



---

# Переменные и типы данных



# Задачи:

---

- Анализировать переменные
- Находить различия между переменными и константами
- Приводить списки различных типов данных и смотреть, как они применяются в различных программах на С
- Анализировать различные арифметические операторы

# Структура простой программы

---

- `#include <iostream>`
- 
- `int main()`
- `{`
- `std::cout << "Hello, world!";`
- `return 0;`
- `}`



# Алфавит

---

Программа на языке программирования записывается с помощью символов, образующих алфавит языка. Алфавит языка C++ включает:

- большие латинские буквы от A до Z
- малые латинские буквы от a до z
- цифры от 0 до 9
- знаки препинания: , . ; : ! ?
- скобки: ( ) [ ] { }
- знаки математических операций: + - \* / < > =

# Алфавит



---

## Специальные символы:

- \ (обратная наклонная черта)
- ~ (волна или тильда)
- & (амперсant)
- # (решетка или диез)
- ' (апостроф или одиночная кавычка)
- " (двойная кавычка)
- ^ (стрелка)
- % (процент)
- \_ (знак подчеркивания)



# Алфавит

---

- Большие и малые буквы считаются различными.
- Знак подчеркивания считается буквой.
- Русские буквы нельзя использовать в операторах языка C++, их можно применять в комментариях и в строках символов, заключенных в двойные кавычки, а также как одиночные символы, заключенные в апострофы.



# Лексемы

---

- Программа представляет собой последовательность лексем.
- **Лексема** – это фрагмент программы, имеющий самостоятельное значение.
  - Различают следующие виды лексем:
    - **ключевые слова,**
    - **идентификаторы,**
    - **числовые константы,**
    - **строковые константы,**
    - **символьные константы,**
    - **знаки препинания и знаки операторов.**



# Лексемы

---

- **Ключевое слово** — это лексема, имеющая некоторое предопределенное значение,

Например:

ключевое слово **int** указывает, что величина, перед которой оно стоит, является целым числом, а ключевое слово **for** используется при организации циклов.

- Ключевые слова нельзя использовать для каких-либо других целей.





# Лексемы

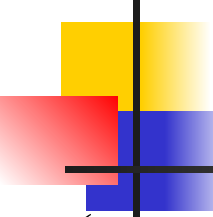
---

- **Идентификатор** – это последовательность букв и цифр, начинающаяся с буквы.
- Идентификаторы используются как **имена элемента** (объекта) программы (переменных, функций, классов).
- Например, в строку программы
- **int x0;**
- входят три лексемы: ключевое слово **int**, имя переменной целого типа
- **x0** и знак препинания (**;**), который завершает данную инструкцию

# Лексемы

- **Числовые константы** – задают в программе конкретные целые или дробные числа, записываемые по обычным правилам, например, 123, 32.1, 3.21e2.
- **Строковые константы** – это последовательности произвольных символов, заключенные в двойные кавычки, например, "Строка символов", "String of characters".
- Символьные константы представляют в программе одиночные символы, при записи заключаются в одиночные апострофы, например, 'a', 'A', '0', '1'. Важно понимать, что '0' это символ цифры ноль, а не число ноль.

# Лексемы



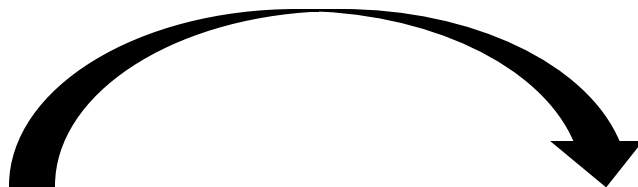
~~К знакам препинания в C++ относятся два символа: (; (точка с запятой) и (,) (запятая), служащие для разделения лексем.~~

- **Символом (;)** оканчивается любое предложение (оператор) программы.
- **Запятая** разделяет переменные при их объявлении, аргументы функций.
- В некоторых случаях запятая является не только чистым знаком препинания, но и оператором (операндом).
- Другие символы (. : ? !), которые в естественных языках являются знаками препинания, в C++ обозначают операторы (операнды).
- Кроме знаков препинания, разделителями лексем являются

