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?

Алгори́тм — 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

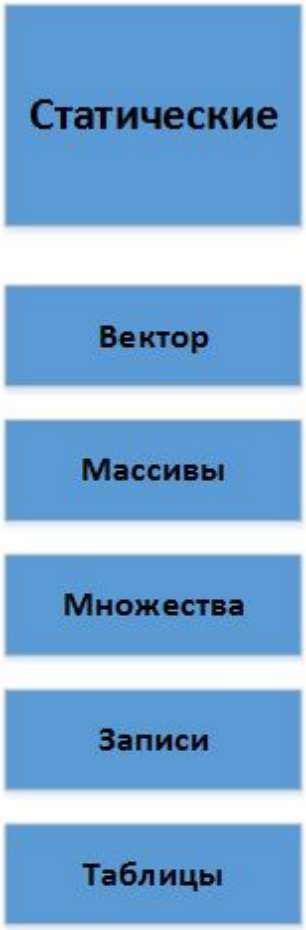
Изменчивость – изменение числа элементов и/или связей между элементами структуры.

- Статические
- Полустатические
- Динамические

СТРУКТУРЫ ДАННЫХ

Структуры данных

Структуры данных



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

Массив – последовательность элементов одного базового типа *

Множество – набор неповторяющихся данных одного и того же типа

Запись – конечное упорядоченное множество полей, характеризующихся различным типом данных

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

* Массив – вектор с индексами элементов

** Таблица – записи с индексами записей

Структуры данных

Последовательность, в которой включение и исключение элемента происходит с одной стороны

LIFO

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

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

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

Динамичес-кие

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**

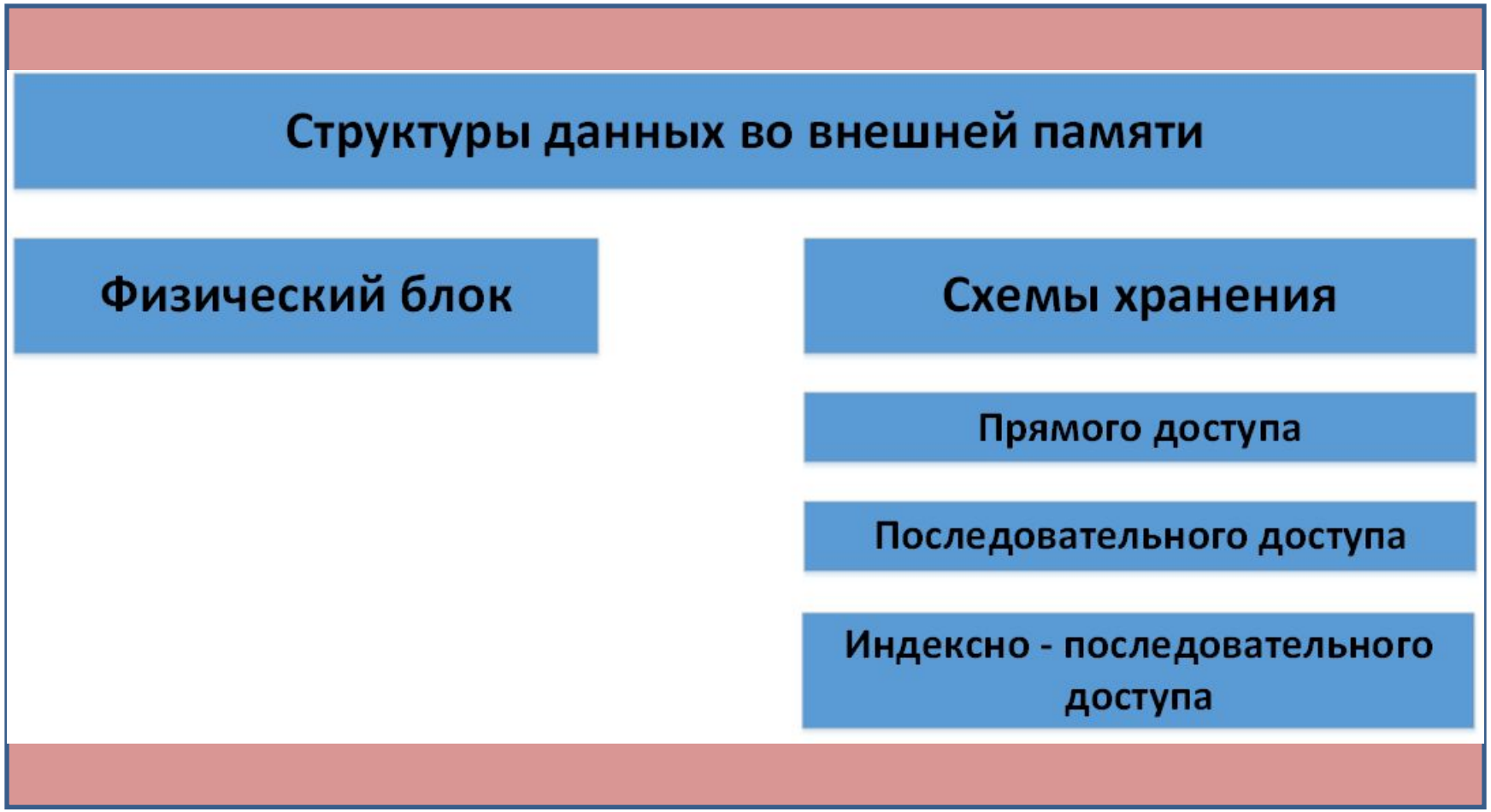
Структуры данных

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

- **Графы**
- **Деревья**
- **Плексы (сплетения)**

Структуры данных

Структуры данных



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*10 ^-38	3,4* 10 ^38	7	4
double	1,7*10^-308	1,7*10^308	15	8
long double	1,7*10^-308	1,7*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
bool, char, unsigned char, signed char, __int8	1 байт
__int16, short, unsigned short, wchar_t, __wchar_t	2 байта
float, __int32, int, unsigned int, long, unsigned long	4 байта
double, __int64, long double, long long	8 байт
__int128	16 байт

Типы данных

bool - требуется всего 1 бит!!!!
false , true

Размерность – 1 байт.*

*** Теоретически 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 переменных

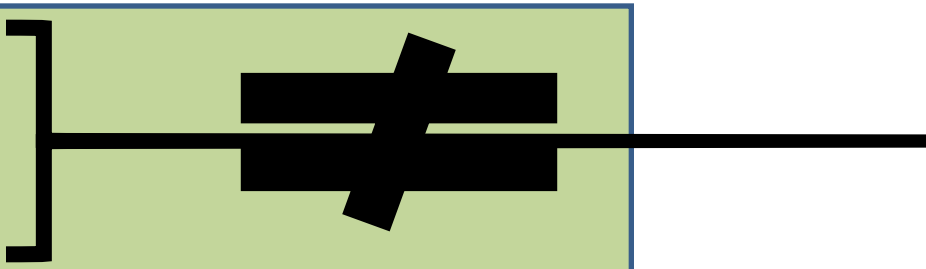
var1, var2 – идентификаторы

_var1, _var2 – идентификаторы

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;

Операторы ПРИСВАИВАНИЯ И ВЫВОД

« = »

```
cout << “var1 + 2 равно ”;
cout << var2 << ‘\n’;
```

```
cout << var2 << endl;
```

Другие целые типы

int – размер аппаратно-зависимый

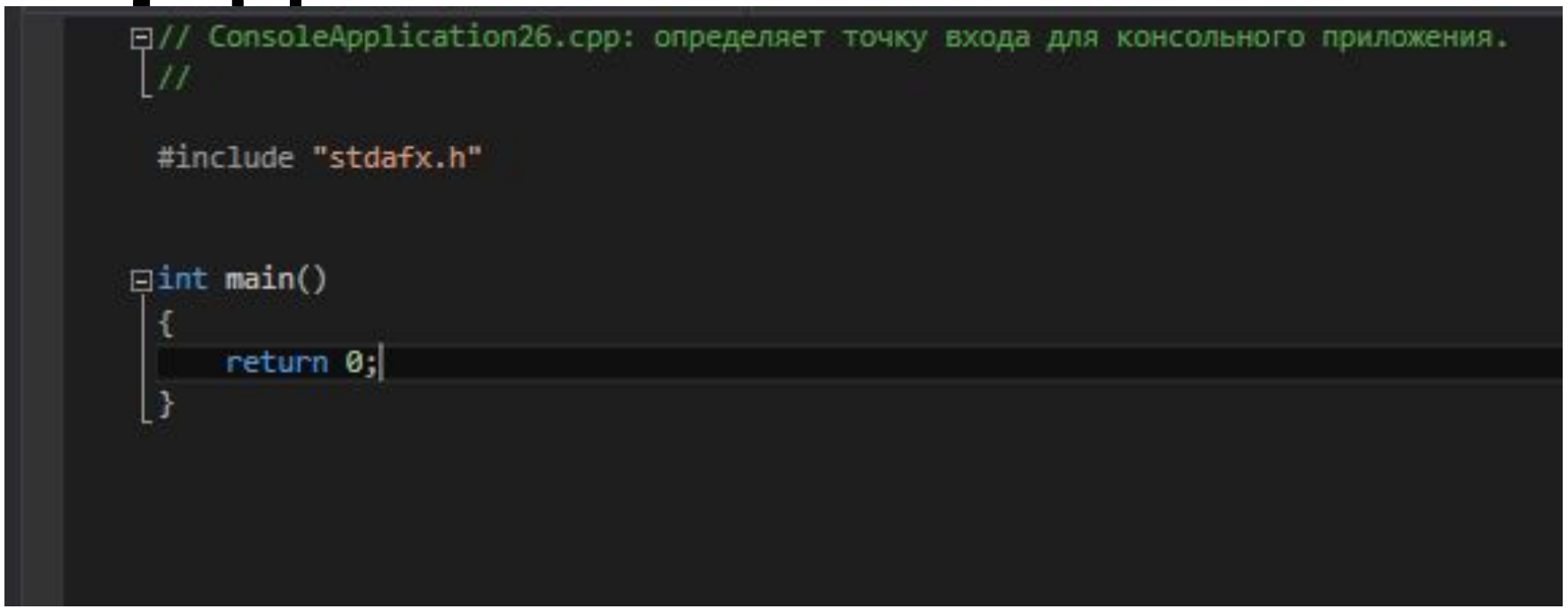
long, short – фиксированный размер *

long, short	⟷	**int – для 32-разрядных систем**
long	⟷	**long – для 16-разрядных систем**
int	⟷	**short – для 16-разрядных систем**

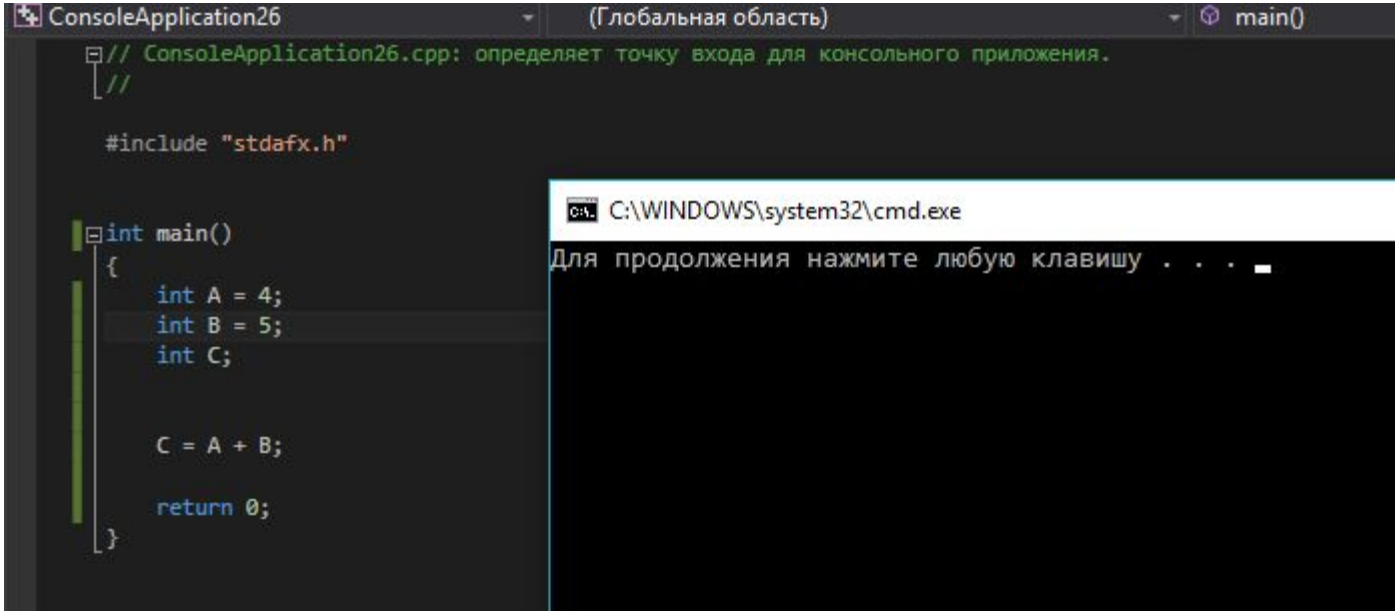
* Размер типа long всегда равен 4 байтам и совпадает с размером типа int в случае 32-разрядных систем, подобных Windows. Это означает, что диапазон значений типа long совпадает с диапазоном типа int: от -2 147 483 648 до 2 147 483 647. Тип long может быть описан как long int между двумя такими описаниями 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

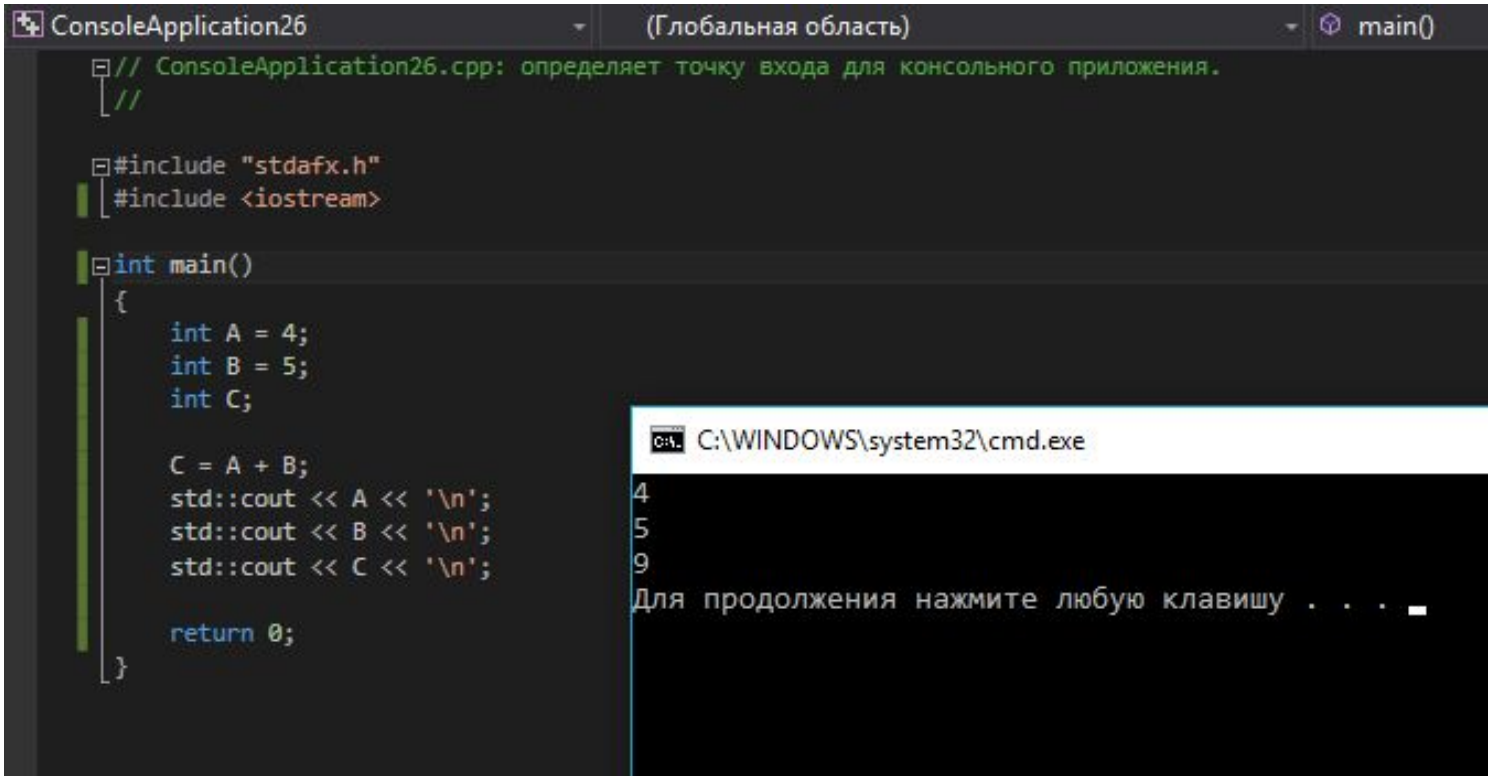
Структура программы C++



Структура программы

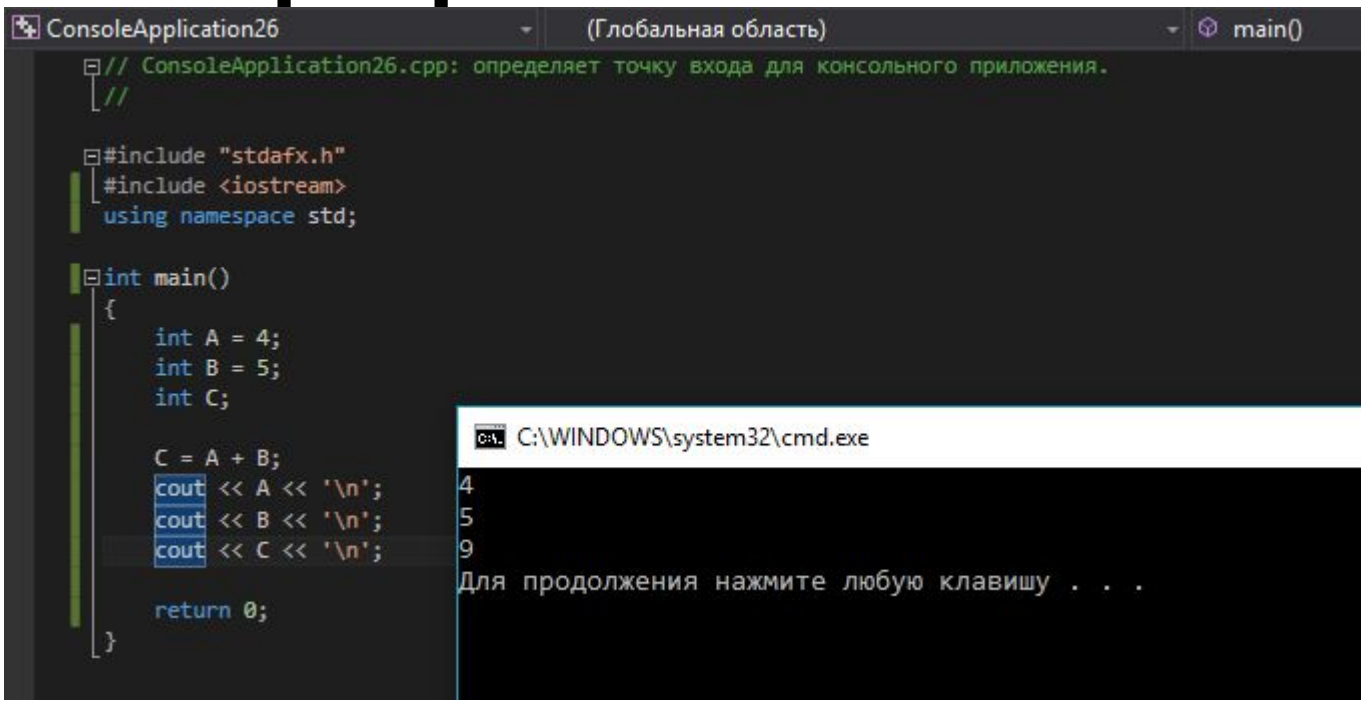


Структура

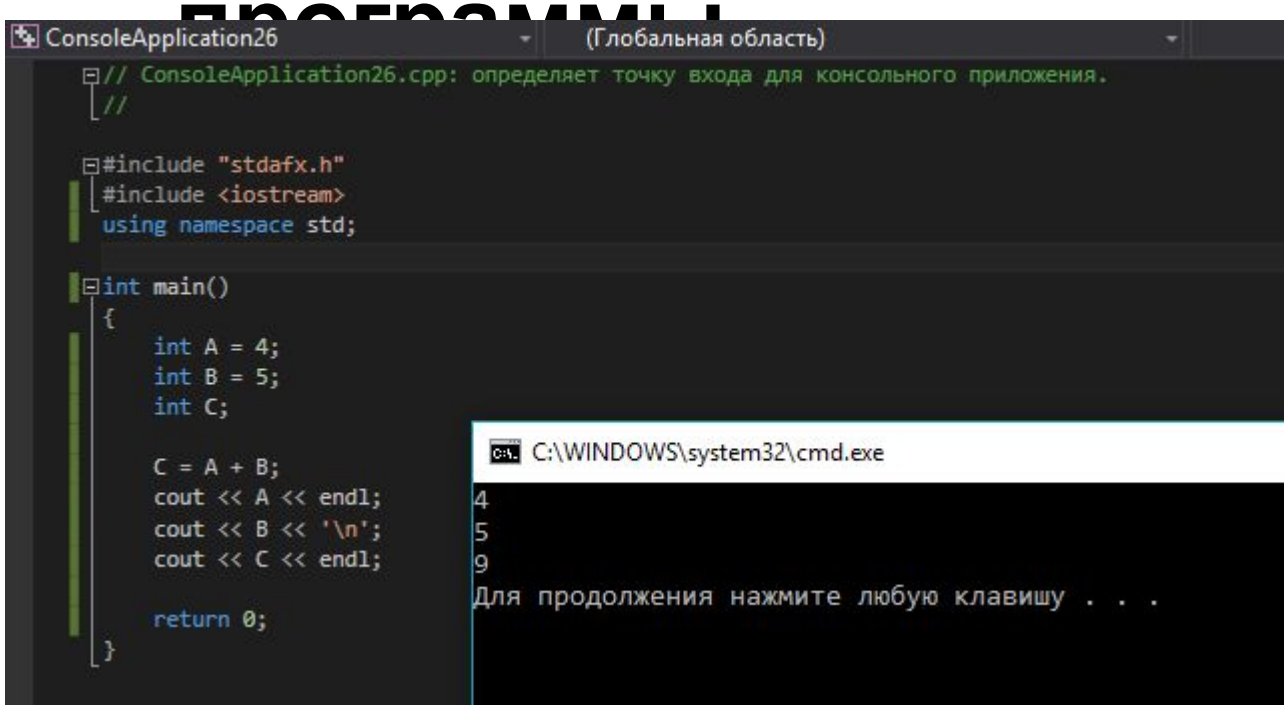


std :: cout << A << ‘\n’;

