
Программирование ч.1

ЛЕПУСТИН А.В.
ст. преп.
Отделение ИТ
Инженерная школа ИТиР

Элементы языка

- Директивы препроцессора (`#include`, `#define`, ...)
- Пространства имён
- Функции
 - Особое место занимает функция `_tmain()` – точка входа
 - Блок `{}` – составной оператор
 - Оператор возврата значения из функции `return`
 - Выход из функции `main` = завершение работы программы
- Потоки ввода/ вывода
 - `cin` / `cout`

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

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    cout << " Hello, world!" << endl;
    return 0;
}
```

Элементы языка

```
int _tmain(int argc, _TCHAR* argv[])
{
    int a, b, s;
    cout << "Введите два числа\n";
    cin >> a >> b;
    s = a + b;
    cout << "Сумма равна: " << s;
    return 0;
}
```

Объявление переменной:

Тип имя;

```
int x;
```

Присвоение значения:

```
x = 10;
```

Объявление с
инициализацией:

Тип имя = значение;

```
int x = 5;
```

Неинициализированные
переменные хранят
случайное значение!

Элементы языка

- Алфавит языка – A..Z, a..z, 0..9, _
- Встроенные типы
 - Целые : **char, int, short, long (signed, unsigned)**
 - С плавающей запятой: **float, double, long double**
 - Логический: **bool**
 - Перечисления: **enum**
 - Пустой: **void**
 - Классы: **structure, union, class**
 - <https://docs.microsoft.com/ru-ru/cpp/cpp/data-type-ranges?view=vs-2019>
- Комментарии
 - // строчный
 - /* блочный */

Элементы языка

- Строки
 - Строки-константы – "мама мыла раму"
 - Строки-переменные – `std::string`
- Директива препроцессора `#define`
- Модификатор `const`
- Выражения
 - Вычисления `a + b`
 - Присвоения `variable = value`
 - ...
 - Общий вид бинарного оператора – **ОПЕРАНД оператор ОПЕРАНД**
Общий вид унарного оператора – **оператор ОПЕРАНД**
 - *оператор* выполняет *операцию* над *операндами*
 - *операторы* можно переопределять
 - Результат вычисления – *l-value* (адресное выражение) или *r-value* (значение без адреса)

```
#define myint int  
#define cycle for
```

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    cycle(myint x = 0; x < 5; x++)  
        cout << x;  
    return 0;  
}
```

Элементы языка

- Присвоение
 - *l-value = value*
 - *value* может быть *l-value* или *r-value*
 - $a = b = c = d + 10;$
 - $a = (b = c + 3) = d + 4;$
 - Может вызывать преобразование типов (явное или неявное)
 - Может быть комбинированным *l-value* оп= *value*
 - $a += 5;$
 - $b *= 18;$
 - Результат – всегда *l-value*

```
int x = 50;  
double y = 13.5;  
y = x + 5.5;  
x = y;
```

Элементы языка

- Арифметические операторы
 - `+` `-` `*` `/` `%` (оператор `%` - только для целых типов)
 - `ТИП` оп `ТИП` => `ТИП`
 - `int` оп `int` => `int`
 - `10 / 4 = 2`
 - `5.4 % 4` – так нельзя!
 - Если типы операндов различны – тип результата = больший тип
- Операторы сравнения `==` `!=` `<` `>` `<=` `>=`
- Логические операторы `!` `&&` `||`
- Побитовые операторы `~` `&` `|` `^` `<<` `>>`

Элементы языка

- Автоувеличение (`++`) и автоуменьшение (`--`)
 - Постфиксный вариант: `x++` (результат – *r-value*)
 - Префиксный вариант: `++x` (результат – *l-value*)
- Тернарная операция (`?:`)
 - `int x = 5;`
 - `int y = (x > 0 ? 100 : -20);`
 - `int max = (x > y ? x : y);`
- Операция следования
 - `int a = 3, b = 8, c; // здесь запятая – разделитель, а не операция`
 - `c = a++, a + b; // здесь a станет равно 4, затем c 12;`
 - `(b--, c) *= 3; // здесь b станет равно 7, затем c 36.`
- Приоритет операторов
 - <https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-built-in-operators-precedence-and-associativity?view=vs-2019>

Условный оператор

- Полная форма

```
if (условие)
{
    ...
}
else
{
    ...
}
```

- Неполная форма

```
if (условие)
{
    ...
}
```

Оператор выбора

```
char sign;
int x, y, z;
cout << "Задайте знак операции + - * / \n";
cin >> sign;
cout << "Задайте x и y \n";
cin >> x >> y;
switch (sign)
{
    case '+': z = x + y;
              break;
    case '-': z = x - y;
              break;
    case '*': z = x * y;
              break;
    case '/': if (y == 0)
              {
                  cout << "Делить на нуль нельзя!\n";
                  exit(1);
              }
              else z = x / y;
              break;
    default: cout << "Неизвестная операция!\n";
              exit(1);
}
```

Операторы цикла

- Цикл *while*

```
int x = 1, y = 5, s = 0;
while (x <= y)
    s += x++;
cout << s;
```