**пробелы и знаки операторов**

# Переменные

Данные **15**



Память

	<b>15</b>		
	Data in memory		

**Расположение в памяти уникально**

**Переменным присваивается имя (идентификатор),  
указывающее на местоположение в памяти**



# Переменные

---

**Переменная обладает характеристиками:**

**- имя (идентификатор)**

**-тип**

**- значение**



# Пример

```
BEGIN  
  DISPLAY 'Enter 2 numbers'  
INPUT A, B  
C = A + B  
  DISPLAY C  
END
```

A, B и C – переменные псевдокода

Переменные избавляют программиста от необходимости указывать доступ к памяти по адресу

Операционная система отвечает за распределение переменных в памяти

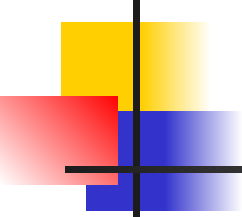
Для обозначения участка памяти достаточно использовать имя переменной



# Константы

---

- Константы никогда не изменяют значения
- Примеры
  - ◆ 5            **число / константа целого типа**
  - ◆ 5.3        **число / константа с плавающей точкой**
  - ◆ 'Black'     **строковая константа**
  - ◆ 'C'         **символьная константа**
- Переменные содержат значения констант

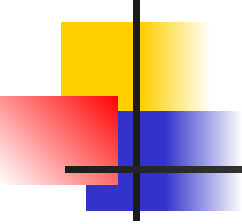


# Имена идентификаторов

---

- Имена переменных, функций, меток и других объектов определенных пользователем называются **идентификаторами**
- Некоторые правильные имена идентификаторов
  - arena
  - s\_count
  - marks40
  - class\_one
- Примеры неправильного использования имен идентификаторов
  - 1sttest
  - oh!god **! is invalid**
  - start... end
- Идентификаторы могут быть любой удобной длины, но количество символов распознаваемых компилятором различается в зависимости от компилятора
- Идентификаторы в C чувствительны к реестру





# Советы для наименования идентификаторов

---

Имена переменных должны начинаться с буквы

После первого символа могут идти как буквы так и цифры

Имена идентификаторов не должны совпадать с именами переменных

Имя переменной должно что то означать и описывать называемый объект

Не должно быть многозначных символов

Должны соблюдаться правила о наименовании стандартных переменных



# Данные

---

- В переменных могут храниться различные типы данных. Вот некоторые примеры:
  - ◆ Числа.
    - Целые числа. Например, 10 или 178993455.
    - Вещественные. Например, 15.22 или 15463452.25.
    - Положительные числа.
    - Отрицательные числа.
  - ◆ Имена. Например, John.
  - ◆ Логические значения. Например, Y или N.



# Данные

---

Тип данных описывает тот тип информации, которая там будет храниться

Имя переменной указывается перед её типом данных

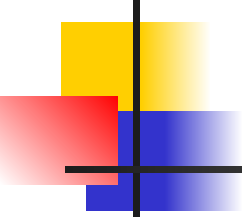
Например, тип данных **int** указывается перед именем **varName**

Тип данных имя переменной

```
int varName
```

# Основные типы данных

Категория	Тип	Значение	Пример
Логический тип данных	bool	true или false	true
Символьный тип данных	char, wchar_t, char16_t, char32_t	Один из ASCII-символов	'с'
Тип данных с плавающей запятой	float, double, long double	Десятичная дробь	3.14159
Целочисленный тип данных	short, int, long, long long	Целое число	64
Пустота	void	Пустота	



# Целочисленный тип данных(int)

---

- Хранит целые числа

```
int num;
```

- Не может хранить другой тип данных, как, например, "Alan" или "abc"
- Имеет размер 16 бит (2 байта)
- Диапазон значений от -32768 до 32767
- Например: 12322, 0, -232