Структура программы



Структура программы



‘\n’ = endl

Манипулятор endl связан с очисткой выходного буфера

Структура программы

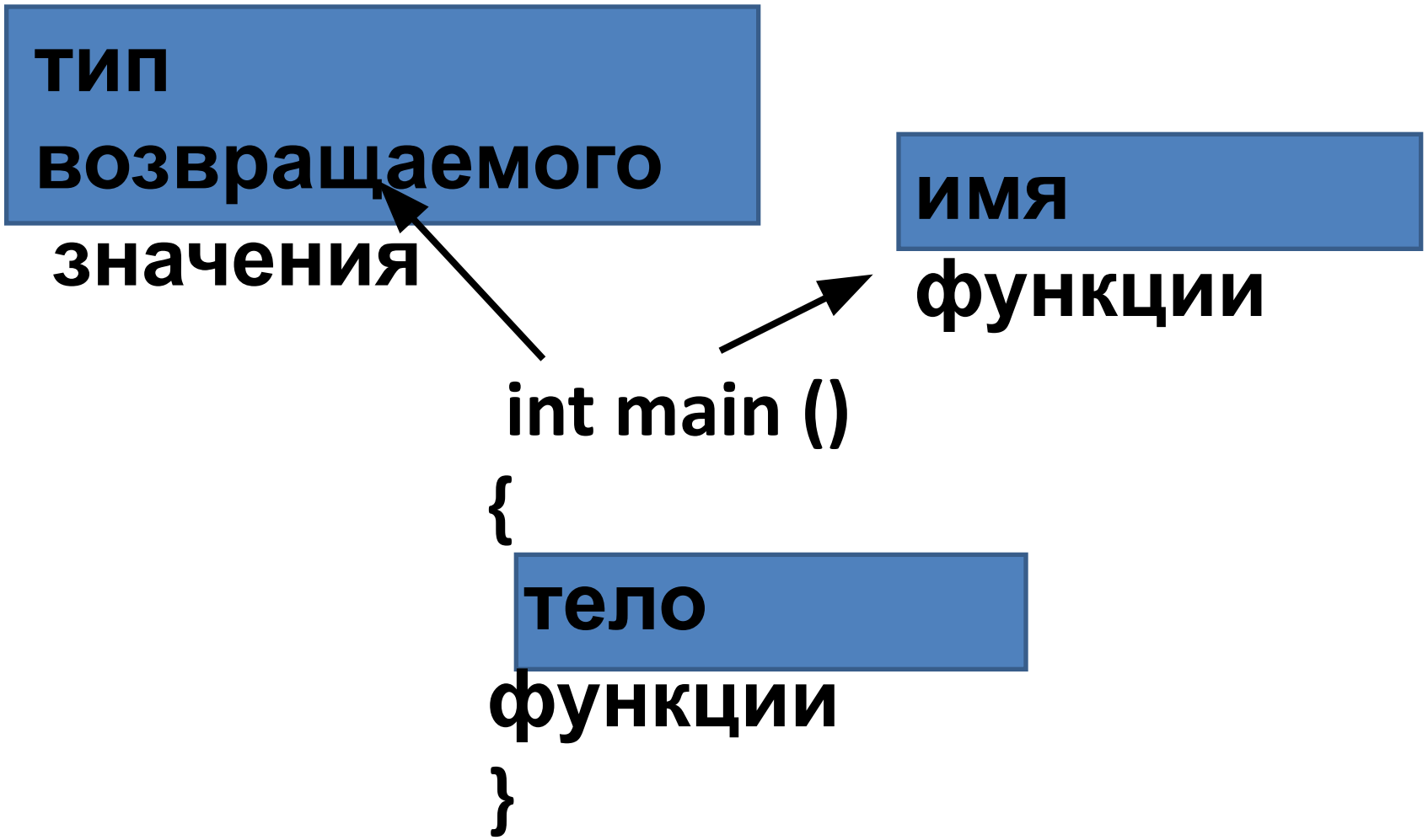
#include – директива препроцессора

#include <iostream>

iostream = iostream.h

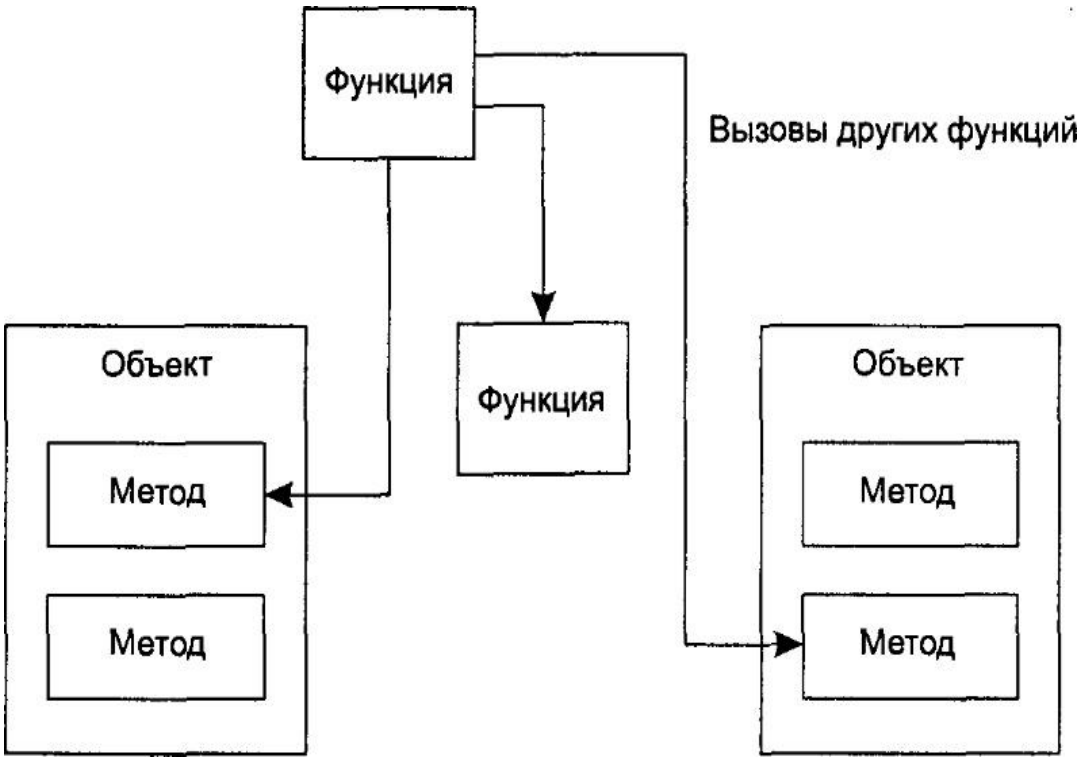
using namespace std; - директива

Структура программы



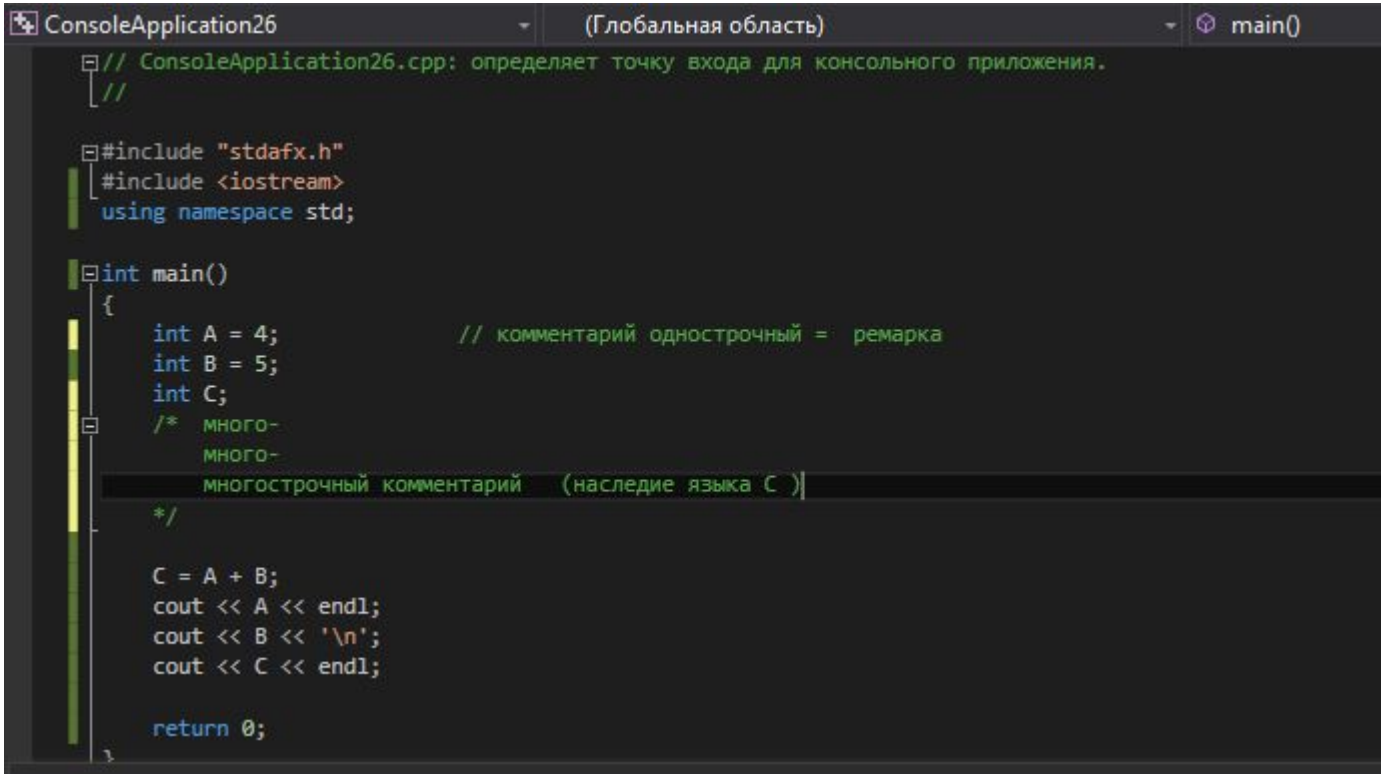
Структура программы

функция main()