- Цикл *do-while*

```
int x = 0, y = 5, s = 0;
do
{
    s += ++x;
} while (x < y);
cout << s;
```

- Цикл *for*

```
int y = 5, s = 0;
for (int x = 1; x <= y; x++)
    s += x;
cout << s;
```

```
for (оператор1; выражение1; выражение2)
{
    оператор2;
}
```

```
оператор1;
while (выражение1)
{
    оператор2;
    выражение2;
}
```

Оператор *continue*

```
for (int x = 0; x < 100; x++)  
{  
    if (x % 2 == 0)  
        continue;  
    cout << x << '\n';  
}
```

```
for (int x = 1; x < 100; x += 2)  
{  
    cout << x << '\n';  
}
```

Оператор *break*

```
char sign;
int x, y, z;
cout << "Задайте знак операции + - * / \n";
cin >> sign;
cout << "Задайте x и y \n";
cin >> x >> y;
switch (sign)
{
    case '+': z = x + y;
              break;
    case '-': z = x - y;
              break;
    case '*': z = x * y;
              break;
    case '/': if (y == 0)
              {
                  cout << "Делить на нуль нельзя!\n";
                  exit(1);
              }
              else z = x / y;
              break;
    default: cout << "Неизвестная операция!\n";
              exit(1);
}
```

Оператор *break*

```
for (int x = 0; x < 100; x++)  
{  
    if (x % 2 == 0)  
        continue;  
  
    if (x == 50)  
        break;  
    cout << x << '\n';  
}
```

Оператор *return*

- Завершение работы функции
- Возврат управления в вызывающую точку
- Передача результата работы функции в вызывающую функцию
- В случае использования в **_tmain** – завершает работу программы

Указатели

```
int x = 20;  
int *px1;  
int *px2;  
px1 = &x;  
px2 = px1;  
*px2 = 50;
```

```
int y = *px1 + 1;  
(*px1)++; // но px1++; и *px1++; - неправильно!
```

```
cout << x << endl;  
cout << y << endl;  
cout << (*px1) << endl;  
cout << (*px2) << endl;
```


Массивы

- Объявление
 - `int a[5];`
- Объявление с инициализацией
 - `int b[5] = { 7, 3, 6, 1, 2 };`
 - `int c[5] = { 0 };`
- Объявление без указания размера
 - `int d[] = { 7, 3, 6, 1, 2, 5, 8, 4 };`
- Обращение к элементам
 - `d[5] = 500;`
 - `d[3] = 300 + d[5]*5;`

Массивы и указатели

- Имя массива – константный указатель на нулевой элемент

```
int a[20];  
int *pa;  
pa = &a[0];    //эквивалентно pa = a;
```

- Обращение по индексу

```
a[5]    *(a+5)    5[a]  
*(указатель + индекс)
```

- Операции над указателями

- сравнение
- вычитание (если указывают на элементы одного массива – количество элементов между указателями)

Работа с одномерными массивами

```
int a[20];  
int s = 0;  
for (int i = 0; i < 20; i++)  
    s += a[i];  
cout << s;
```

Многомерные массивы

- Двумерный массив – массив, элементы которого – одномерные массивы
- Трехмерный массив – массив, элементы которого – двумерные массивы
- ...
- Для массива `int a[3][4]`:
 - `a[2][2]` - объект типа `int` (элемент массива)
 - `a[2]` - объект типа `int*` – одномерный массив из 4 целых
 - `a` - сам двумерный массив.
 - `a[0][0]` `a[0][1]` `a[0][2]` `a[0][3]` `a[1][0]` `a[1][1]` `a[1][2]` `a[1][3]` `a[2][0]` `a[2][1]` `a[2][2]` `a[2][3]`
- Для массива `int b[10][20][30]`:
 - `b[3][4][5]` - объект типа `int` (элемент массива)
 - `b[5][7]` - объект типа `int*` – одномерный массив из 30 целых
 - `b[8]` - двумерный массив из $20 \cdot 30 = 600$ целых
 - `b` - сам трехмерный массив.

Инициализация массива

- `int d[2][3] = { 10, 12, 14, 16, 18, 20 };`
- `int e[][3] = { 2, 3, 5, 8, 13, 21 }; //массив e[2][3];`
- `int f[2][3] = { { 1, 7 }, { -5, 3 } };`

Многомерные массивы и указатели

- Для массива `int c[10][20][30]`:
 - `c[0]` \square `c[1]` в байтах: $20 * 30 * \text{sizeof}(\text{int}) = 2400$
- `c[4]` \square `*(a+4)`
- `c[4][5]` \square `*(*(a+4)+5)`
- `c[4][5][6]` \square `*(***(a+4)+5)+6)`

Ссылки

- Ссылка – синоним имени
- Ссылки – константные объекты (псевдонимы)

```
int i = 0;
```

```
int& iref = i;
```

```
iref++;           // то же, что и i++
```

```
int *ip = &iref; // то же, что и ip=&i
```

- `double d = 0.0;`
- `int& ir = d; // Создан анонимный объект типа int;`
- `ir = 3.0; // d – не меняется!`

