

# Программирование в C++

Часть 1

# Программирование

Для представления целостного взгляда на "программирование" предлагается разделить на главное (фундаментальное) и второстепенное (заменяемое):

- фундаментальное (типы данных, структуры данных, программные управляющие конструкции, алгоритмы)
- Заменяемое: языки программирования технологии)

# Программа

## Что такое программа?

Описать процесс – это значит определить последовательность состояний заданной информационной среды. Если мы хотим, чтобы по заданному описанию требуемый процесс порождался *автоматически* на каком-либо компьютере, необходимо, чтобы это описание было *формализованным*. Такое описание называется *программой*.

# Программа

Что такое программа?

- С точки зрения пользователя

(пользователем – потребителем может быть так же процесс)

Изменение начальных (входных) данных до конечного ожидаемого\* результата

# Программа

Что такое программа?

- С точки зрения программиста  
Набор обобщенных\* операторов, выстроенных в определенном порядке, изменяющих информационное пространство (набор мест хранения и значения хранимых данных) по определенному алгоритму.

# Программа

Что такое программа?

- С точки зрения компьютера

Набор адресных мест хранения (временных и постоянных) доступных для изменения с заданным порядком обращения (чтение, запись, исполнение).  
Регистры, стек.

# Из чего состоят данные

Содержимое мест хранения (временное, постоянное) до применения полного набора команд, во время применения отдельной команды, и после полного применения полного набора команд.

# Из чего состоит программа?

- Набор данных (данные)
- Набор действий (список команд)
- Порядок выполнения команд (алгоритм)

# Из чего состоит команда

- Описание процесса изменения данных на языке программирования (синтаксис)

# Из чего состоит алгоритм?

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи **за конечное число действий**.

В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Это связано с тем, что работа каких-то инструкций алгоритма может быть зависима от других инструкций или результатов их работы. Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшие независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые процессор и операционная система.

# Структуры данных

- Физическая структура – способ физического представления данных в памяти машины (структура хранения, внутренняя структура, структура памяти – синонимы).
- Логическая (абстрактная) структура – структура данных без учета ее представления в машинной памяти.

# Структуры данных

- Процедуры отображения логической структуры в физическую
  - Процедуры отображения физической структуры в логическую
- \* Эти процедуры обеспечивают доступ к физическим структурам, выполнение над ними операций.
- \*\* Применительны к обеим видам структур.

# Структуры данных

- Простые (базовые, примитивные) – не могут быть расчленены на составные части, большие чем биты.
- Интегрированные (структурированные, композитные, сложные) – состоят из простых и/или интегрированных структур

# Структуры данных

Отсутствие или наличие явно заданных связей.

**Несвязные структуры** – векторы, массивы, строки, стеки, очереди.

**Связные структуры** – связные списки.

# Структуры данных

Изменчивость – изменение числа элементов и/или связей между элементами структуры.

- Статические
- Полустатические
- Динамические

# СТРУКТУРЫ ДАННЫХ

Простые базовые

Статические

Полустатические

Динамические

Числовые

Вектор

Стеки

Линейные  
связные списки

Символьные

Массивы

Очереди

Разветвленные  
связные списки

Логические

Множества

Деки

Графы

Перечисления

Записи

Строки

Деревья

Интервал

Таблицы

Указатели

Файловые

Последовательные

Прямого доступа

Комбинированного доступа

Организованные разделами

Для  
внешней  
памяти

Оперативные структуры

# Структуры данных

Простые  
базовые

Числовые

Символьные

Логические

Перечисления

Интервал

Указатели

# Структуры данных

Статические

Вектор

Вектор – структура данных с фиксированным количеством элементов одного и того же типа

Массивы

Массив – последовательность элементов одного базового типа \*

Множества

Множество – набор неповторяющихся данных одного и того же типа

Записи

Запись – конечное упорядоченное множество полей, характеризующихся различным типом данных

Таблицы

Таблица – последовательность записей, которые имеют одну и ту же организацию \*\*

\* Массив – вектор с индексами

элементов  
\*\* Таблица – записи с индексами

записей

# Структуры данных

Полустатические

Стеки

Очереди

Деки

Строки

Последовательность, в которой включение и исключение элемента происходит с одной

стороны

Последовательность, в которую включают элементы с одной стороны, а исключают - с другой

Последовательность, в которой включение и исключение элементов происходит с двух стороны

**LIFO**

**FIFO**

**Строчные структуры – одномерные, динамические изменяемые структуры данных, различающиеся способами включения и исключения элементов**

# Структуры данных

Списковые структуры – логический порядок данных определяется указателями.

Каждый элемент состоит из двух полей:

- Элемент данных или указатель на него
  - Указатель на следующий элемент списка
- Список – упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения и исключения. Линейный список – отражающий отношения соседства между элементами.**

# Структуры данных

Динамические

Линейные  
связные списки

Разветвленные  
связные списки

Графы

Деревья

Связный список – структура данных, элементами которой являются записи, связанные между собой указателями, хранящимися в самих элементах.

Граф – совокупность двух множеств: вершин и ребер.

Дерево – совокупность элементов, называемых *узлами* (один из которых определен как *корень*), и отношений, образующих иерархическую структуру узлов.

# Структуры данных



# Структуры данных

**Линейные структуры** – структуры, в которых связи элементов не зависят от выполнения какого-либо условия

- Строчные структуры
- Стек
- Очередь
- Дек

# Структуры данных

**Нелинейные структуры** – у которых связи между элементами зависят от выполнения определенного условия.

- Графы
- Деревья
- Плексы  
(сплетения)

# Структуры данных

## Структуры данных в оперативной памяти

Полуслово

Слово

Двойное  
слово

Схемы хранения

Прямоугольные

Связные

# Структуры данных

## Структуры данных во внешней памяти

Физический блок

Схемы хранения

Прямого доступа

Последовательного доступа

Индексно - последовательного  
доступа

# Информация по каждому типу данных однозначно определяет:

- 1) Структуру хранения данных указанного типа, т.е. выделение памяти и представление данных в ней и интерпретирование двоичного представления;
- 2) Множество допустимых значений, которые может иметь тот или иной объект описываемого типа;
- 3) Множество допустимых операций, которые применимы к объекту описываемого типа;

Над всеми структурами и типами данных должны

## **обязательно**

должны выполняться следующие операции:

- **Операция создания** заключается в выделении памяти для структуры данных;
- **Операция уничтожения** противоположна операции создания. Помогает эффективно использовать память;
- **Операция выбора** используется для доступа к данным внутри самой структуры. *Метод доступа* – одно из наиболее важных свойств структуры, потому что имеет непосредственное отношение к выбору конкретной структуры данных;
- **Операция обновления** позволяет изменить значения данных в структуре данных. Примеры: операция присваивания и передача параметров.

- Процедуры отображения логической структуры в физическую
- Процедуры отображения физической структуры в логическую
- Операция создания;
- Операция уничтожения;
- Операция выбора;
- Операция обновления.

**Определяют выбор языка  
программирования**

# Типы данных

Элементы структур данных состоят из простых базовых типов.

Любая программа оперирует с переменными и константами определенного типа данных.

# Таблица типов

Название типа	Н.Г. диапазона	В.Г. диапазона	Точность	Размер (байт)
bool	False	True	нет	1
char	-128	127	нет	1
short	-32 768	32 767	нет	2
int	-2 147 483 648	2 147 483 647	нет	4
long	-2 147 483 648	2 147 483 647	нет	4
float	$3,4 \cdot 10^{-38}$	$3,4 \cdot 10^{38}$	7	4
double	$1,7 \cdot 10^{-308}$	$1,7 \cdot 10^{308}$	15	8
long double	$1,7 \cdot 10^{-308}$	$1,7 \cdot 10^{308}$	15	8

signed        - **знаковый тип**  
unsigned     - **беззнаковый тип**

# Основные типы языка C++ \*

Категория	Тип	
Целые числа	char	
	bool	
	short	
	int	
	__int8, __int16, __int32, __int64, __int128	
	long	
	long long	
	wchar_t, __wchar_t	
	С плавающей запятой	float
		double
long double		

\* - подробности в раздаточном материале

# Размеры основных типов

Тип	Размер
<code>bool, char, unsigned char, signed char, __int8</code>	1 байт
<code>__int16, short, unsigned short, wchar_t, __wchar_t</code>	2 байта
<code>float, __int32, int, unsigned int, long, unsigned long</code>	4 байта
<code>double, __int64, long double, long long</code>	8 байт
<code>__int128</code>	16 байт

# Типы данных

**bool - требуется всего 1**

**бит!!!!**

**false , true**

## Размерность – 1

**байт\*** фактически размер переменной типа bool равен 1 биту (не байту!), но большинство компиляторов на практике выделяет под такие переменные 1 байт, поскольку доступ к целому байту осуществляется быстрее, чем к отдельному биту. Чтобы получить доступ к биту, необходимо произвести операцию его извлечения из того байта, в котором он содержится, что увеличивает время доступа.

- **ОБЪЯВЛЕНИЕ ПЕРЕМЕННОЙ – УКАЗАНИЕ ИМЕНИ ПЕРЕМЕННОЙ И ЕЕ ТИПА.**
- **ОПРЕДЕЛЕНИЕ ПЕРЕМЕННОЙ – УКАЗАНИЕ ИМЕНИ ПЕРЕМЕННОЙ С ВЫДЕЛЕНИЕМ ПАМЯТИ ПОД**

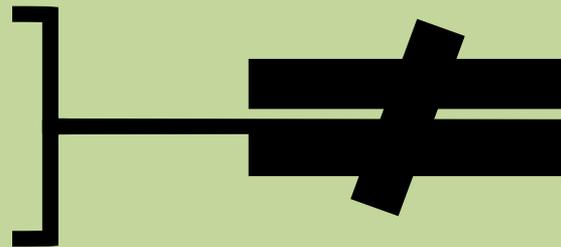
```
int var1;  
int var2;
```

# Имена

var1, var2 – идентификаторы  
**Переменных**

var1 и var1 – различные  
идентификаторы

TempDate –  
идентификатор;  
tempDate –  
идентификатор;



**НЕЛЬЗЯ ИСПОЛЬЗОВАТЬ КЛЮЧЕВЫЕ  
СЛОВА!!**

asm; auto; bool; break; case; catch; char; class; const; const\_cast; continue; default;  
delete; do; double; dynamic\_cast; else; enum; explicit; export; extern; false; float; for;  
friend; goto; if; inline; int; long; main; mutable; namespace; new; operator; private;  
protected; public; register; reinterpret\_cast; **return**; short; signed; sizeof; static;  
static\_cast; struct; switch; template; this; true; try; typedef; typeid; typename; union;  
unsigned; using; virtual; **void**; volatile; wchar\_t; **while**

# Переменные целого

типа;  
var1 = 1;

var2 = var1 + 2;

## Целые

**КОНСТАНТЫ**

"1" – константа целого

типа;

## Оператор

**ПРИСВАИВАНИЕ**

« = »

**ВЫВОД**

cout << "var1 + 2 равно

**Д**

cout << var2 << '\n';

cout << var2 << endl;

# Другие целые

**типы** размер аппаратно-зависимый

long, short – фиксированный

размер \*

long, short



int – для 32-разрядных

long



long – для 16-разрядных

int



short – для 16-разрядных

**СИСТЕМ**

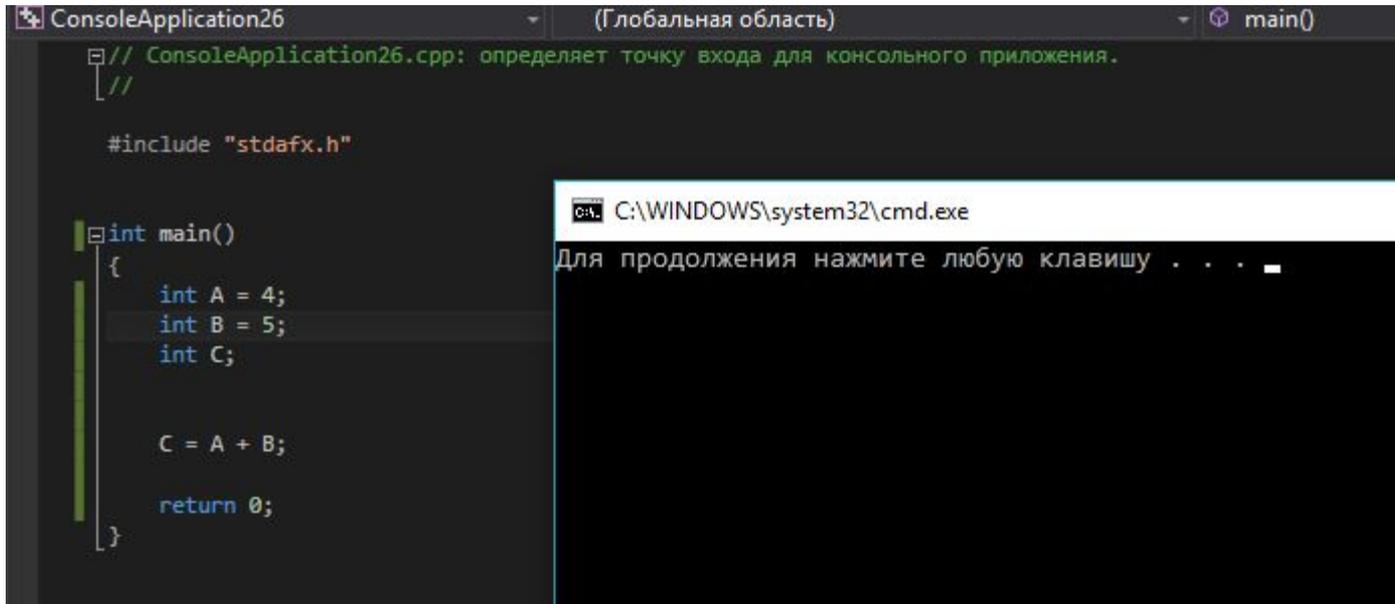
\* Размер типа long всегда равен 4 байтам и совпадает с размером типа int в случае 32-разрядных систем, подобных Windows. Это означает, что диапазон значений типа long совпадает с диапазоном типа int: от -2 147 483 648 до 2 147 483 647. Тип long может быть описан как long int между двумя такими описаниями нет разницы. Если ваша операционная система 32-разрядная, то не важно, какой тип использовать — int или long, но в 16-разрядной системе тип long сохранит свой диапазон значений, в то время как тип int уже будет иметь диапазон, совпадающий с типом short.

Тип short в любой операционной системе имеет размер, равный двум байтам. Диапазон значений типа short — от -32 768 до 32 767. Использование типа short не имеет особого смысла на современных 32-разрядных операционных системах, за исключением тех случаев, когда необходима экономия памяти. Несмотря на вдвое больший размер по сравнению с типом short, обработка переменных типа int происходит быстрее

# Структура программы

```
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.  
//  
#include "stdafx.h"  
  
int main()  
{  
    return 0;  
}
```

# Структура программы



The image shows a screenshot of a C++ IDE with two windows. The top window, titled 'ConsoleApplication26', displays the source code for a simple console application. The code includes a comment in Russian, an include directive for 'stdafx.h', and a 'main' function that declares three integer variables (A, B, and C), assigns values to A and B, calculates the sum of A and B into C, and returns 0. The bottom window, titled 'C:\WINDOWS\system32\cmd.exe', shows the command prompt output: 'Для продолжения нажмите любую клавишу . . .', which is a standard prompt for a console application to wait for user input before exiting.

```
ConsoleApplication26 (Глобальная область) main()
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"

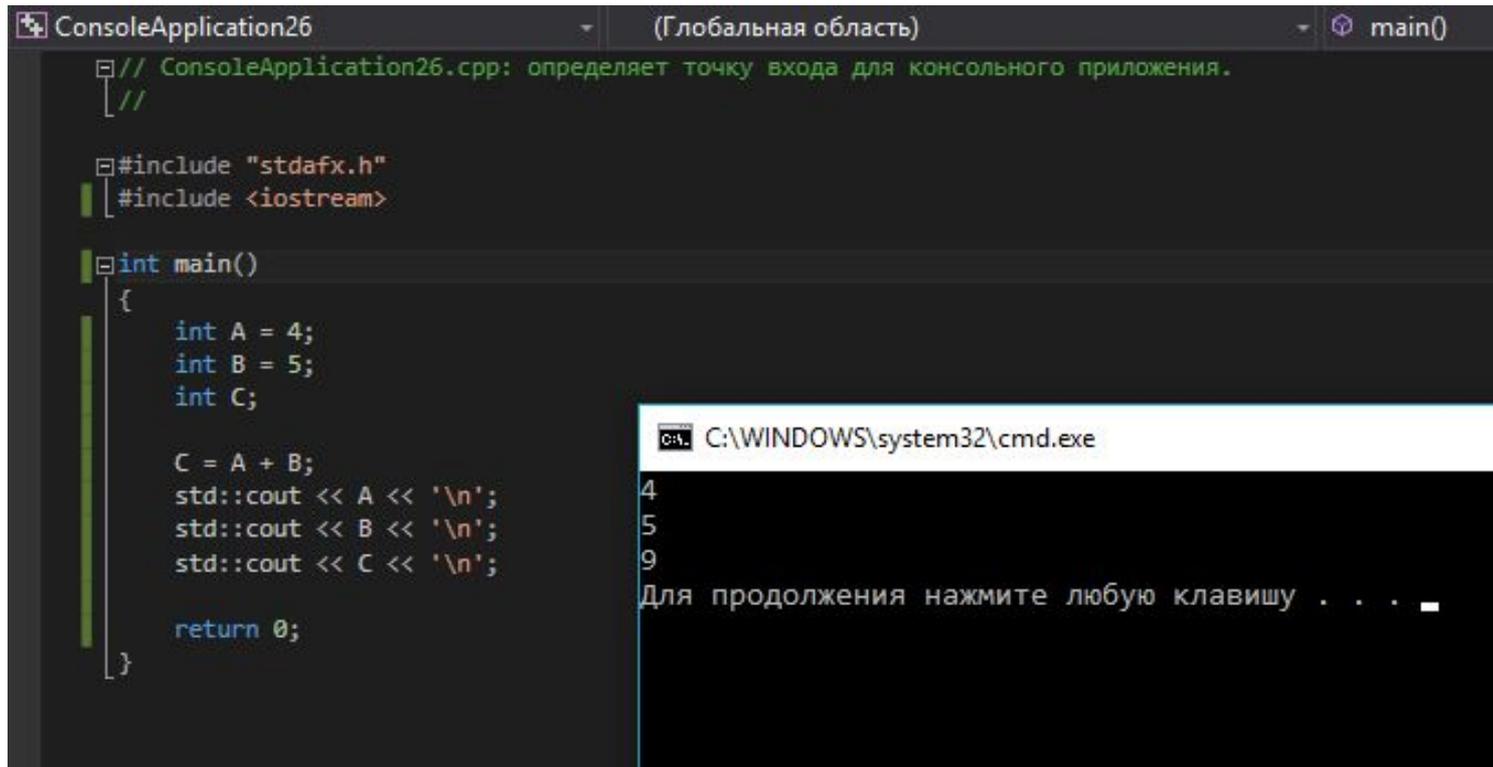
int main()
{
    int A = 4;
    int B = 5;
    int C;

    C = A + B;

    return 0;
}
```

C:\WINDOWS\system32\cmd.exe  
Для продолжения нажмите любую клавишу . . .