Структура программы

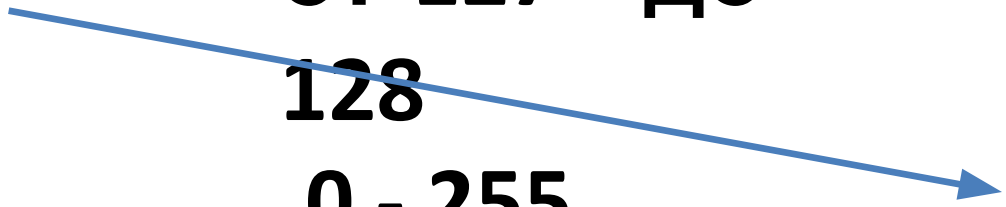
написание комментариев



Символьные переменные

char от-127 до 128 — 1 байт

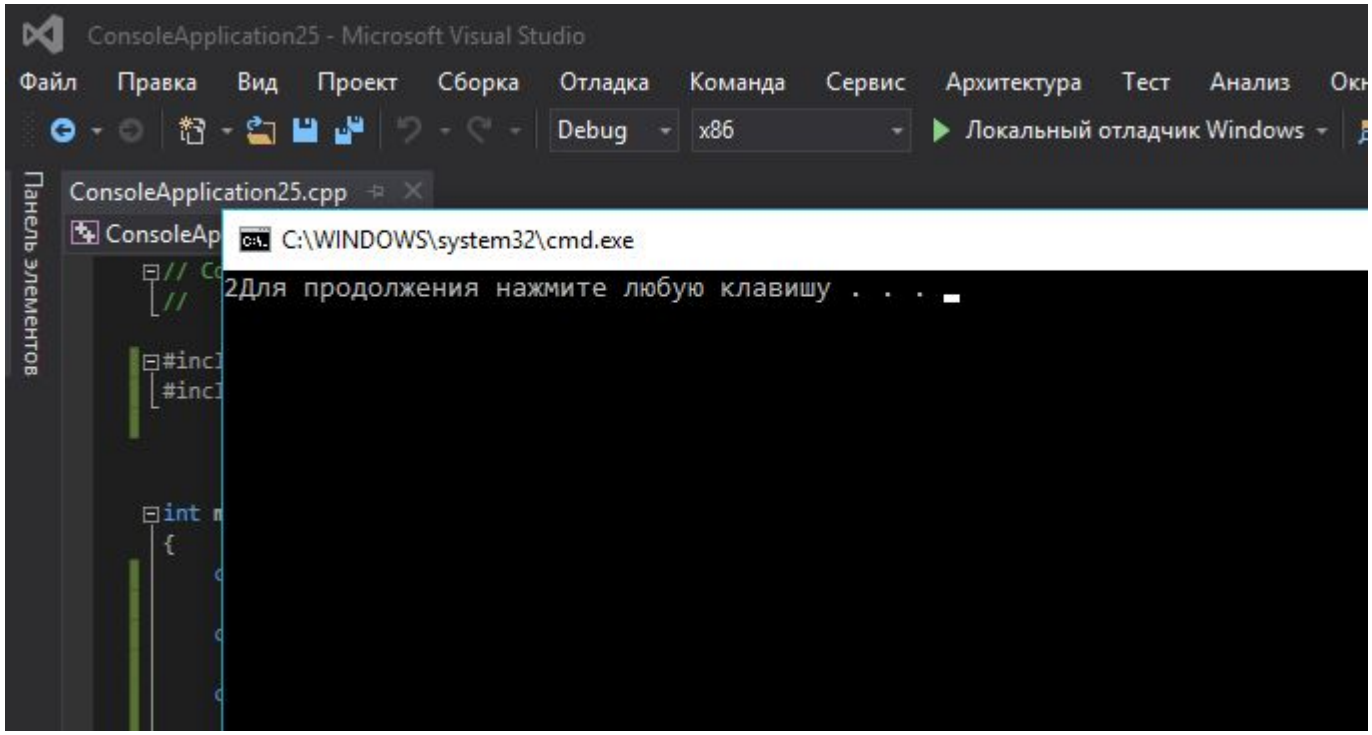
ASCII 0 - 255 — wchar_t

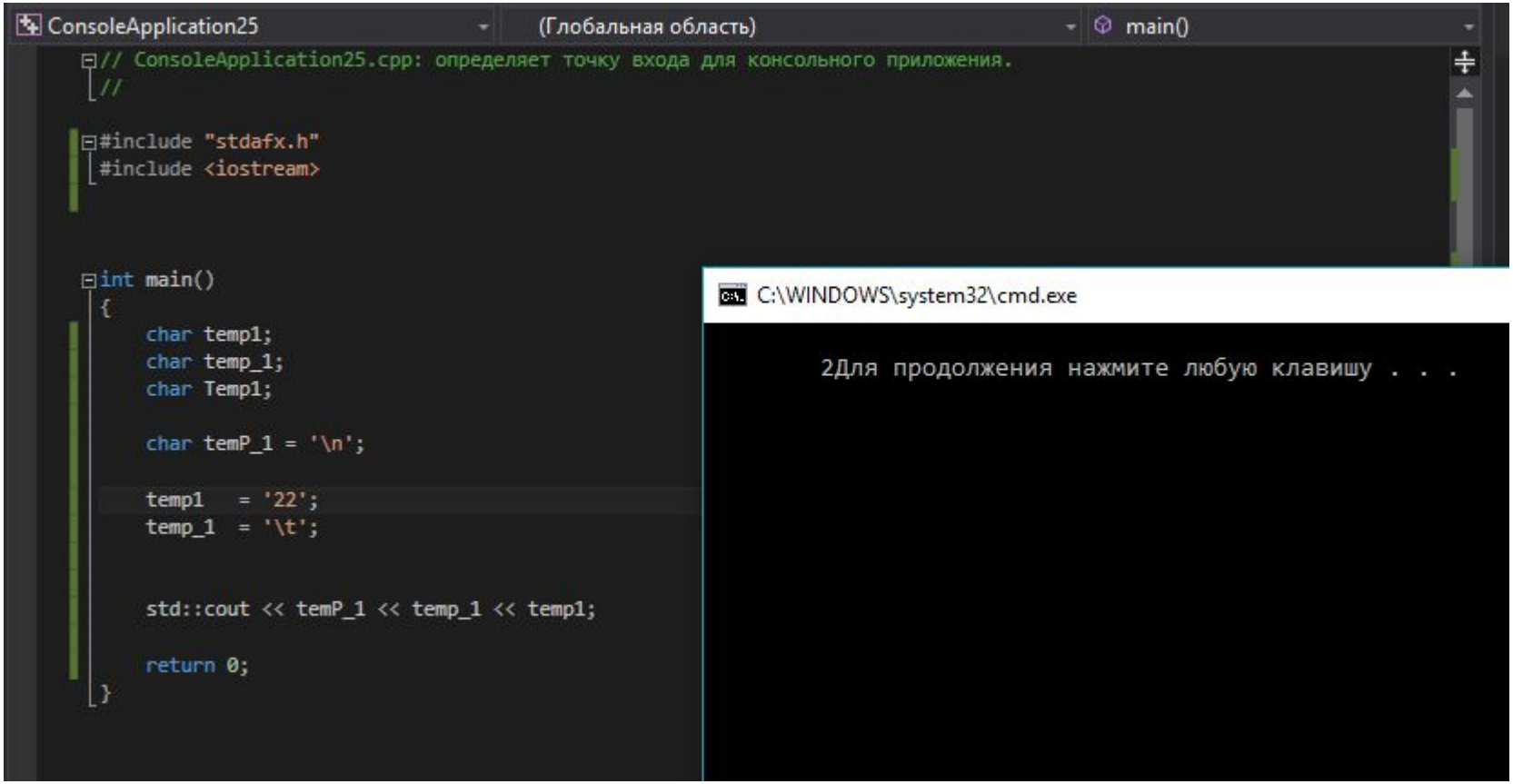


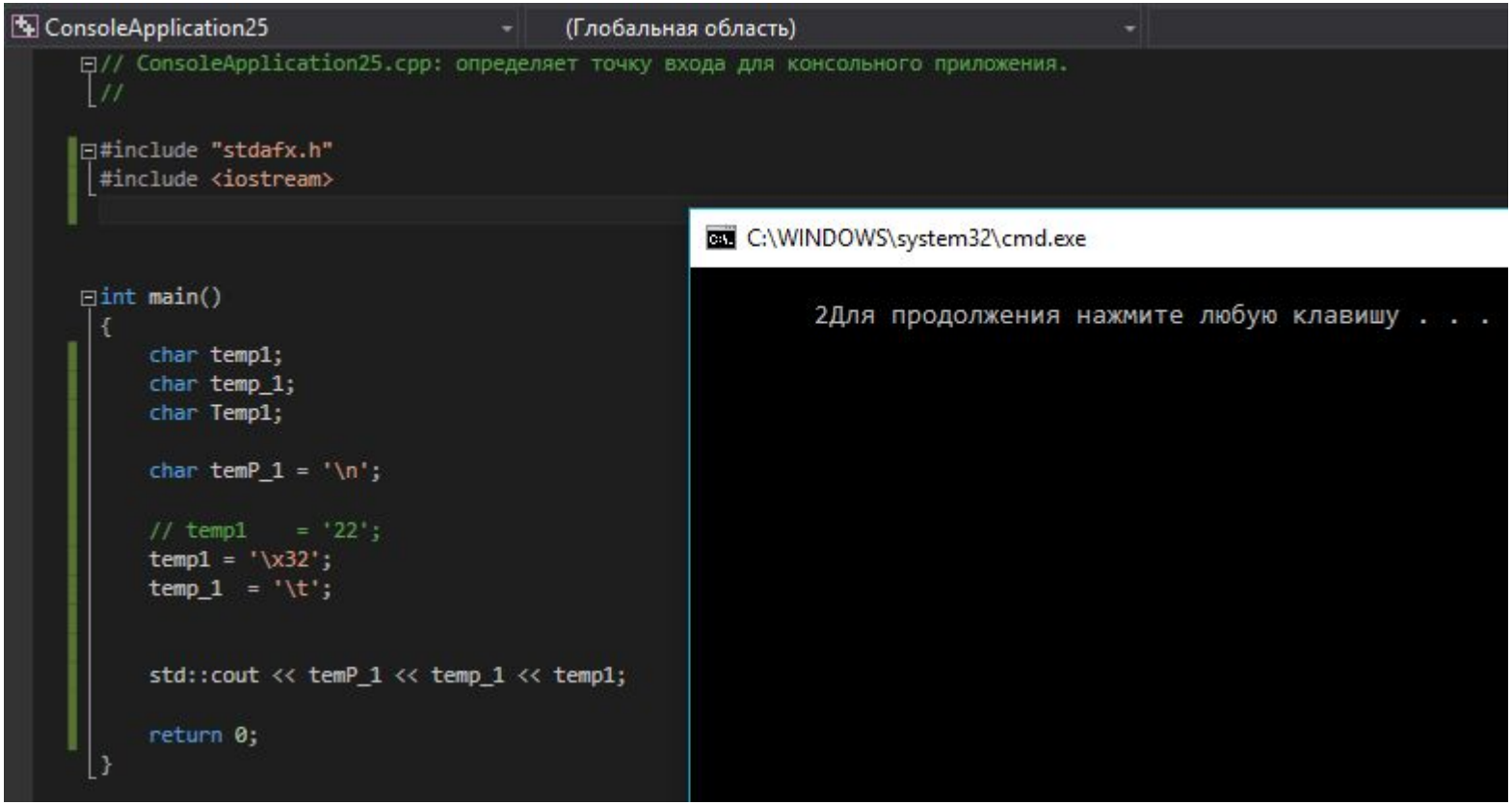
Символьные константы

```
char var1 = ‘2’;
char var2;
var2 =  ‘2’;
```

```cpp
#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 - Microsoft Visual Studio
Файл Правка Вид Проект Сборка Отладка Команда Сервис Архитектура Тест Анализ Окн
Debug
x86
Локальный отладчик Windows
Панель элементов
ConsoleApplication25.cpp
C:\WINDOWS\system32\cmd.exe
2Для продолжения нажмите любую клавишу . . .


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
2Для продолжения нажмите любую клавишу . . .



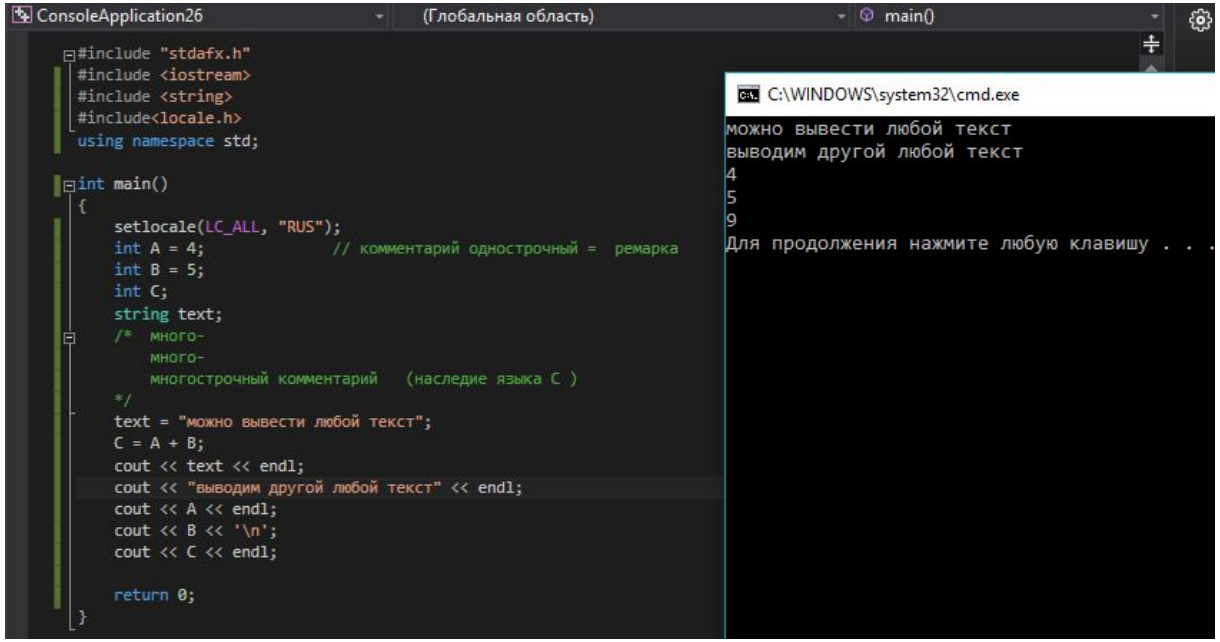
Каскадирование операции “<<“

Управляющие последовательности

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> “