Функции

- Объявление:
 - Тип имя (список описаний аргументов){ операторы }

```
int max(int a, int b)
{
    return(a >= b) ? a : b;
}
```


Передача аргумента в функцию по значению

```
int max(int a, int b)
{
    int c = (a >= b ? a : b);
    a++;
    b++;
    return c;
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;
    int y = 20;
    int z = max(x, y);
    return 0;
}
```

Передача аргумента в функцию по ссылке

```
int max(int& a, int& b)
{
    int c = (a >= b ? a : b);
    a++;
    b++;
    return c;
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;
    int y = 20;
    int z = max(x, y);
    return 0;
}
```

Передача аргумента в функцию по указателю

```
int max(int* a, int* b)
{
    int c = (*a >= *b ? *a : *b);
    (*a)++;
    (*b)++;
    return c;
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;
    int y = 20;
    int z = max(&x, &y);
    return 0;
}
```

Аргументы по умолчанию

```
int inc(int& val, int step = 1)
{
    return (val += step);
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;

    inc(x, 3);    //теперь x = 13
    inc(x);      //теперь x = 14
    inc(x);      //теперь x = 15

    return 0;
}
```

Аргументы по умолчанию

```
void PrintValues(int a = 10, int b = 20, int c = 30)
{
    cout << a << " " << b << " " << c << endl;
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    PrintValues(5, 6, 7);           //на экране - «5 6 7»
    PrintValues(15, 16);          //на экране - «15 16 30»
    PrintValues(25);              //на экране - «25 20 30»
    PrintValues();                //на экране - «10 20 30»

    return 0;
}
```

Аргументы по умолчанию

```
void PrintValues(int a, int b = 20, int c = 30)
{
    cout << a << " " << b << " " << c << endl;
}

int _tmain(int argc, _TCHAR* argv[])
{
    PrintValues(5, 6, 7);           //на экране - «5 6 7»
    PrintValues(15, 16);          //на экране - «15 16 30»
    PrintValues(25);              //на экране - «25 20 30»
    //PrintValues();              //ошибка!

    return 0;
}
```

Возврат значения по ссылке

```
int& max(int& a, int& b)
{
    return (a > b ? a : b);
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;
    int y = 20;

    max(x, y) = 0;
    cout << x << " " << y << endl; //на экране - «10 0»
    max(x, y) = 5;
    cout << x << " " << y << endl; //на экране - «5 0»

    return 0;
}
```

Возврат значения по ссылке

```
int& max(int a, int b)
{
    return (a > b ? a : b);
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;
    int y = 20;

    max(x, y) = 0;
    cout << x << " " << y << endl;           //на экране - «10 20»
    max(x, y) = 5;
    cout << x << " " << y << endl;           //на экране - «10 20»

    return 0;
}
```


Возврат значения по указателю

Тоже возможен 😊

Перегрузка функций

```
int max(int a, int b)
{
    return (a > b ? a : b);
}

double max(double a, double b)
{
    return (a > b ? a : b);
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    int x = 10;    double k = 2.5;
    int y = 20;    double p = 5.5;
    int z;         double m;
    float r = 1.2f;

    z = max(x, y);
    m = max(k, p);
    m = max(r, k);
    //max(x, k);    //ошибка!
    return 0;
}
```

1. Точное соответствие
2. (min!) Стандартные преобразования (без преобразований цел.↔плав.)
3. (min!) Любые стандартные преобразования
4. (min!) Все преобразования + преобразования, реализованные программистом

Операции new и delete

```
int* p1 = new int(2);  
//создаётся два объекта - указатель и безымянное целое
```

```
int* p2 = p1;  
//теперь два указателя на один объект
```

```
int a = 5;  
p1 = &a;  
//p2 по-прежнему указывает на безымянный int!
```

```
p2 = p1;  
//безымянный int по-прежнему существует, это мусор!
```

Операции new и delete

```
double *mas = new double[50];  
//создаётся указатель и безымянный массив  
  
delete mas;  
//память массива освобождается
```

Динамические многомерные массивы

```
int m=5, n=10;
```

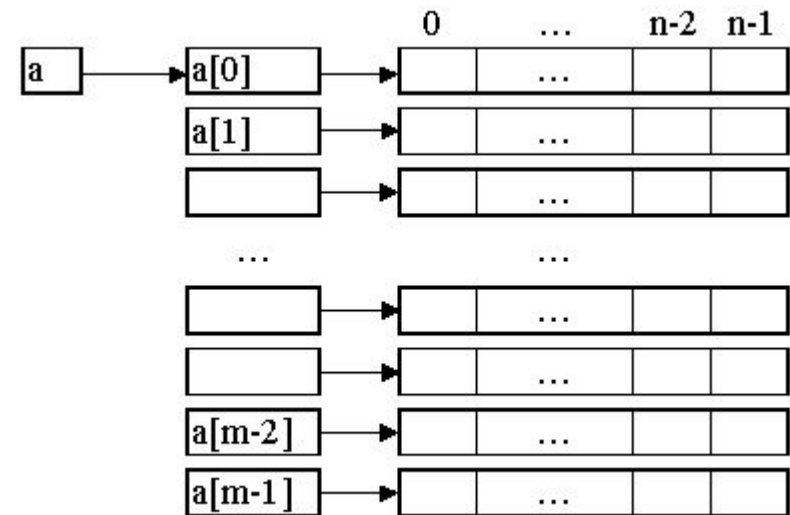
```
int **a = new int *[m];
```

```
for (int i = 0; i < m; i++)  
    a[i] = new int[n];
```

```
a[3][4] = 100;
```

```
for (int i = 0; i < m; i++)  
    delete a[i];
```

```
delete a;
```



