

**ФГБОУ ВО ЧГУ им. И.Н. Ульянова**  
**факультет радиоэлектроники и автоматики**  
**кафедра автоматики и управления в технических системах**

# **Общие сведения о языке C++**

*Лекция 2.1.*  
**доцент Васильева Л.Н.**

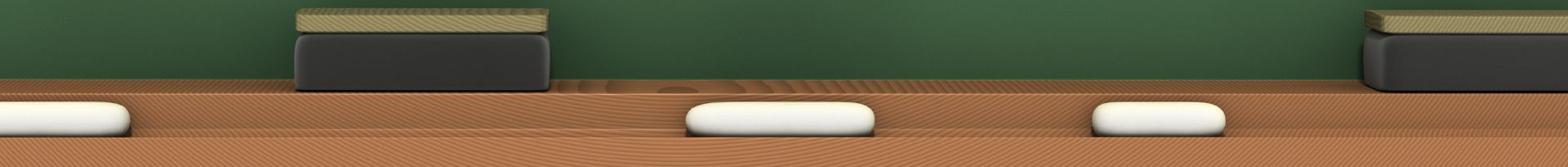
# История развития языка C/C++

Язык C был создан в начале 70-х годов Дэннисом Ритчи сотрудником компании Bell Telephone Laboratories.

Родословная языка берет свое начало от языка Алгол и включает в себя Паскаль и PL/I.

В конце 1970-х годов C начал вытеснять Бейсик с позиции ведущего языка для программирования микрокомпьютеров.

В 1980-х годах он был адаптирован для использования в IBM PC, что привело к резкому росту его популярности.



**C++ компилируемый** язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

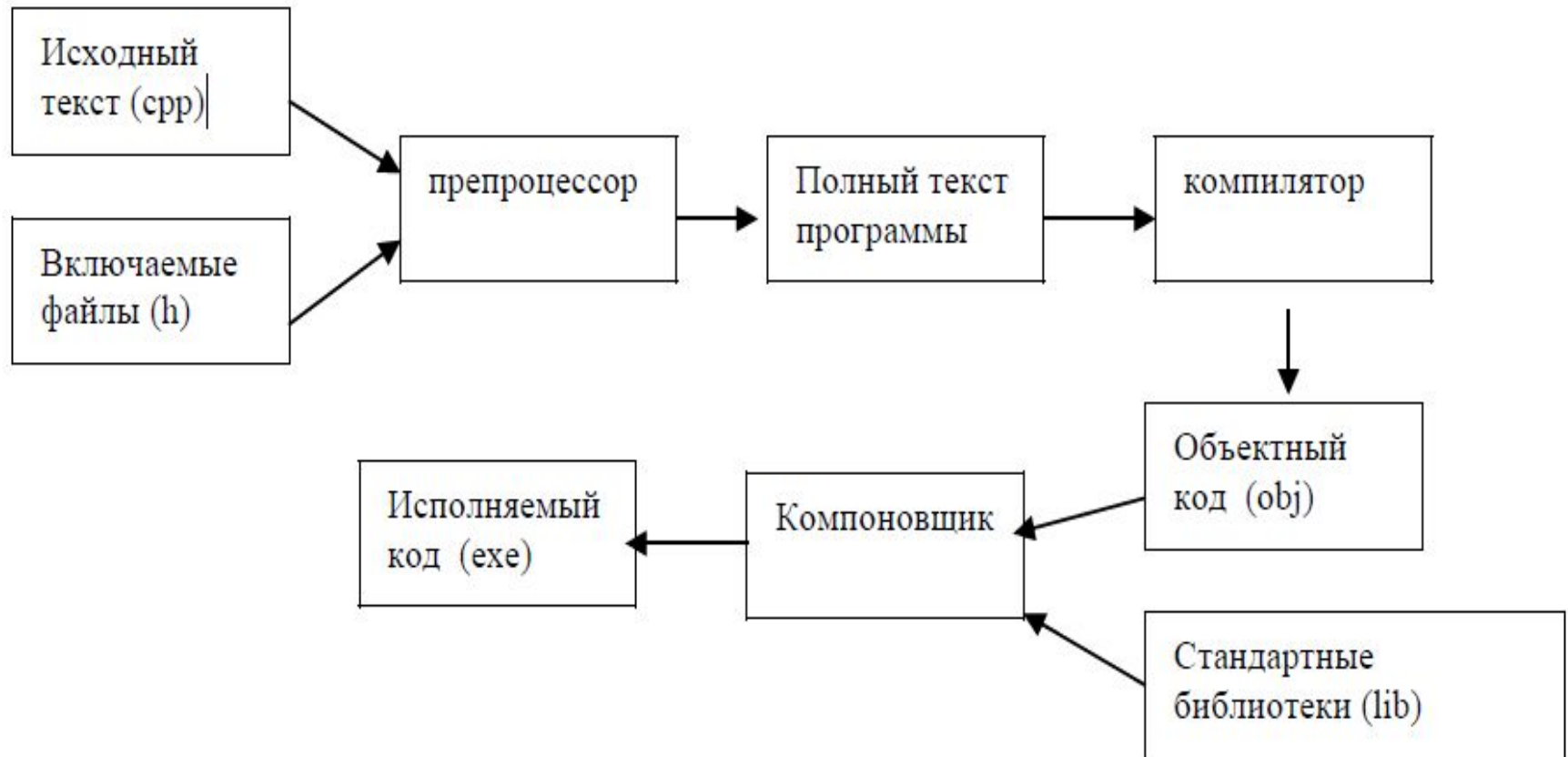
Язык программирования C++ широко **используется** для разработки программного обеспечения: создание разнообразных прикладных программ, разработка операционных систем, драйверов устройств, а также видео игр и многое другое.