```
#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;
}
```

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

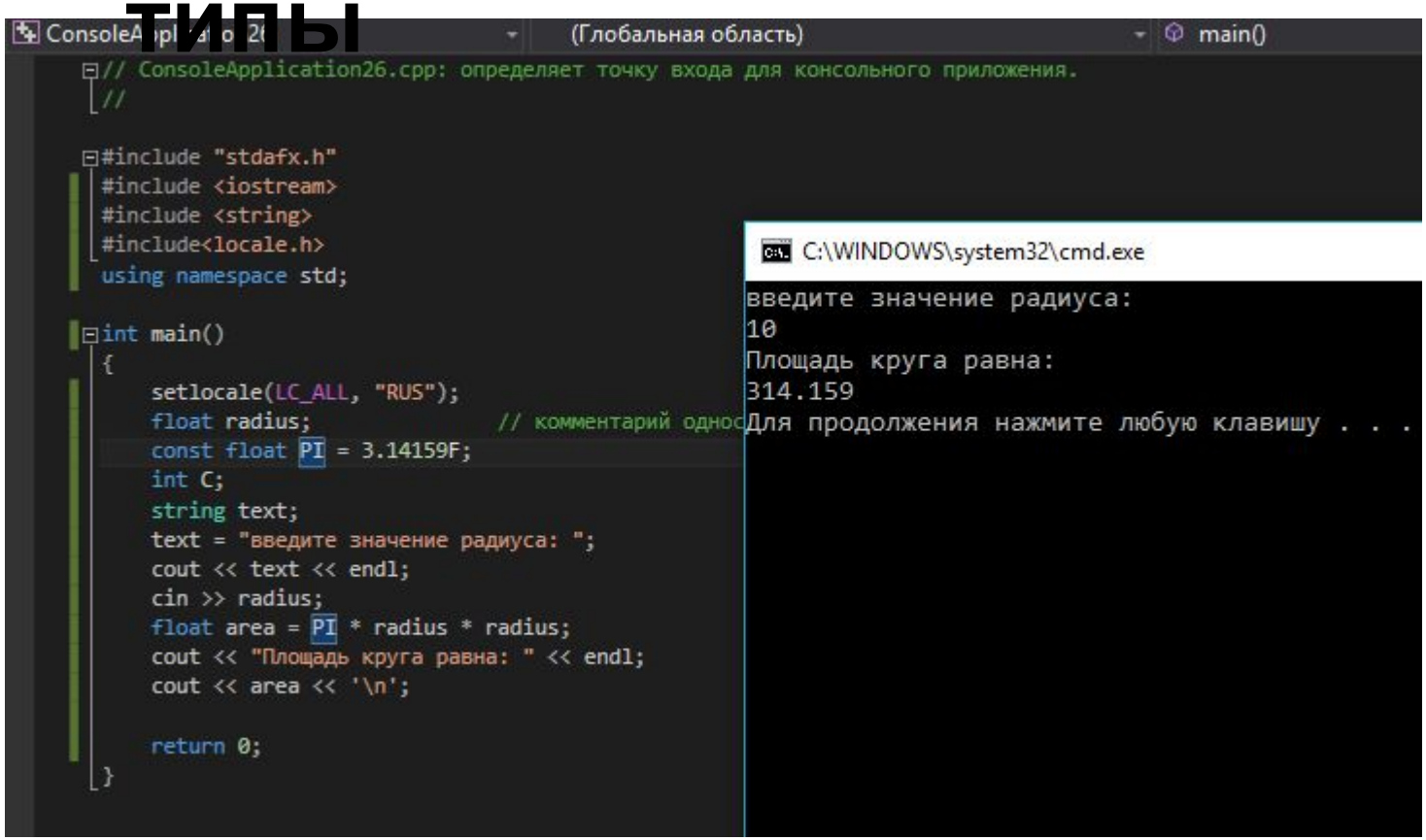
В одном операторе могут присутствовать несколько выражений

Структура программы

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

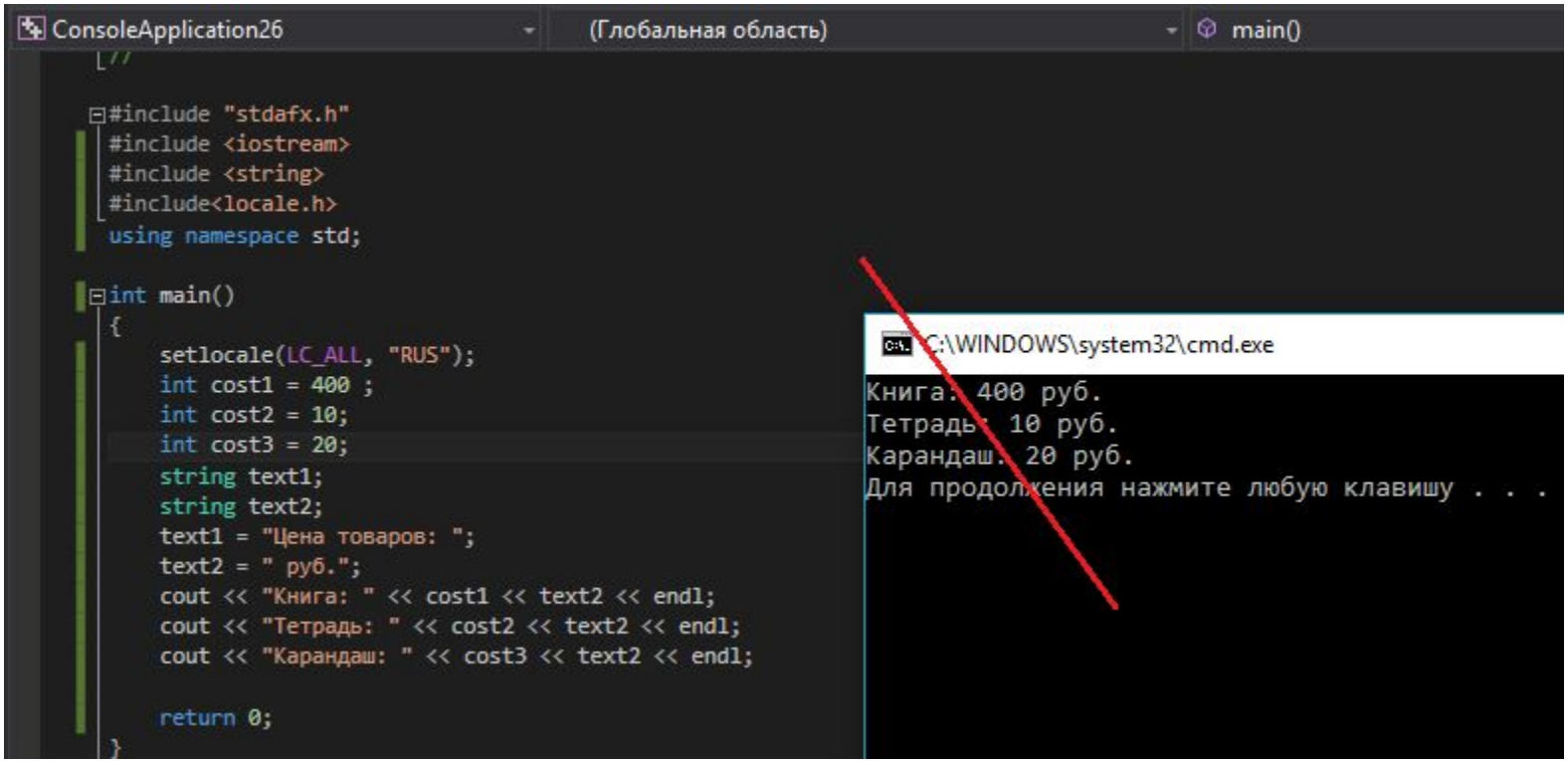
Переменные вещественного типа предназначены для хранения вещественных чисел — тех чисел, которыми измеряются непрерывные величины: температура, расстояние, площадь. Вещественные числа, как правило, имеют ненулевую дробную часть.

Вещественные типы

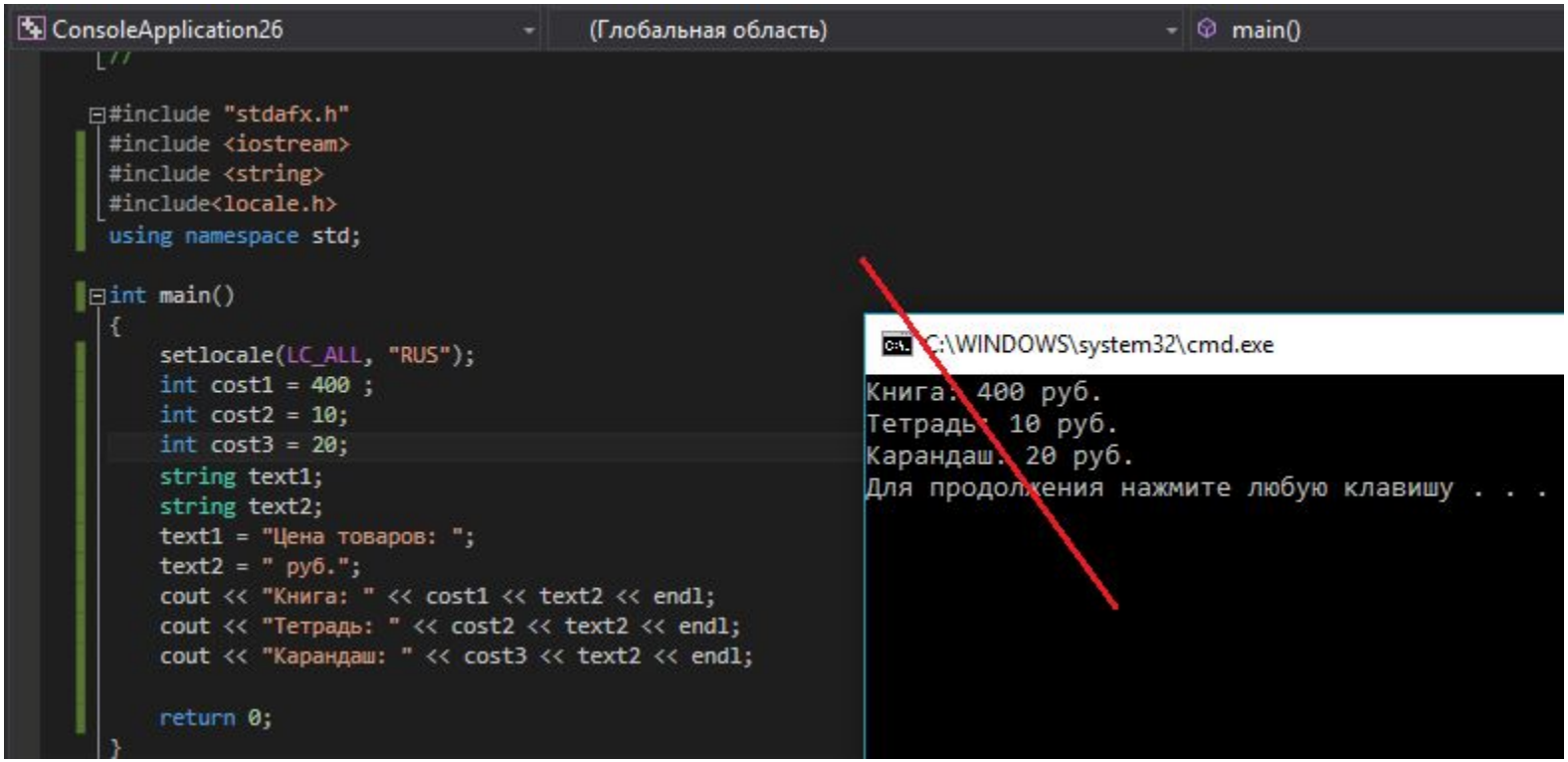


Префикс const гарантирует, что программа не сможет случайно изменить значение переменной

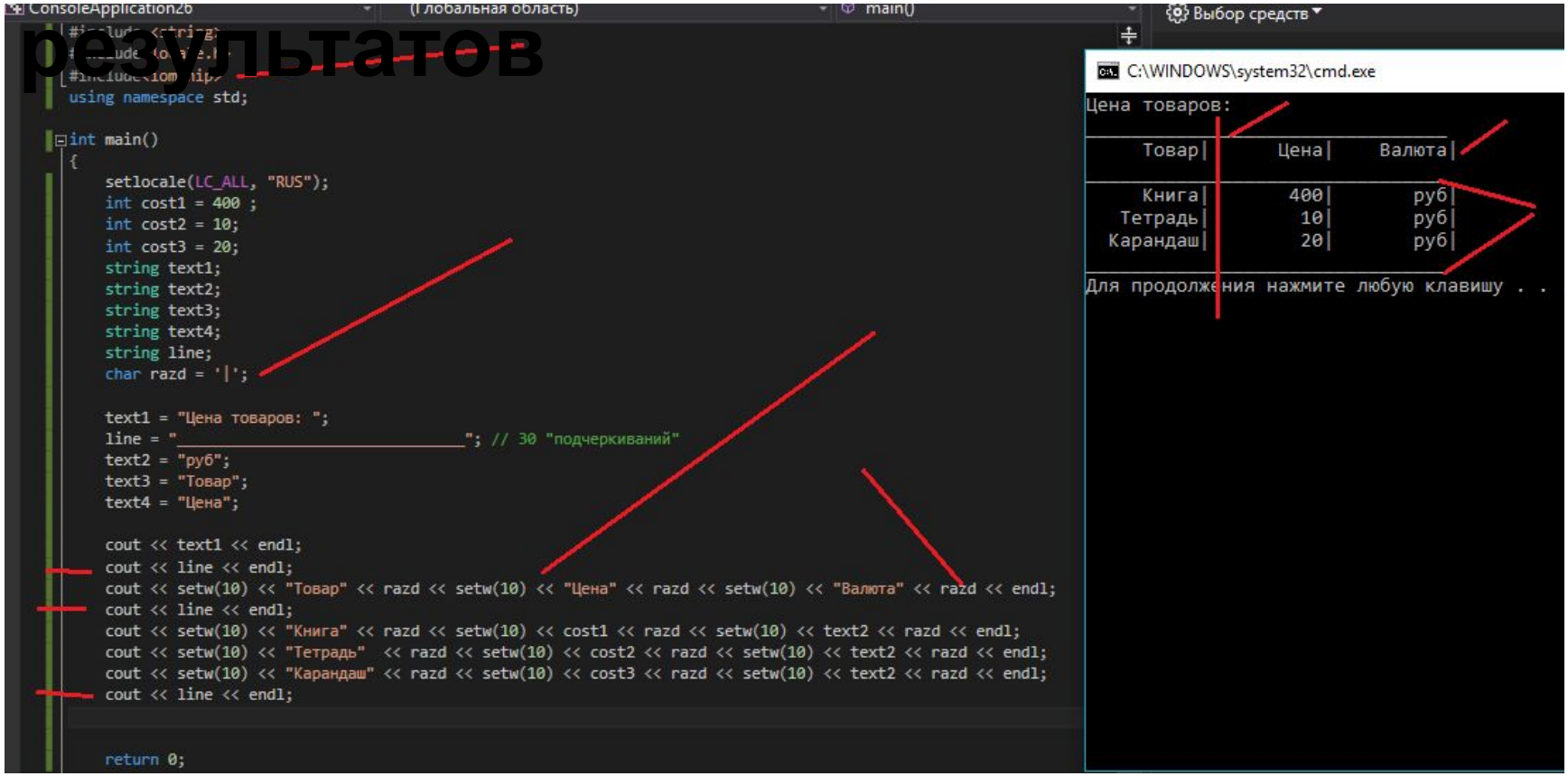
Вывод результатов



Вывод результатов

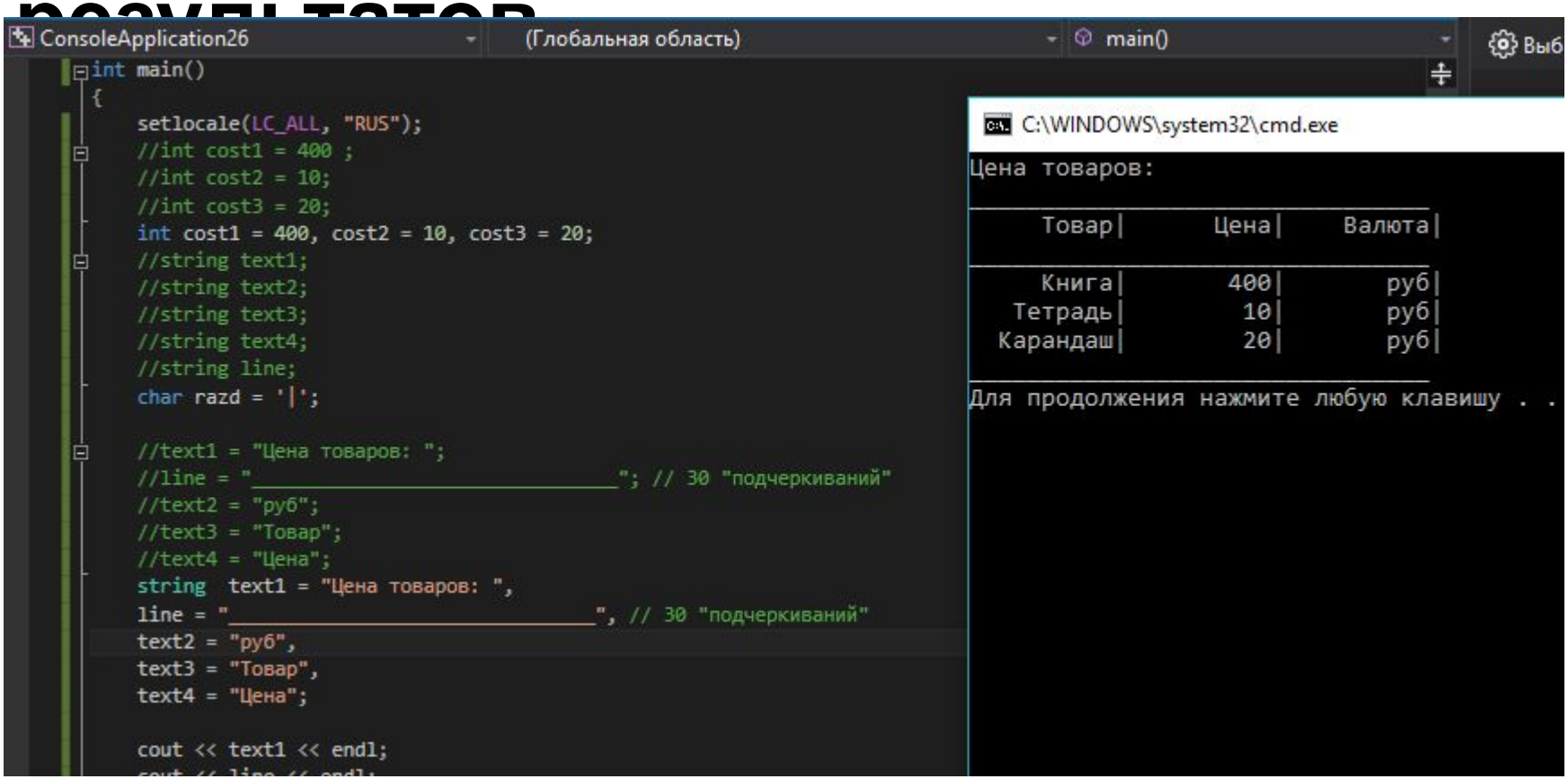


Вывод результатов

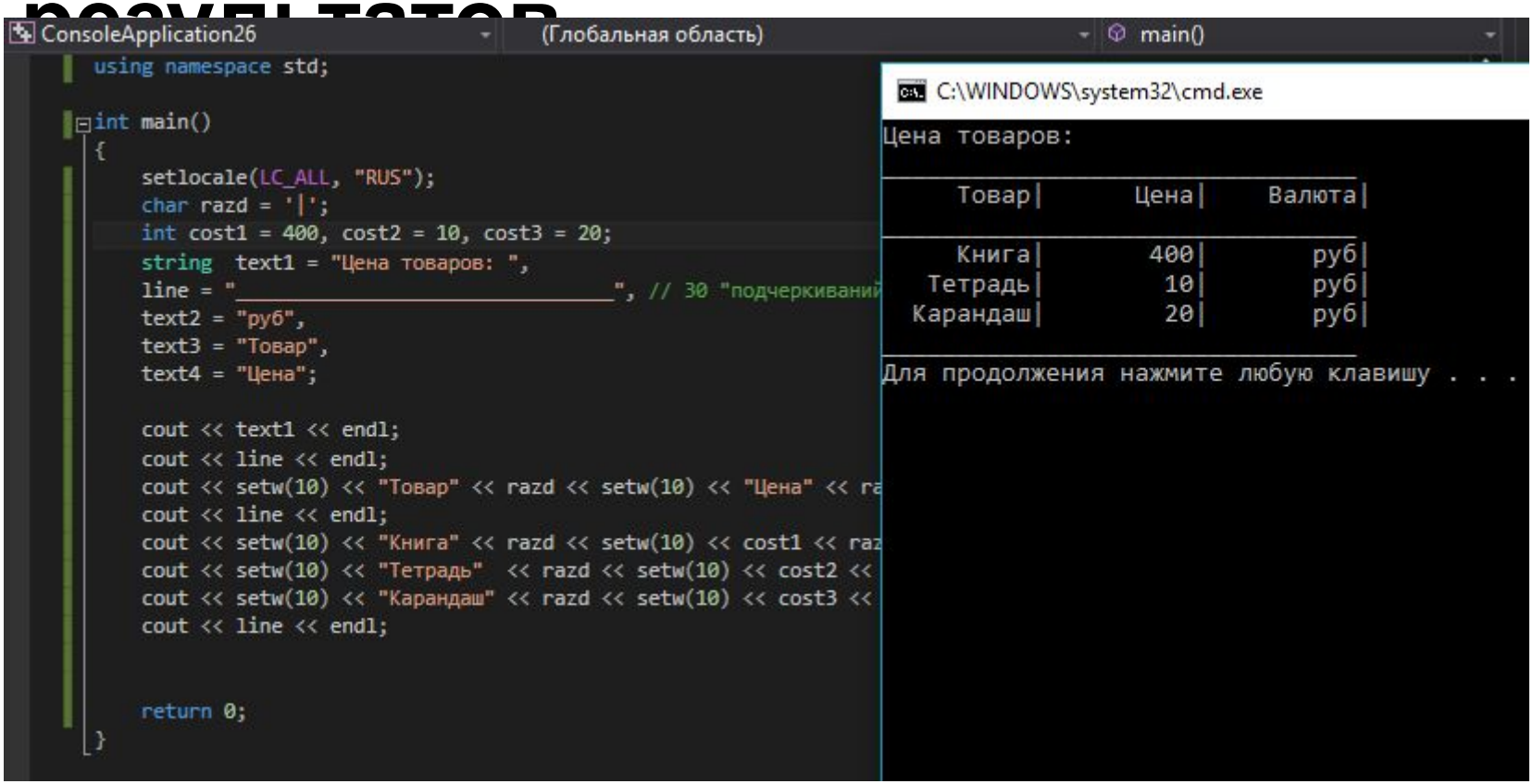


#include<iomanip>
setw(N) N – размерность «поля»

Вывод результатов



Вывод результатов



Использовали множественное определение

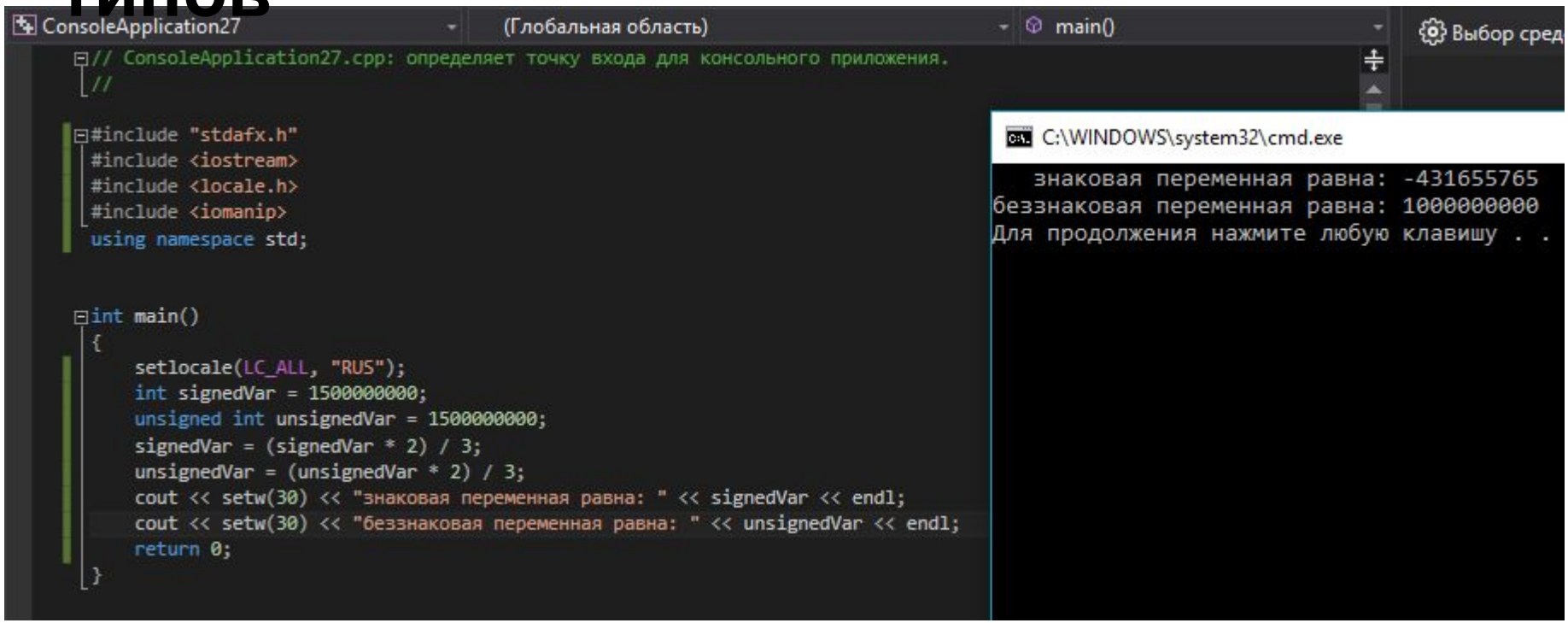
Типы данных

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	$3.4*10^{-38}$	$3.4*10^{38}$	7	4
double	$1.7*10^{-308}$	$1.7*10^{308}$	15	8

Беззнаковые типы данных

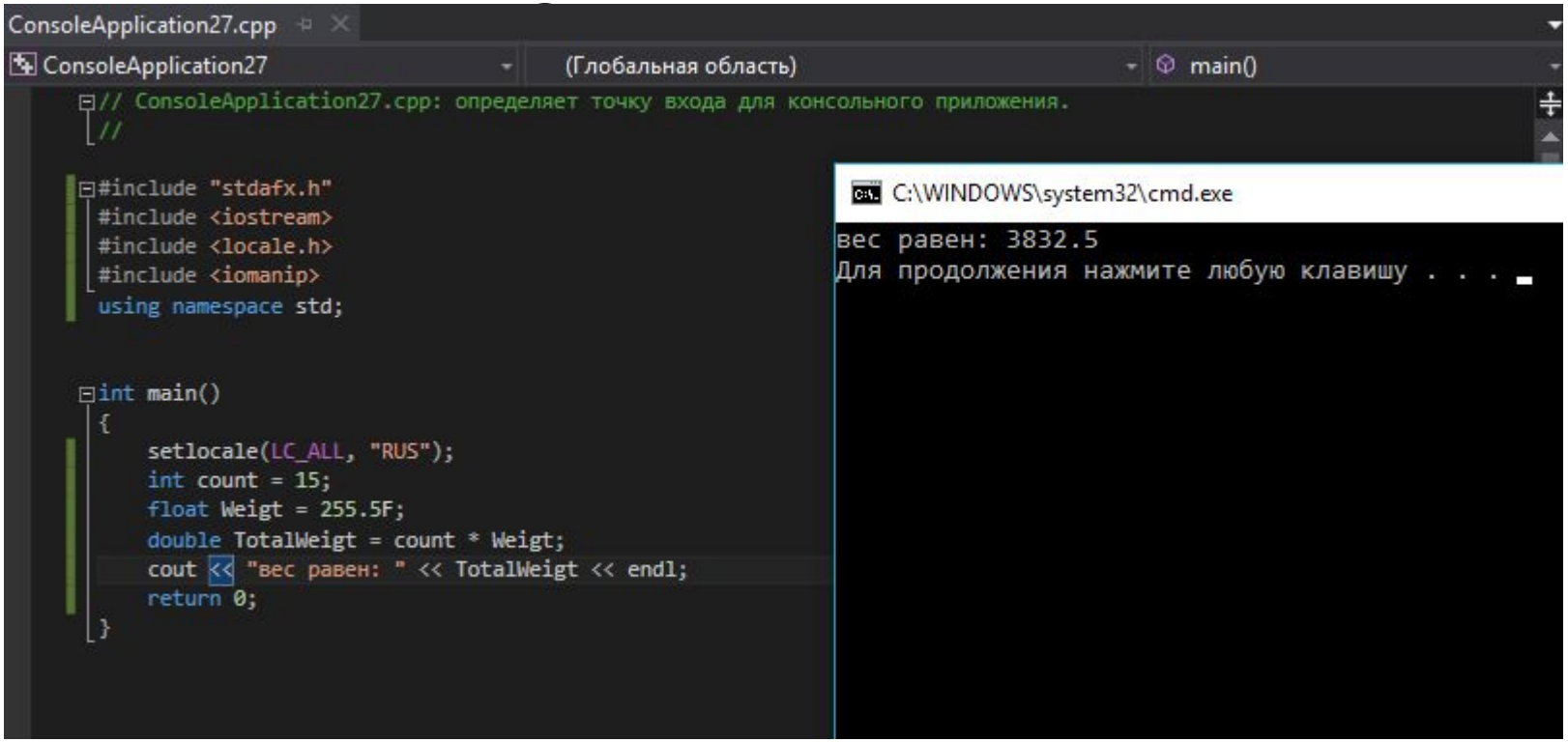
Беззнаковые целые типы			