Динамические многомерные массивы

```
int m=5, n=10;
```

```
int **a = new int *[m];
```

```
for (int i = 0; i < m; i++)  
    a[i] = new int[n];
```

```
a[3][2] = 100;
```

```
for (int i = 0; i < m; i++)  
    delete a[i];
```

```
delete a;
```

Битовые операции

- Операнды – только целые!
- Операторы: \sim $\&$ $|$ \wedge \ll \gg
- Общий вид:
 - Общий вид унарного оператора – оператор ОПЕРАНД
 - Общий вид бинарного оператора – ОПЕРАНД оператор ОПЕРАНД
 - Пример использования унарного: $\sim a$
 - Бинарные : $a \ll 4$ $b \wedge c$...

Системы счисления

- Перевод чисел в 10-ю СС:
 - Пронумеровать разряды справа налево, начиная с 0
 - Вычислить вес каждого разряда, возведя основание в степень номера разряда
 - Для каждого разряда найти произведение цифры в нём на его вес
 - Найти сумму произведений

$$\begin{array}{cccc} 3 & 2 & 1 & 0 \\ 2 & 8 & 6 & 3 \\ 1000 & 100 & 10 & 1 \end{array} 10 = 2 \cdot 1000 + 8 \cdot 100 + 6 \cdot 10 + 3 \cdot 1 = 2863_{10}$$

$$\begin{array}{cccc} 3 & 2 & 1 & 0 \\ 2 & 4 & 6 & 3 \\ 343 & 49 & 7 & 1 \end{array} 7 = 2 \cdot 343 + 4 \cdot 49 + 6 \cdot 7 + 3 \cdot 1 = 927_{10}$$

Системы счисления

- Перевод чисел в 10-ю СС:
 - Пронумеровать разряды справа налево, начиная с 0
 - Вычислить вес каждого разряда, возведя основание в степень номера разряда
 - Для каждого разряда найти произведение цифры в нём на его вес
 - Найти сумму произведений

$$\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array} \cdot 2 = 128 + 64 + 16 + 4 + 1 = 213_{10}$$

Системы счисления

- Перевод из 10й СС:
 - Деление *исходного числа* нацело с остатком на основание целевой СС
 - Деление *полученного частного* нацело с остатком на основание целевой СС
 - Деление продолжается до получения в частном значения 0
 - Составление из остатков (в обратном порядке) числа в целевой СС

$$\begin{array}{r|l} 530_{10} & 12_{10} \\ \underline{48} & 44 \\ 50 & \\ \underline{48} & \\ 2 & \end{array}$$

$$\begin{array}{r|l} 44_{10} & 12_{10} \\ \underline{36} & 3 \\ 8 & \end{array}$$

$$\begin{array}{r|l} 3_{10} & 12_{10} \\ \underline{0} & 0 \\ 3 & \end{array}$$

Ответ: 382_{12}

Системы счисления

- $20 : 2 = 10$ (ост. 0)
- $10 : 2 = 5$ (ост. 0)
- $5 : 2 = 2$ (ост. 1)
- $2 : 2 = 1$ (ост. 0)
- $1 : 2 = 0$ (ост. 1)

- $20_{10} = 10100_2$

Операции

- $\&$ - логическое И

- $00 \Rightarrow 0$
- $01 \Rightarrow 0$
- $10 \Rightarrow 0$
- $11 \Rightarrow 1$

- $|$ - логическое ИЛИ

- $00 \Rightarrow 0$
- $01 \Rightarrow 1$
- $10 \Rightarrow 1$
- $11 \Rightarrow 1$

- \wedge - исключающее ИЛИ

- $00 \Rightarrow 0$
- $01 \Rightarrow 1$
- $10 \Rightarrow 1$
- $11 \Rightarrow 0$

- \sim - логическое НЕ

- $0 \Rightarrow 1$
- $1 \Rightarrow 0$

Битовые операции

```
unsigned char a = 27;
```

```
unsigned char b = 92;
```

```
unsigned char r;
```

```
r = a & b;
```

```
r = a | b;
```

```
r = a ^ b;
```

```
r = ~b;
```

```
00011011
```

```
01011100
```

```
00011000 □ 24
```

```
01011111 □ 95
```

```
01000111 □ 71
```

```
10100011 □ 163
```