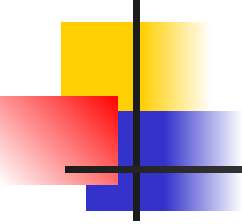
# Число с плавающей точкой(float)

---

- Содержит значения, имеющие десятичную точку

`float num;`

- Может хранить значения с точностью до 6 знаков после запятой
- Имеет размер 32 бита (4 байта) памяти
- Примеры: 23.05, 56.5, 32



# Число с плавающей точкой двойной точности(**double**)

---

- Содержит значения, имеющие десятичную точку

```
double num;
```

- Может хранить значения с точность до 10 знаков после запятой
- Имеет размер 64 бита (8 байт)
- Примеры: 32.54, -152.369



# Символьный тип(char)

---

- Хранит один символ информации

```
char gender;
```

```
gender='M';
```

- Имеет размер 8 бит (1 байт)

- Примеры: 'a', 'm', '\$' '%', '1', '5'





# Процедурный тип Void

---

- Этот тип данных не хранит никакой информации
- Показывает компилятору, что здесь не присутствует никакой информации

# Установленный тип данных

Тип данных модификатора  
в

+

Основной тип данных

=

Наследуемый тип данных

**unsigned**

+

**int**

=

**unsigned int**  
(Допускаются только положительные числа)

**short**

+

**int**

=

**short int**  
(Занимает меньше места в памяти чем int)

**long**

+

**int/double**

=

**Long int / longdouble**  
(Занимает больше места в памяти чем int/double)

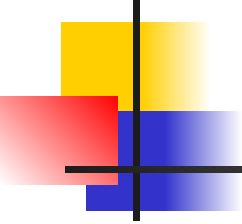
# Подписанные и

# неподписанные типы данных

- **Неподписанные типы данных (`unsigned`)** точно определяют, что переменная может содержать только положительные значения

```
unsigned int varNum;  
varNum=23123;
```

- Переменная `varNum` занимает 2 байта
- Модификаторы могут быть использованы как с целочисленными, так и с данными с плавающей точкой
- Неподписанные целочисленные данные поддерживают диапазон от 0 до 65535



# Короткие и длинные ТИПЫ ДАННЫХ

---

- **Короткое целое**(short int) занимает 8 бит (1 байт)
  - ◆ Позволят хранить числа в диапазоне от -128 до 127
- **Длинное целое**(long int) занимает 32 бита (4 байта)
  - ◆ -2,147,483,647 and 2,147,483,647
- **Длинное целое двойной точности**(long double) занимает 128 бит (16 байт)

# Типы данных и их диапазоны-1

Тип	Аппроксимированный размер в битах	Минимальный диапазон
char	8	-128 to 127
unsigned	8	0 to 255
signed char	8	-128 to 127
int	16	-32,768 to 32,767
unsigned int	16	0 to 65,535
signed int	16	Same as int
short int	16	Same as int
unsigned short int	8	0 to 65, 535

# Типы данных и их диапазоны -2

Тип	Аппроксимированный размер в битах	Минимальный диапазон
signed short int	8	Тоже что и short int
signed short int	8	Тоже что и short int
long int	32	-2,147,483,647 to 2,147,483,647
signed long int	32	0 to 4,294,967,295
unsigned long int	32	0 to 4,294,967,295
float	32	6-ти значная точность после запятой
double	64	10-ти значная точность после запятой
long double	128	10-ти значная точность после запятой

# Примеры объявления переменных

```
int main()
{
    char abc;      /*abc символьного типа */
    int xyz;      /*xyz целочисленного типа */
    float length; /*length как число с плавающей
точкой */
    double area;  /*area длинное целое двойной
точности */
    long liteyrs; /*liteyrs как длинное целое */
    short arm;    /*arm как короткое целое*/
}
```