Название	Нижняя граница диапазона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned short	0	65 535	2
unsigned int	0	4 294 967 295	4
unsigned long	0	4 294 967 295	4

Использование знаковых и беззнаковых типов



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

Преобразования



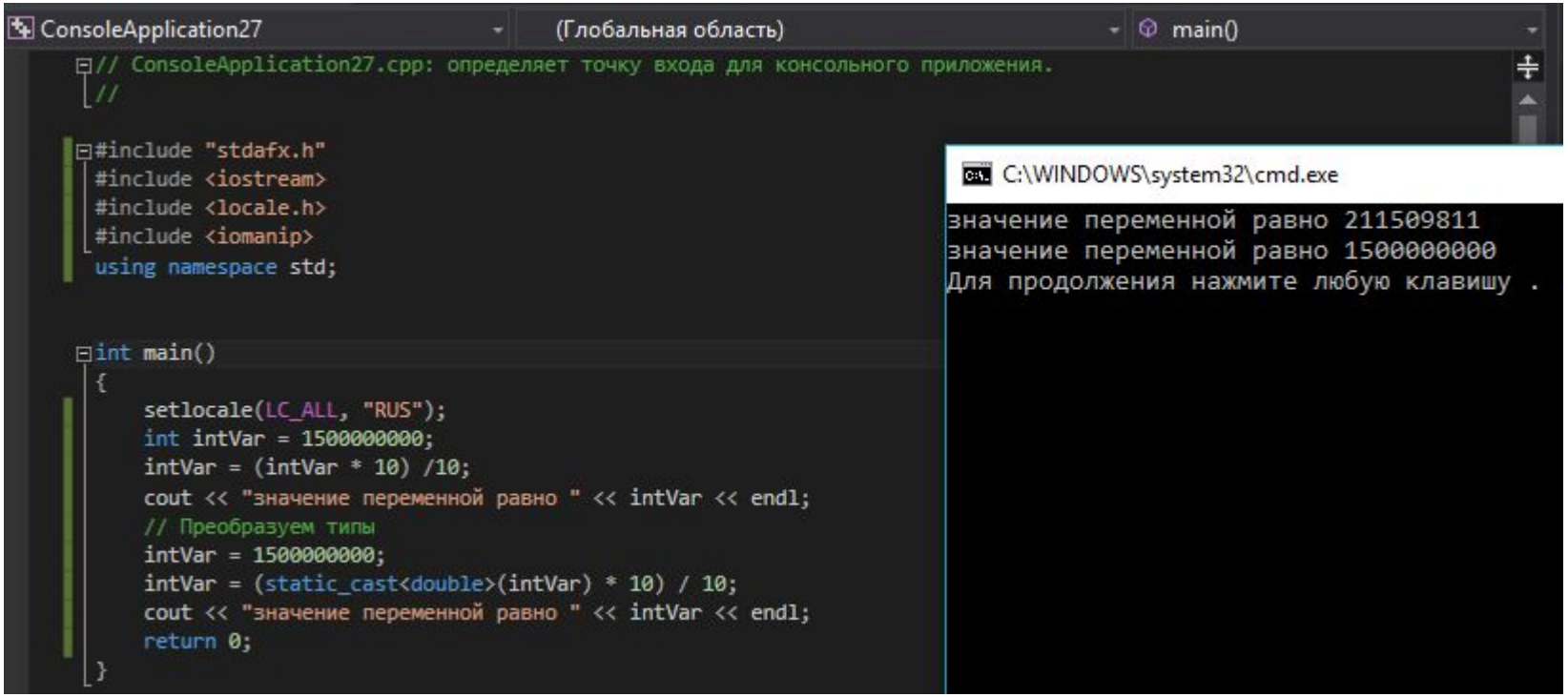
Неявные преобразования типов

Иерархия типов

Тип данных	Старшинство
long double	Высший
double	
float	
long	
int	
short	
char	Низший

Явные преобразования типов

intVar = static_cast<double>(intVar);



Приведения типов – только в осознанных случаях!!!

Арифметические операции

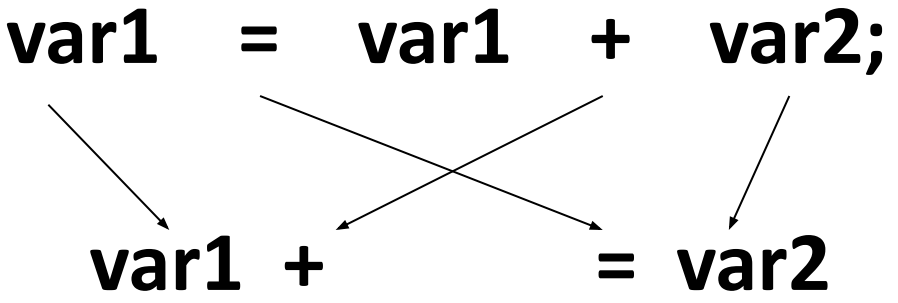
“+” – сложение;
“-” – вычитание;
“*” – умножение;
“/” – деление.

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



‘-=‘

‘*=‘

‘/=‘

‘%=‘

Инкремен

Т

var1 = var1 + 1

var1 += 1

++var1

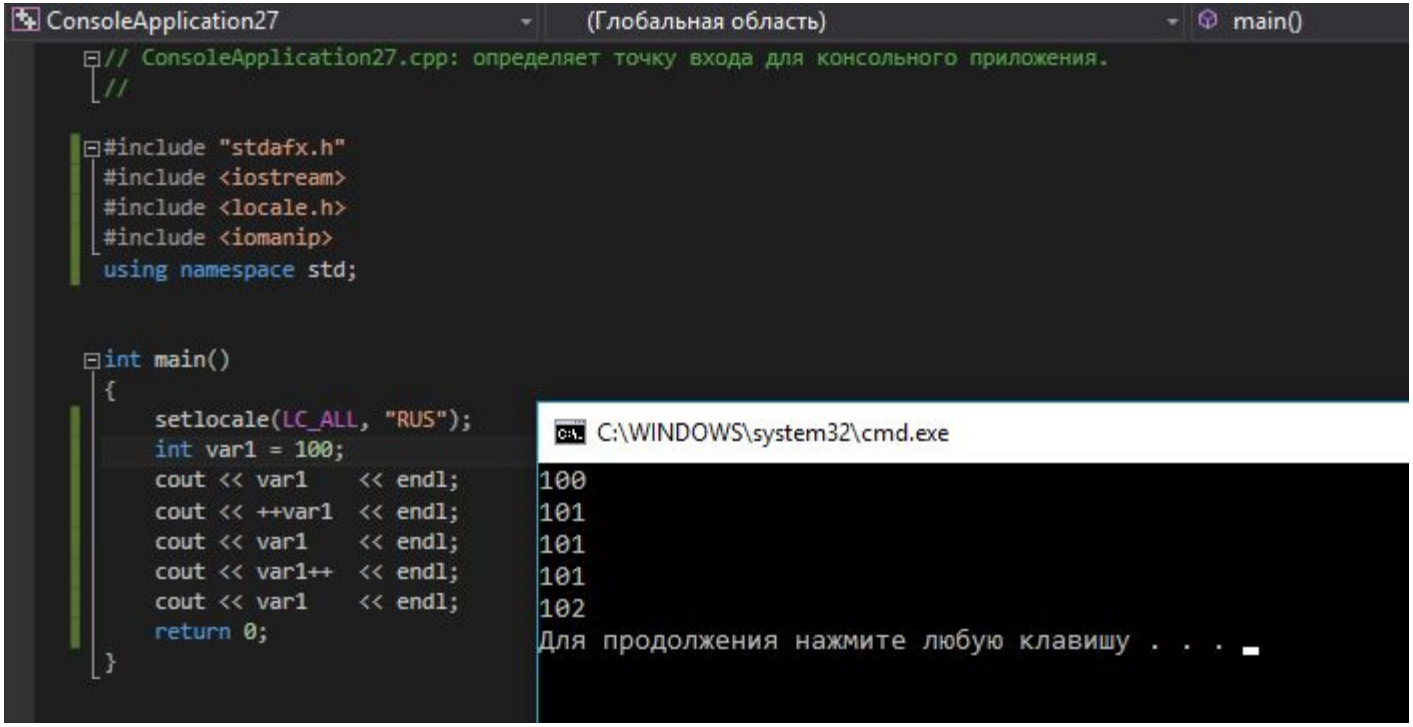
Префиксная форма

++var1

Постфиксная форма

var1++

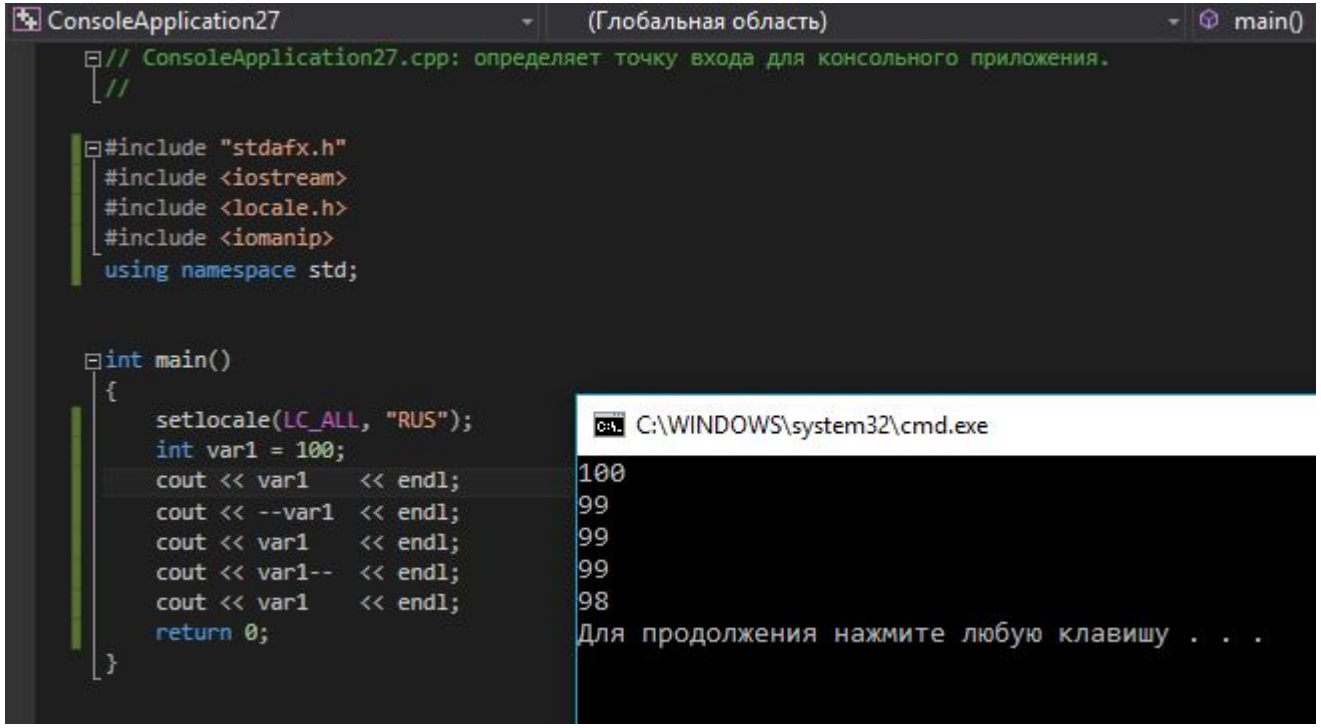
Инкремент


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

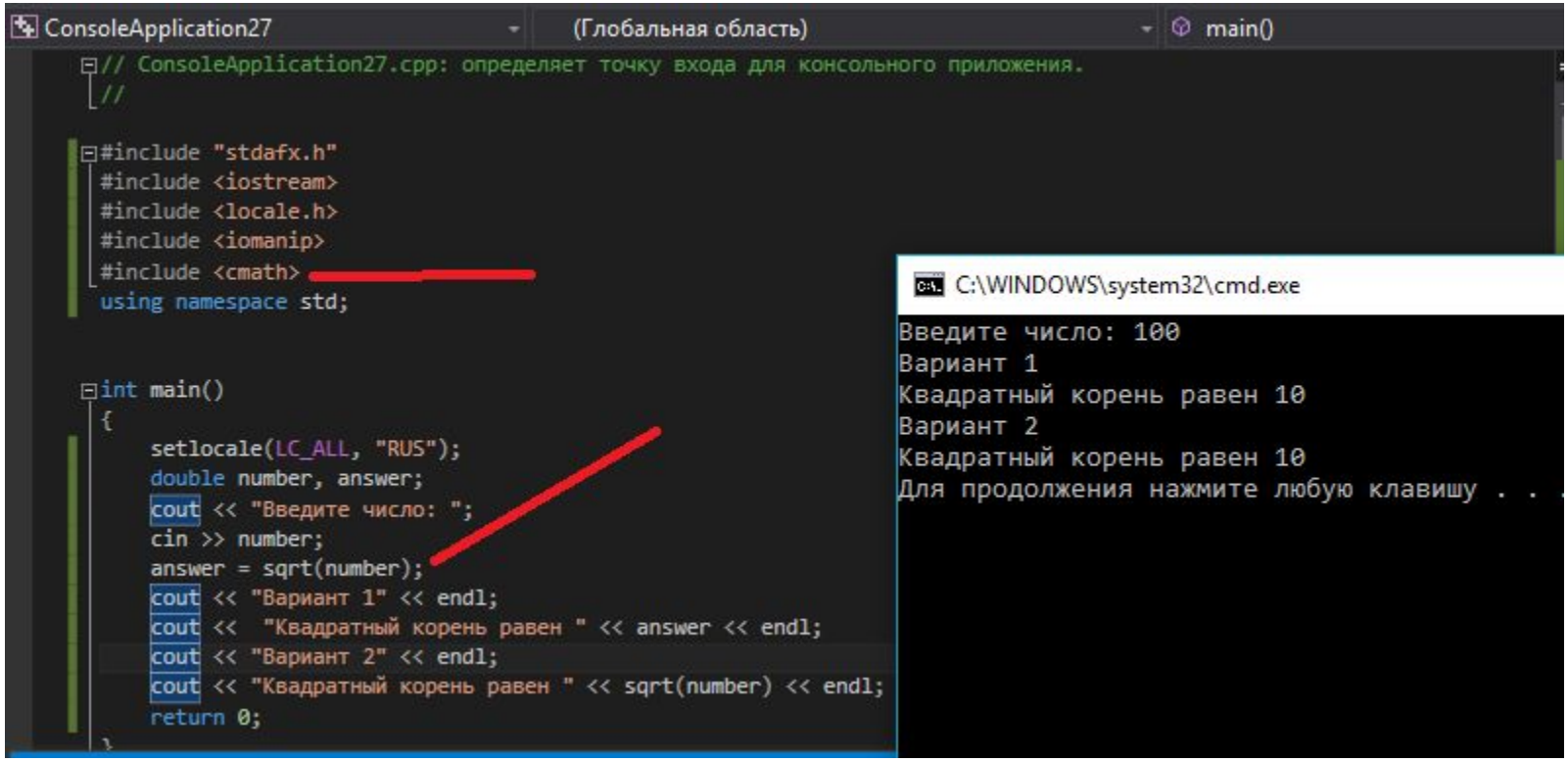
Декремен

--var1;

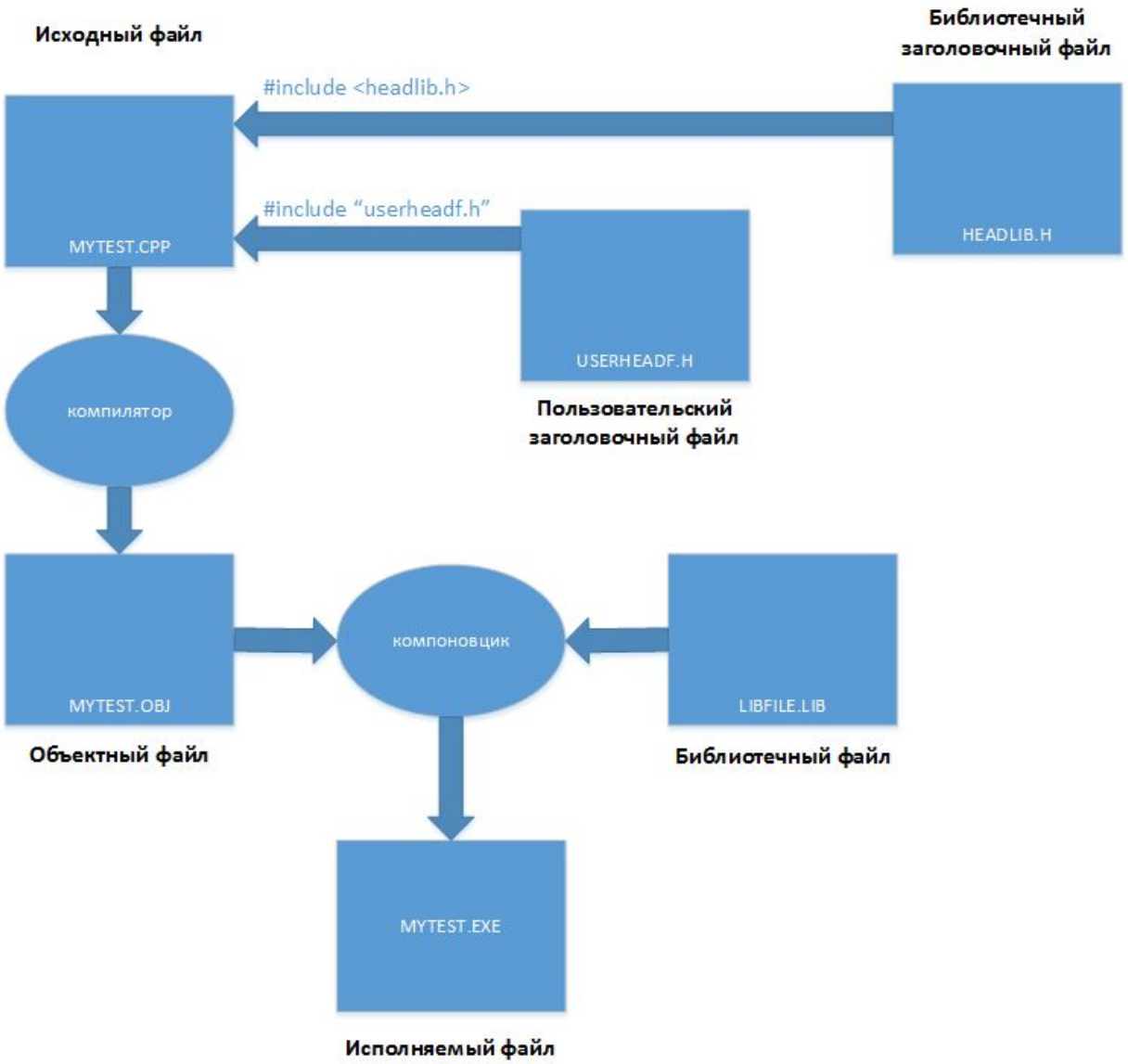
var1--;



Библиотечные



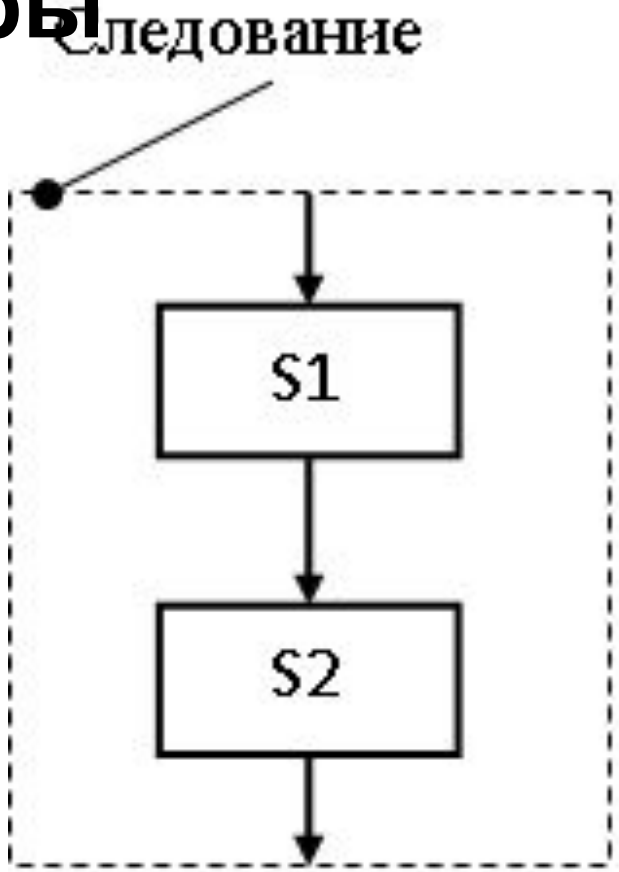
Заголовочные и библиотечные файл



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

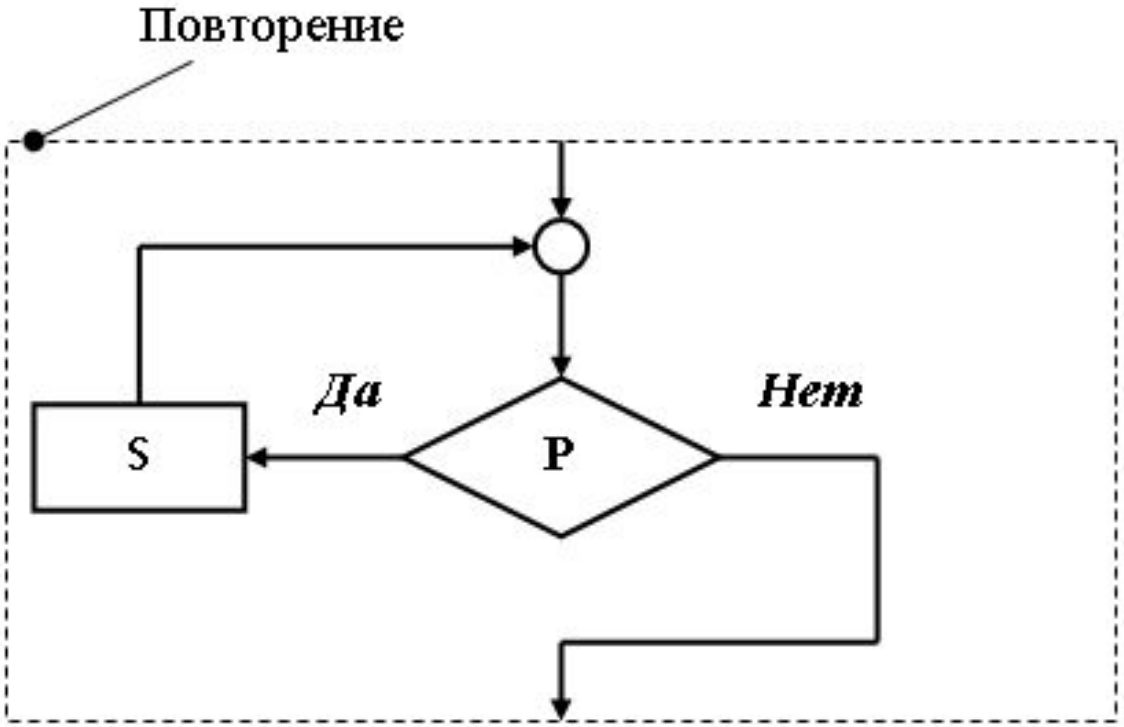
ConsoleApplication27 (Глобальная область)