# Структура



The image shows a screenshot of a C++ IDE. The main window displays the source code for a console application named 'ConsoleApplication26'. The code defines a `main()` function that declares three integer variables: `A = 4`, `B = 5`, and `C`. It then calculates `C = A + B` and prints the values of `A`, `B`, and `C` to the console using `std::cout`. The output window shows the results: `4`, `5`, and `9`, followed by a prompt for the user to press any key to continue.

```
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <iostream>

int main()
{
    int A = 4;
    int B = 5;
    int C;

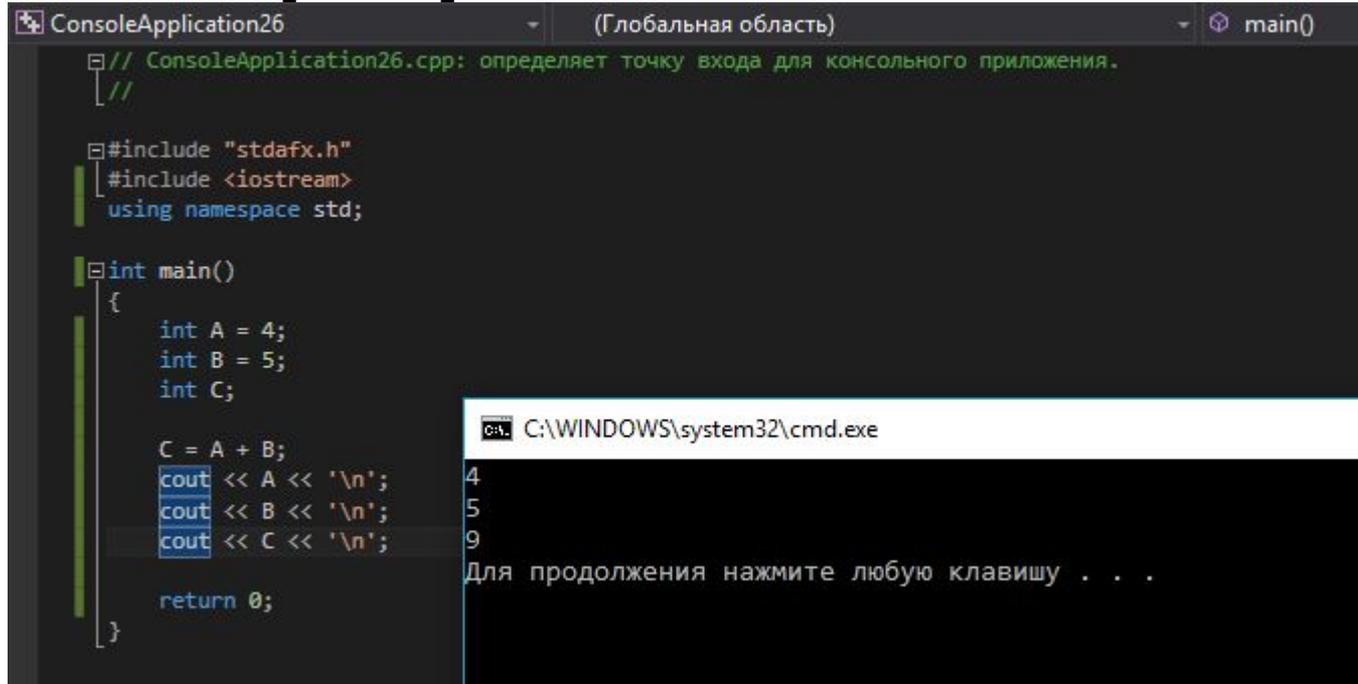
    C = A + B;
    std::cout << A << '\n';
    std::cout << B << '\n';
    std::cout << C << '\n';

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
4
5
9
Для продолжения нажмите любую клавишу . . .
```

**`std :: cout << A << '\n';`**

# Структура программы



The image shows a screenshot of a C++ development environment. The top part displays the source code for a console application named 'ConsoleApplication26'. The code includes headers, uses the 'std' namespace, and defines a 'main' function. Inside 'main', variables A, B, and C are declared. A is assigned 4, B is assigned 5, and C is assigned the sum of A and B (9). The program then prints the values of A, B, and C on separate lines. The bottom part of the screenshot shows the command prompt output, which matches the program's output: 4, 5, and 9, followed by a prompt for the user to press a key to continue.

```
ConsoleApplication26 (Глобальная область) main()
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
using namespace std;

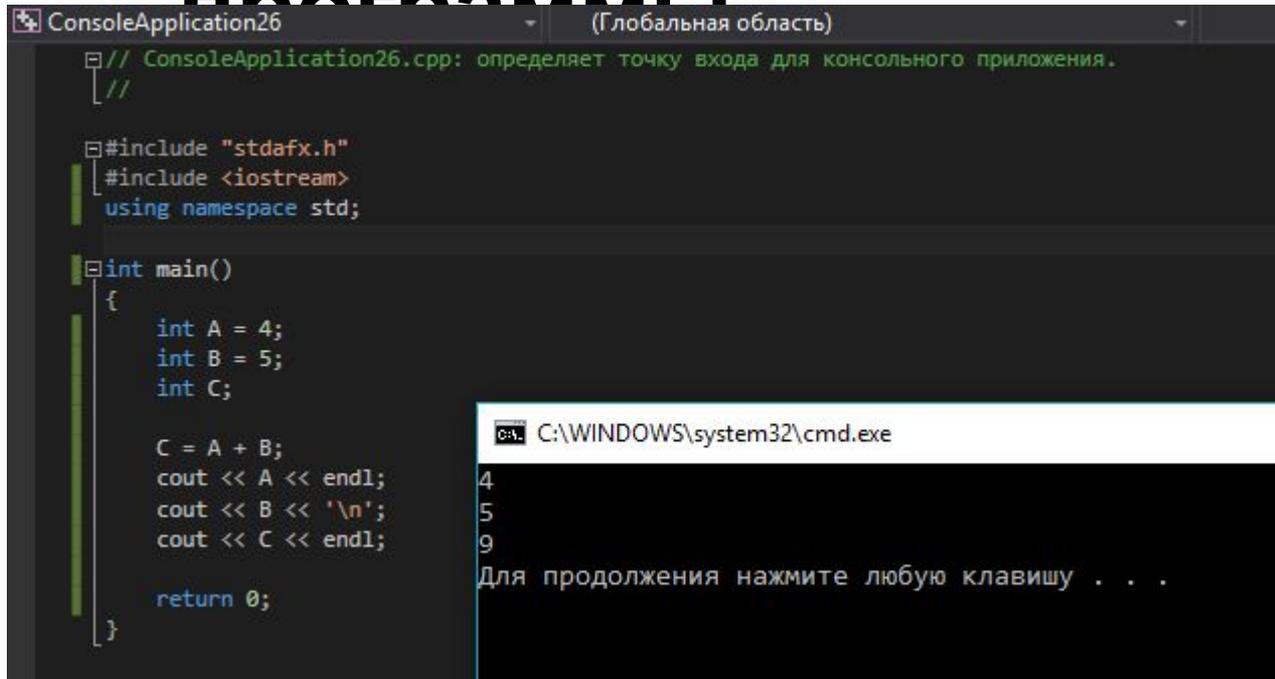
int main()
{
    int A = 4;
    int B = 5;
    int C;

    C = A + B;
    cout << A << '\n';
    cout << B << '\n';
    cout << C << '\n';

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
4
5
9
Для продолжения нажмите любую клавишу . . .
```

# Структура ПРОГРАММЫ



The image shows a screenshot of a C++ IDE window titled 'ConsoleApplication26'. The code in the editor is as follows:

```
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.  
//  
  
#include "stdafx.h"  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int A = 4;  
    int B = 5;  
    int C;  
  
    C = A + B;  
    cout << A << endl;  
    cout << B << '\n';  
    cout << C << endl;  
  
    return 0;  
}
```

Below the code editor, a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' shows the output of the program:

```
4  
5  
9  
Для продолжения нажмите любую клавишу . . .
```

**'\n' = endl**

**Манипулятор endl связан с очисткой выходного буфера**

# Структура

программы

`#include` – директива

препроцессора

`#include <iostream>`

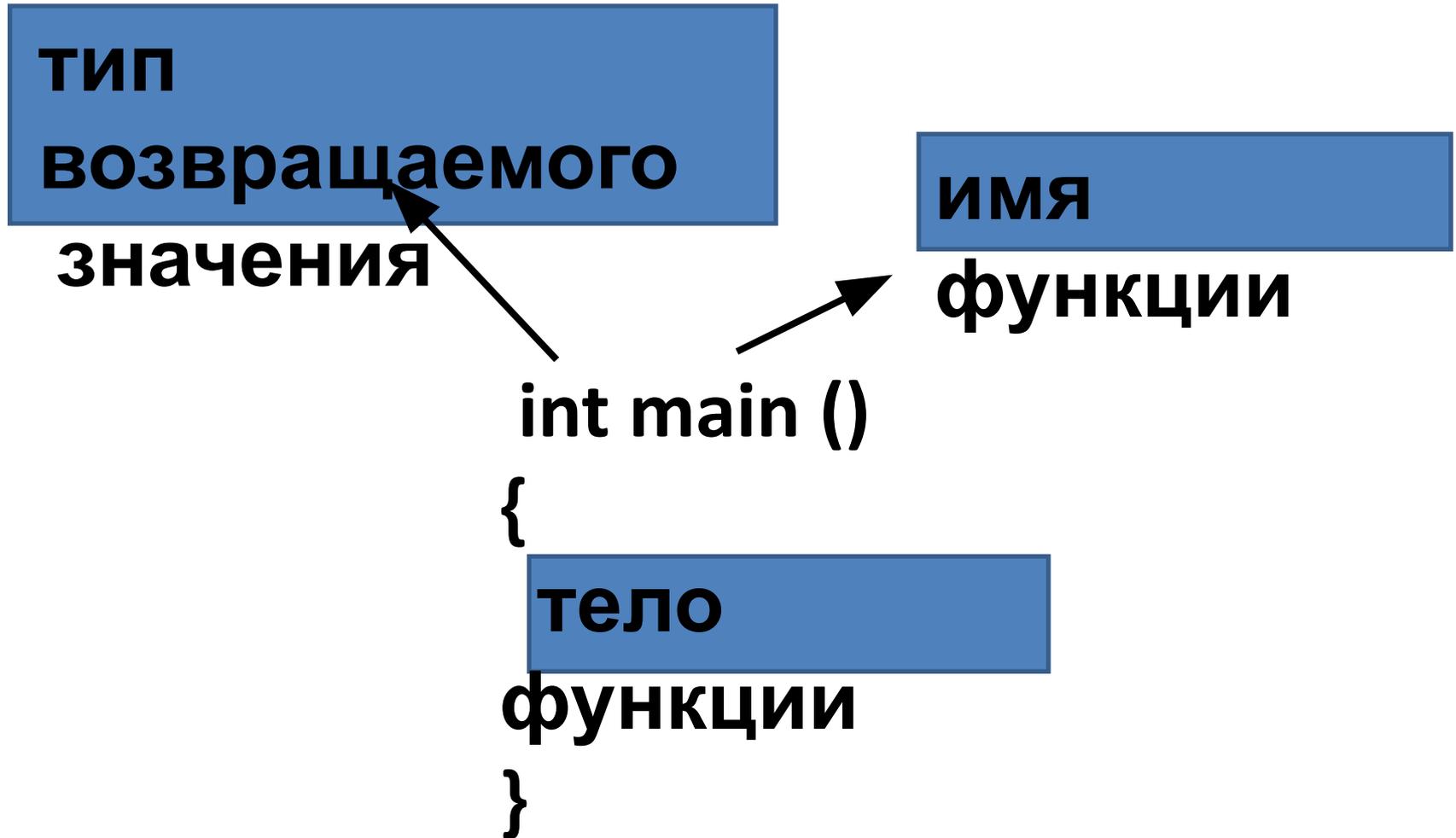
`iostream = iostream.h`

C++

C

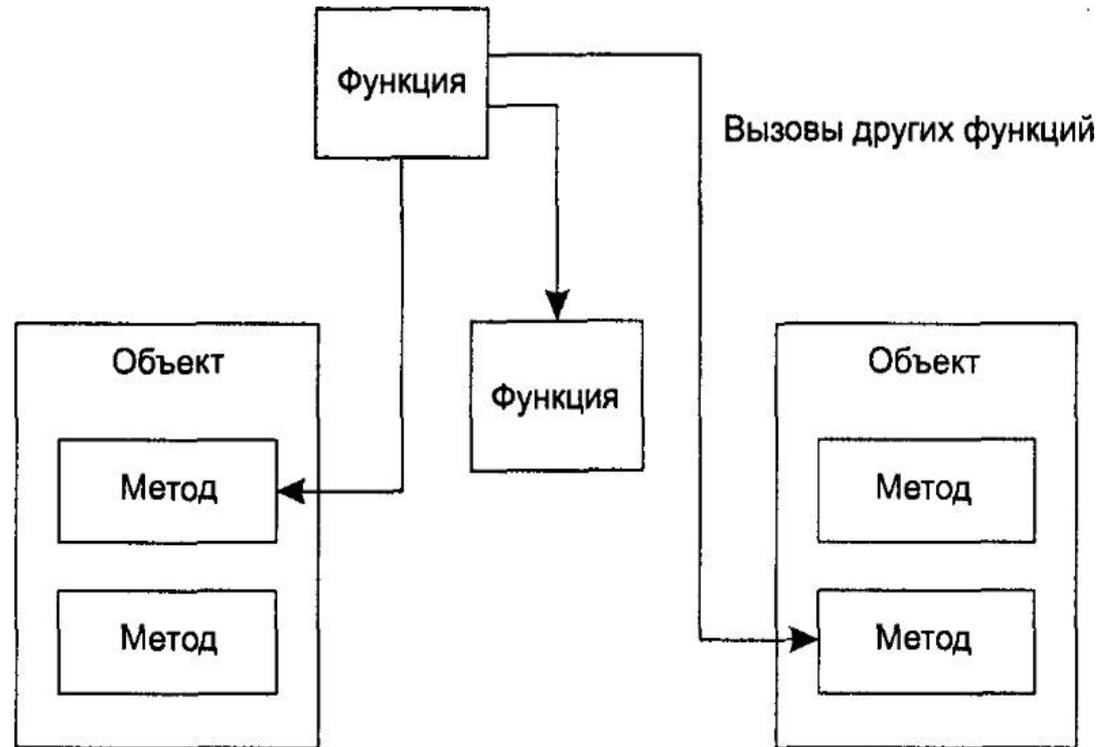
`using namespace std;` -  
директива

# Структура программы



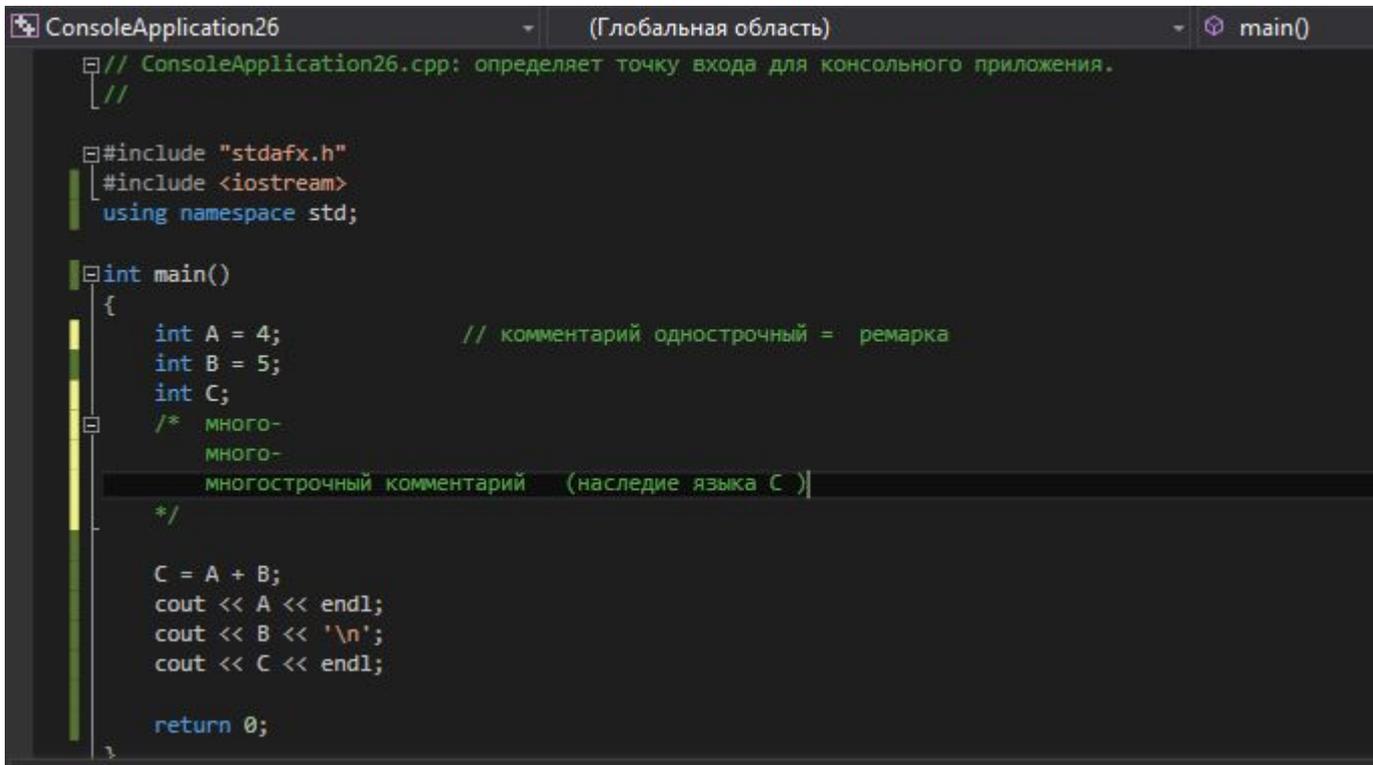
# Структура программы

функция  
main()



