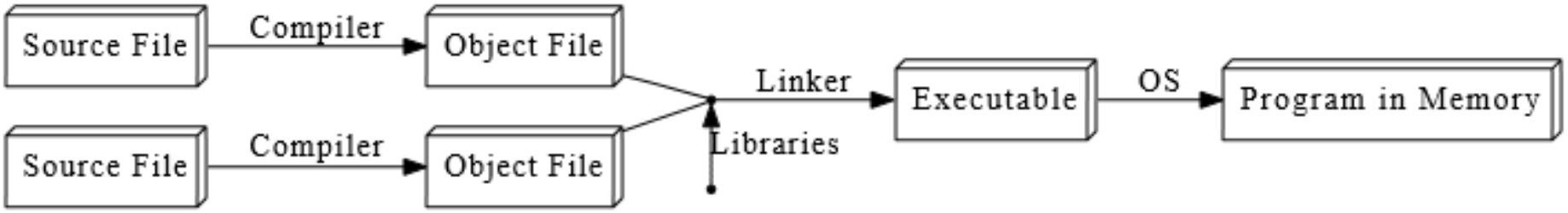
Совместимость стандартов

Обратная совместимость сохраняется на уровне исходного кода.

Бинарная совместимость не гарантируется.

Компиляция всех частей программы, написанных по стандарту C++98, включая библиотеки, с помощью компилятора, поддерживающего стандарт C++11, не должна вызывать трудностей.

Редактирование связей между кодом, скомпилированным по стандарту C++11 и кодом, созданным компилятором, поддерживающим C++98, может завершиться неудачно!!!



Общие сведения

Поддерживается ООП

Регистрозависимый язык

Язык строгой типизации

Переменная (объект) должна быть описана / определена в любом месте программы, но до ее использования

Доступ к памяти, тип указатель, арифметика указателей

Гибкость – сложно искать ошибки

Использование указателей и памяти типа «куча»- проблема утечки памяти

Время на изучение – n лет, n > 10

Терминология

Идентификатор – последовательность из букв латинского алфавита, десятичных цифр и символов подчёркивания, начинающаяся не с цифры.

Ключевые (служебные) слова – это идентификаторы, зарезервированные в языке для специального использования.

Выражение – это последовательность операндов, разделителей и знаков операций, задающая вычисления, то есть это правила для получения значения

Знаки операций обеспечивают формирование и последующее вычисление выражений.

Директива (команда) препроцессора

Определение [*declaration]*

Описание [*specification]*

ПодбельскийВ.В. Язык Си++. -М.: Финансы и статистика, 2003. –560 с.

ПРОСТАЯ ПРОГРАММА

Из чего состоит программа

1. Программа на C++ состоит из
   - функций,
   - описаний, определений
   - директив препроцессора.
2. Обязательно должна быть одна функция с именем `main()`
3. Функции вызываются либо из других функций, либо из главной функции с именем `main().`
4. Не допускаются вложенные функции.
5. Блоки – часть кода в фигурных скобках
6. Функции ввода/вывода – из стандартных библиотек,
   - в стиле C и
   - в стиле C++.

Лучше не использовать то и другое в одной программе

1. Программа может состоять из нескольких модулей (файлов). При этом функция `main()` может быть только в одном файле.

Структура программы

```
#include <iostream>
#include  <Libr.h>
```

Подключение библиотек и заголовочных файлов

```
<тип> Function1 (<список
  параметров>);
   int N=1000;
   using namespace std;
```

Описание

- прототипов функций,
- глобальных переменных,
- пространства имен

```
void main()
   {
  …
   }
```

Функция main

```
<тип> Function1 (<список
  параметров>)
   {
  …
   }
```

Другие функции

Пример программы на C++

```
#include <iostream>
using namespace std;
void main()
 {
      int a, b;
      cout << "Hello World!" << endl;
      cout <<"Enter two numbers" << '\n';
      cin >> a >> b;
      cout << "sum:" << a+b << endl;
      system("pause");
 }
```