```
#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

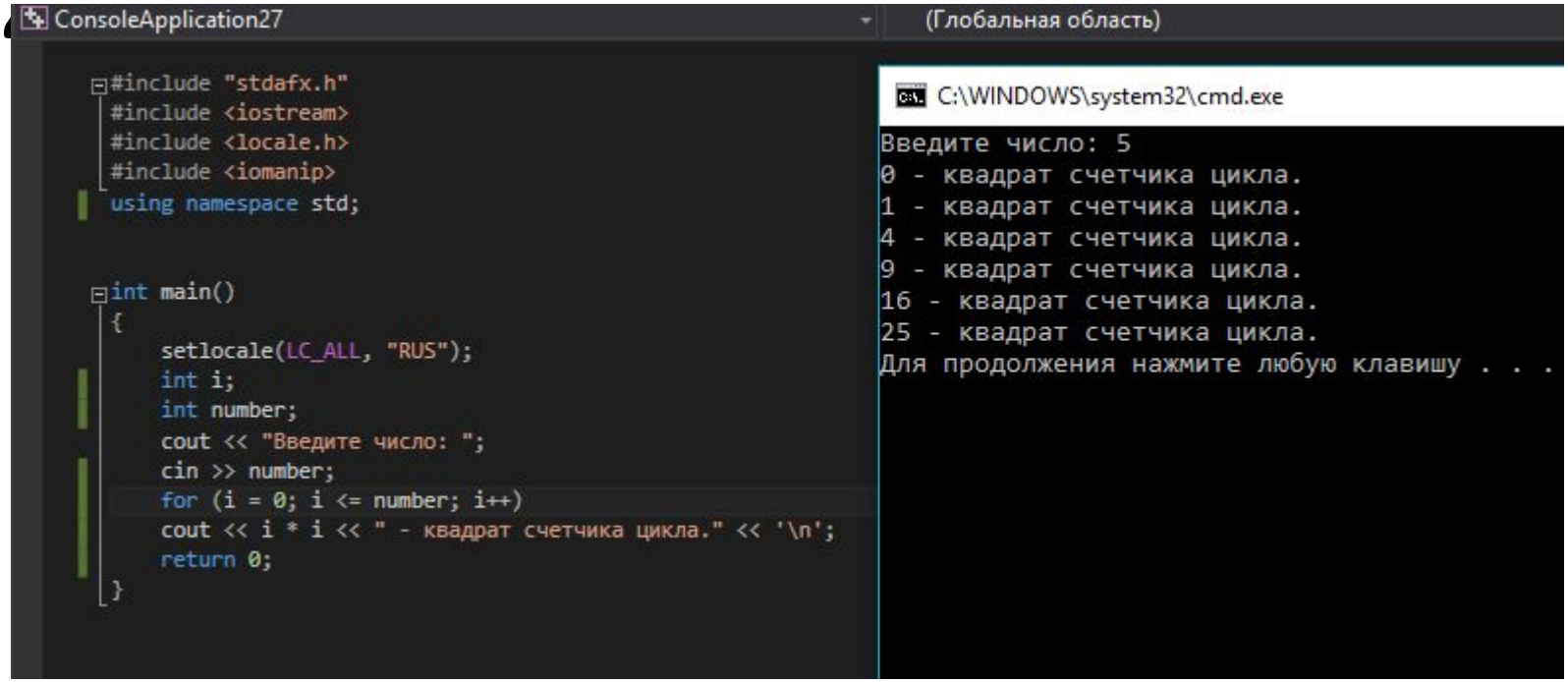
```
Введите число: 10
10 < 10 равно 0
10 <= 10 равно 0
10 > 10 равно 0
10 >= 10 равно 0
10 == 10 равно 1
Для продолжения нажмите любую клавишу . .
```

Циклы



- for
- while
- do

Циклы



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

i = 0 инициализирующее
i <= number условие проверки
i++ инкрементирующее

Счетчик i определен в теле цикла

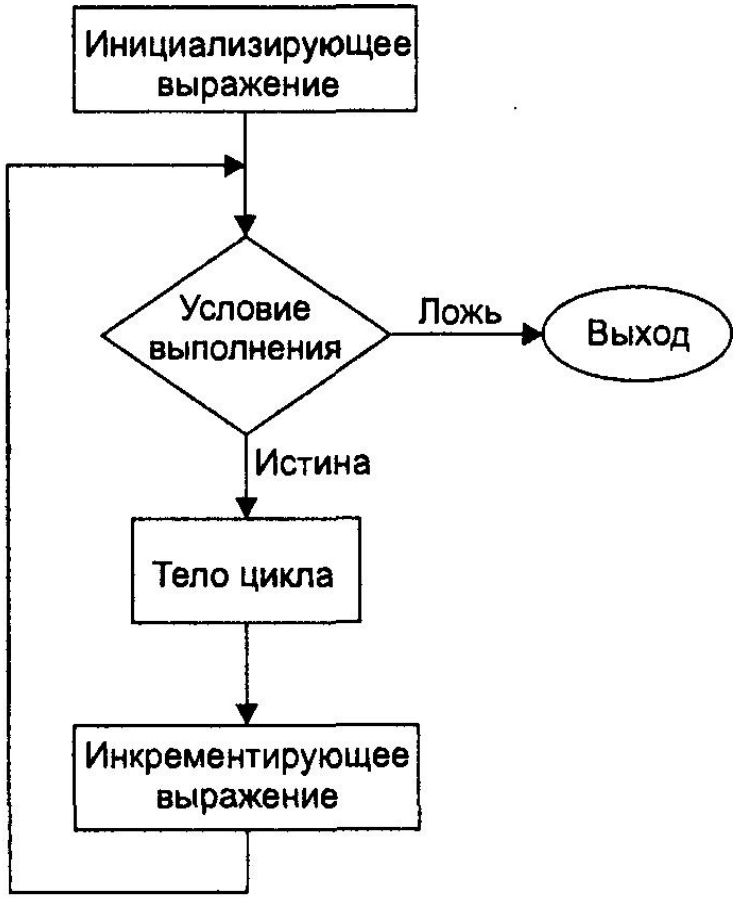
Циклы “for”

```
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;
}
```

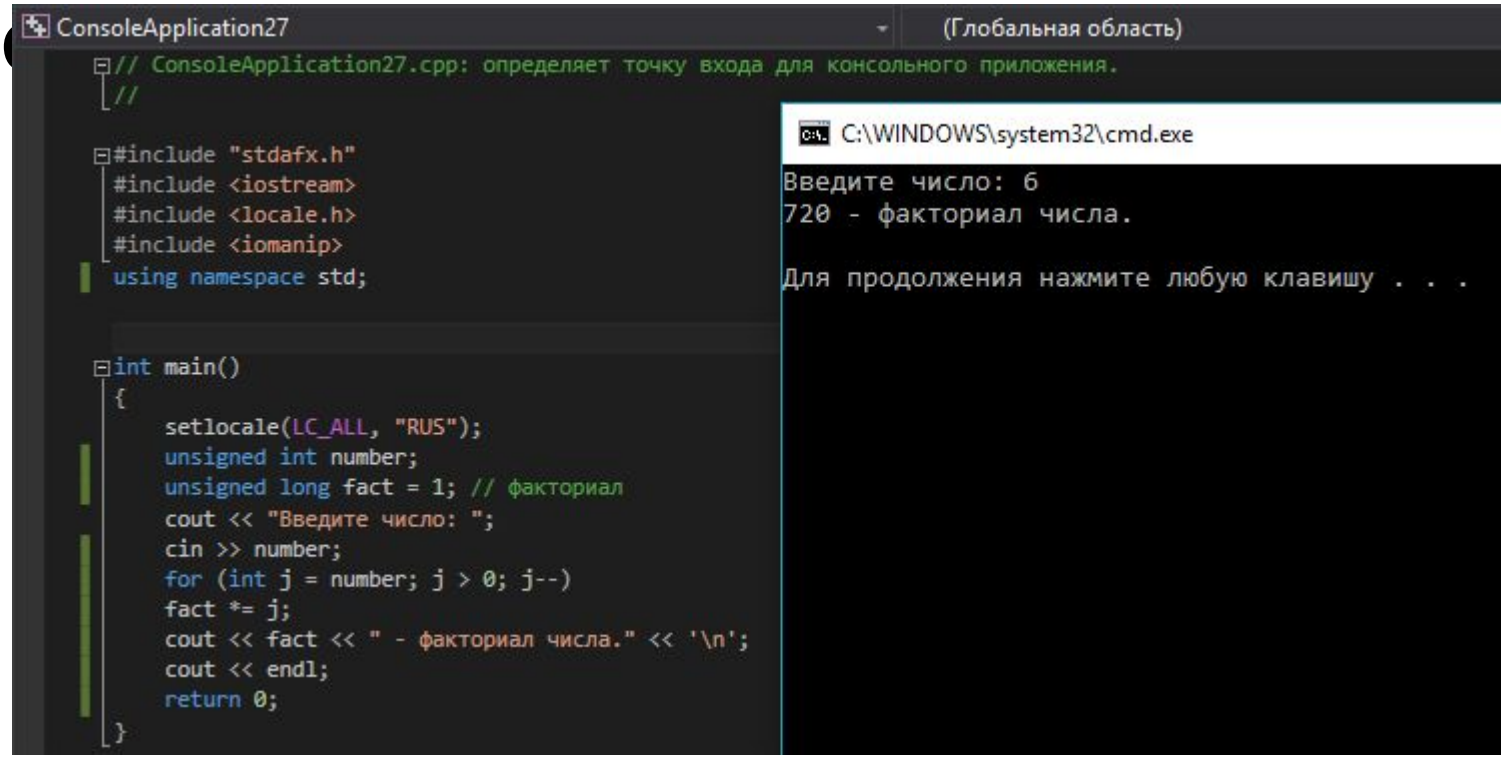
new_count переменные, определенные внутри него, ***невидимы*** вне этого блока. Невидимость означает, что программа не имеет доступа к переменной

Циклы ‘for’

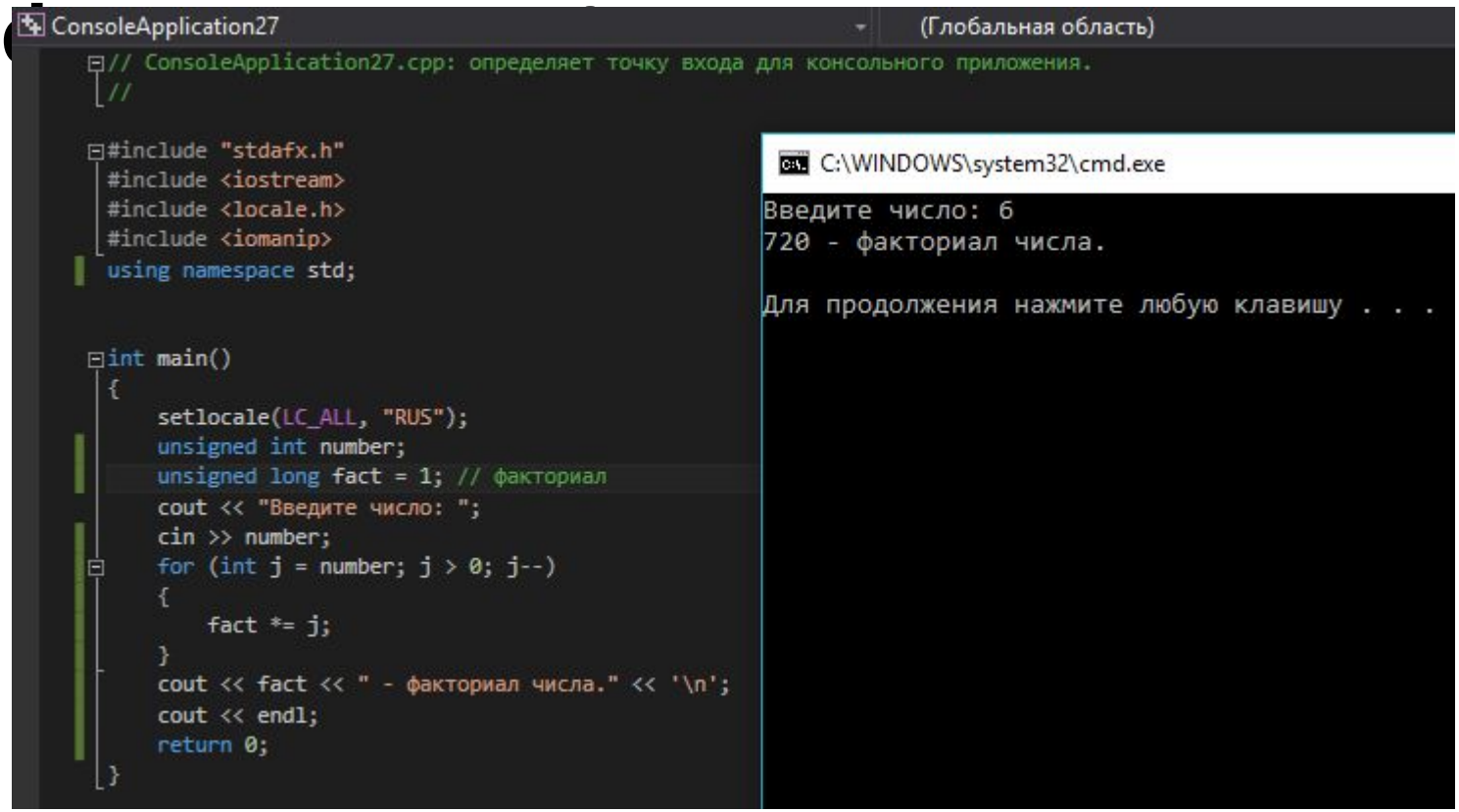
Присваивая счетчику начальное значение, равное 0, в качестве условия продолжения цикла ставят сравнение счетчика с желаемым числом выполнений и увеличивают счетчик на единицу каждый раз, когда исполнение тела цикла завершается.



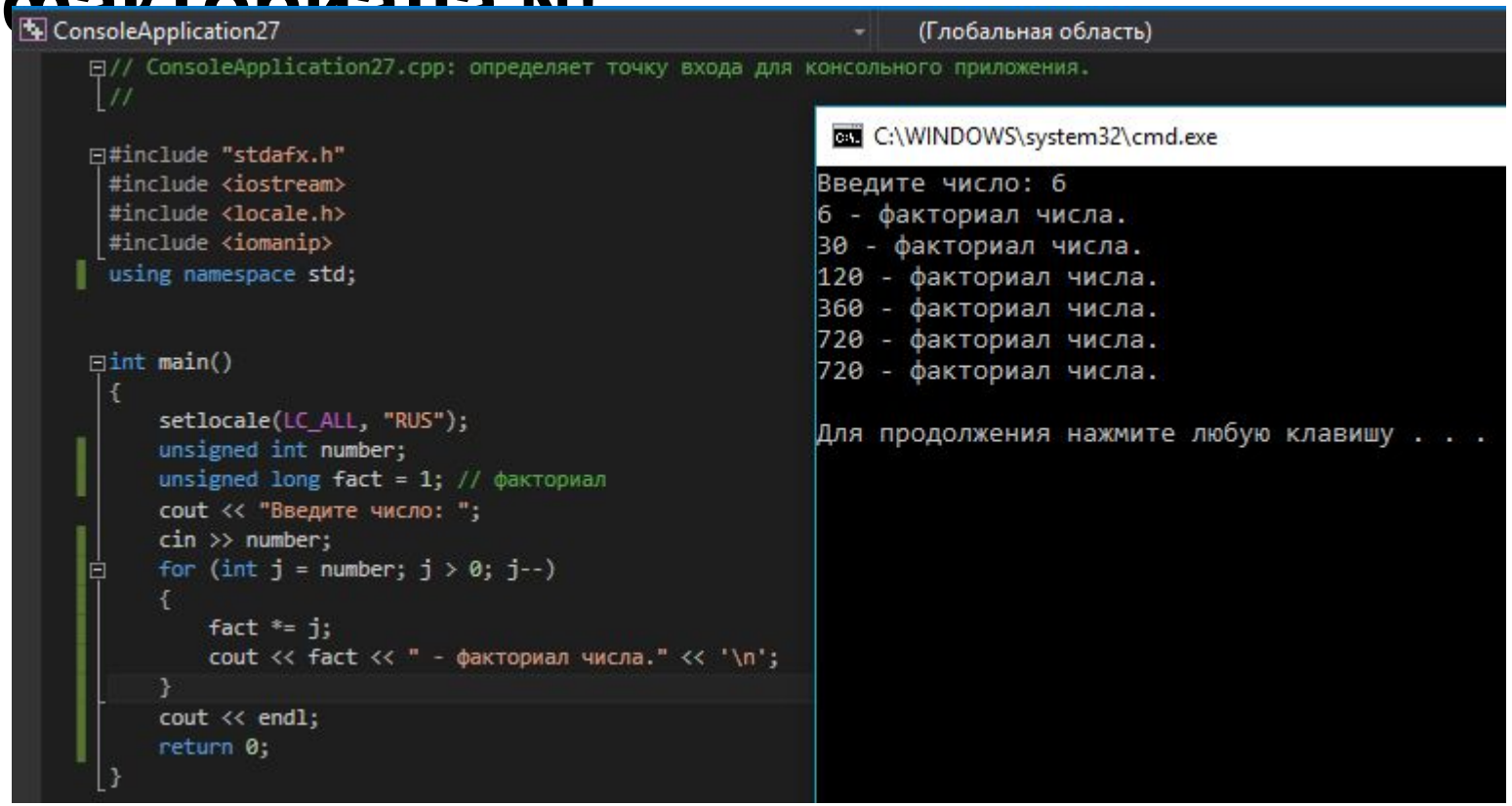
Вычисление


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!