# Структура программы

## написание комментариев



```
ConsoleApplication26 (Глобальная область) main()
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    int A = 4;           // комментарий однострочный = ремарка
    int B = 5;
    int C;
    /* много-
    много-
    многострочный комментарий (наследие языка C) */

    C = A + B;
    cout << A << endl;
    cout << B << '\n';
    cout << C << endl;

    return 0;
}
```

# Символьные переменные

<code>char</code>	от-127 до 128	1 байт
ASCII	0 - 255	<code>wchar_t</code>

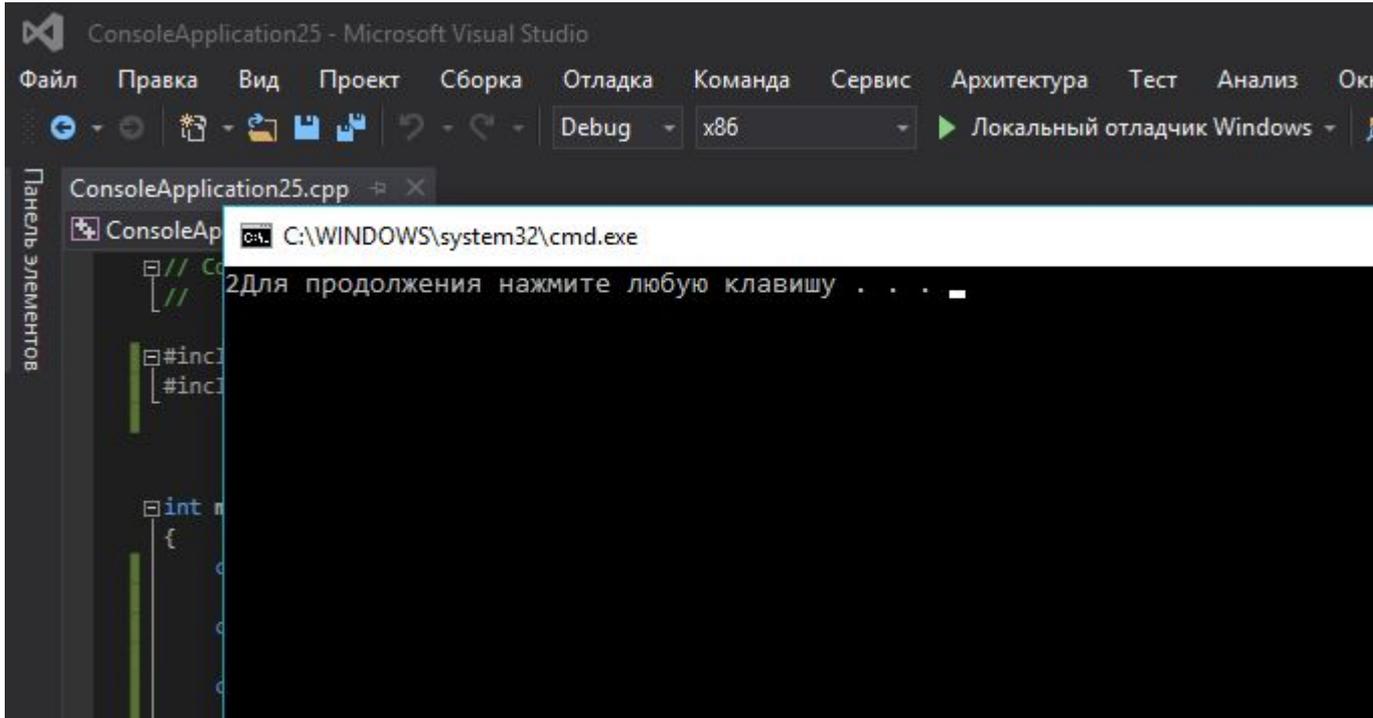


# Символьные константы

```
char var1 = '2';  
char var2;  
var2 = '2';
```

```
#include "stdafx.h"  
#include <iostream>
```

```
int main()  
{  
    char temp1;  
  
    char temp_1;  
  
    char Temp1;  
  
    char temP_1 = '+';  
  
    temp1 = '2';  
  
    std::cout << temp1;  
  
    return 0;  
}
```



```
ConsoleApplication25 (Глобальная область) main()
// ConsoleApplication25.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <iostream>

int main()
{
    char temp1;
    char temp_1;
    char Temp1;

    char temp_1 = '\n';

    temp1 = '22';
    temp_1 = '\t';

    std::cout << temp_1 << temp_1 << temp1;

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe

Для продолжения нажмите любую клавишу . . .
```

```
ConsoleApplication25 (Глобальная область)
// ConsoleApplication25.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>

int main()
{
    char temp1;
    char temp_1;
    char Temp1;

    char temp_1 = '\n';

    // temp1 = '22';
    temp1 = '\x32';
    temp_1 = '\t';

    std::cout << temp_1 << temp_1 << temp1;

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe

2Для продолжения нажмите любую клавишу . . .
```

# Каскадирование операции

“<<”

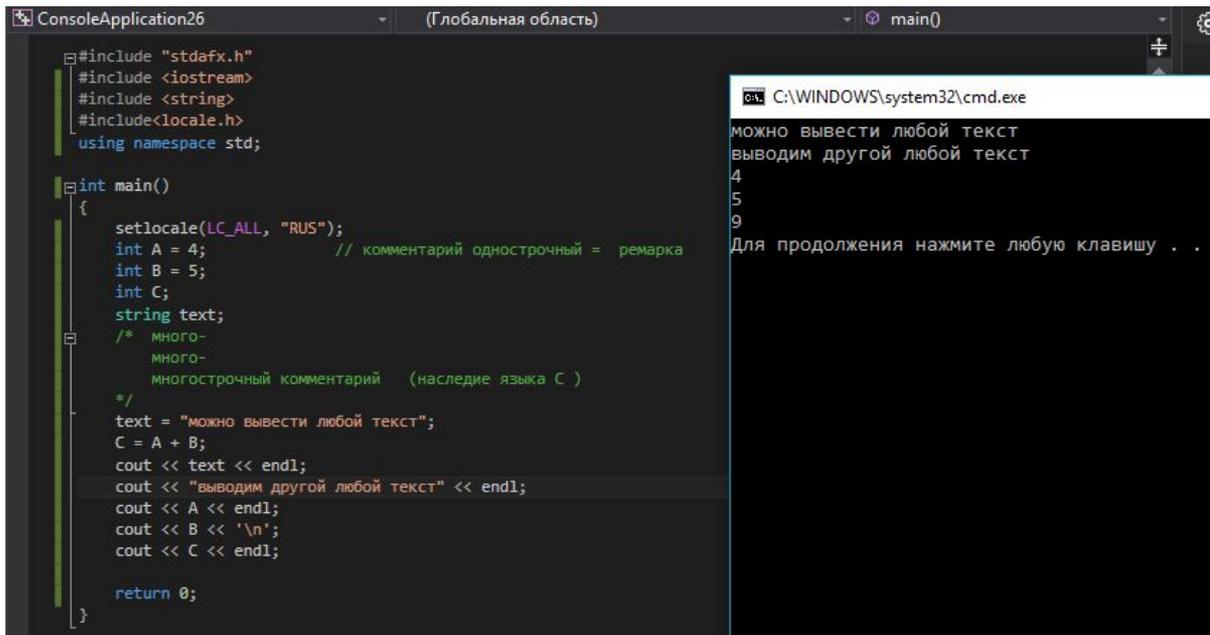
# Управляющие

## последовательности (escape – последовательности)

Управляющая последовательность	Символ
\a	Сигнал
\b	Возврат на одну позицию
\f	Перевод страницы
\n	Перевод в начало следующей строки
\r	Возврат каретки
\t	Табуляция горизонтальная
\v	Табуляция вертикальная
\\	Обратная косая черта
\'	Одинарные кавычка
\"	Двойные кавычка
\?	Вопросительный знак
\xdd	Шестнадцатеричный код символа

# Строковые переменные и константы

“ #include <string> ”



The screenshot shows a C++ IDE window titled 'ConsoleApplication26' with a file named 'main()'. The code in the editor includes headers for `stdafx.h`, `iostream`, `string`, and `locale`, and uses the `std` namespace. The `main` function sets the locale to Russian, declares variables `A`, `B`, and `C`, and a `string` variable `text`. It then outputs the text, the sum of `A` and `B`, and the values of `A`, `B`, and `C` on separate lines. A multi-line comment is also present. To the right, a terminal window shows the program's output: 'можно вывести любой текст', 'выводим другой любой текст', '4', '5', '9', and a prompt to press any key to continue.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int A = 4;           // комментарий однострочный = ремарка
    int B = 5;
    int C;
    string text;
    /* много-
    много-
    многострочный комментарий (наследие языка C)
    */
    text = "можно вывести любой текст";
    C = A + B;
    cout << text << endl;
    cout << "выводим другой любой текст" << endl;
    cout << A << endl;
    cout << B << '\n';
    cout << C << endl;

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
можно вывести любой текст
выводим другой любой текст
4
5
9
Для продолжения нажмите любую клавишу . . .
```

#include<locale.h>

...

setlocale(LC\_ALL,"RUS");

...

# Ввод

# cin

```
ConsoleApplication26 (Глобальная область) main()
#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int A = 4; // комментарий
    int B;
    int C;
    string text;
    text = "введите значение B ";
    cout << text << endl;
    cin >> B;
    C = A + B;
    cout << A << endl;
    cout << B << '\n';
    cout << C << endl;

    return 0;
}
```

cmd.exe. Выбрать C:\WINDOWS\system32\cmd.exe

```
введите значение B
6
4
6
10
Для продолжения нажмите любую клавишу . . .
```

# **Структура программы**

**Выражение – определяет совокупность  
вычислений**

**Операторы – указание компьютеру  
выполнить какое-либо действие**

**В одном операторе могут  
присутствовать несколько выражений**

# Структура

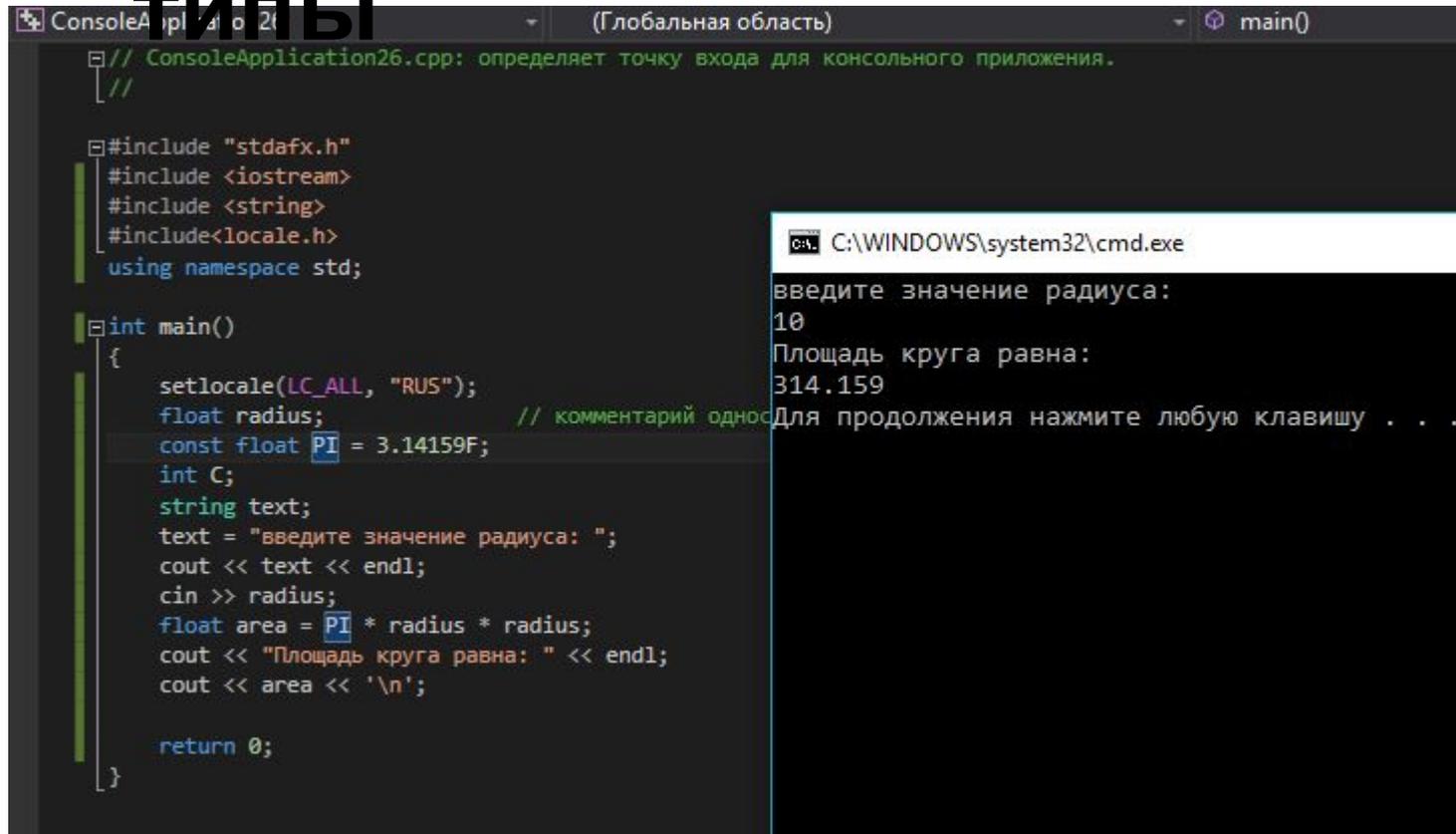
## программы

Вещественные типы: **float, double, long double**

Переменные вещественного типа предназначены для хранения вещественных чисел — тех чисел, которыми измеряются непрерывные величины: температура, расстояние, площадь. Вещественные числа, как правило, имеют ненулевую дробную часть.

# Вещественные

## ТИПЫ



```
// ConsoleApplication26.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    float radius; // комментарий однос
    const float PI = 3.14159F;
    int C;
    string text;
    text = "введите значение радиуса: ";
    cout << text << endl;
    cin >> radius;
    float area = PI * radius * radius;
    cout << "Площадь круга равна: " << endl;
    cout << area << '\n';

    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
введите значение радиуса:
10
Площадь круга равна:
314.159
Для продолжения нажмите любую клавишу . . .
```

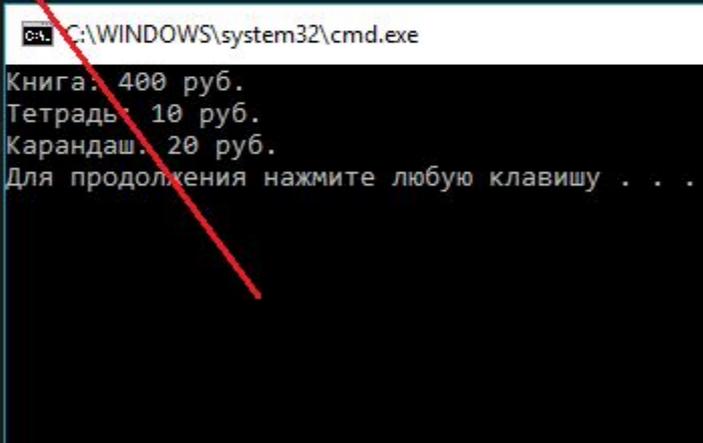
**Префикс const гарантирует, что программа не сможет случайно изменить значение переменной**

# Вывод результатов