**C++ разработан Бьерном Строустропом** сотрудником научно-исследовательского центра AT&T Bell Laboratories (Нью-Джерси, США) в 1979 году. Он придумал ряд усовершенствований к языку программирования C, для собственных нужд. Страуструп добавил возможность работы с классами и объектами.

Ранние версии языка C++, известные под именем и «C с классами», начали появляться с 1980 года. В 1983 году переименован на «язык программирования C++».

**Язык программирования C++ является свободным, то есть никто не обладает на него правами**

# Этапы создания исполняемого кода



# Структура программы на C++

#директивы препроцессора

.....

#директивы препроцессора

функция a( )

{операторы;}

функция b( )

{операторы;}

int **main** ( ) //функция, с которой начинается выполнение программы

{

**операторы:** описания, присваивания, функция, пустой оператор, составной, выбора, циклов, перехода;

return 0;

}

**Директивы препроцессора – управляют преобразованием текста программы до ее компиляции.**

Директива начинается со значка # (pound).

**#define** – указывает правила замены в тексте.

```
#define ZERO 0.0
```

**#include** – предназначена для включения в текст программы текста из каталога «Заголовочных файлов», поставляемых вместе со стандартными библиотеками.

```
#include<iostream>
```

- Употребление директивы **include** не подключает соответствующую стандартную библиотеку, а только позволяют вставить в текст программы описания из указанного заголовочного файла.



# Элементы языка C++



## 1. Алфавит языка C++

прописные и строчные латинские буквы и знак подчеркивания; арабские цифры от 0 до 9; специальные знаки "{},| []()+-/%\*.\':;&?<>=!#^; пробельные символы .

## 2. Лексемы языка

- **идентификаторы** – имена объектов .

**PROG1, prog1 и Prog1 – три различных идентификатора!!!**

Первым символом должна быть буква или знак подчеркивания (**!не цифра**).

- **ключевые** (зарезервированные) **слова** – это слова, которые имеют специальное значение для компилятора. Их нельзя использовать в качестве идентификаторов.

- **3. Знаки операций** – это один или несколько символов, определяющих действие над операндами.

- **4. Константы** – это неизменяемые величины (целые, вещественные, символьные и строковые константы).

- **5. Разделители** – скобки, точка, запятая пробельные символы.



# Типы данных в C++

Тип данных определяет:

- 1) внутреннее представление данных в памяти компьютера;
- 2) множество значений, которые могут принимать величины этого типа;
- 3) операции и функции, которые можно применять к данным этого типа.

| Простые типы данных |   |
|---------------------|---|
| <i>int</i>          | <i>целый</i>                            |
| <i>char</i>         | <i>символьный</i>                       |
| <i>wchar_t</i>      | <i>расширенный символьный</i>           |
| <i>bool</i>         | <i>логический</i>                       |
| <i>float</i>        | <i>вещественный</i>                     |
| <i>double</i>       | <i>вещественный с двойной точностью</i> |
| <i>void</i>         | <i>пустое множество</i>                 |

| спецификаторы типа |                    |
|--------------------|--------------------|
| <i>short</i>       | <i>короткий</i>    |
| <i>long</i>        | <i>длинный</i>     |
| <i>signed</i>      | <i>знаковый</i>    |
| <i>unsigned</i>    | <i>беззнаковый</i> |

# Переменные в C++

**Переменная в C++** - именованная область памяти, в которой хранятся данные определенного типа.

У переменной есть имя и значение.