Циклы

“for”

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)

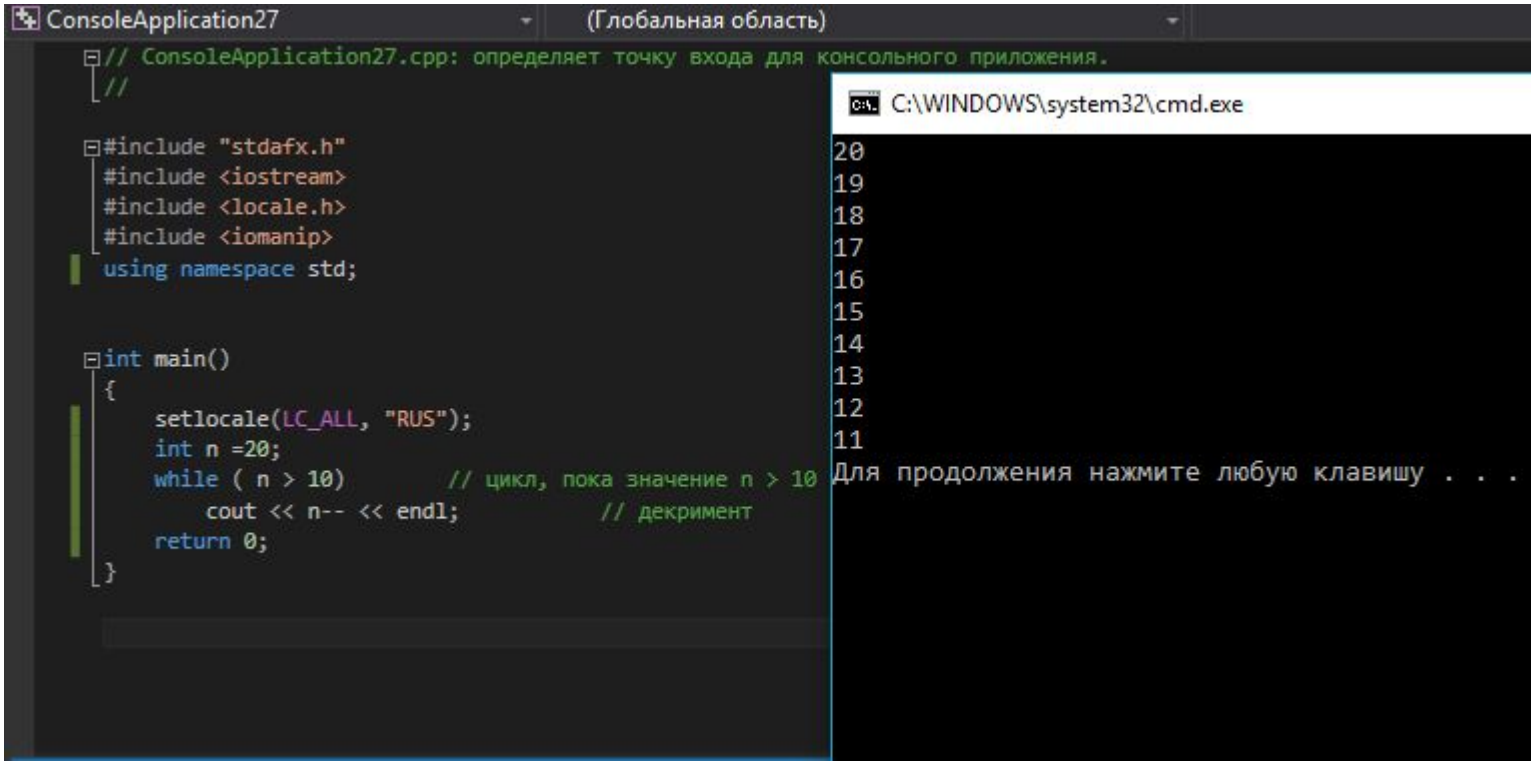
```
#include <iomanip>
using namespace std;


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

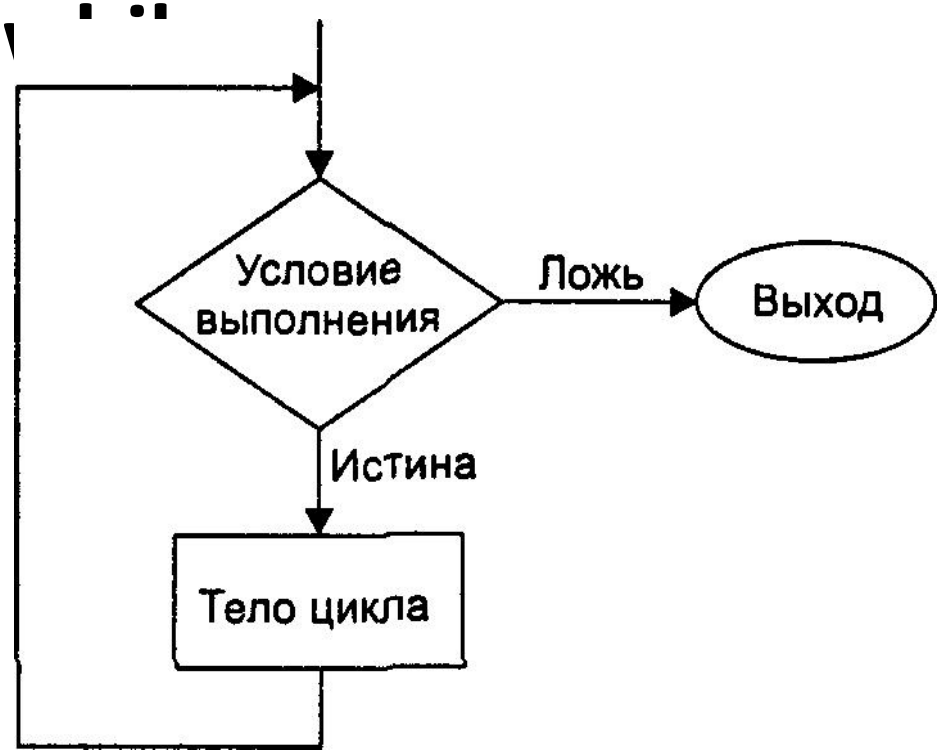
```
Введите число: 5
5 - факториал числа.номер цикла - 1
20 - факториал числа.номер цикла - 2
60 - факториал числа.номер цикла - 3
120 - факториал числа.номер цикла - 4
120 - факториал числа.номер цикла - 5

Для продолжения нажмите любую клавишу .
```

Циклы ‘while’

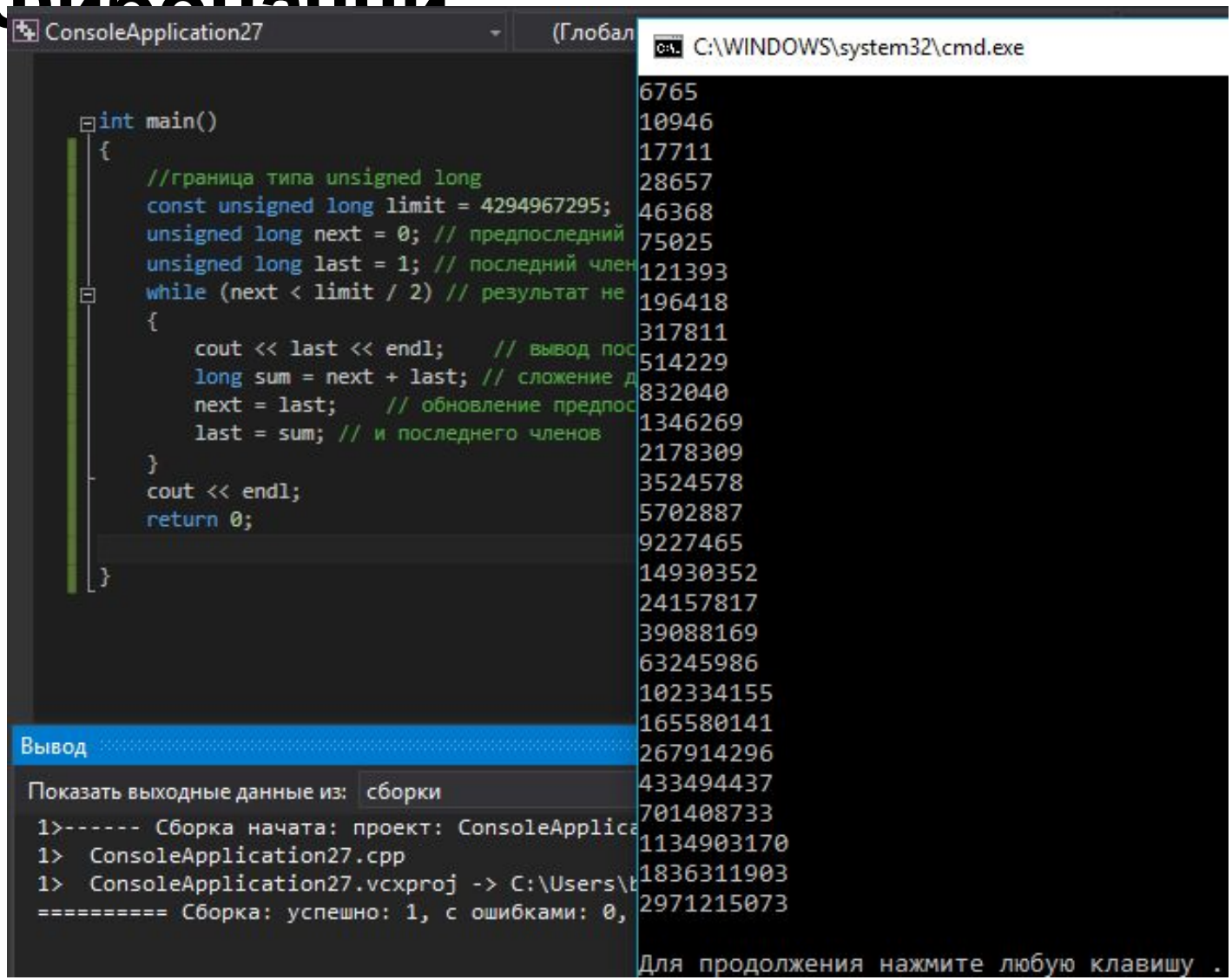
- Когда неизвестно какое количество циклов выполнить



Циклы

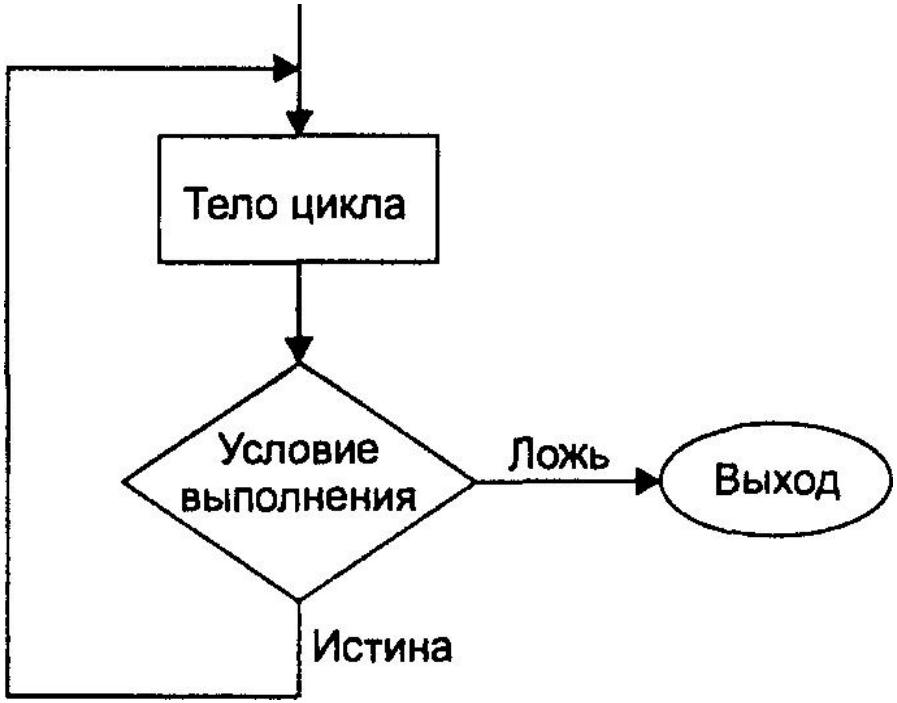


Вычисления ряда чисел Фибоначчи



Цикл ‘do’

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




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

Цикл с постусловием!

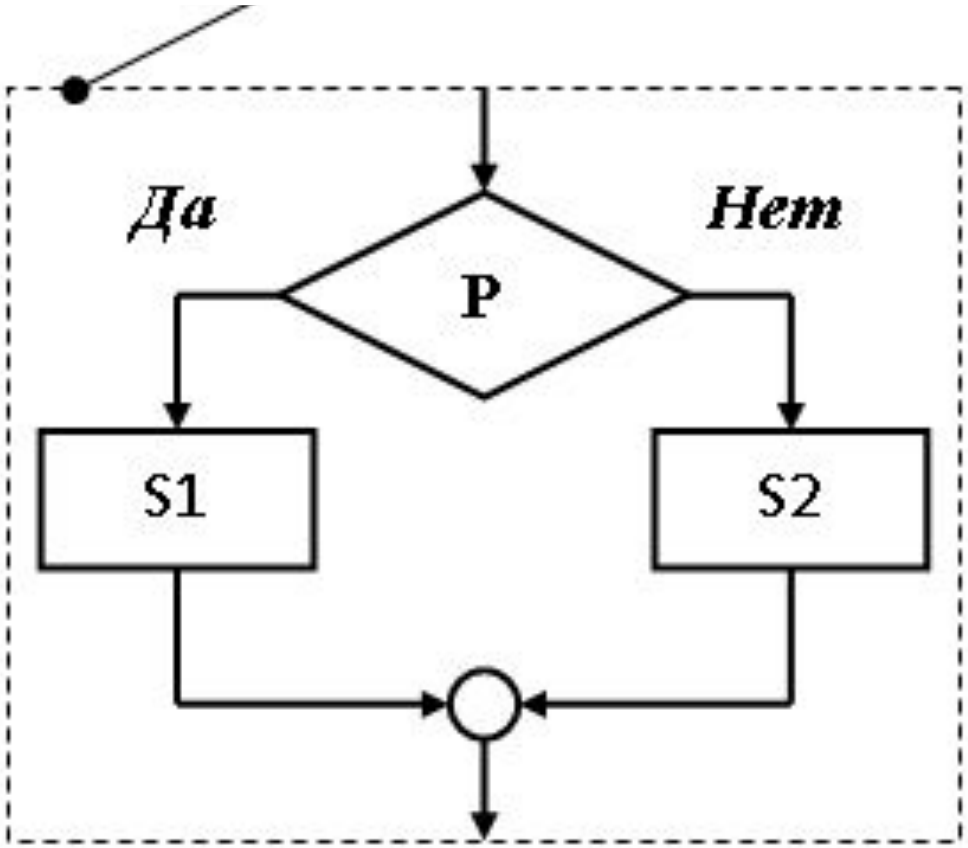
Выполнится хотя бы 1 раз

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

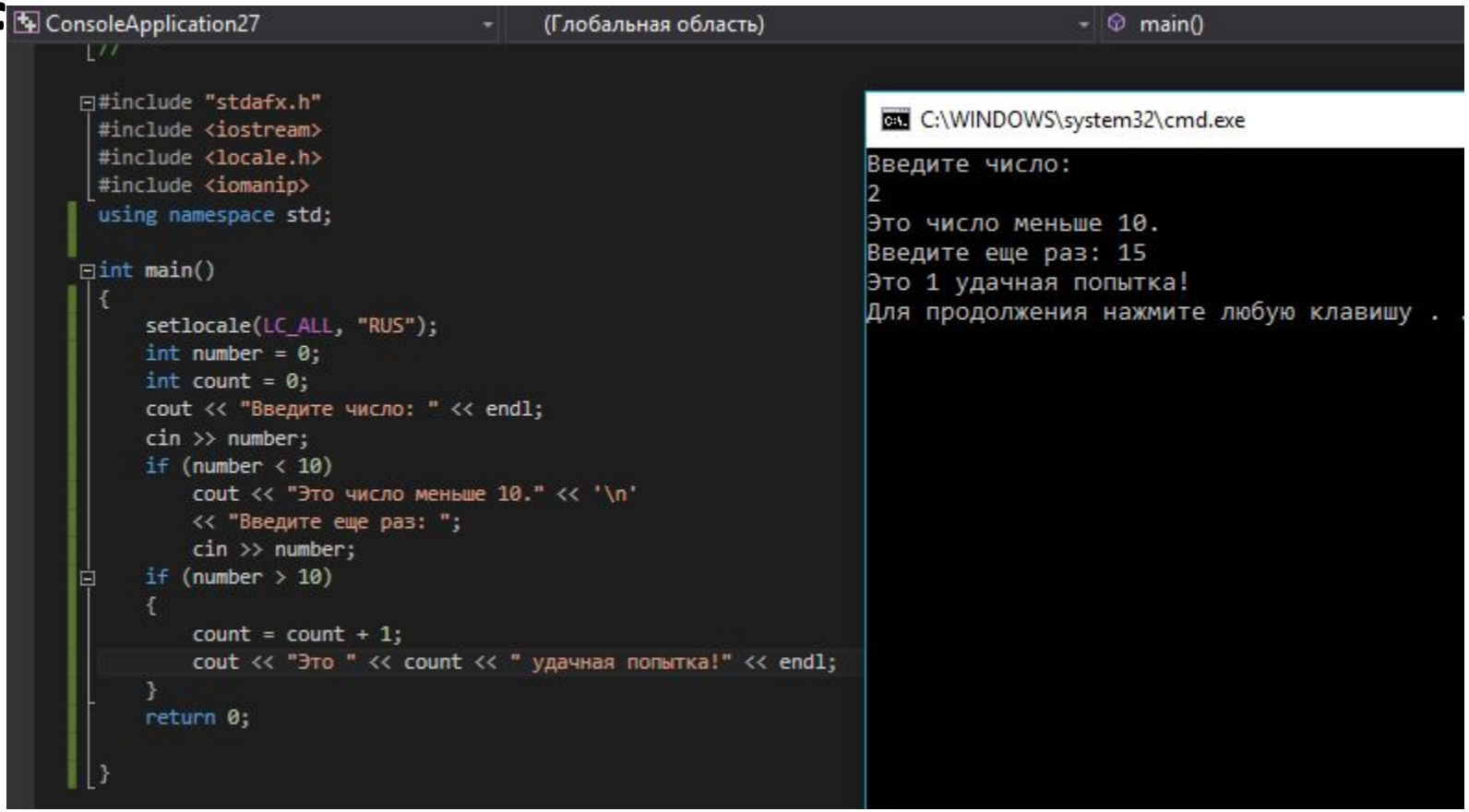


Ветвление

- if …
- if … else …
- switch…. case …

Ветвления

if



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 . . .
```

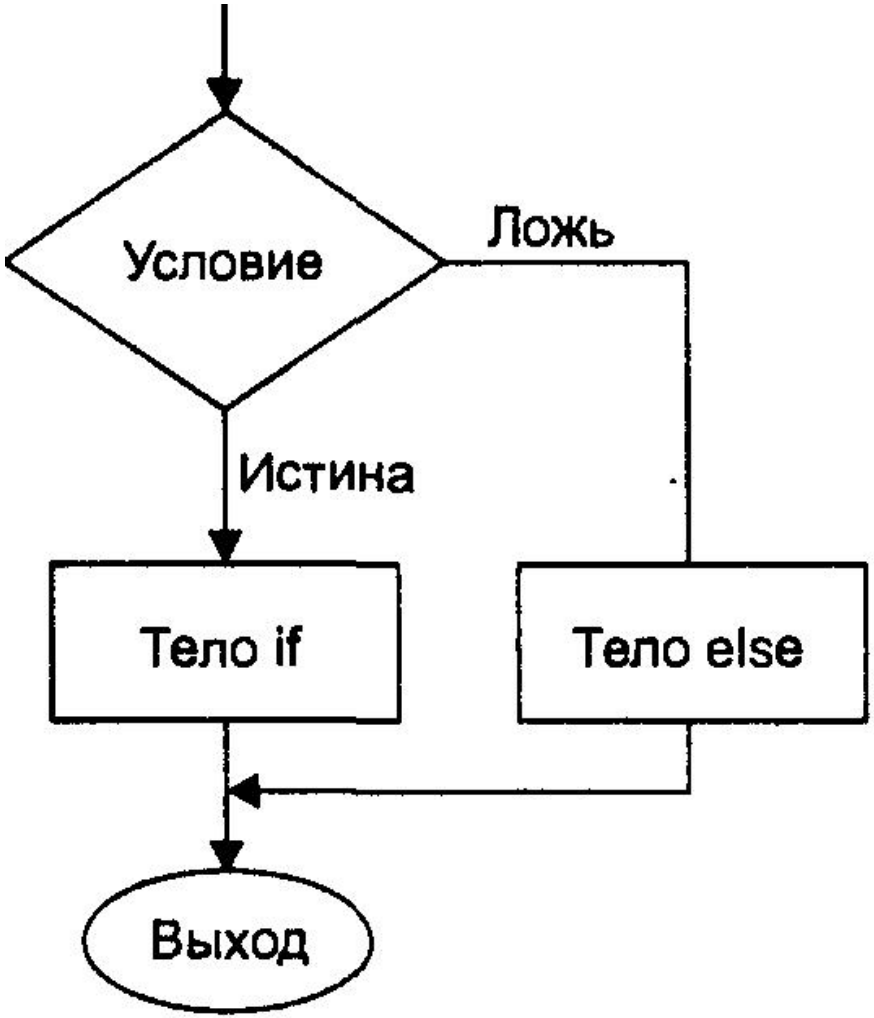
тело цикла не заключено в фигурные скобки. Это объясняется тем, что оператор if и операторы тела ветвления на самом деле являются одним оператором

Выход из программы

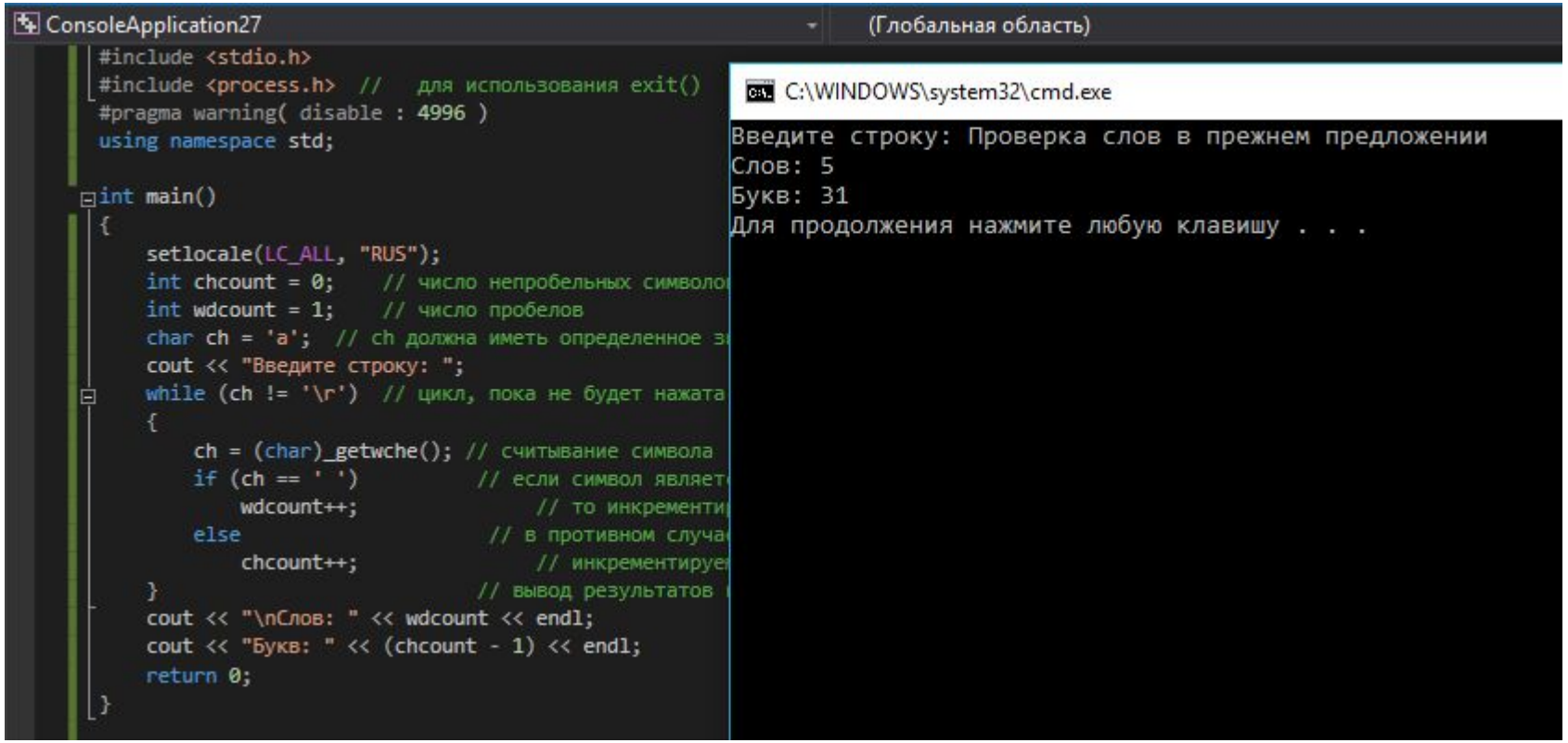
Функция **exit()** – немедленный выход из программы, независимо в каком месте исполнения она находится.