# Примеры объявления переменных

При объявлении переменной мы можем присвоить ей значение в этот же момент. Это называется **инициализацией переменной**.

```
int main()  
{  
    int nValue = 5; // копирующая инициализация  
    int mValue(5); // прямая инициализация  
}
```



# Примеры объявления переменных

В попытке обеспечить единый механизм инициализации, который будет работать со всеми типами данных, в C++11 добавили новый способ инициализации, который называется `uniform-инициализация`:

```
int main()
{
    int value1{5};
    int value2{}; /* инициализация переменной по
умолчанию значением 0 (ноль) */
}
```

# Примеры объявления переменных

В `uniform`-инициализации есть еще одно дополнительное преимущество: вы не сможете присвоить переменной значение, которое не поддерживает её тип данных — компилятор выдаст предупреждение или сообщение об ошибке. Например:

```
int main()
{
    int value{4.5}; /* ошибка: целочисленная
переменная не может содержать нецелочисленные
значения*/
}
```

# Примеры объявления переменных

Когда переменной присваивается значение после её объявления (не в момент объявления), то это **копирующее присваивание** (или просто *«присваивание»*):

```
int main()
{
    int nValue;
    nValue = 5; // копирующее присваивание
}
```

# Примеры объявления переменных

В одном операторе можно объявить сразу несколько переменных одного и того же типа данных, разделяя их имена запятыми. Например, следующие 2 фрагмента кода выполняют одно и то же:

```
int main()  
{  
    int a, b;  
}
```

```
int main()  
{  
    int a;  
    int b;  
}
```

# Примеры объявления переменных

Можно инициализировать несколько переменных в одной строке

```
int main()  
{  
    int a = 5, b = 6;  
    int c(7), d(8);  
    int e{9}, f{10};  
}
```



# Ошибки объявления переменных

---

**Ошибка №1:** Указание каждой переменной одного и того же типа данных при инициализации нескольких переменных в одном описании. Это не критичная ошибка, так как компилятор легко её обнаружит и сообщит вам об этом:

```
int a, int b; // неправильно (ошибка компиляции)
```

```
int a, b; // правильно
```

# Ошибки объявления переменных

**Ошибка №2:** Использование разных типов данных в одном описании. Переменные разных типов должны быть объявлены в разных описаниях. Эту ошибку компилятор также легко обнаружит

```
int a, double b; // неправильно (ошибка компиляции)
```

```
int a; double b; // правильно (но не рекомендуется)
```

```
// Правильно и рекомендуется (+ читабельнее)
```

```
int a;  
double b;
```

# Ошибки объявления переменных



---

**Ошибка №3:** Инициализация двух переменных с помощью одной операции

**`int a, b = 5; //` неправильно (переменная `a` остаётся неинициализированной)**

**`int a = 5, b = 5; //` правильно**



# Объявление переменных в программе

Более старые версии компиляторов языка Си

вынуждали пользователей объявлять все переменные в верхней части функции:

```
#include <iostream>
int main()
{
    // Все переменные в самом верху функции
    int x;
    int y;
    // А затем уже весь остальной код
    std::cout << "Enter a number: ";
    std::cin >> x;
    std::cout << "Enter another number: ";
    std::cin >> y;
    std::cout << "The sum is: " << x + y << std::endl;
    return 0;
}
```

# Объявление переменных в программе

Современные компиляторы не требуют, чтобы все переменные обязательно были объявлены в самом верху функции. В языке C++ возможно объявление (а также инициализация) переменных как можно ближе к их первому использованию:

```
#include <iostream>
int main()
{
    std::cout << "Enter a number: ";
    int x; // мы используем x в следующей строке, поэтому объявляем эту
    переменную здесь (как можно ближе к её первому использованию)
    std::cin >> x; // первое использование переменной x
    std::cout << "Enter another number: ";
    int y; // переменная y понадобилась нам только здесь, поэтому здесь
    её и объявляем
    std::cin >> y; // первое использование переменной y
    std::cout << "The sum is: " << x + y << std::endl;
    return 0;
}
```