```
ConsoleApplication26 (Глобальная область) main()
//
#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int cost1 = 400 ;
    int cost2 = 10;
    int cost3 = 20;
    string text1;
    string text2;
    text1 = "Цена товаров: ";
    text2 = " руб.";
    cout << "Книга: " << cost1 << text2 << endl;
    cout << "Тетрадь: " << cost2 << text2 << endl;
    cout << "Карандаш: " << cost3 << text2 << endl;

    return 0;
}
```



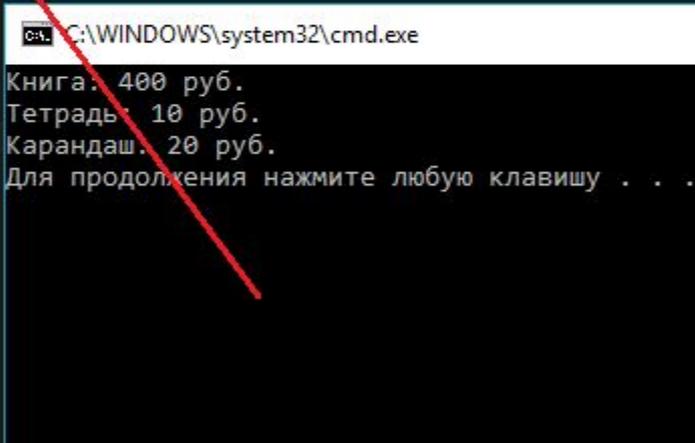
```
C:\WINDOWS\system32\cmd.exe
Книга: 400 руб.
Тетрадь: 10 руб.
Карандаш: 20 руб.
Для продолжения нажмите любую клавишу . . .
```

# Вывод результатов

```
ConsoleApplication26 (Глобальная область) main()
//
#include "stdafx.h"
#include <iostream>
#include <string>
#include <locale.h>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int cost1 = 400 ;
    int cost2 = 10;
    int cost3 = 20;
    string text1;
    string text2;
    text1 = "Цена товаров: ";
    text2 = " руб.";
    cout << "Книга: " << cost1 << text2 << endl;
    cout << "Тетрадь: " << cost2 << text2 << endl;
    cout << "Карандаш: " << cost3 << text2 << endl;

    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
Книга: 400 руб.
Тетрадь: 10 руб.
Карандаш: 20 руб.
Для продолжения нажмите любую клавишу . . .
```

# Вывод

## результатов

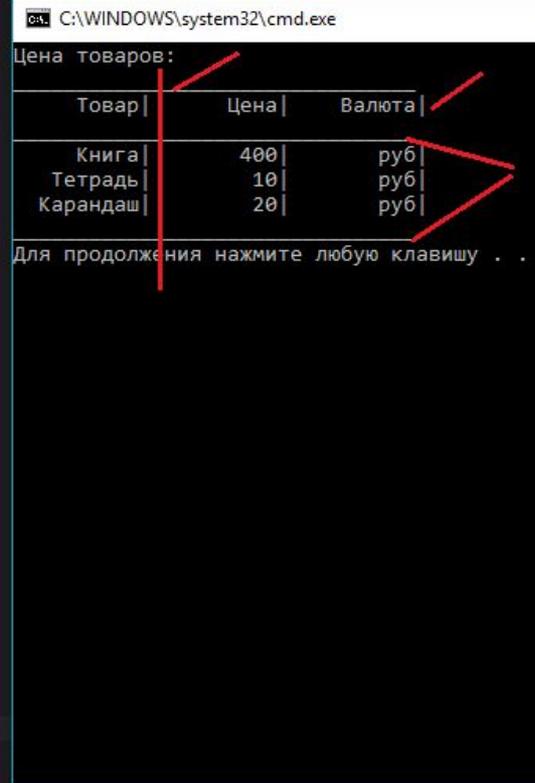
```
ConsoleApplication26 (Глобальная область) main()
#include <string>
#include <locale>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int cost1 = 400;
    int cost2 = 10;
    int cost3 = 20;
    string text1;
    string text2;
    string text3;
    string text4;
    string line;
    char razd = '|';

    text1 = "Цена товаров: ";
    line = "_____"; // 30 "подчеркиваний"
    text2 = "руб";
    text3 = "Товар";
    text4 = "Цена";

    cout << text1 << endl;
    cout << line << endl;
    cout << setw(10) << "Товар" << razd << setw(10) << "Цена" << razd << setw(10) << "Валюта" << razd << endl;
    cout << line << endl;
    cout << setw(10) << "Книга" << razd << setw(10) << cost1 << razd << setw(10) << text2 << razd << endl;
    cout << setw(10) << "Тетрадь" << razd << setw(10) << cost2 << razd << setw(10) << text2 << razd << endl;
    cout << setw(10) << "Карандаш" << razd << setw(10) << cost3 << razd << setw(10) << text2 << razd << endl;
    cout << line << endl;

    return 0;
}
```



**#include<iomanip>**  
**setw(N) N – размерность**

**«ПОДЯ»**

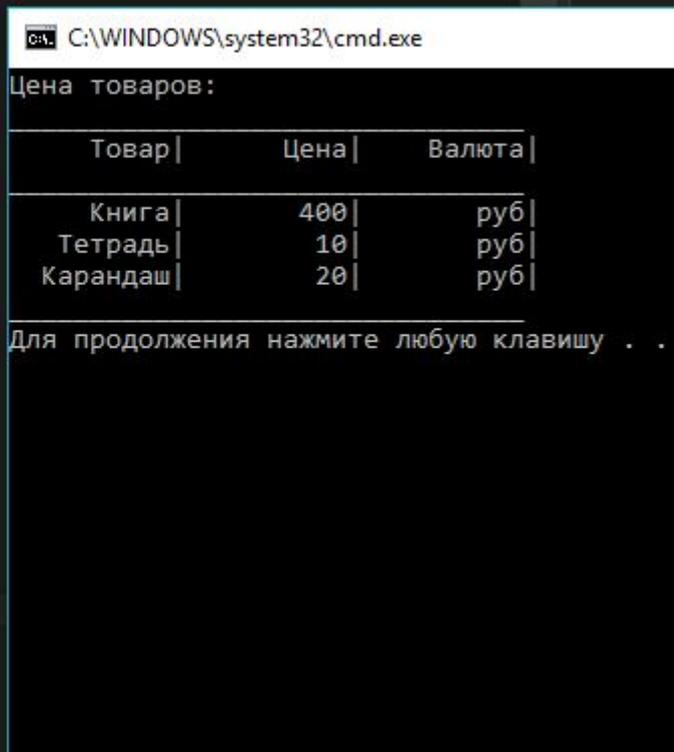
# Вывод

## РЕЗУЛЬТАТ

```
ConsoleApplication26 (Глобальная область) main()
int main()
{
    setlocale(LC_ALL, "RUS");
    //int cost1 = 400 ;
    //int cost2 = 10;
    //int cost3 = 20;
    int cost1 = 400, cost2 = 10, cost3 = 20;
    //string text1;
    //string text2;
    //string text3;
    //string text4;
    //string line;
    char razd = '|';

    //text1 = "Цена товаров: ";
    //line = "_____"; // 30 "подчеркиваний"
    //text2 = "руб";
    //text3 = "Товар";
    //text4 = "Цена";
    string text1 = "Цена товаров: ",
    line = "_____"; // 30 "подчеркиваний"
    text2 = "руб",
    text3 = "Товар",
    text4 = "Цена";

    cout << text1 << endl;
    cout << line << endl;
```



# Вывод

## результатов

```
ConsoleApplication26 (Глобальная область) main()
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    char razd = '|';
    int cost1 = 400, cost2 = 10, cost3 = 20;
    string text1 = "Цена товаров: ",
    line = "_____"; // 30 "подчеркиваний"
    text2 = "руб",
    text3 = "Товар",
    text4 = "Цена";

    cout << text1 << endl;
    cout << line << endl;
    cout << setw(10) << "Товар" << razd << setw(10) << "Цена" << razd << endl;
    cout << line << endl;
    cout << setw(10) << "Книга" << razd << setw(10) << cost1 << razd << endl;
    cout << setw(10) << "Тетрадь" << razd << setw(10) << cost2 << razd << endl;
    cout << setw(10) << "Карандаш" << razd << setw(10) << cost3 << razd << endl;
    cout << line << endl;

    return 0;
}
```

C:\WINDOWS\system32\cmd.exe

Цена товаров:

Товар	Цена	Валюта
Книга	400	руб
Тетрадь	10	руб
Карандаш	20	руб

Для продолжения нажмите любую клавишу . . .

Использовали множественное  
определение

# Типы данных

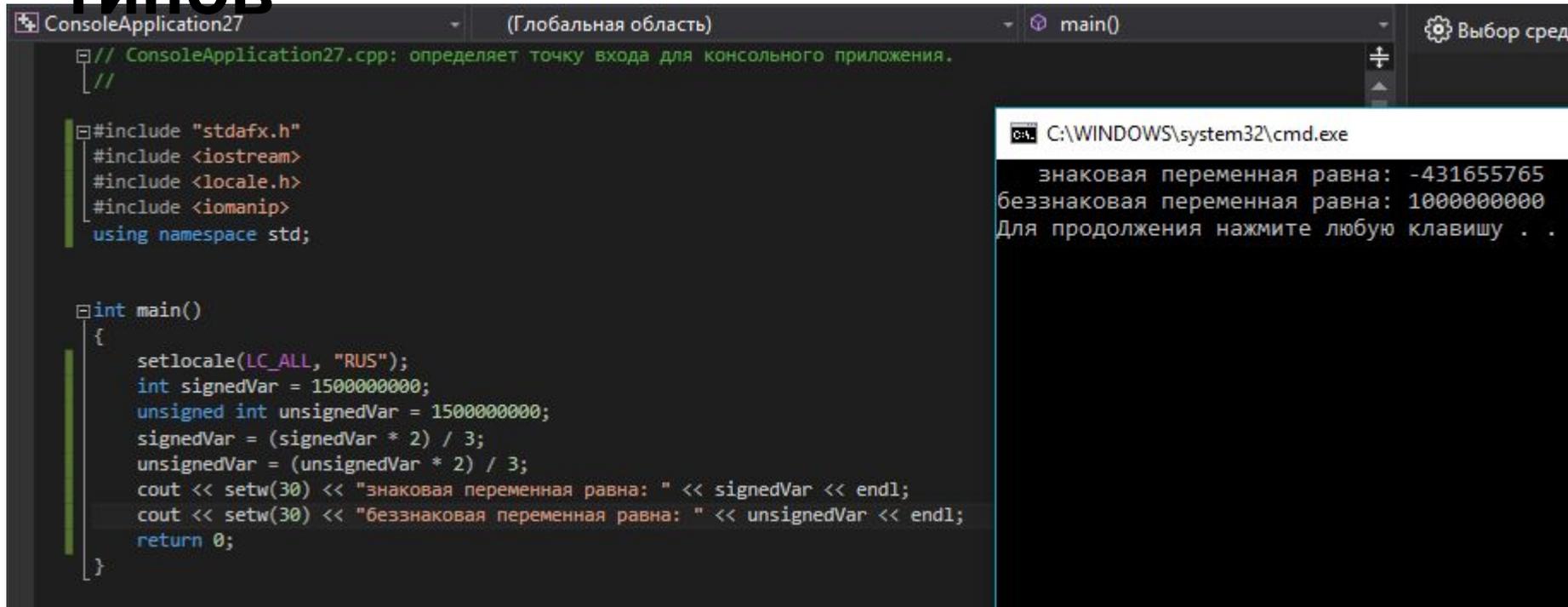
Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Точность	Размер в байтах
bool	False	True	Нет	1
char	-128	127	Нет	1
short	-32 768	32 767	Нет	2
int	-2 147 483 648	2 147 483 647	Нет	4
long	-2 147 483 648	2 147 483 647	Нет	4
float	$3.4 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$	7	4
double	$1.7 \cdot 10^{-308}$	$1.7 \cdot 10^{308}$	15	8

## Беззнаковые типы

### данные

Беззнаковые целые типы			
Название	Нижняя граница диапазона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned short	0	65 535	2
unsigned int	0	4 294 967 295	4
unsigned long	0	4 294 967 295	4

# Использование знаковых и беззнаковых ТИПОВ



The screenshot shows a C++ IDE with a source file named 'ConsoleApplication27.cpp'. The code defines a 'main' function that sets the locale to Russian and performs calculations on signed and unsigned integers. The output window shows the results of these calculations.

```
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//

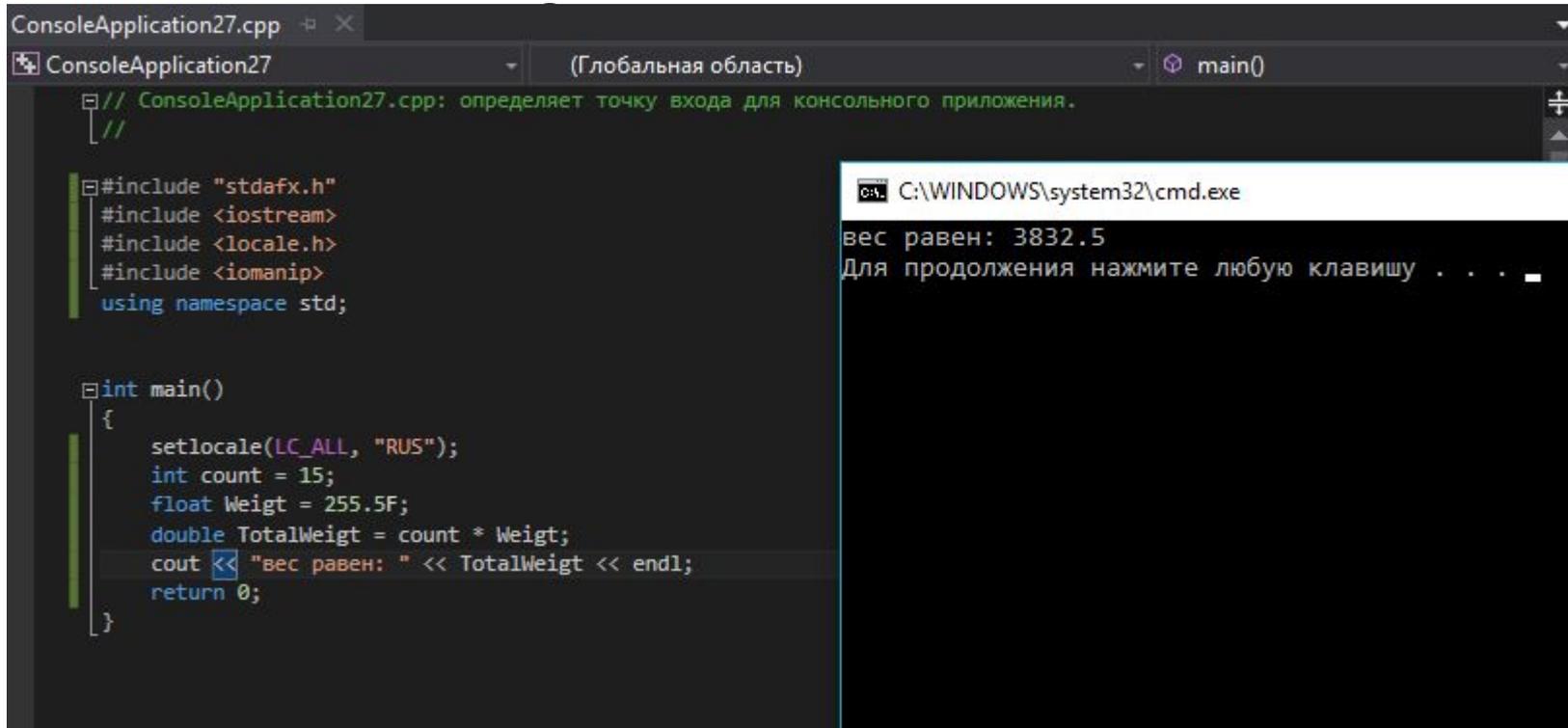
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int signedVar = 1500000000;
    unsigned int unsignedVar = 1500000000;
    signedVar = (signedVar * 2) / 3;
    unsignedVar = (unsignedVar * 2) / 3;
    cout << setw(30) << "знаковая переменная равна: " << signedVar << endl;
    cout << setw(30) << "беззнаковая переменная равна: " << unsignedVar << endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
знаковая переменная равна: -431655765
беззнаковая переменная равна: 1000000000
Для продолжения нажмите любую клавишу . .
```

**1 500 000 000 \* 2 = 3 000 000 000 (предел 2 147 483 647)**

# Преобразования



The image shows a screenshot of a C++ IDE with two windows. The left window displays the source code for 'ConsoleApplication27.cpp'. The right window shows the output of the program.