Требуется #include <process.h>

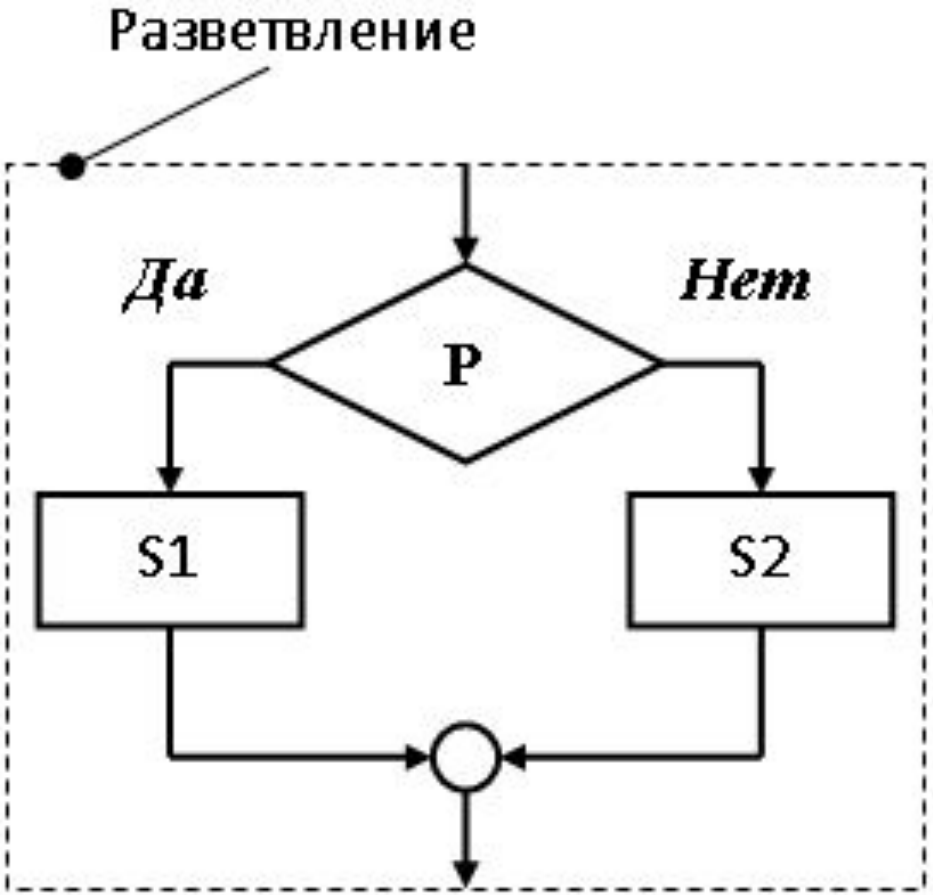
Ветвления if … else …



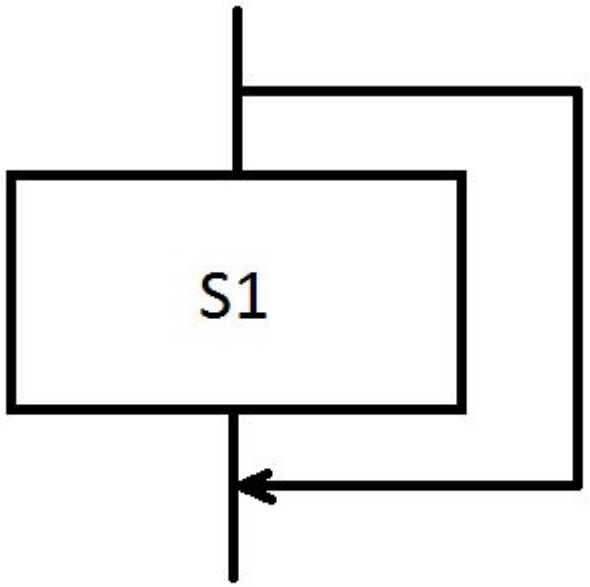
Ветвления if … else …



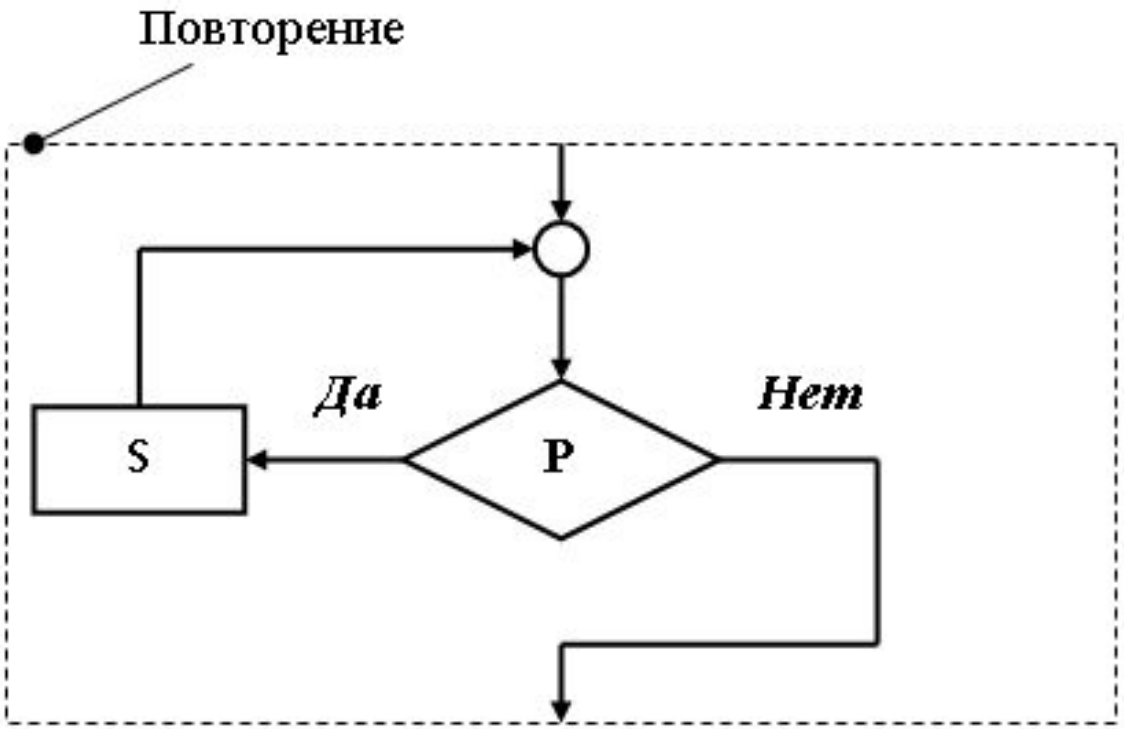
Разветвление



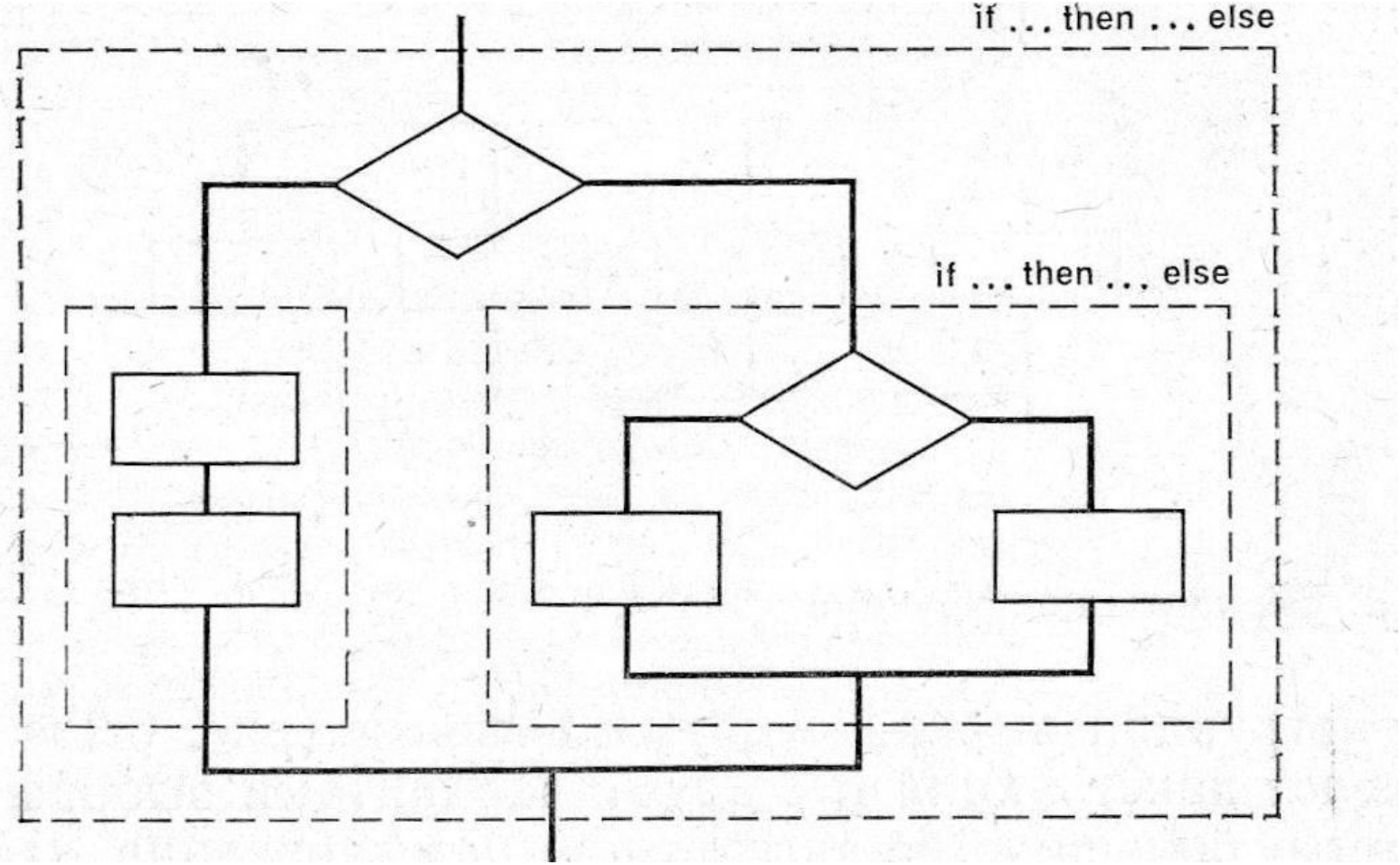
Переход (GOTO…)



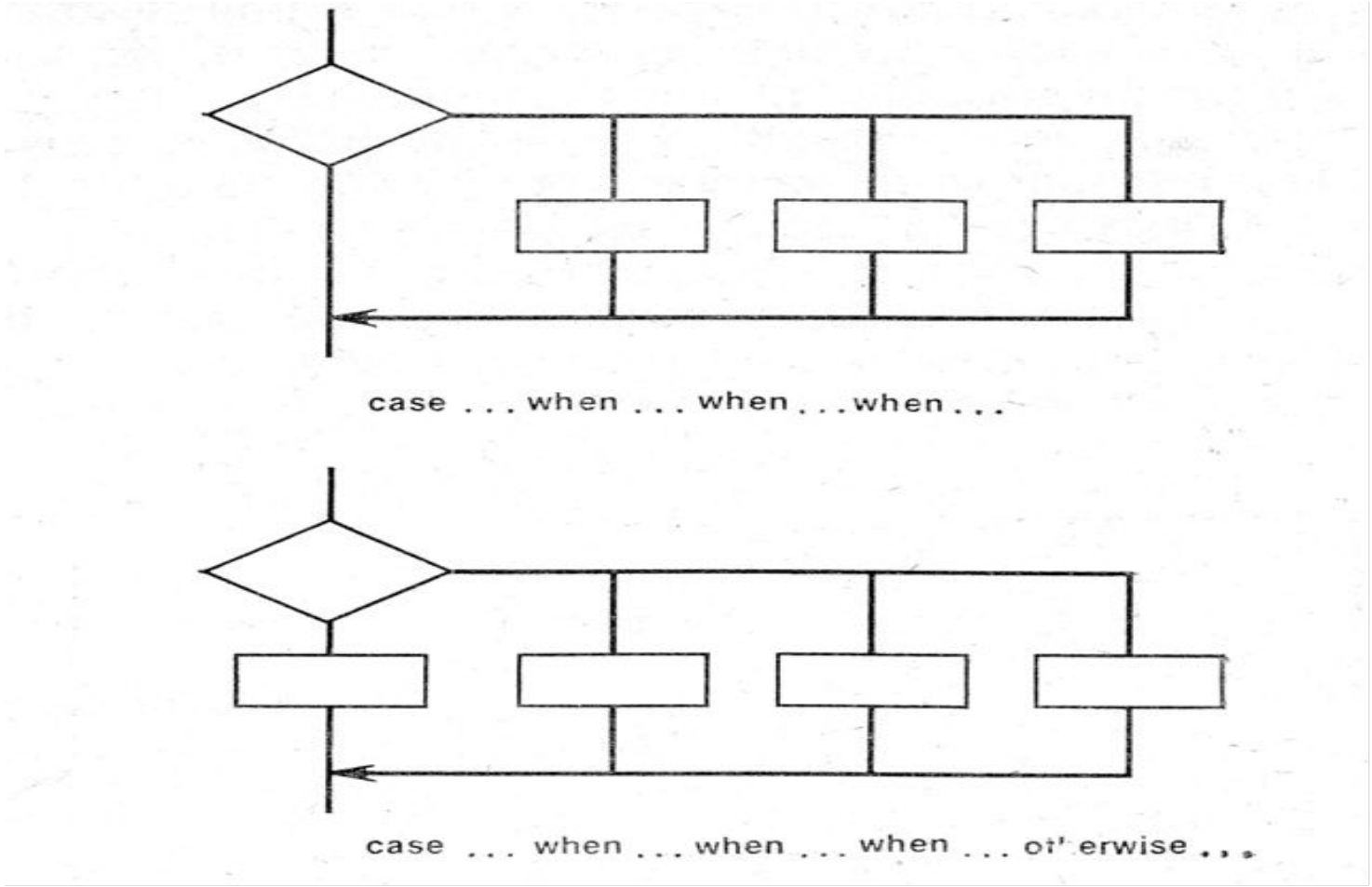
Цикл (повторение)



Вложенные конструкции



Конструкция “CASE”



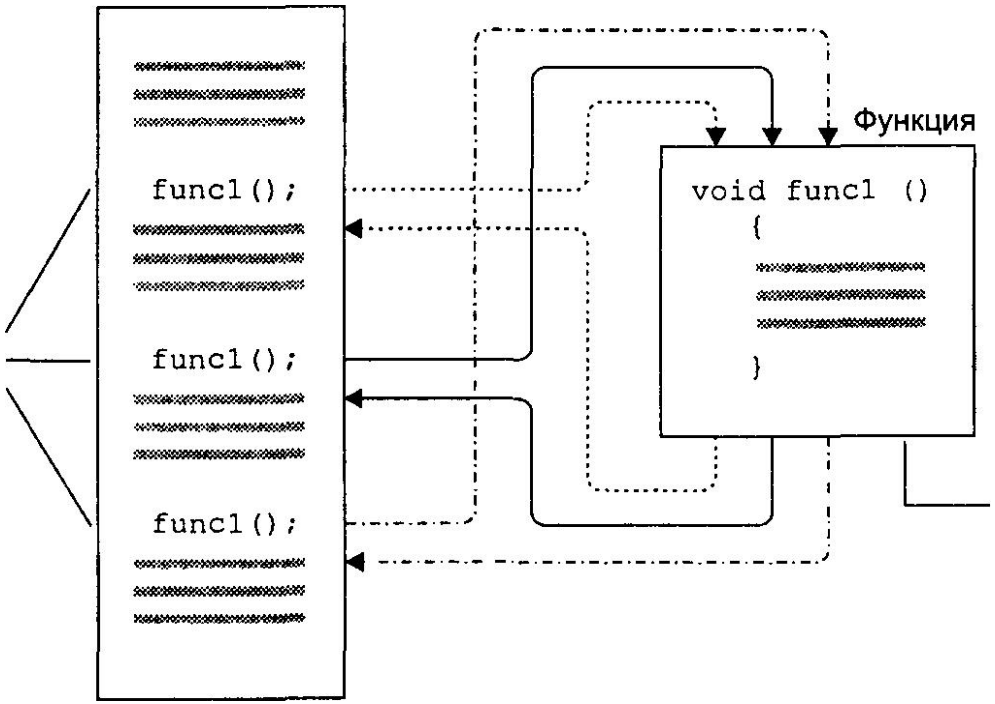
Функции

Название	Назначение	Пример
Объявление (прототип)	Содержит имя функции, типы ее аргументов и возвращаемого значения. Указывает компилятору на то, что определение функции будет сделано позднее	void func();
Вызов	Указывает на то, что необходимо выполнить функцию	func();
Заголовок	Первая строка определения	void func()

Определение	Является собственно функцией. Содержит код, предназначенный для исполнения	void func()
		{
		// операторы
		}

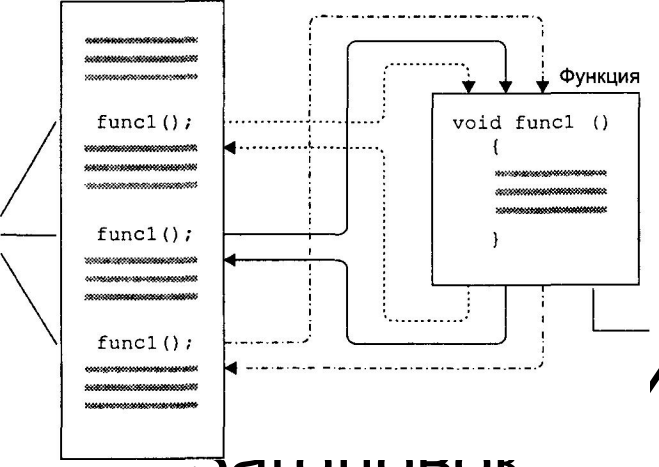
ФУНКЦИИ

Объявление void func1()



Вызов

Определение



Функция

ЛЕ

Заголовок

Тело

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

Передача аргументов в функцию

Передача констант в функцию.