Примеры:

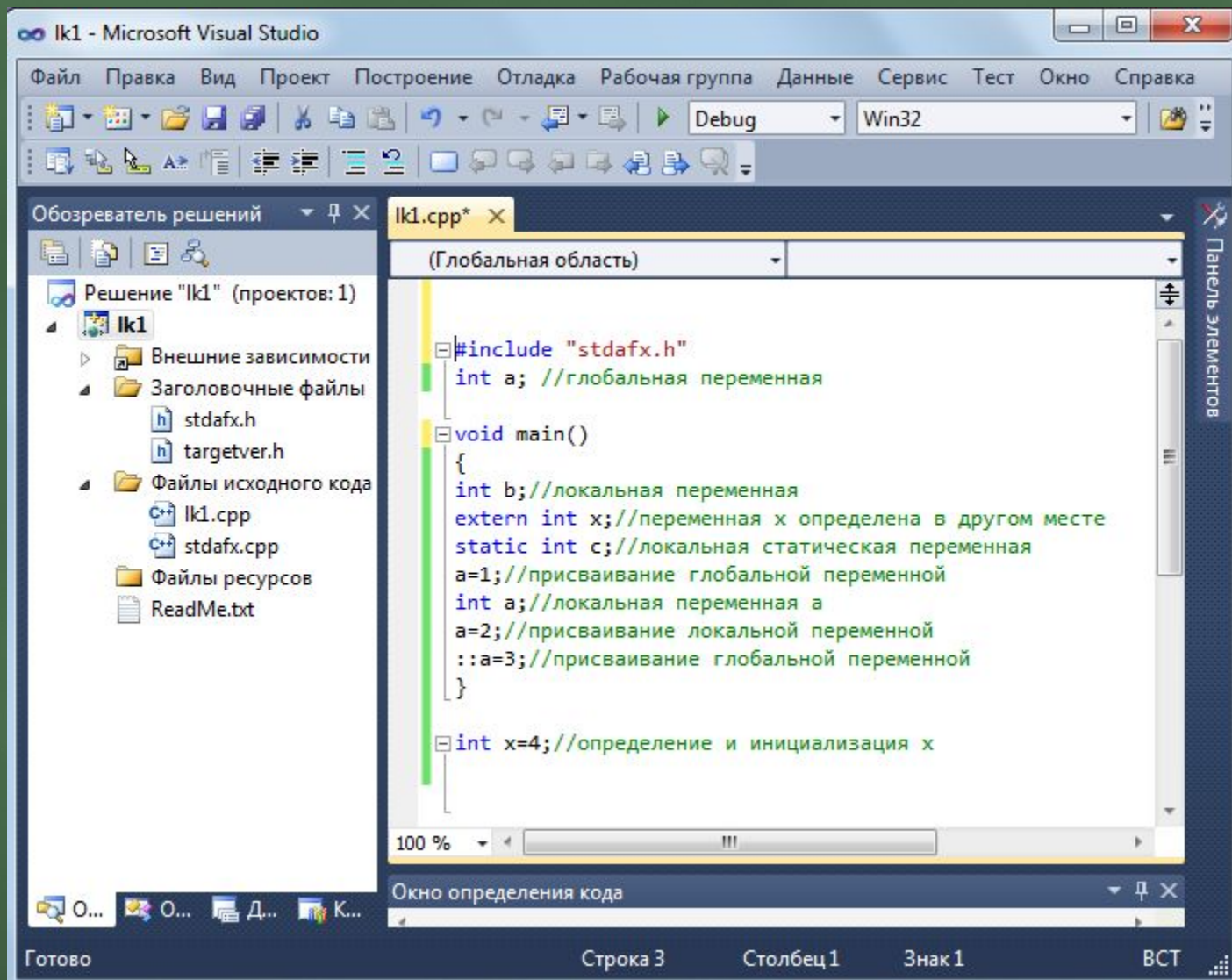
```
int a;  
float x=10.2;
```

**Общий вид оператора описания:**

*[класс памяти][const]тип имя [инициализатор];*

• *Класс памяти* определяет время жизни и область видимости переменной.