```
ConsoleApplication27.cpp
ConsoleApplication27 (Глобальная область) main()
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int count = 15;
    float Weigt = 255.5F;
    double TotalWeigt = count * Weigt;
    cout << "вес равен: " << TotalWeigt << endl;
    return 0;
}
```

Output window (C:\WINDOWS\system32\cmd.exe):

```
вес равен: 3832.5
Для продолжения нажмите любую клавишу . . .
```

# Неявные преобразования

## Иерархия **ТИПОВ**

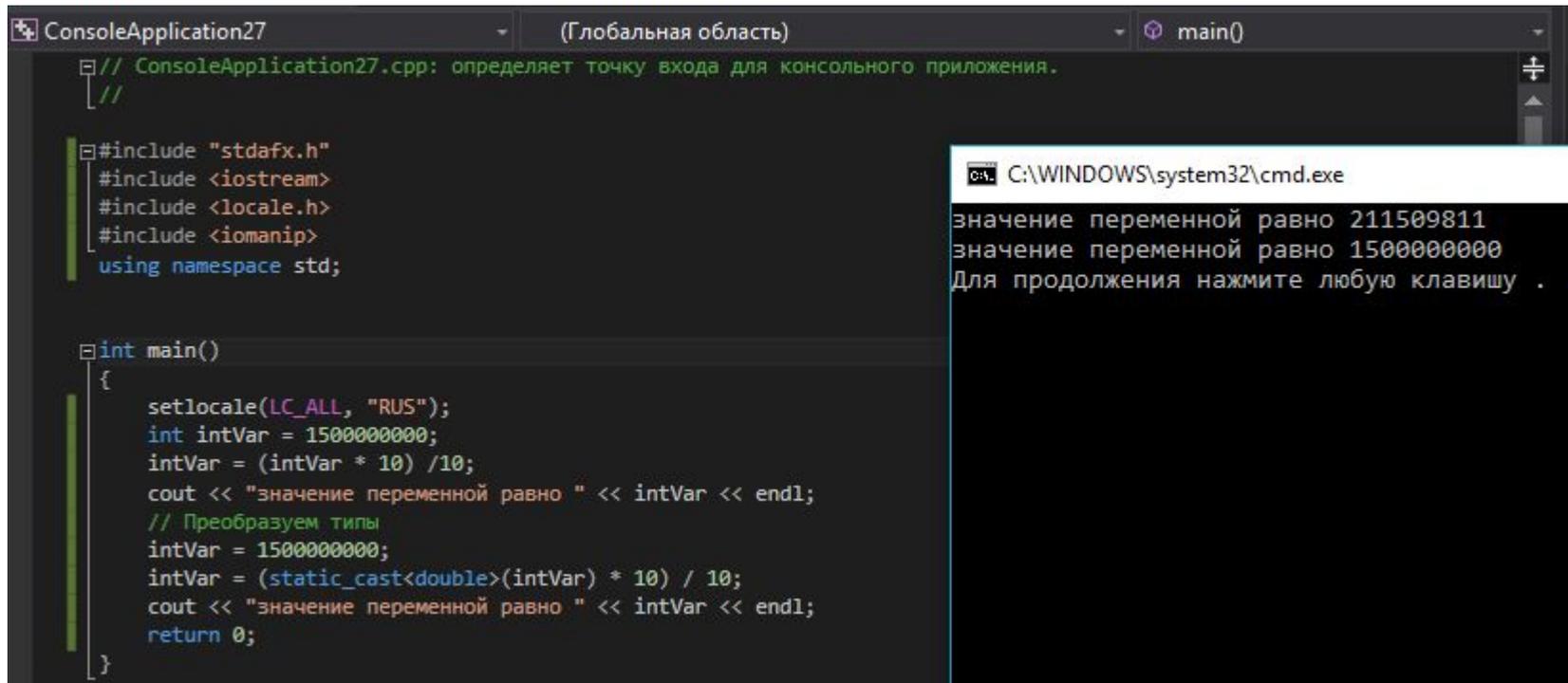
<u>Тип данных</u>	<u>Старшинство</u>
long double	Высший
double	
float	
long	
int	
short	
char	Низший



# Явные преобразования

## ТИПОВ

```
intVar = static_cast<double>(intVar);
```



The screenshot shows a C++ IDE window titled 'ConsoleApplication27' with a file named 'ConsoleApplication27.cpp'. The code defines a `main` function that sets the locale to Russian, initializes an integer variable `intVar` to 1500000000, and prints its value. It then performs a type cast to `double` and prints the result. The output window shows the program's execution, displaying the values 211509811 and 1500000000.

```
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int intVar = 1500000000;
    intVar = (intVar * 10) / 10;
    cout << "значение переменной равно " << intVar << endl;
    // Преобразуем типы
    intVar = 1500000000;
    intVar = (static_cast<double>(intVar) * 10) / 10;
    cout << "значение переменной равно " << intVar << endl;
    return 0;
}
```

C:\WINDOWS\system32\cmd.exe  
значение переменной равно 211509811  
значение переменной равно 1500000000  
Для продолжения нажмите любую клавишу .

**Приведения типов – только в осознанных случаях!!!**

# Арифметические операции

**“+” – сложение;**

**“-” – вычитание;**

**“\*” – умножение;**

**“/” – деление.**

# Остаток от деления

**“%” – остаток от деления (взятие по  
модулю)**

$$8 \% 10 = 8;$$

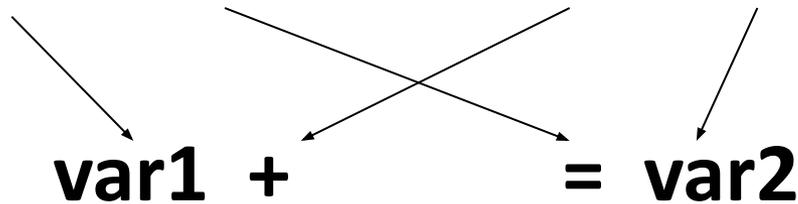
$$10 \% 10 = 0;$$

$$11 \% 10 = 1;$$

```
cout << 11 % 10 ;
```

# Арифметические операции с присваиванием

**var1 = var1 + var2;**



**'-='**

**'\*='**

**'/='**

**'%='**

# Инкремент

**T**  
**var1 = var1 + 1**

**var1 += 1**

**++var1**

**Префиксная форма**

**++var1**

**Постфиксная форма**

**var1++**

# Инкремент

```
ConsoleApplication27 (Глобальная область) main()
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

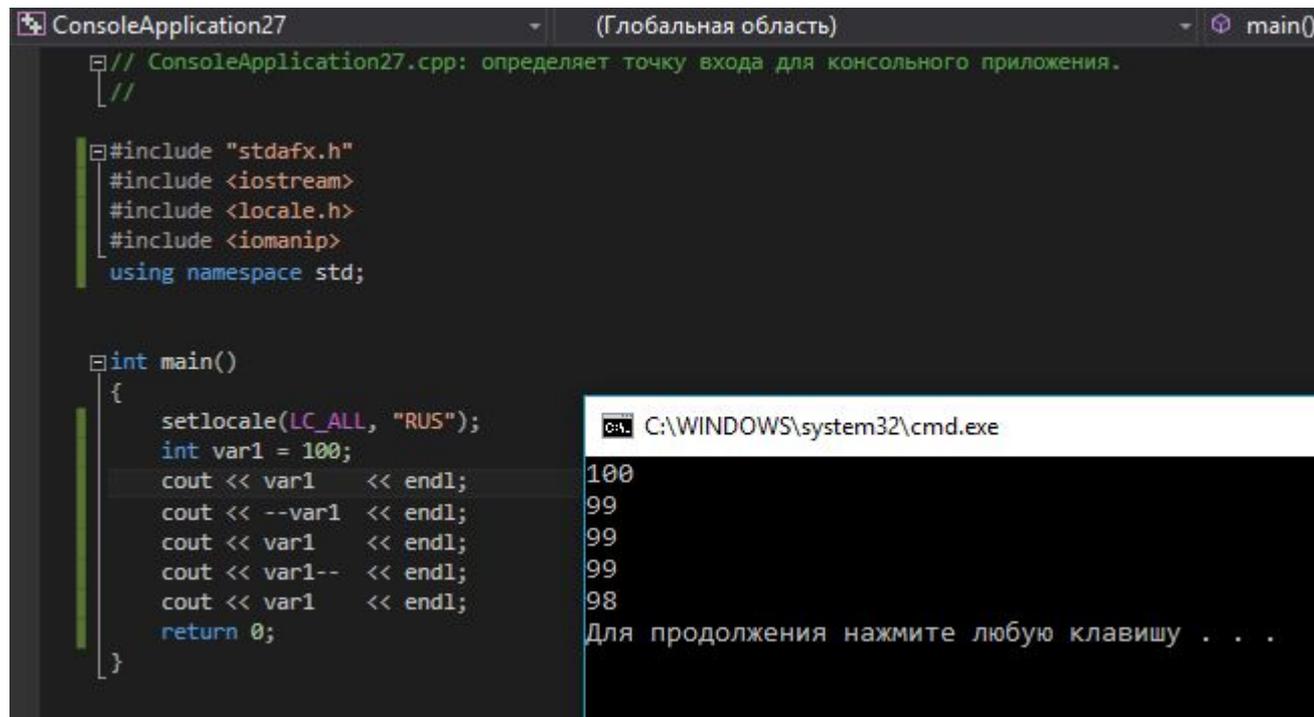
int main()
{
    setlocale(LC_ALL, "RUS");
    int var1 = 100;
    cout << var1 << endl;
    cout << ++var1 << endl;
    cout << var1 << endl;
    cout << var1++ << endl;
    cout << var1 << endl;
    return 0;
}
```

```
cmd. C:\WINDOWS\system32\cmd.exe
100
101
101
101
101
102
Для продолжения нажмите любую клавишу . . .
```

# Декремент

**↑**var1;

var1**↓**;



The image shows a screenshot of a C++ IDE window titled "ConsoleApplication27" with a sub-window for "main()". The code in the editor is as follows:

```
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//

#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int var1 = 100;
    cout << var1 << endl;
    cout << --var1 << endl;
    cout << var1 << endl;
    cout << var1-- << endl;
    cout << var1 << endl;
    return 0;
}
```

Below the code, a terminal window titled "cmd.exe" shows the output of the program:

```
100
99
99
99
98
Для продолжения нажмите любую клавишу . . .
```

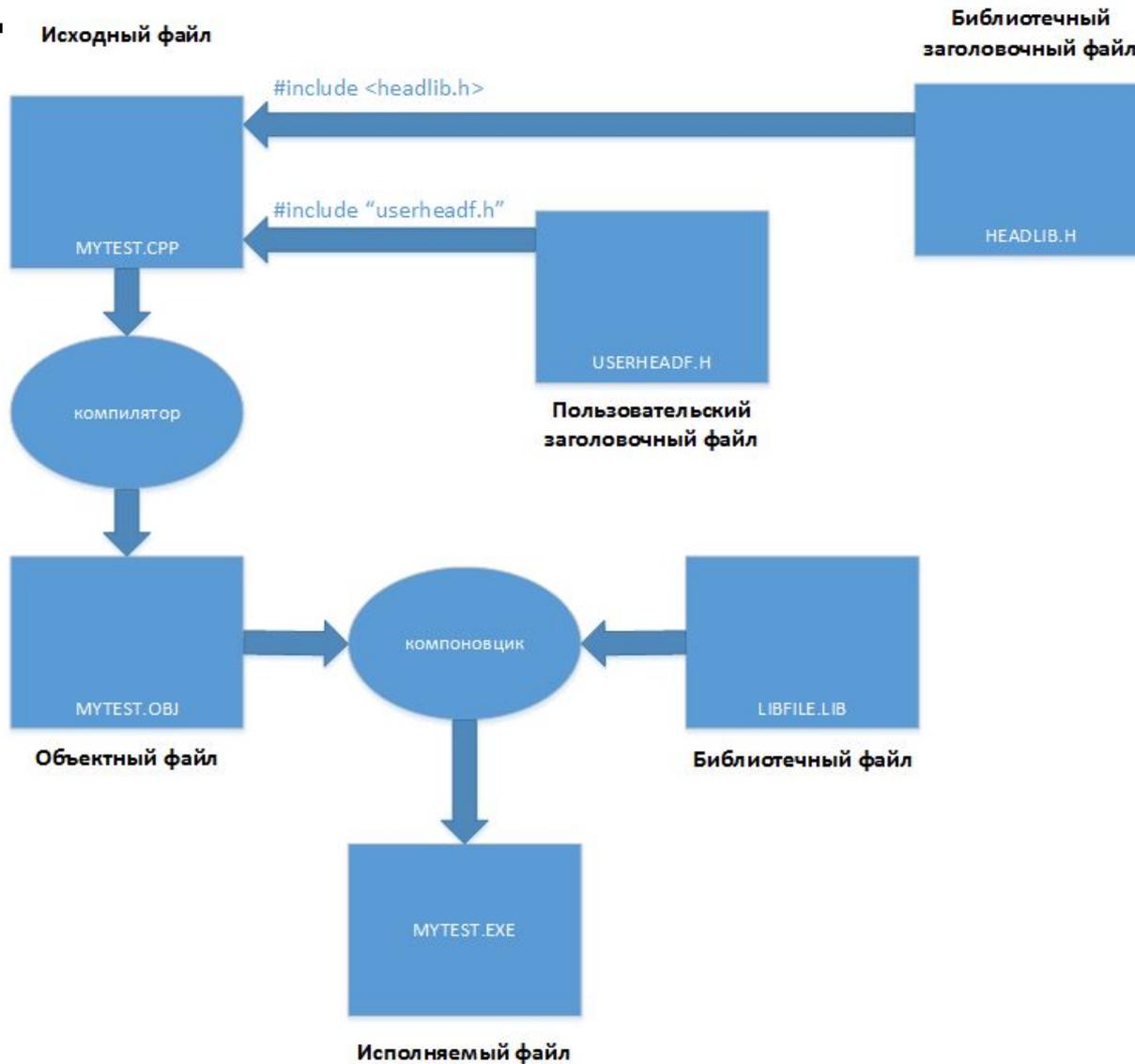
# Библиотечные

```
ConsoleApplication27 (Глобальная область) main()
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    double number, answer;
    cout << "Введите число: ";
    cin >> number;
    answer = sqrt(number);
    cout << "Вариант 1" << endl;
    cout << "Квадратный корень равен " << answer << endl;
    cout << "Вариант 2" << endl;
    cout << "Квадратный корень равен " << sqrt(number) << endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
Введите число: 100
Вариант 1
Квадратный корень равен 10
Вариант 2
Квадратный корень равен 10
Для продолжения нажмите любую клавишу . . .
```

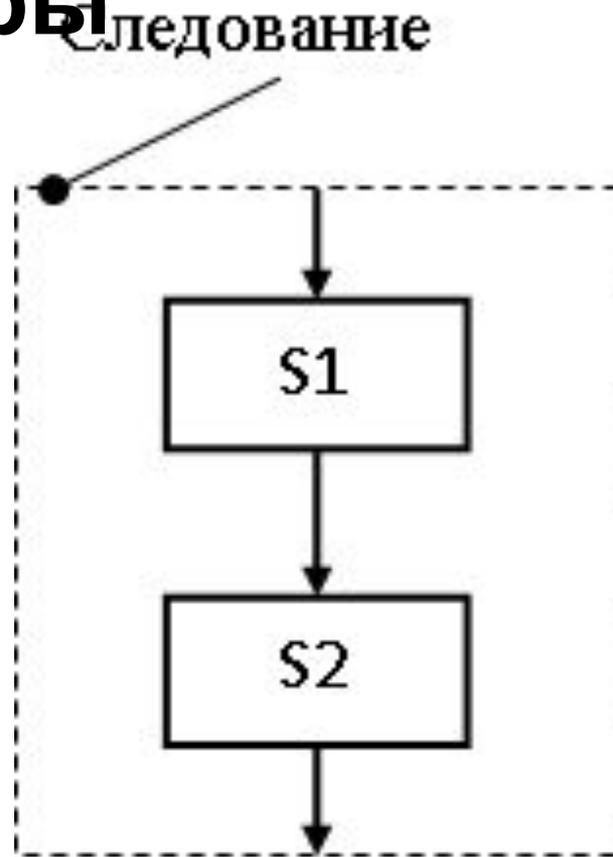
# Заголовочные и библиотечные файлы



# Базовые управляющие структуры

- Следовани  
е;
- Циклы;
- Ветвления;
- Переходы.

# Базовые управляющие структуры



# Базовые управляющие

## Структуры Операции отношения

Операция	Название
>	больше
<	меньше
==	равно
!=	не равно
>=	больше или равно
<=	меньше или равно

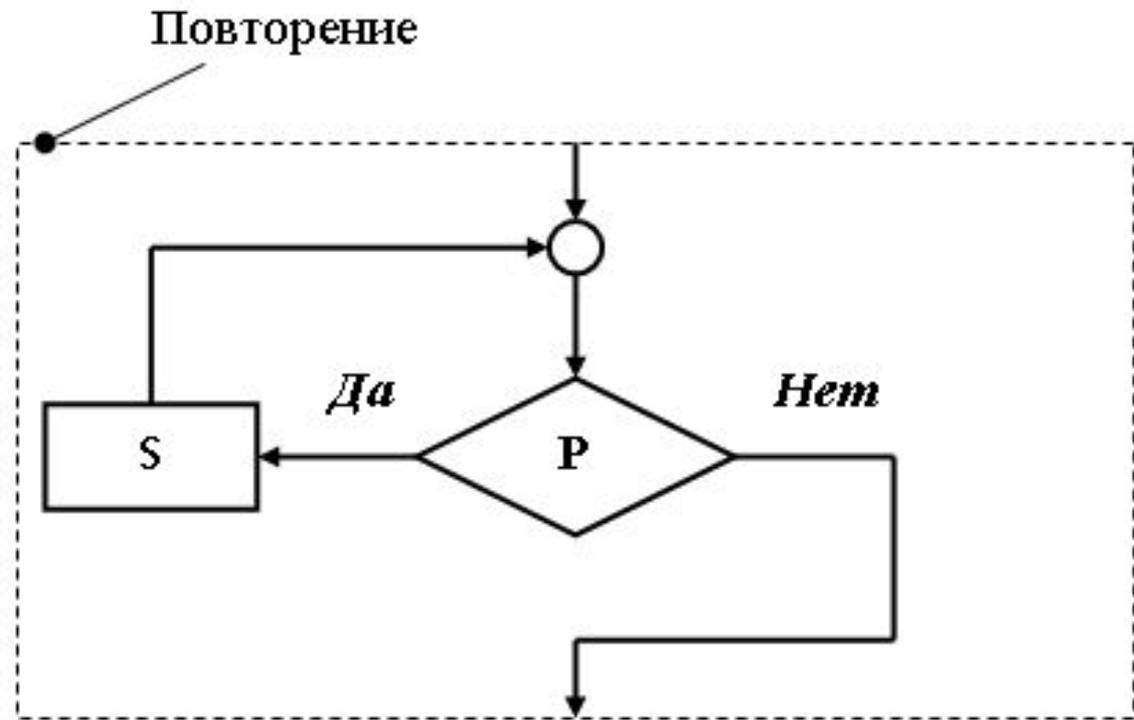
```
ConsoleApplication27 (Глобальная область)