Указатели на функцию

```
int max(int a, int b)
{
    return (a > b ? a : b);
}

int min(int a, int b)
{
    return (a < b ? a : b);
}

int _tmain(int argc, _TCHAR* argv[])
{
    int x = 3, y = 5, z;
    int (*f)(int, int);           //указатель на функцию!

    f = max;
    z = f(x, y);

    f = min;
    z = f(x, y);

    return 0;
}
```

Указатели на функцию

```
int max(int a, int b)
{
    return (a > b ? a : b);
}
```

```
int min(int a, int b)
{
    return (a < b ? a : b);
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    int x = 3, y = 5, z;
    int (*f)(int, int);           //указатель на функцию!

    if (x % 2 == 0)
        f = max;
    else f = min;

    z = f(x, y);
}
```

Прототип функции

```
int min(int a, int b);    // прототип !
```

```
int minXOR(int a, int b)
{
    return min(a % 2, b % 2);
}
```

```
int min(int a, int b)
{
    return (a < b ? a : b);
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    int x = 3, y = 5, z;
    z = minXOR(x, y);
}
```


Область существования имени

```
int _tmain(int argc, _TCHAR* argv[])
{
    ...

    if (val == 0)
    {
        int a = 573 * val;    // ОС а – БЛОК
        cout << a;
    }

    ...
}
```

Область существования имени

```
int _tmain(int argc, _TCHAR* argv[])
{
    ...

    int b;           // ОС b – ФУНКЦИЯ
    cout << b;

    ...
}
```

Область существования имени

```
int min(int a, int b);    // ОС а и b – прототип!
```

Область существования имени

```
int val;           // область видимости – ФАЙЛ!
```

```
int _tmain(int argc, _TCHAR* argv[])
```

```
{
```

```
    val = 10;      // глобальная переменная
```

```
    cout << val;
```

```
}
```

Область видимости имени

```
int val;           // область видимости – ФАЙЛ!  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
    val = 10;      // глобальная переменная  
  
    int val = 20;  // локальная переменная  
  
    cout << val;  
    cout << ::val;  
}
```

Область видимости имени

```
int val = 10;
```

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    ...  
    int val = 20;  
  
    if (...)  
    {  
        ...  
        int val = 30;  
        ...  
    }  
}
```

Классы памяти

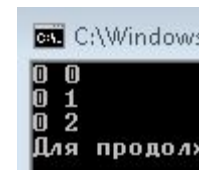
- Автоматические данные размещаются в стеке
- Динамические данные размещаются в динамической памяти с помощью операций **new** и **delete**
- Статические данные размещаются в сегменте данных, существуют в течение всего времени выполнения программы (глобальные и локальные переменные, объявленные со служебным словом **static**)

```
int g_value;           //глобальная переменная, статический объект
int _tmain(int argc, _TCHAR* argv[])
{
    int a_value;       //локальная переменная, автоматический объект
    static int s_value; //локальная переменная, статический объект
}
```

Классы памяти

```
void f()  
{  
    int a = 0;  
    static int b = 0;  
    cout << (a++) << " " << (b++) << endl;  
}
```

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    f();  
    f();  
    f();  
}
```



Шаблоны функций

```
template <class type>  
type abs(type x)  
{  
    return x > 0 ? x : -x;  
}
```

```
abs(-10);  
    int abs(int x)
```

```
abs(-5.5);  
    double abs(double x)
```

Шаблоны функций

```
template <class T>  
void swap(T& x, T& y)  
{  
    T z = x;  
    x = y;  
    y = z;  
}
```

Явные преобразования типов

```
double d = (double)5;
```

```
int i = int(d);
```

```
int *ip = &i;
```

```
float fp = (float*)ip;
```

Неявные преобразования типов

- а) типы `char`, `short`, `enum` преобразуются к типу `int`, а `unsigned short` - к `unsigned int`;
- б) затем, если один из операндов имеет тип `long double`, то и второй преобразуется к `long double`;
- в) иначе, если один из операндов имеет тип `double`, то и второй преобразуется к `double`;
- г) иначе, если один из операндов имеет тип `float`, то и второй преобразуется к `float`;
- д) иначе, если один из операндов имеет тип `unsigned long`, то и второй преобразуется к `unsigned long`;
- е) иначе, если один из операндов имеет тип `unsigned`, то и второй преобразуется к `unsigned`;
- ж) иначе, если один из операндов имеет тип `long`, то и второй преобразуется к `long`;
- з) иначе оба операнда имеют тип `int`.

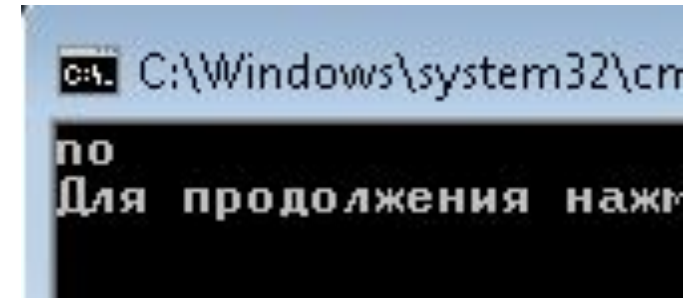
```
int g = 10, t = 5, t2 = t*t / 2;  
double s = g*t2;           // s  станет равно 120.  
double s0 = g*t*t / 2.0;  // s0 станет равно 125.
```