- **auto** – автоматическая *локальная переменная*. Спецификатор auto может быть задан только при определении объектов блока, например, в теле функции. Этим переменным память выделяется при входе в блок и освобождается при выходе из него. Вне блока такие переменные не существуют.
- **extern** – *глобальная переменная*, она находится в другом месте программы (в другом файле или далее по тексту). Используется для создания переменных, которые доступны во всех файлах программы.
- **static** – *статическая переменная*, она существует только в пределах того файла, где определена переменная.
- **register** – аналогичны *auto*, но память под них выделяется в регистрах процессора. Если такой возможности нет, то переменные обрабатываются как auto.



# Операции и выражения в C++

**Выражение** задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций.

$$a+b*\sin(\cos(x)).$$

**Операции** делятся на унарные, бинарные и т.д.



| Операция                 | Описание                                 |
|--------------------------|--|
| <b>Унарные операции</b>  |  |
| <b>++</b><br>инкремент   | увеличение значения на единицу           |
| <b>--</b><br>декремент   | уменьшение значения на единицу           |
| <b>~</b>                 | поразрядное отрицание                    |
| <b>!</b>                 | логическое отрицание                     |
| <b>-</b>                 | арифметическое отрицание (унарный минус) |
| <b>+</b>                 | унарный плюс                             |
| <b>&amp;</b>             | взятие адреса                            |
| <b>*</b>                 | разадресация                             |
| <b>(type)</b>            | преобразование типа                      |
| <b>Бинарные операции</b> |  |
| <b>+</b>                 | сложение                                 |
| <b>-</b>                 | вычитание                                |
| <b>*</b>                 | умножение                                |



| Операция | Описание                     |
|----------|------------------------------|
| /        | деление                      |
| %        | остаток от деления           |
| <<       | сдвиг влево                  |
| >>       | сдвиг вправо                 |
| <        | меньше                       |
| >        | больше                       |
| <=       | меньше или равно             |
| >=       | больше или равно             |
| ==       | равно                        |
| !=       | не равно                     |
| &        | поразрядная конъюнкция (И)   |
| ^        | поразрядное исключающее ИЛИ  |
|          | поразрядная дизъюнкция (ИЛИ) |
| &&       | логическое И                 |
|          | логическое ИЛИ               |
| =        | присваивание                 |
| *=       | умножение с присваиванием    |
| /=       | деление с присваиванием      |
| +=       | сложение с присваиванием     |

Операции присваивания имеет вид:

*имя\_переменной=значение;*

Множественное присваивание в общем виде может быть записано следующим образом:

*имя\_переменной1= имя\_переменной2=...=  
имя\_переменнойN=значение;*

Пример

$a=b=c=3.14159;$

Составным присваиванием являются операции  $+=$ ,  $-=$ ,  $*=$ ,  $/=$ .

$x+=r;$  // Увеличение  $x$  на  $r$ , то же что и  $x=x+r$ .

$x-=r;$  // Уменьшения  $x$  на  $r$ , то же что и  $x=x-r$ .

$x*=r;$  // Умножение  $x$  на  $r$ , то же что и  $x=x*r$ .

$x/=r;$  // Деление  $x$  на  $r$ , то же что и  $x=x/r$ .

Операции инкремента ++ и декремента -- выполняют увеличение и уменьшение на единицу значения переменной.

Эти операции имеют две формы записи  
**префиксную и постфиксную.**

### Пример

оператор `r=r+1;`

можно представить в префиксной форме `++r;`

и в постфиксной `r++;`

### Пример

`x=12; y=++x; //у будет иметь значение 13.`

`x=12; y=x++; //у будет иметь значение 12`

# Операции битовой арифметики

**Арифметическое И (&)** Оба операнда переводятся в двоичную систему, затем над ними происходит логическое поразрядное умножение операндов по следующим правилам:

$$1 \& 1 = 1, \quad 1 \& 0 = 0, \quad 0 \& 1 = 0, \quad 0 \& 0 = 0.$$

## Пример

$A = 13_{10} = 00000000000001101_2$  и  $B = 23_{10} = 00000000000010111_2$

$$\begin{array}{r} 00000000000001101 \\ \& \quad 00000000000010111 \\ \hline 0000000000000101 = 5_{10} \end{array}$$