#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int number;
    cout << "Введите число: ";
    cin >> number;
    cout << number << " < 10 равно " << (number<10) << '\n';
    cout << number << " <= 10 равно " << (number<=10) << '\n';
    cout << number << " > 10 равно " << (number>10) << '\n';
    cout << number << " >= 10 равно " << (number>=10) << '\n';
    cout << number << " == 10 равно " << (number==10) << '\n';
    return 0;
}
```

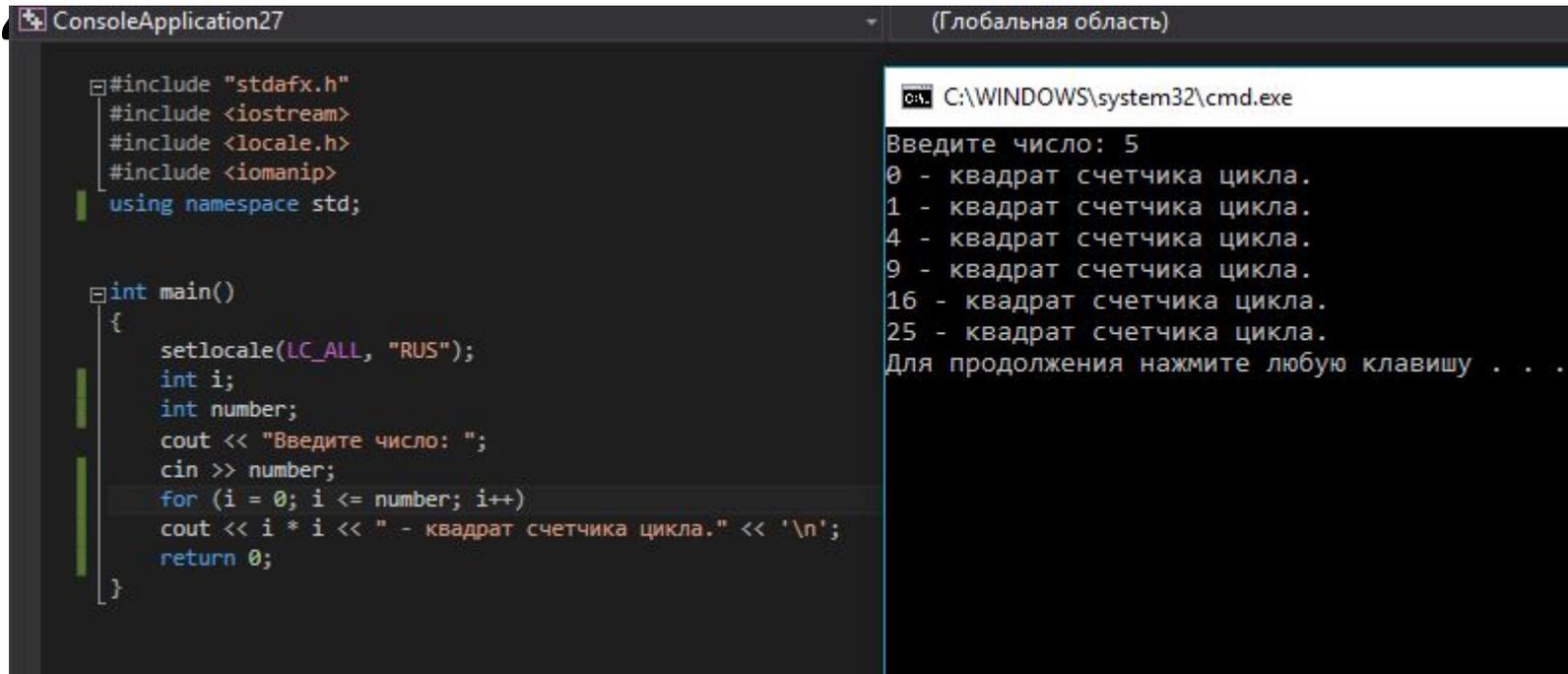
```
C:\WINDOWS\system32\cmd.exe
Введите число: 10
10 < 10 равно 0
10 <= 10 равно 1
10 > 10 равно 0
10 >= 10 равно 1
10 == 10 равно 1
Для продолжения нажмите любую клавишу . . .
```

# Циклы



- for
- while
- do

# Циклы



```
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int i;
    int number;
    cout << "Введите число: ";
    cin >> number;
    for (i = 0; i <= number; i++)
        cout << i * i << " - квадрат счетчика цикла." << '\n';
    return 0;
}
```

C:\WINDOWS\system32\cmd.exe

Введите число: 5  
0 - квадрат счетчика цикла.  
1 - квадрат счетчика цикла.  
4 - квадрат счетчика цикла.  
9 - квадрат счетчика цикла.  
16 - квадрат счетчика цикла.  
25 - квадрат счетчика цикла.  
Для продолжения нажмите любую клавишу . . .

**for (i = 0; i <= number; i++ )**

**i = 0**      инициализирующее

**i <= number**    условие проверки

**i++**      инкрементирующее

**Счетчик i**  
**определен в**  
**теле**  
**цикла**

# Циклы

## “for”

```
for (i = 0; i <= number; i++)
cout << i * i << " - квадрат счетчика цикла." <<
'\n';
for (i = 0; i <= number; i++)
{
cout << i << " - значение счетчика цикла." << '\n';
cout << i * i << " - квадрат счетчика цикла." << '\n';
cout << i * i * i << "- 3-я степень счетчика цикла." <<
'\n';
int new_count = count * 2;
cout << new_count << endl;
}
```

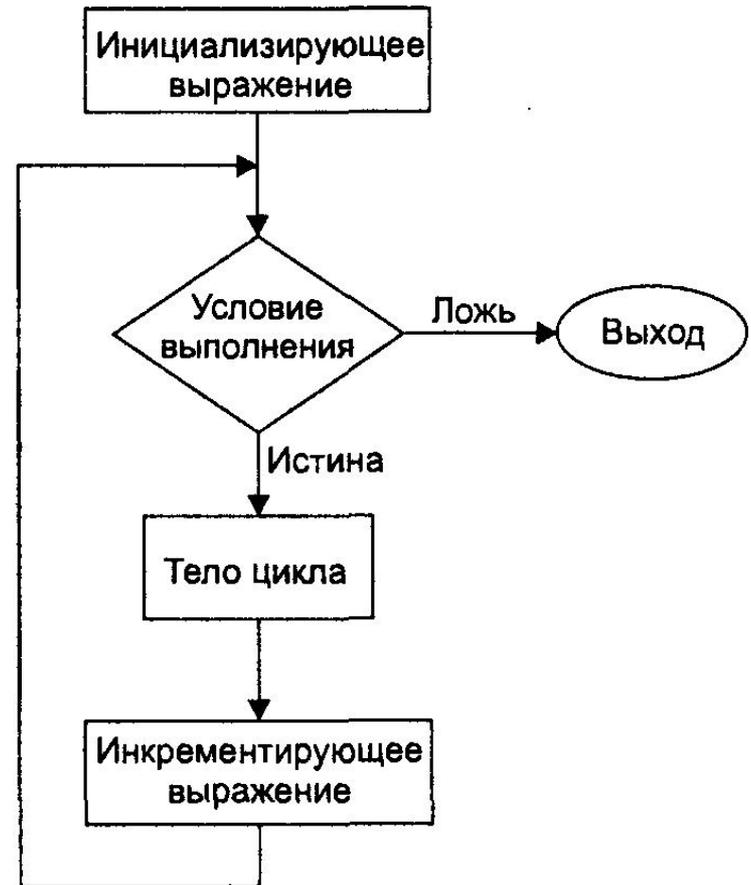
`new_count`

переменные, определенные внутри него, *невидимы* вне этого блока. Невидимость означает, что программа не имеет доступа к переменной

# Циклы

## 'for'

Присваивая счетчику начальное значение, равное 0, в качестве условия продолжения цикла ставят сравнение счетчика с желаемым числом выполнений и увеличивают счетчик на единицу каждый раз, когда исполнение тела цикла завершается.



# Вычисление

```
ConsoleApplication27 (Глобальная область)
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned int number;
    unsigned long fact = 1; // факториал
    cout << "Введите число: ";
    cin >> number;
    for (int j = number; j > 0; j--)
        fact *= j;
    cout << fact << " - факториал числа." << '\n';
    cout << endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
Введите число: 6
720 - факториал числа.

Для продолжения нажмите любую клавишу . . .
```

# Вычисление

```
ConsoleApplication27 (Глобальная область)
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned int number;
    unsigned long fact = 1; // факториал
    cout << "Введите число: ";
    cin >> number;
    for (int j = number; j > 0; j--)
    {
        fact *= j;
    }
    cout << fact << " - факториал числа." << '\n';
    cout << endl;
    return 0;
}
```

C:\WINDOWS\system32\cmd.exe

Введите число: 6  
720 - факториал числа.

Для продолжения нажмите любую клавишу . . .

# Вычисление факториала N

The image shows a screenshot of a C++ IDE with two windows. The left window, titled 'ConsoleApplication27', displays the source code for a program that calculates the factorial of a number. The code includes headers for `stdafx.h`, `iostream`, `locale`, and `iomanip`, and uses the `std` namespace. The `main` function sets the locale to Russian, prompts the user for a number, and then uses a `for` loop to calculate the factorial, printing the result for each step. The right window, titled 'C:\WINDOWS\system32\cmd.exe', shows the program's execution. It prompts the user to enter a number, and the user has entered '6'. The output shows the factorial calculation for 6, with intermediate results: 6, 30, 120, 360, 720, and 720. The prompt 'Для продолжения нажмите любую клавишу . . .' is visible at the bottom.

```
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned int number;
    unsigned long fact = 1; // факториал
    cout << "Введите число: ";
    cin >> number;
    for (int j = number; j > 0; j--)
    {
        fact *= j;
        cout << fact << " - факториал числа." << '\n';
    }
    cout << endl;
    return 0;
}
```

C:\WINDOWS\system32\cmd.exe

Введите число: 6  
6 - факториал числа.  
30 - факториал числа.  
120 - факториал числа.  
360 - факториал числа.  
720 - факториал числа.  
720 - факториал числа.

Для продолжения нажмите любую клавишу . . .

# Циклы

**'for' ( ; ; )**

Несколько инициализирующих выражений и условий цикла. Вместо одного инициализирующего выражения в операторе цикла for можно использовать несколько выражений, разделяемых запятыми. Подобным же образом можно использовать более одного инкрементирующего выражения. Лишь условие продолжения цикла всегда должно быть одно.

```
for (j = 0, alpha = 100; j < 50; j++, beta--)  
{  
    // тело цикла  
}
```

# Циклы

## 'for'

```
ConsoleApplication27 (Глобальная область)
#include <iostream.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned int number;
    unsigned long fact = 1; // факториал
    int i = 0;
    cout << "Введите число: ";
    cin >> number;
    for (int j = number, i = 1; j > 0; j--, i++)
    {
        fact *= j;
        cout << fact << " - факториал числа." << "номер цикла - " << i << '\n';
    }
    cout << endl;

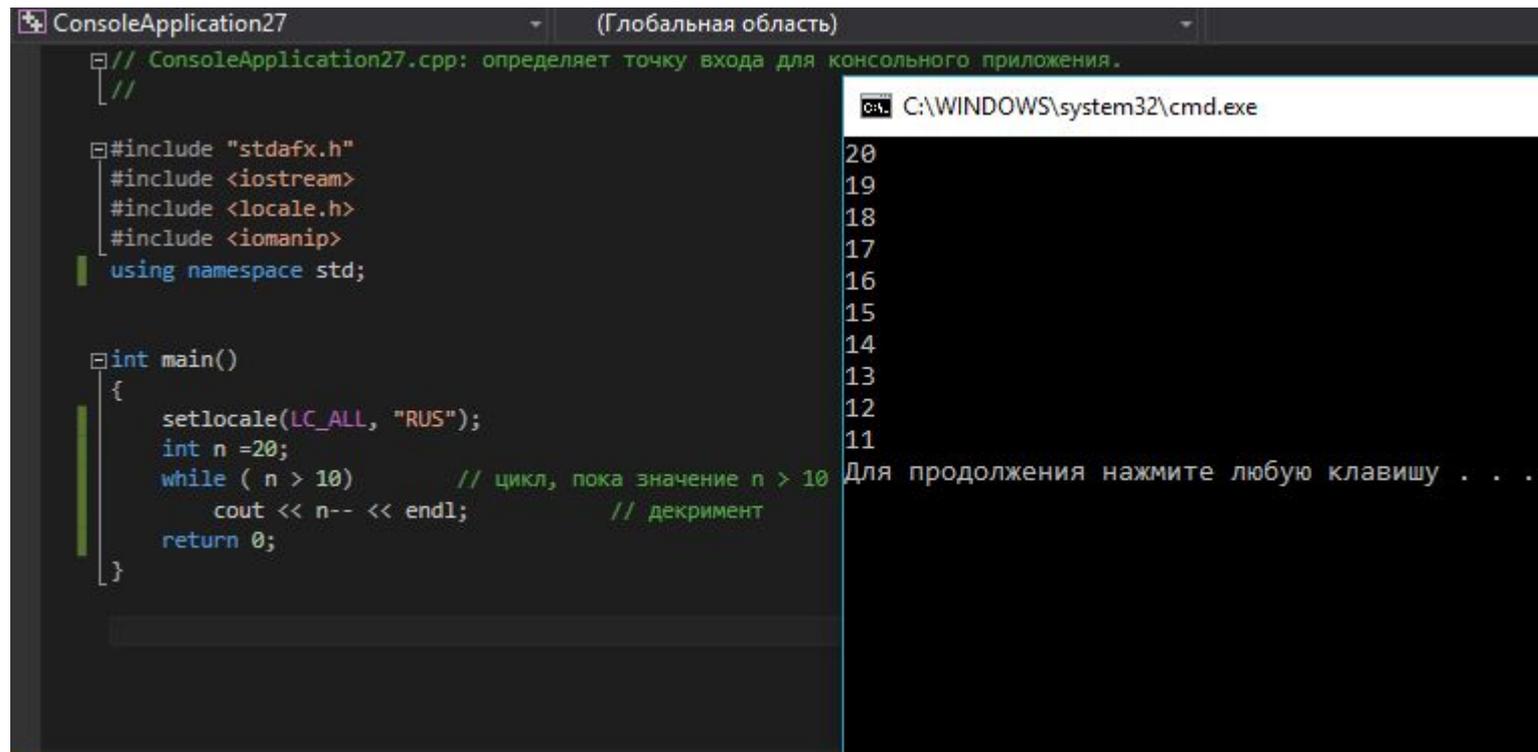
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
Введите число: 5
5 - факториал числа.номер цикла - 1
20 - факториал числа.номер цикла - 2
60 - факториал числа.номер цикла - 3
120 - факториал числа.номер цикла - 4
120 - факториал числа.номер цикла - 5