Подключаем библиотеку ввода/вывода

Стандартное пространство имен

Функция не возвращает значения, имеет тип void

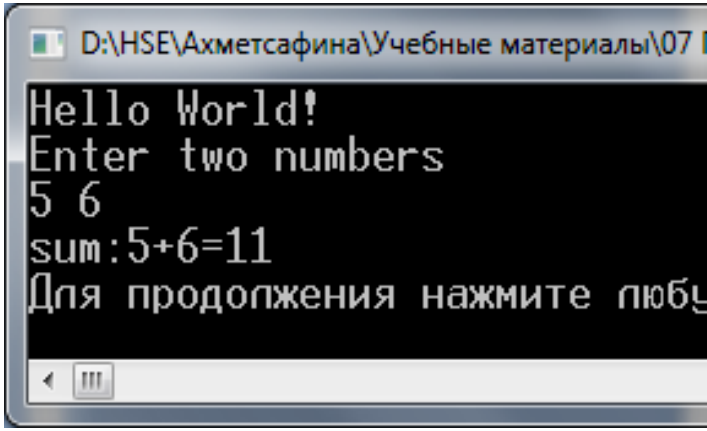
вывод текста, конец строки

ввод двух чисел

system("pause") - системная команда «пауза», ожидает нажатие любой клавиши, нужна для того, чтобы мы успели увидеть результат работы программы.

Ввод и вывод

```
int a, b;
cout << "Hello World!" << endl;
 cout <<"Enter two numbers" << endl;
 cin >> a >> b;
 cout << "sum:" << a << '+'<< b <<
'=' << a + b << endl;
 system("pause");
```



cin ввод данных с консоли. В коде элементы ввода разделяются **>>.**

При исполнении вводятся через пробел или Enter

cout - вывод данных на консоль.

Элементы вывода разделяются **<<.**

Элементами вывода могут быть

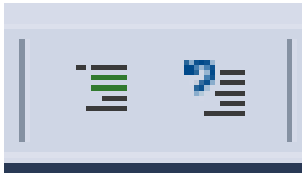
- константы, выводятся значения;
- переменные – выводятся значения;
- выражения – выводится результат вычисления.

cin и **cout** - из стандартного пространства имен, его надо указать **using namespace std;**

- **endl** или **\n** – перевод на след. строку

Комментарии

```
// однострочный комментарий
int a, b;
cout << "Hello World!" << endl;
 cout <<"Enter two numbers" << endl;
 cin >>a >>b;
 cout << "sum:" <<a<<'+'<<b<<'='<<
a+b << endl;
/* многострочный
Комментарий */
 system("pause");
```



В VisualStudio кнопки «закомментарить» «раскомментарить»

О переменных

Переменная (объект) имеет

- Имя
- Значение

(Не забудем про типы)

Просто?

Переменная может иметь не одно имя

Ссылки – псевдонимы переменных

Указатели – хранят адреса переменных, по ним можно обращаться к значениям

О переменных

1. **Время жизни** переменной (объекта) м.б. ***постоянным*** (в течение выполнения программы) и ***временным*** (в течение выполнения блока { })

2. **Область действия** переменной – локальная и глобальная.

   - Если переменная объявлена внутри блока { } , она ***локальная***. Область действия – от точки описания до конца блока, включая вложенные блоки.
   - Если переменная объявлена вне блоков – она ***глобальная***. Область действия – файл, в котором она определена, от точки описания до конца.

О переменных

3. **Область видимости** переменной (объекта) – часть текста программы, из которой допустим обычный доступ к связанной с переменной областью памяти.

Часто совпадает с областью действия.

Исключение - когда во вложенном блоке описана переменная с тем же идентификатором. Тогда внешняя переменная невидима.

К внешней глобальной переменной можно обратиться, используя операцию доступа к области видимости `::`

Начальные сведения

ПРОСТЫЕ ТИПЫ ДАННЫХ

Типы данных в C++

Типы данных определяют

1) Диапазоны значений. Определяется двумя составляющими:
   - Формат хранения
   - Выделенная память
2) Операции (операторы и функции)

Целое без знака, 4 байта

$$10000000\ 00000000\ 00000000\ 00000001_2 = A_{10}$$

Целое со знаком, 4 байта

$$10000000\ 00000000\ 00000000\ 00000001_2 = B_{10}$$

Вещественное с плавающей точкой, 4 байта

$$10000000\ 00000000\ 00000000\ 00000001_2 = C_{10}$$

см. тип union позже

Пока значения неизвестны

Ждем с нетерпением продолжения

Типы данных в C++

Типы C++ - **простые и составные**.