Неочевидные особенности вещественных чисел

```
int _tmain(int argc, _TCHAR* argv[])
{
    float x = 0.1;
    if (x == 0.1)
        cout << "yes" << endl;
    else cout << "no" << endl;
}
```

Неочевидные особенности вещественных чисел

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    float x = 0.1;  
    if (x == 0.1)  
        cout << "yes" << endl;  
    else cout << "no" << endl;  
}
```

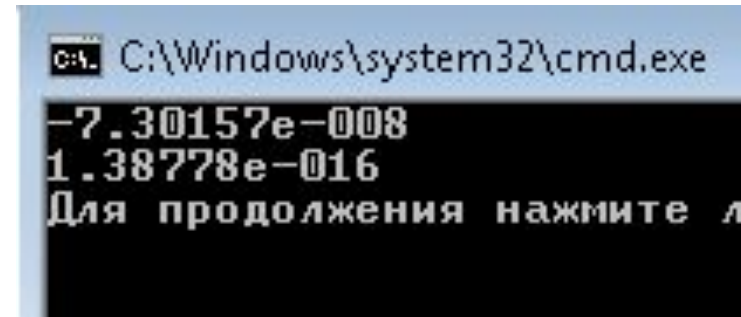


Неочевидные особенности вещественных чисел

```
int _tmain(int argc, _TCHAR* argv[])
{
    float x = 1;
    double y = 1;
    for (int i = 0; i < 10; i++)
    {
        x -= 0.1;
        y -= 0.1;
    }
    cout << x << endl;
    cout << y << endl;
}
```

Неочевидные особенности вещественных чисел

```
int _tmain(int argc, _TCHAR* argv[])
{
    float x = 1;
    double y = 1;
    for (int i = 0; i < 10; i++)
    {
        x -= 0.1;
        y -= 0.1;
    }
    cout << x << endl;
    cout << y << endl;
}
```



```
C:\Windows\system32\cmd.exe
-7.30157e-008
1.38778e-016
Для продолжения нажмите л
```


Неочевидные особенности вещественных чисел

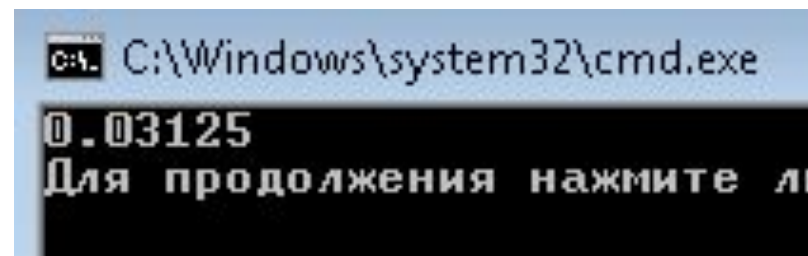
$$S = 10^{-9} + 10^{-9} + \dots + 10^{-9}$$
$$10^9$$

```
int _tmain(int argc, _TCHAR* argv[])
{
    float s = 0;
    float p = 1e-9;
    for (long int i = 0; i < 1000000000; i++)
        s += p;
    cout << s << endl;
}
```

Неочевидные особенности вещественных чисел

$$S = 10^{-9} + 10^{-9} + \dots + 10^{-9} \\ 10^9$$