Для продолжения нажмите любую клавишу .
```

# Циклы

‘while’  
Когда известно какое количество циклов  
ВЫПОЛНИТЬ

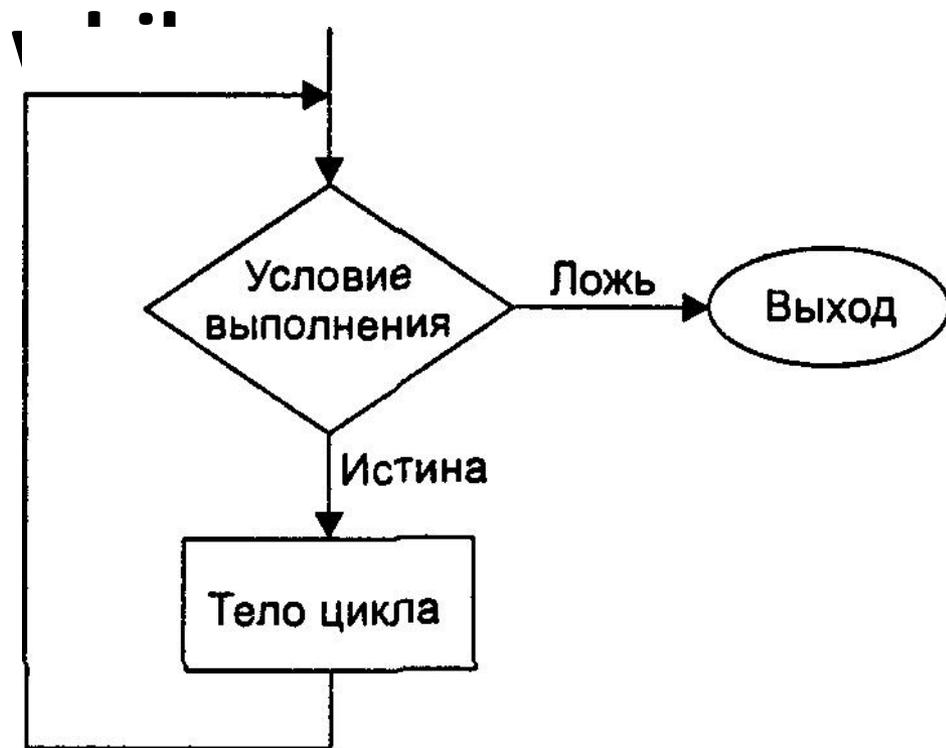


```
ConsoleApplication27 (Глобальная область)
// ConsoleApplication27.cpp: определяет точку входа для консольного приложения.
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int n = 20;
    while ( n > 10) // цикл, пока значение n > 10
        cout << n-- << endl; // декримент
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
20
19
18
17
16
15
14
13
12
11
Для продолжения нажмите любую клавишу . . .
```

# Циклы



# Вычисления ряда чисел

## Фибоначчи

```
ConsoleApplication27 (Глобал) C:\WINDOWS\system32\cmd.exe
int main()
{
    //граница типа unsigned long
    const unsigned long limit = 4294967295;
    unsigned long next = 0; // предпоследний
    unsigned long last = 1; // последний член
    while (next < limit / 2) // результат не
    {
        cout << last << endl; // вывод пос
        long sum = next + last; // сложение д
        next = last; // обновление предпос
        last = sum; // и последнего членов
    }
    cout << endl;
    return 0;
}

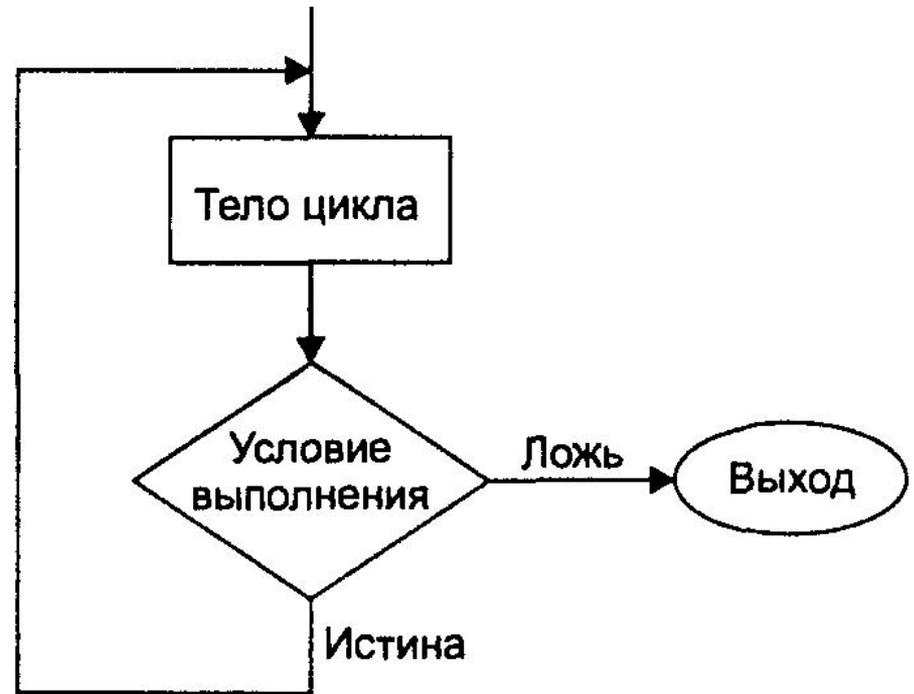
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
2971215073

Вывод
Показать выходные данные из: сборки
1>----- Сборка начата: проект: ConsoleApplica
1> ConsoleApplication27.cpp
1> ConsoleApplication27.vcxproj -> C:\Users\l
===== Сборка: успешно: 1, с ошибками: 0,
Для продолжения нажмите любую клавишу .
```

# Цикл 'do'

```
do  
    procedure;  
while ( var1 != var2);
```

```
do  
{  
    procedure1;  
    procedure2;  
}  
while ( var1 != var2);
```



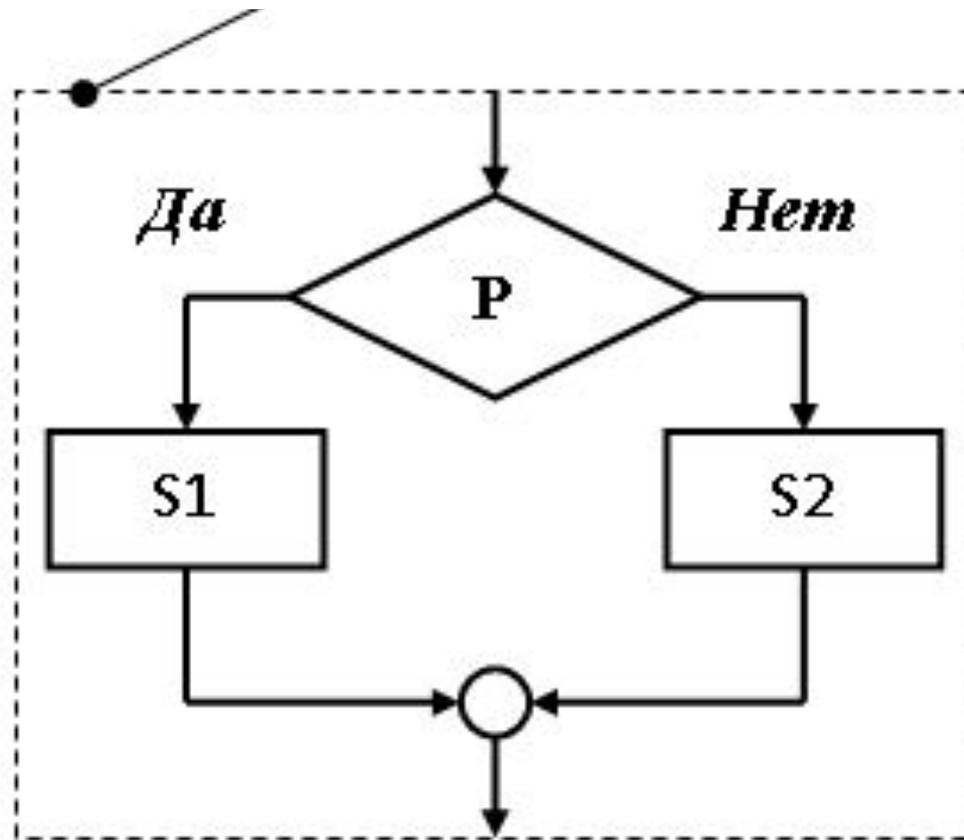
**Цикл с  
постусловием!**  
Выполнится хотя бы 1  
раз

# Циклы **for**, **while**, **do**

Рекомендации:

Цикл **for** подходит для тех случаев, когда мы заранее знаем, сколько раз нам потребуется его выполнение. Циклы **while** и **do** используются в тех случаях, когда число итераций цикла заранее не известно, причем цикл **while** подходит в тех случаях, когда тело цикла может быть не исполненным ни разу, а цикл **do** — когда обязательно хотя бы однократное исполнение тела цикла.

# Ветвления



# **Ветвлен**

## **ия**

**if ...**

**if ... else ...**

**switch.... case**

**...**

# Ветвления

if

```
ConsoleApplication27 (Глобальная область) main()
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    int number = 0;
    int count = 0;
    cout << "Введите число: " << endl;
    cin >> number;
    if (number < 10)
        cout << "Это число меньше 10." << '\n'
        << "Введите еще раз: ";
        cin >> number;
    if (number > 10)
    {
        count = count + 1;
        cout << "Это " << count << " удачная попытка!" << endl;
    }
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
Введите число:
2
Это число меньше 10.
Введите еще раз: 15
Это 1 удачная попытка!
Для продолжения нажмите любую клавишу .
```

# Ветвления if

## Пример поиска простых чисел

```
ConsoleApplication27 (Глобальная область)
//
#include "stdafx.h"
#include <iostream>
#include <locale.h>
#include <iomanip>
#include <process.h> // для использования exit()
using namespace std;

int main()
{
    setlocale(LC_ALL, "RUS");
    unsigned long n, j;
    cout << "Введите число: ";
    cin >> n; // ввод проверяемого числа
    for (j = 2; j <= n / 2; j++) // деление на целые числа,
        if (n%j == 0) // начиная с 2; если остаток
        { //нулевой, то число не простое
            cout << "Число не простое; делится на " << j << endl;
            exit(0); // выход из программы
        }
    cout << "Число является простым\n";
    return 0;
}
```

C:\WINDOWS\system32\cmd.exe  
Введите число: 123  
Число не простое; делится на 3  
Для продолжения нажмите любую клавишу . . .

C:\WINDOWS\system32\cmd.exe  
Введите число: 43567  
Число не простое; делится на 19  
Для продолжения нажмите любую клавишу . . .

C:\WINDOWS\system32\cmd.exe  
Введите число: 13  
Число является простым  
Для продолжения нажмите любую клавишу . . .

C:\WINDOWS\system32\cmd.exe  
Введите число: 22243  
Число не простое; делится на 13  
Для продолжения нажмите любую клавишу . . .

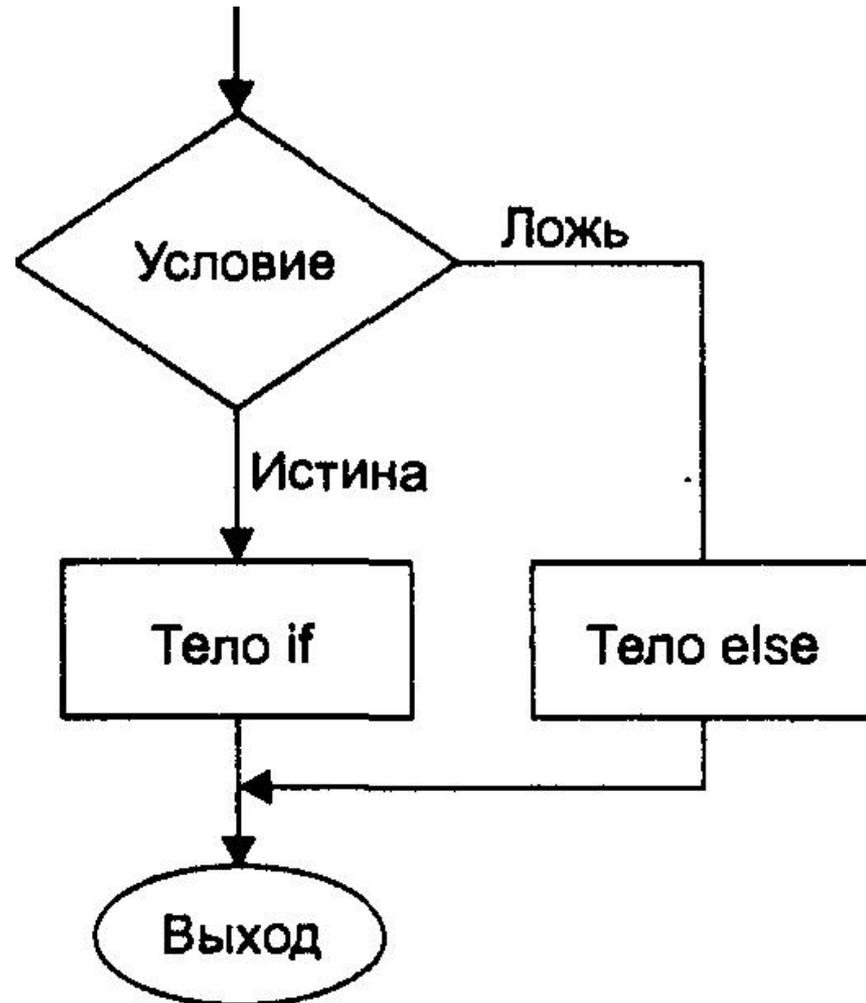
тело цикла не заключено в фигурные скобки. Это объясняется тем, что оператор if и операторы тела ветвления на самом деле являются одним оператором

# Выход из программы

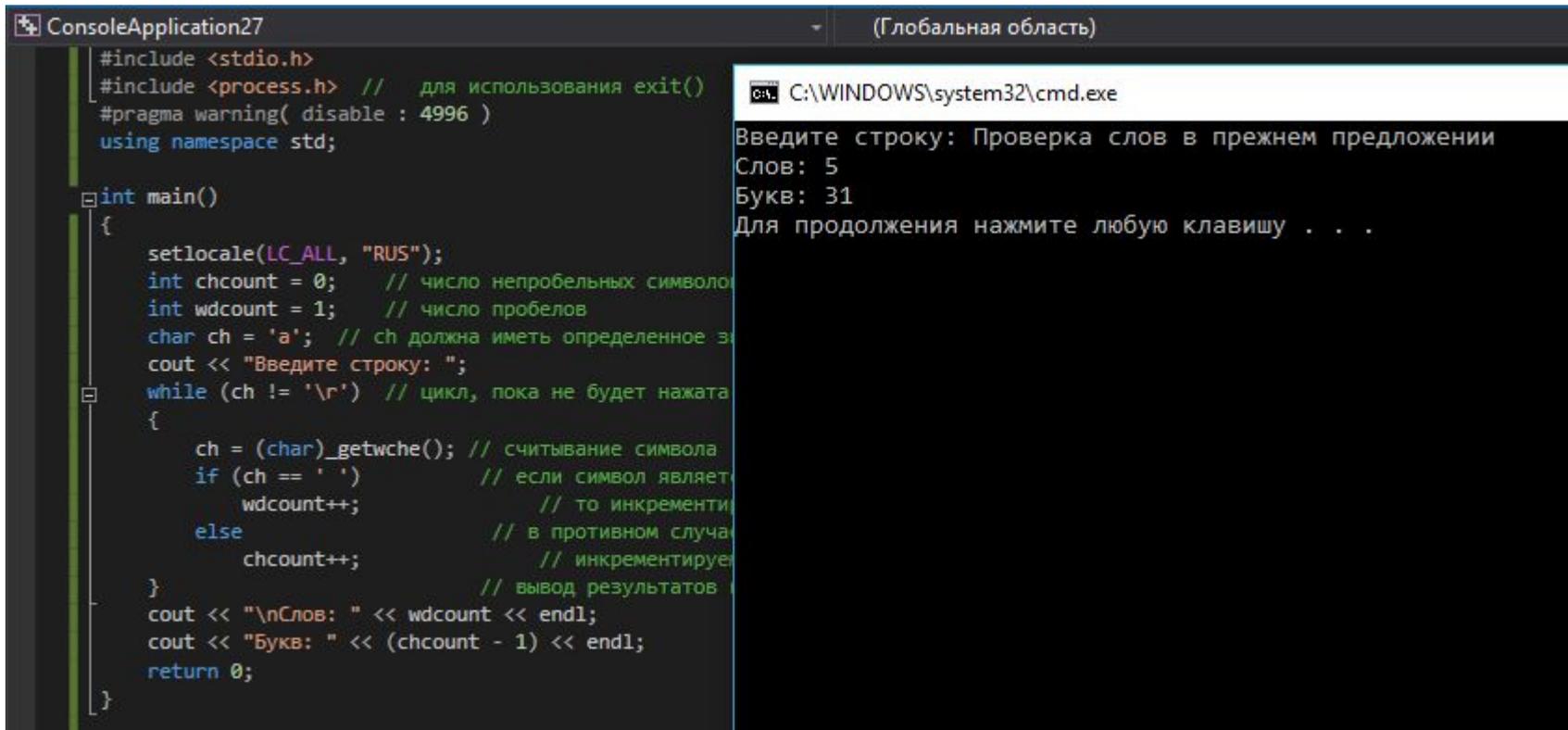
Функция **exit()** – немедленный выход из программы, независимо в каком месте исполнения она находится.

Требуется **#include <process.h>**

# Ветвления if ... else ...



# Ветвления if ... else ...



The image shows a screenshot of a Visual Studio IDE. On the left, a code editor window titled 'ConsoleApplication27' displays C++ code. The code includes headers for `<stdio.h>` and `<process.h>`, uses the `std` namespace, and defines a `main` function. The `main` function sets the locale to Russian, counts non-space characters (`chcount`) and spaces (`wdcount`) in a user input string, and prints the results. The code uses `if` and `else` statements to increment the counters based on the character type.