К **простым** относят типы, которые характеризуются одним значением. Их шесть:

1. `int` (целый)
2. `char` (символьный)
3. `wchar_t` (расширенный символьный)
4. `bool` (логический)
5. `float` (вещественный)
6. `double` (вещественный с двойной точностью)

Целочисленные

с плавающей точкой

Типы данных в C++

Спецификаторы типа уточняют внутреннее представление и диапазон стандартных типов

Спецификатор	Применяется к типу
`short` (короткий)	`int`
`long` (длинный)	`int`, `double`
`signed` (знаковый)	`int`, `char`
`unsigned` (беззнаковый)	`int`, `char`

Тип	Размер в битах	диапазон
char	8	-128 … 127
unsigned char	8	0 … 255
signed char	8	-128 … 127
int	32	-2147483648--2147483647
unsigned int	32	0… 4294967295
signed int	32	см int
short int	16	-32768 … 32767
unsigned short int	16	0 … 65535
signed short int	16	-32768 … 32767
long int	32	int
signed long int	32	signed int
unsigned long int	32	unsigned int
float	32	3.4E-38 – 3.4E+38
double	64	1.7e-308 – 1.7E+308
long double	80	3.4E-4932—1.1E+4932

И отрицательные числа

Простые типы данных

Описание переменных

```
[Класс памяти] [const] тип имя [инициализатор];
```

Описание с инициализацией

имя = значение;

имя (значение) ;

```
тип имя_пер1 = нач_знач1, имя пер2 (нач_знач2) …,
    имя_перN;
```

Константа должна быть инициализирована при объявлении, ее значение не меняется.

Простые типы данных

Примеры

```
bool Flag; // декларация , описание
int a, f, k; // декларация , описание
k = 5; a = 12 * k;          // инициализация
int b = 7, c (10); //описание с инициализацией
const int d = 12;   // константа
char ch1 ='a', ch2 (54); // символьная переменная
float bal = 123.23F;
```

Классы памяти

auto (в стеке, для глобальных не используется, для локальных – по умолчанию)

extern – переменная определяется в другом месте программы

static - статическая переменная, время жизни постоянное, инициализируется один раз, м.б. глобальной или локальной

register – размещается в регистрах процессора. Если нет возможности, то как auto

Начальные сведения

ОПЕРАЦИИ

Операции

Арифметические операции:

умножение (*),
деление (/), для целых – целая часть частного
остаток от деления (%), оба операнда д.б. целыми
сложение (+),
вычитание (-).

Формат операции простого ***присваивания*** (=):

```
операнд_1 = операнд_2
```

Пример:

```
a = b = c = 100
```

выполняется справа налево, сначала `c = 100`,
затем число 100 присвоится переменной b, далее переменной a.
Все три переменные будут равны 100.

Операции

Сложные операции присваивания:

*= – умножение с присвоением,
/= - деление с присвоением
%= - остаток от деления с присвоением
+= – сложение с присвоением
-= – вычитание с присвоением

Пример: компактная запись

```
x += y
```

означает, что к операнду_1 прибавляется операнд_2 и результат записывается в операнд_1, *т.е.* `x = x + y`

Операции

Операции увеличения (инкремента) и уменьшения (декремента) на единицу (++ и --);

Пример:

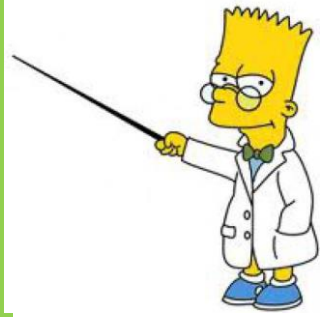
x = x + 1

компактная запись

x++ (постфиксная запись)

++x (префиксная запись)

Упражнение 1 – вывести на печать значения переменных до выполнения постфиксной и префиксной операции инкремента и после выполнения операций в выражениях, содержащих и другие арифметические операции (см. ниже приоритеты операций)