**Арифметическое ИЛИ (|)** Оба операнда переводятся в двоичную систему, после чего над ними происходит логическое поразрядное сложение операндов по следующим правилам:

$$1 | 1 = 1, \quad 1 | 0 = 1, \quad 0 | 1 = 1, \quad 0 | 0 = 0.$$

## Пример

$$\begin{array}{r} 00000000000001101 \\ | \quad 00000000000010111 \\ \hline 00000000000011111 = 31_{10} \end{array}$$

**Арифметическое исключающее ИЛИ (^)** Оба операнда переводятся в двоичную систему, после чего над ними происходит логическая поразрядная операция ^ по следующим правилам:  
 $1 \wedge 1 = 0$ ,  $1 \wedge 0 = 1$ ,  $0 \wedge 1 = 1$ ,  $0 \wedge 0 = 0$ .

**Арифметическое отрицание (~)** Операция ~ вызывает побитную инверсию двоичного представления числа

Пример  $\sim 13$

$\sim 00000000000001101 = 11111111111110010$

**Сдвиг влево ( $M \ll L$ )** Двоичное представление числа M сдвигается влево на L позиций.

Пример  $17 \ll 3$ .

$17_{10} = 10001_2$ .

$10001000 = 136_{10}$ . Итак,  $17 \ll 3 = 136$ .

**Сдвиг влево** на один разряд соответствует умножению на 2, на два разряда умножению на 4, на три умножению на 8. Таким образом, операция  $M \ll L$  эквивалентна умножению числа M на  $2^L$ .

**При сдвиге вправо ( $M \gg L$ )** двоичное представление числа M сдвигается вправо на L позиций, что эквивалентно целочисленному делению числа M на  $2^L$ .

Пример  $25 \gg 1 = 12$ ,  $25 \gg 3 = 8$ .

# Условная операция

Для организации ветвлений в простейшем случае можно использовать *условную операцию* ?:

Операция имеет три операнда и в общем виде может быть представлена так:

*условие ? выражение1 : выражение2;*

Если условие истинно, то результатом будет выражение1, в противном случае выражение2.

## Пример

$y = x < 0 ? -x : x;$  // **вычисляется абсолютное знач. x**



# Операция преобразования типа

Для приведения выражения к другому типу данных в C++ существует **операция преобразования типа**:

*(тип) выражение;*

## Пример

```
x=5;
```

```
y=x/2;
```

```
z=(float) x/2;
```

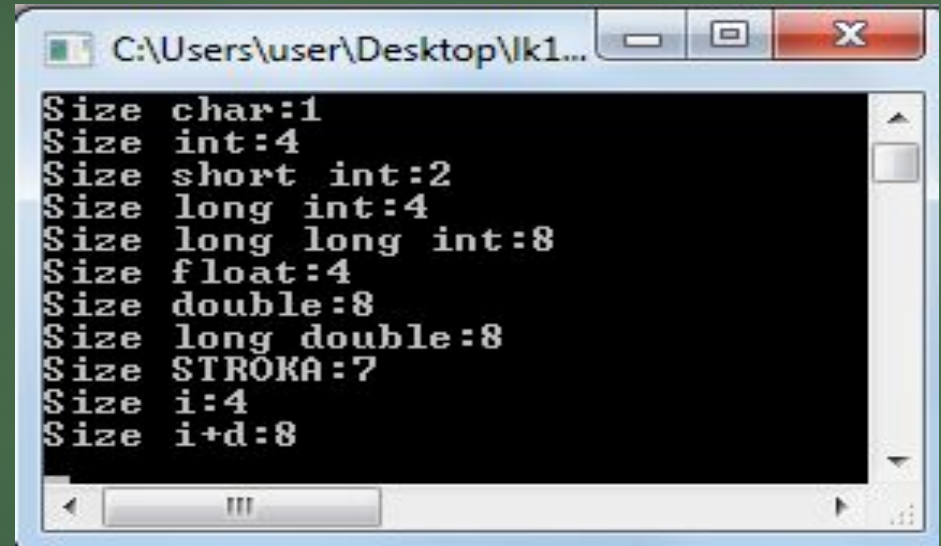
переменная y примет значение равное 2, а переменная z = 2.5.