```
int _tmain(int argc, _TCHAR* argv[])
{
    float s = 0;
    float p = 1e-9;
    for (long int i = 0; i < 1000000000; i++)
        s += p;
    cout << s << endl;
}
```



Неочевидные особенности вещественных чисел

Результат: $0,03125 = 0,00001_2 = 1,0_2 \cdot 2^{-5}$

При типе **double** результат равен $0,999999992539932880$

- Сложение $0,03125$ и $1 \cdot 10^{-9}$. Выравнивание порядков:
 - $1.000000 \cdot 10^{-9}$ $3,125000 \cdot 10^{-2}$
 - $0.100000 \cdot 10^{-8}$ $3,125000 \cdot 10^{-2}$
 - $0.010000 \cdot 10^{-7}$ $3,125000 \cdot 10^{-2}$
 - $0.001000 \cdot 10^{-6}$ $3,125000 \cdot 10^{-2}$
 - $0.000100 \cdot 10^{-5}$ $3,125000 \cdot 10^{-2}$
 - $0.000010 \cdot 10^{-4}$ $3,125000 \cdot 10^{-2}$
 - $0.000001 \cdot 10^{-3}$ $3,125000 \cdot 10^{-2}$
 - $0.000000 \cdot 10^{-2}$ $3,125000 \cdot 10^{-2}$

Неочевидные особенности вещественных чисел

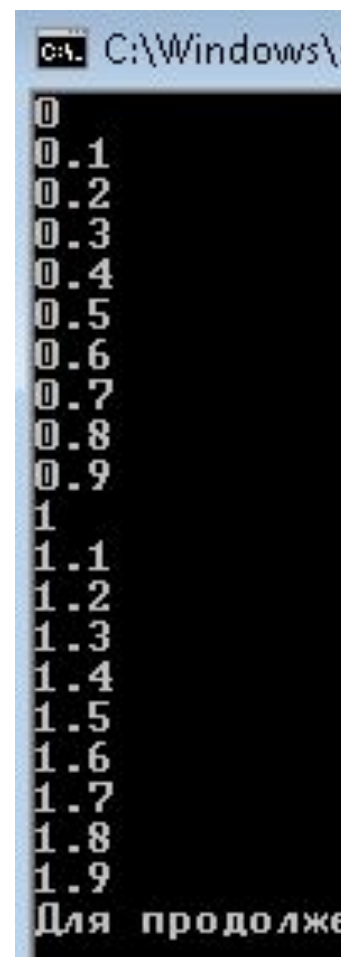
```
int _tmain(int argc, _TCHAR* argv[])
{
    for (double x = 0; x <= 2; x += 0.1)
        cout << x << endl;
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    for (double x = 0; x < 3; x += 0.3)
        cout << x << endl;
}
```

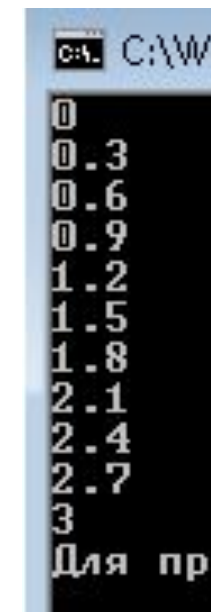
Неочевидные особенности вещественных чисел

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    for (double x = 0; x <= 2; x += 0.1)  
        cout << x << endl;  
}
```

```
int _tmain(int argc, _TCHAR* argv[])  
{  
    for (double x = 0; x < 3; x += 0.3)  
        cout << x << endl;  
}
```



```
C:\Windows\s  
0  
0.1  
0.2  
0.3  
0.4  
0.5  
0.6  
0.7  
0.8  
0.9  
1  
1.1  
1.2  
1.3  
1.4  
1.5  
1.6  
1.7  
1.8  
1.9  
Для продолже
```



```
C:\Win  
0  
0.3  
0.6  
0.9  
1.2  
1.5  
1.8  
2.1  
2.4  
2.7  
3  
Для про
```

Классы

- Класс – это набор из одной или более переменных и функций, возможно, различных типов, сгруппированных под одним именем
- Объявление:

```
ключ_класса имя_класса  
{  
    список_членов  
};
```

Классы

```
ключ_класса имя_класса  
{  
    список_членов  
};
```

- **ключ_класса** – одно из служебных слов `struct`, `union`, `class`
- **имя_класса** – идентификатор
- **список_членов** – определения и описания членов класса – данных и методов (функций)

Классы

Объявление и использование классов и их объектов:

```
struct MyDate  
{  
    int day;  
    int month;  
    int year;  
};
```

```
MyDate d1, arr[5];  
MyDate *p = &d1;
```


Классы

Размер класса не всегда совпадает с суммой размера полей:

```
struct MyDate
{
    int day;
    int month;
    int year;
};
```

```
sizeof(MyDate)
□ 12
```

```
struct MyDate2
{
    int day;
    int month;
    int year;
    char x;
};
```

```
sizeof(MyDate2)
□ 16
```

```
struct MyDate3
{
    int day;
    int month;
    int year;
    char x, y;
};
```

```
sizeof(MyDate3)
□ 16
```

Классы

Обращение к полям объекта класса:

```
struct MyDate  
{  
    int day;  
    int month;  
    int year;  
};
```

```
MyDate d1;  
d1.day = 30;  
d1.month = 11;  
d1.year = 2019;
```

Классы

Обращение к полям объекта класса:

```
struct MyDate  
{  
    int day;  
    int month;  
    int year;  
};
```

```
MyDate *p = new MyDate();  
p->day = 30;  
p->month = 11;  
p->year = 2019;
```

Классы

```
struct MyDate  
{  
    int day;  
    int month;  
    int year;  
};
```

```
MyDate d1, d2;  
d1.day = 30;  
d1.month = 11;  
d1.year = 2019;  
d2 = d1;           //побайтовое копирование содержимого памяти
```

Классы

Копирование полей-указателей

```
struct MyDate  
{  
    int day;  
    int month;  
    int year;  
    int *p;  
};
```

```
int k = 10;
```

```
MyDate d1, d2;  
d1.day = 30;  
d1.month = 11;  
d1.year = 2019;  
d1.p = &k;
```

```
d2 = d1;    //два указателя на k
```

Классы

Поля и методы класса:

```
struct MyDate
{
    int day;
    int month;
    int year;

    void print()
    {
        cout << day << "."
              << month << "."
              << year << endl;
    }
};
```

```
MyDate d1;
d1.day = 30;
d1.month = 11;
d1.year = 2019;
d1.print();
```

Классы

Поля и методы класса:

```
struct MyDate
{
    int day;
    int month;
    int year;

    void print();
};
void MyDate::print()
{
    cout << day << "."
         << month << "."
         << year << endl;
}
```

```
MyDate d1;
d1.day = 30;
d1.month = 11;
d1.year = 2019;
d1.print();
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s = new char[20];

    cin >> s;
    cout << s << endl;

    return 0;
}
```


Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s = new char[20];

    s = "some string";           //указатель изменён!
    cout << s << endl;

    return 0;
}
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s1 = "some string";
    char *s2 = new char[20];
    s2 = s1; // два указателя на одну строку!

    cout << s1 << endl;
    cout << s2 << endl;

    return 0;
}
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s = "some string"; //указатель на константу
    s[1] = 'a';              //ошибка: попытка изменить константу
    cout << s;

    return 0;
}
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char s[] = "some string"; // теперь это инициализация массива
    s[1] = 'a';                // копией строки, ошибки нет
    cout << s;

    return 0;
}
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s1 = "some string";
    char *s2 = new char[20];
    s2 = s1; // два указателя на одну строку!

    cout << s1 << endl;
    cout << s2 << endl;

    return 0;
}
```

Работа со строками (ASCII-Z строки)

```
#include <iostream>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    char *s1 = "some string";
    char *s2 = new char[20];
    strcpy(s2, s1);           // копирование содержимого строк!
    s2[1] = 'a';

    cout << s1 << endl;     // "some string"
    cout << s2 << endl;     // "same string"

    return 0;
}
```

Работа со строками (класс string)

```
#include <iostream>
#include <string>

using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    string s;
    cin >> s;           // чтение до первого пробельного символа
    cout << s << endl;
    return 0;
}
```

Работа со строками (класс string)

```
#include <iostream>
#include <string>

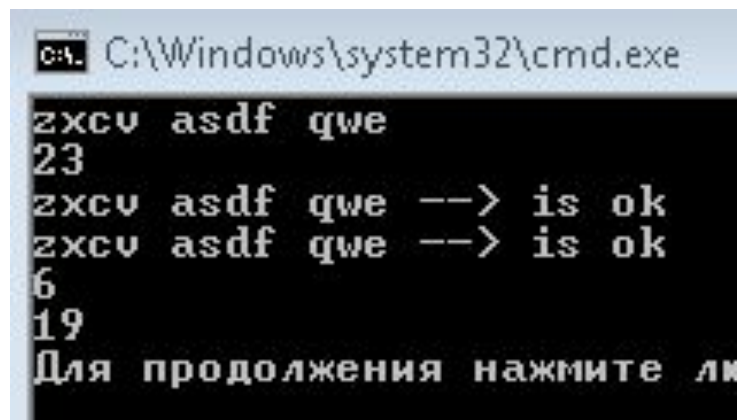
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    string s;
    getline(cin, s);    // чтение строки до \n
    cout << s << endl;
    return 0;
}
```


Работа со строками (класс string)

```
#include <iostream>
#include <string>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    string s;
    getline(cin, s);
    s += " --> is ok";
    cout << s.length() << endl;
    cout << s << endl;
    cout << s.c_str() << endl;
    cout << s.find_first_of("s") << endl;
    cout << s.find_last_of("s") << endl;
    return 0;
}
```



```
cmd.exe C:\Windows\system32\cmd.exe
zxcv asdf qwe
23
zxcv asdf qwe --> is ok
zxcv asdf qwe --> is ok
6
19
Для продолжения нажмите лк
```

Форматирование потоков вывода

```
double values[] = { 1.23, 35.36, 653.7, 4358.24 };
```

```
for (int i = 0; i < 4; i++)  
{  
    cout.width(10);  
    cout << values[i] << endl;  
}
```

```
1.23  
35.36  
653.7  
4358.24
```

Форматирование потоков вывода

```
double values[] = { 1.23, 35.36, 653.7, 4358.24 };
```

```
cout.fill('*');
```

```
for (int i = 0; i < 4; i++)
```

```
{
```

```
    cout.width(10);
```

```
    cout << values[i] << endl;
```

```
}
```

```
*****1.23  
*****35.36  
*****653.7  
***4358.24
```

Форматирование потоков вывода

```
#include <iomanip>
```

```
double values[] = { 1.23, 35.36, 653.7, 4358.24 };  
char *names[] = { "Zoot", "Jimmy", "Al", "Stan" };
```

```
for (int i = 0; i <4; i++)  
cout << setiosflags(ios::left)  
    << setw(6) << names[i]  
    << resetiosflags(ios::left)  
    << setw(10) << values[i] << endl;
```

```
Zoot      1.23  
Jimmy     35.36  
Al        653.7  
Stan     4358.24
```

Работа с файлами

```
MyDate d1;
```

```
d1.day = 30;
```

```
d1.month = 11;
```

```
d1.year = 2019;
```

```
ofstream f("123.txt");
```

```
f << d1.day << " " << d1.month << " " << d1.year;
```

```
f.close();
```

Работа с файлами

```
MyDate d1;
```

```
d1.day = 30;
```

```
d1.month = 11;
```

```
d1.year = 2019;
```

```
ofstream f("123.txt");
```

```
f.write( (char*)&d1, sizeof(d1) );
```

```
f.close();
```