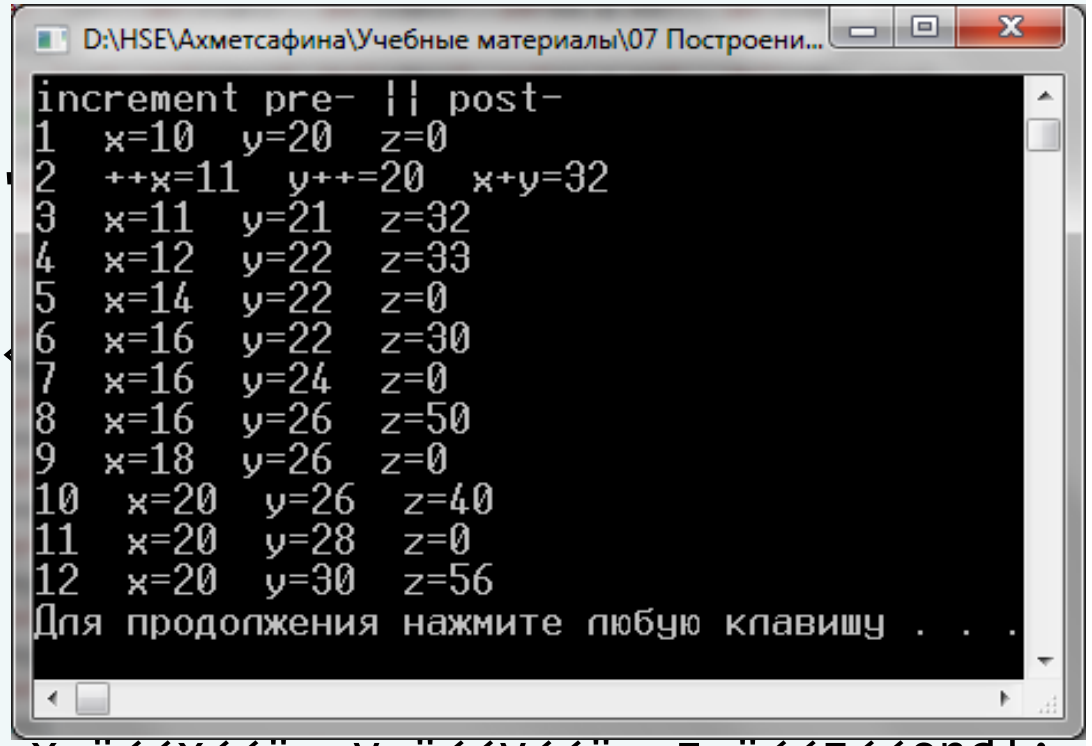


Пример упражнения 1

```
int x=10,y=20,z=0;
   cout<<"increment pre- || post-"<<"\n";
   cout<<"1  x="<<x<<"  y="<<y<<"  z="<< z << endl;
   cout<<"2  ++x="<<++x<<"  y++="<<y++<<"  x+y="<<x+y<< endl;
   z= x + y; cout<<"3  x="<<x<<"  y="<<y<<"  z="<<z<< endl;
   z= x++ + ++y; cout<<"4  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= ++x - x++; cout<<"5  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= ++x + x++; cout<<"6  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= y++ - ++y; cout<<"7  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= y++ + ++y; cout<<"8  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= ++x - ++x; cout<<"9  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= ++x + ++x; cout<<"10 x="<<x<<"  y="<<y<<" z="<<z<<endl;
   z= y++ - y++; cout<<"11 x="<<x<<"  y="<<y<<"  z="<<z<<endl;
   z= y++ + y++; cout<<"12  x="<<x<<" y="<<y<<"  z="<<z<<endl;
```

Пример упражнения 1

```
int x=10,y=20,z=0;
    cout<<"increment pre- |
    cout<<"1  x="<<x<<"  y="
    cout<<"2  ++x="<<++x<<"
    z= x + y; cout<<"3  x="
    z= x++ + ++y; cout<<"4
    z= ++x - x++; cout<<"5
    z= ++x + x++; cout<<"6
    z= y++ - ++y; cout<<"7
    z= y++ + ++y; cout<<"8  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
    z= ++x - ++x; cout<<"9  x="<<x<<"  y="<<y<<"  z="<<z<<endl;
                                                              ;
    z= y++ + y++; cout<<"12  x="<<x<<" y="<<y<<"  z="<<z<<endl;
```



Упражнение 1а – получить результаты в разных компиляторах.

Операции

Операции отношения: (<, <=, >, >=, ==, !=)

Результатом операций являются значения

`true (не 0)`,

`false (0)`.

Логические операции (&& и ||)

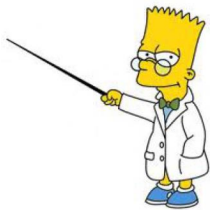
И (&&) - возвращает значение истина тогда и только тогда, когда оба операнда принимают значение истина, в противном случае операция возвращает значение ложь.

ИЛИ (||) - возвращает значение истина тогда и.т. тогда, когда хотя бы один операнд принимает значение истина, в противном случае – ложь

-логические операции выполняются слева направо;
-приоритет операции && выше ||.

Логические поразрядные операции
(&, ^, |)

```
int y = 7 & 17;
int z = 7 | 17;
int w = 7 ^ 17;
```



Приоритеты операций

1	**() [ ] -> :: .**	**9**	**&**
2	! ~ ++ -- - * & sizeof new delete typeid приведение типа	10	^
3	. * ->*	11	|
4	* / %	12	&&
5	+ -	13	||
6	>> <<	14	?:
7	< <= > >=	15	= += -= *= /= %= >>= <<= &= ^= |=
8	== !=	16	,

Начальные сведения

ОПЕРАТОРЫ

Операторы

Операторы – определяют действия программы на каждом шаге ее исполнения.