# Операция определения размера

Вычисляет размер объекта или типа в байтах

*sizeof (тип) или sizeof выражение*

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{int i=3; double d=0.2;
//Вычисление размеров различных типов данных:
cout<<"Size char:"<<sizeof (char)<<"\n";
cout<<"Size int:"<<sizeof (int)<<"\n";
cout<<"Size short int:"<<sizeof (short int)<<"\n";
cout<<"Size long int:"<<sizeof (long int)<<"\n";
cout<<"Size long long int:"<<sizeof (long long int)<<"\n";
cout<<"Size float:"<<sizeof (float)<<"\n";
cout<<"Size double:"<<sizeof (double)<<"\n";
cout<<"Size long double:"<<sizeof (long double)<<"\n";
cout<<"Size STROKA:"<<sizeof "STROKA"<<"\n";
cout<<"Size i:"<<sizeof i<<"\n";
cout<<"Size i+d:"<<sizeof (i+d)<<"\n";
getch();
return 0;}
```



```
Size char:1
Size int:4
Size short int:2
Size long int:4
Size long long int:8
Size float:4
Size double:8
Size long double:8
Size STROKA:7
Size i:4
Size i+d:8
```

# Приоритеты операций в выражениях

| Ранг | Операции                          |
|------|-----------------------------------|
| 1    | () []                             |
| 2    | ! ~ ++ -- & * (тип) sizeof тип( ) |
| 3    | * / %                             |
| 4    | + -                               |
| 5    | << >>                             |
| 6    | < > <= >=                         |
| 7    | == !=                             |
| 8    | &                                 |
| 9    | ^                                 |
| 10   |                                   |
| 11   | &&                                |
| 12   |                                   |
| 13   | ?:                                |
| 14   | = *= /= %= -= &= ^=  = <<= >>=    |

# Контрольные вопросы

Чему будет равно значение выражений:

1. `int z=x/y++;` если `int x=1, y=2;`

ответ: 0

2. `int w=x/++y;` если `int x=2, y=1;`

ответ: 1

3. `int a=++m+n++*sizeof(int);` если `int m=1, n=2;`

ответ: 10

4. `float a=4*m/0.3*n;` если `float m=1.5; int n=5;`

ответ: 100

5. `int ok=int(0.5*y)<short(x++);` если `int x=10, y=3;`

ответ: 1

# Функции ввода и вывода данных

Необходима директива `#include <stdio.h>`

`printf`(строка форматов, список выводимых переменных)

## Пример:

```
printf ("Значение числа Пи равно%f\n", pi);
```

Форматная строка может содержать: символы печатаемые текстуально; спецификации преобразования; управляющие символы.

## Спецификации:

`%d, %i; %f; %e,%E ;%u; %c ; %s` и др.

## Управляющие символы:

`\n; \t` и др.

**Модификаторы** – числа, которые указывают минимальное количество позиций для вывода значения и количество позиций для вывода дробной части числа:

`% [-] m [ . p ] C`

`scanf` (строка форматов, список адресов выводимых переменных);

## Пример:

```
scanf ("%d%f", &x, &y);
```

# Объектно-ориентированные средства ВВОДА-ВЫВОДА

Используется библиотечный файл `iostream`, в котором определены стандартные потоки ввода данных от клавиатуры `cin` и вывода данных на экран дисплея `cout`, а также соответствующие операции

`<<` - операция записи данных в поток;

`>>` - операция чтения данных из потока.

## Пример:

```
#include <iostream>
```

```
Using namespace std;
```

```
.....
```

```
Int n;
```

```
cout << "\nВведите количество элементов: ";
```

```
cin >> n;
```

```
.....
```



# Контрольные вопросы

1. Что такое форматная строка? Что содержит форматная строка функции printf, функции scanf?
2. Что такое спецификация преобразования?
3. Что будет выведено функцией  
`printf("\nСреднее арифметическое  
последовательности чисел равно: %10.5f  
\nКоличество четных элементов  
последовательности равно%10.5d ", S/n, k);`
4. Как записать вывод результатов из вопроса 3 с помощью оператора cout?
5. Как выполнить ввод переменных x и y, где x типа long int, а y типа double с помощью функции scanf? С помощью операции >> ?

# Литература

- *Дејтел Х.М.* Как программировать на С++ / Х.М. Дејтел, П.Дж. Дејтел. – М.: Бином, 2007.
- *Павловская Т.А.* С/С++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб.: Питер, 2005.
- *Подбельский В.В.* Язык С++ / В.В. Подбельский. – М.: Финансы и статистика, 2006.
- *Труб И.И.* Объектно-ориентированное моделирование на С++: учебный курс / И.И. Труб. – СПб.: Питер, 2006.
- *Франка П.* С++: учебный курс / П. Франка. – СПб.: Питер, 2006.