```
#include <stdio.h>
#include <process.h> // для использования exit()
#pragma warning( disable : 4996 )
using namespace std;

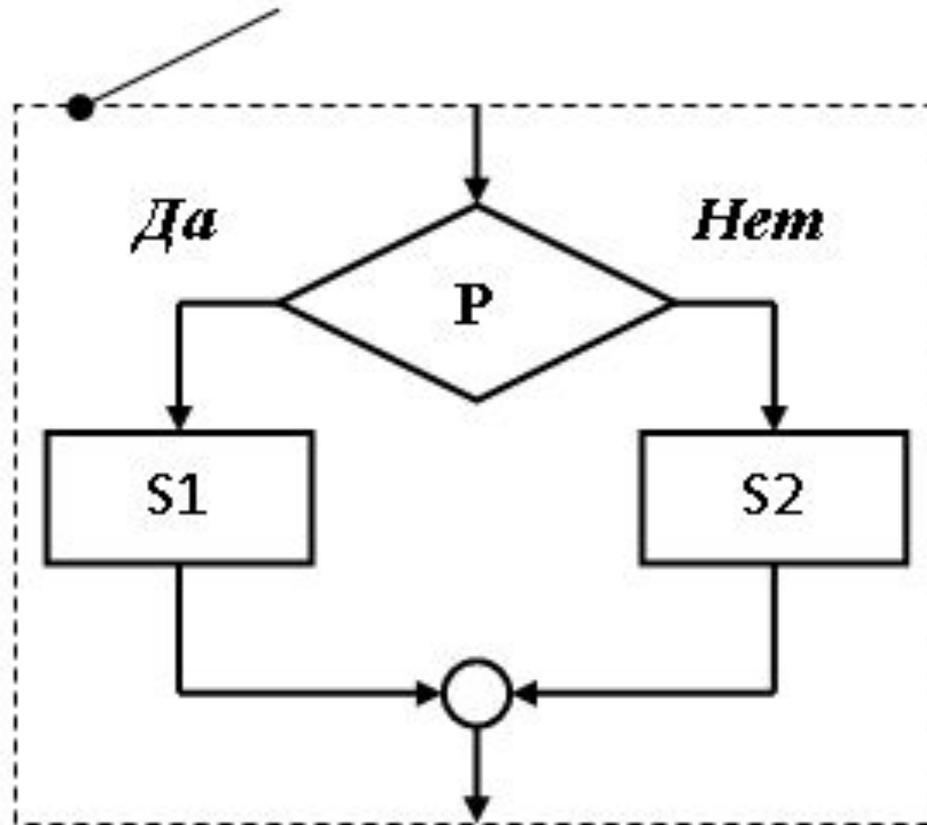
int main()
{
    setlocale(LC_ALL, "RUS");
    int chcount = 0; // число непробельных символов
    int wdcount = 1; // число пробелов
    char ch = 'a'; // ch должна иметь определенное значение
    cout << "Введите строку: ";
    while (ch != '\r') // цикл, пока не будет нажата клавиша Enter
    {
        ch = (char)_getwche(); // считывание символа
        if (ch == ' ') // если символ является пробелом
            wdcount++; // то инкрементируем счетчик пробелов
        else // в противном случае
            chcount++; // инкрементируем счетчик непробельных символов
    }
    cout << "\nСлов: " << wdcount << endl;
    cout << "Букв: " << (chcount - 1) << endl;
    return 0;
}
```

On the right, a command prompt window titled 'C:\WINDOWS\system32\cmd.exe' shows the output of the program. It prompts the user to enter a string, then displays the word count (5) and the character count (31), and finally prompts the user to press any key to continue.

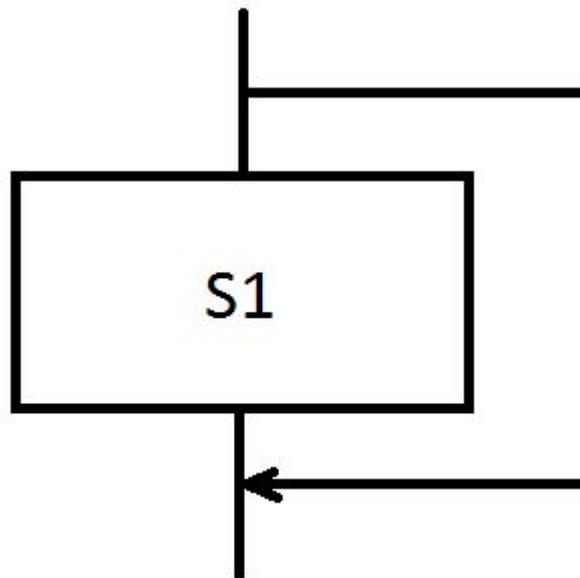
```
C:\WINDOWS\system32\cmd.exe
Введите строку: Проверка слов в прежнем предложении
Слов: 5
Букв: 31
Для продолжения нажмите любую клавишу . . .
```

# Разветвление

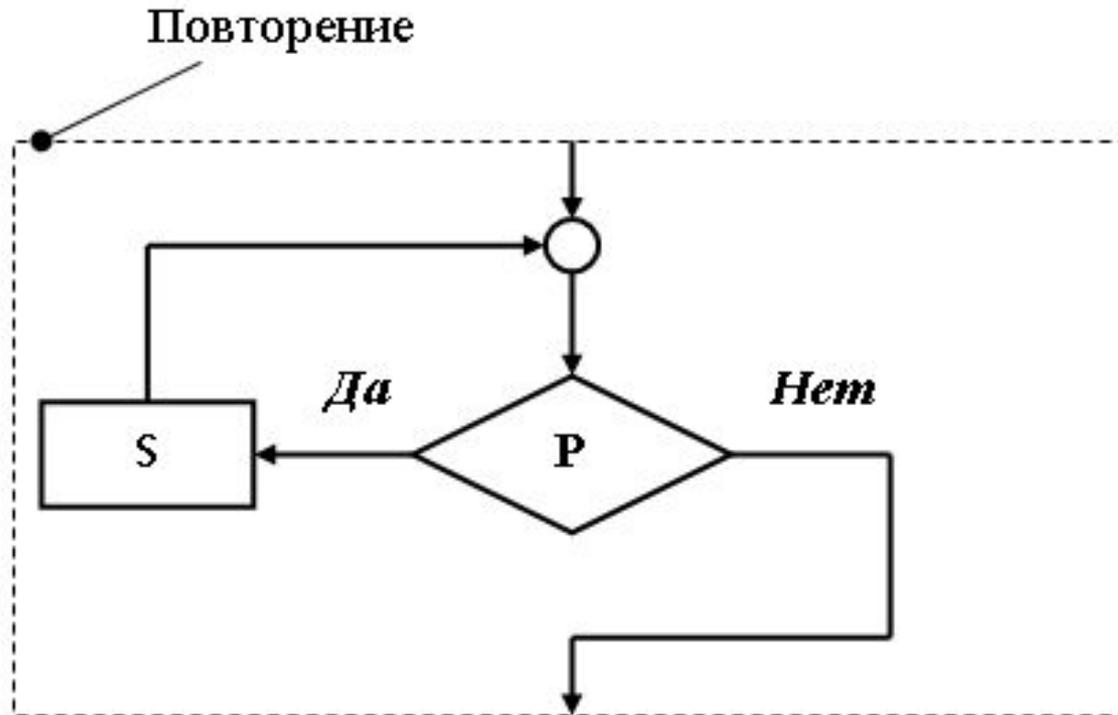
Разветвление



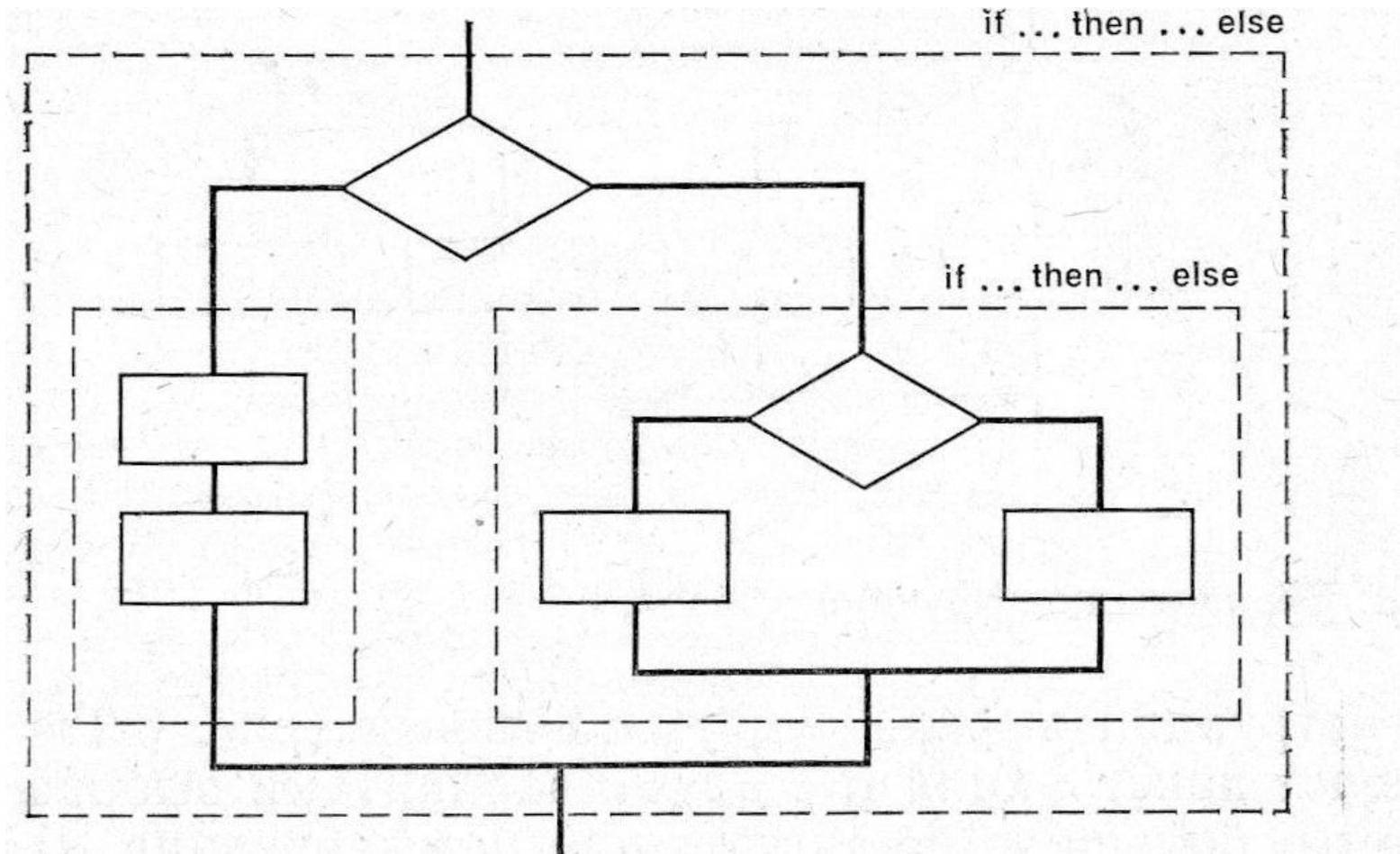
# Переход (GOTO...)



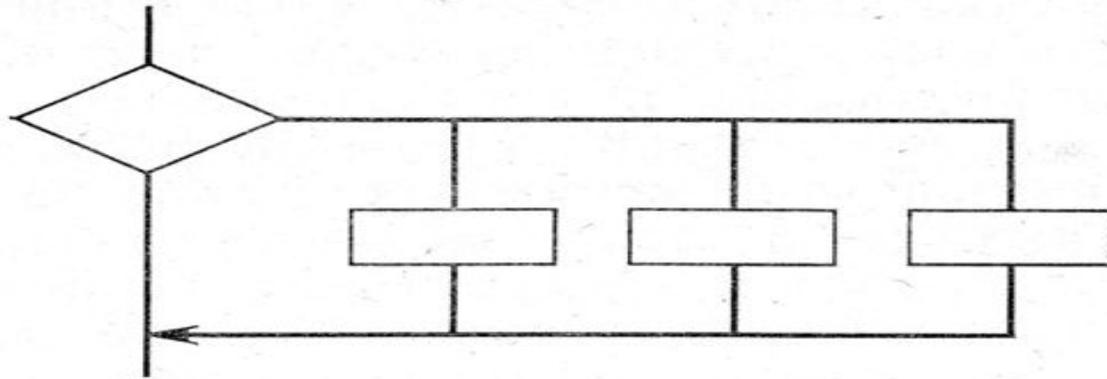
# Цикл (повторение)



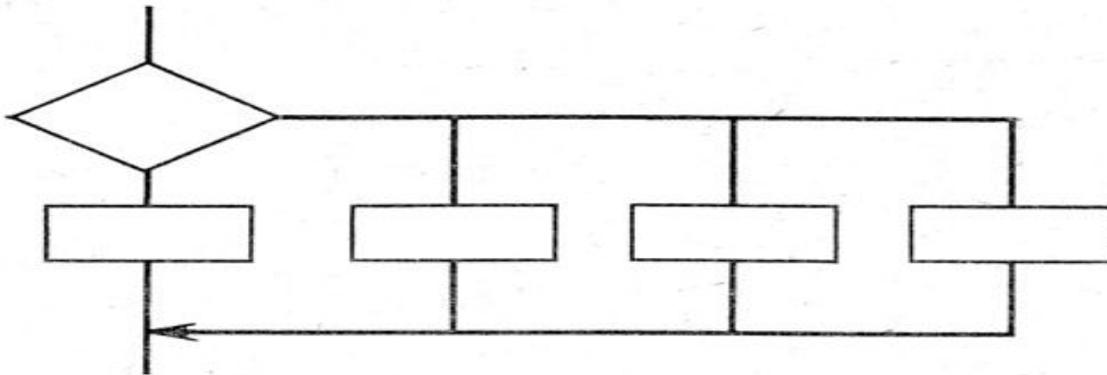
# Вложенные конструкции



# Конструкция "CASE"



case ... when ... when ... when ...



case ... when ... when ... when ... otherwise ...

# ФУНКЦИИ

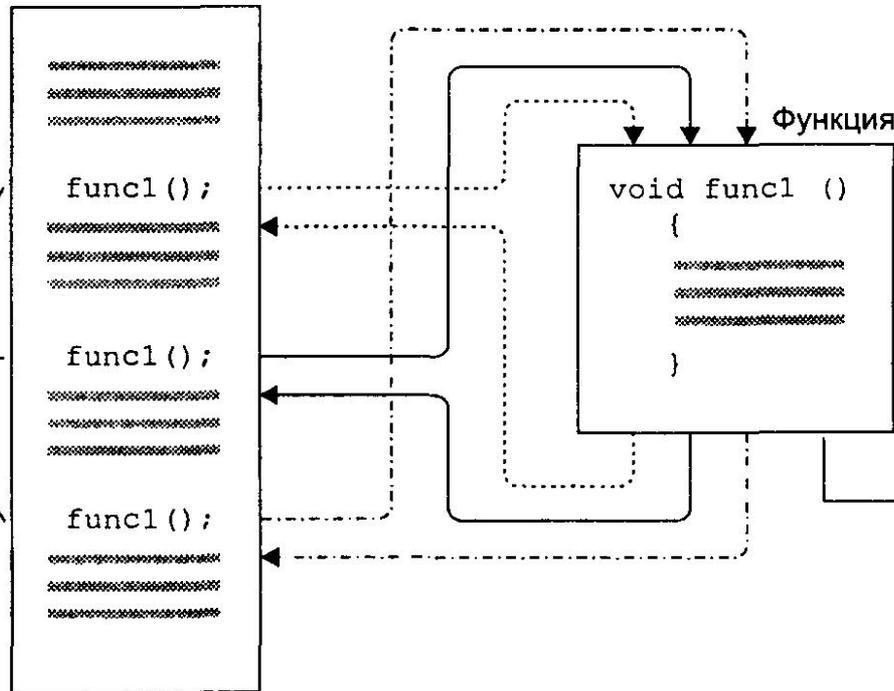
Название	Назначение	Пример
Объявление (прототип)	Содержит имя функции, типы ее аргументов и возвращаемого значения. Указывает компилятору на то, что определение функции будет сделано позднее	<code>void func();</code>
Вызов	Указывает на то, что необходимо выполнить функцию	<code>func();</code>
Заголовок	Первая строка определения	<code>void func()</code>
Определение	Является собственно функцией. Содержит код, предназначенный для исполнения	<code>void func() {  // операторы }</code>

# ФУНКЦИИ

Объявление

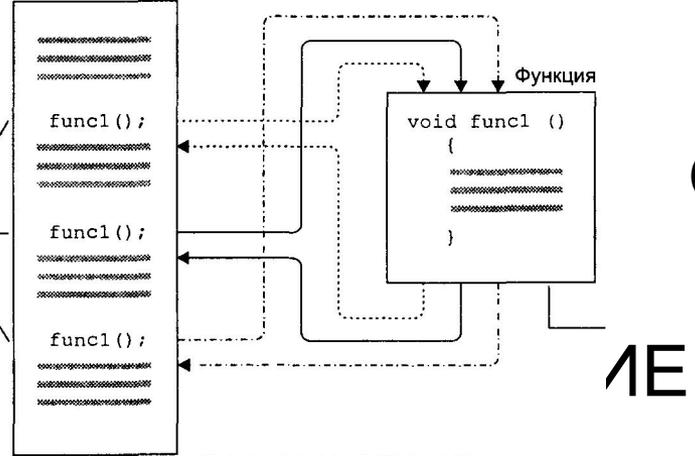
`void func1()`

Вызо  
в



Определен  
ие

# Функция



Заголовок  
Тело

```
void printline()  
{  
  for (int i = 0; i <=60; i++)  
    cout << '=';  
    cout << endl;  
}
```

# Передача аргументов в функцию

Передача констант в функцию.