1. Оператор «выражение»

Любое выражение, заканчивающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается в вычислении этого выражения.

Частным случаем выражения является пустой оператор ; (точка с запятой).

Преобразование и приведение типов

Если в выражении смешаны различные типы, компилятор преобразует их к типу самого большого операнда пооперационно (расширение типа) - **неявное преобразование типа**.

Операция **явного приведения типа**

```
(<тип>) выражение;
```

```
int i = 7, N = 3;
cout << (double)   i / N ;
cout << (double)  (i / N);
cout << ((double)  i)/ N;

(double) i;
```

Тернарный оператор

2. Составные операторы

К составным операторам относят

- собственно составные операторы;
- блоки.

Блок - это последовательность операторов, заключенная в фигурные скобки.

```
{
        оператор1;
        оператор 2;
        …
}
```

Блок отличается от составного оператора наличием описаний / определений в теле блока.

3. Операторы ветвления

Условный оператор

```
if (выражение) инструкция;
else инструкция;
```

```
if (выражение)
        {
                последовательность инструкций;
        }
        else
        {
                последовательность инструкций;
        }
```

Задание:
если y = 5, то x = 8, в остальных случаях значение x не меняется.

Типичная ошибка

```
int y = 0, x = 3;
if (y = 5) x = 8;
```

3. Операторы ветвления

Условный оператор

Вложенная `else`-инструкция относится к ближайшей `if`-инструкции, которая находится внутри того же блока, но еще не связана ни с какой другой `else`-инструкцией.

```
if (i) {
       if (j) оператор1;
       if (k) оператор2;
       else оператор3;
        }
else оператор4;
```

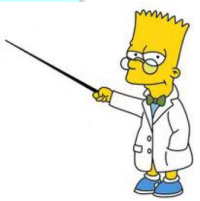
3. Операторы ветвления

Переключатель

```
switch (выражение) {
    case константа1:
            последовательность инструкций;
            break;
   case константа2:
            последовательность инструкций;
            break;
   …
       default:
             последовательность инструкций;
}
```

Попробуйте убрать **break**, использовать одинаковые константы, список констант и т.п.

Упражнение 2 – написать программу перевода 10-балльной оценки в 5-балльную. УПРАЖНЕНИЕ НАДО ВЫПОЛНИТЬ!!!



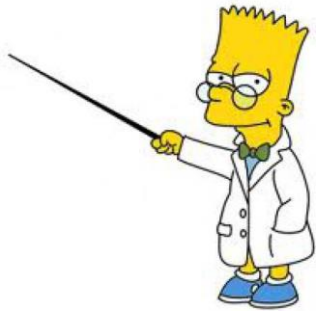
4. Операторы циклов

Цикл с параметром

```
for ( инициализации; условия; список выражений )
инструкция;
```

```
for (инициализации; условия; список выражений )
     {
            последовательность инструкций;
      }
```

В инициализации можно писать операторы



4. Операторы циклов

Цикл с параметром, примеры

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

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

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

4. Операторы циклов

Цикл с предусловием:

```
while (выражение) инструкция;
```

```
while (выражение)
{
  последовательность инструкций;
}
```

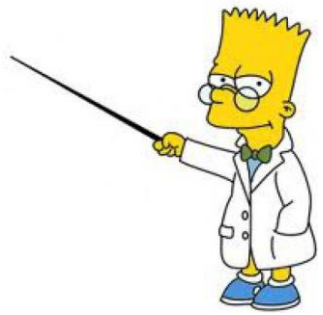
4. Операторы циклов

Цикл с постусловием

```
do  {
            инструкции;
}
while (выражение);
```

В этом операторе цикла инструкции выполняются всегда хотя бы один раз.

ВНИМАНИЕ!!! В псевдокодах используется
`repeat … until` (выражение),
в котором цикл выполняется по **`false.`**
В программах на C++, цикл выполняется по **`true.`**



5. Операторы перехода

Операторы перехода выполняют безусловную передачу управления.

- **`break`** – оператор прерывания цикла.
- **`continue`** – переход к следующей итерации цикла.
- **`return`** – оператор возврата из функции. Он всегда завершает выполнение функции и передает управление в точку ее вызова.
- **`goto`** <**`метка`**> – передает управление оператору, который содержит метку. В теле той же функции должна присутствовать конструкция:

<**`метка`**> : **`оператор`**;