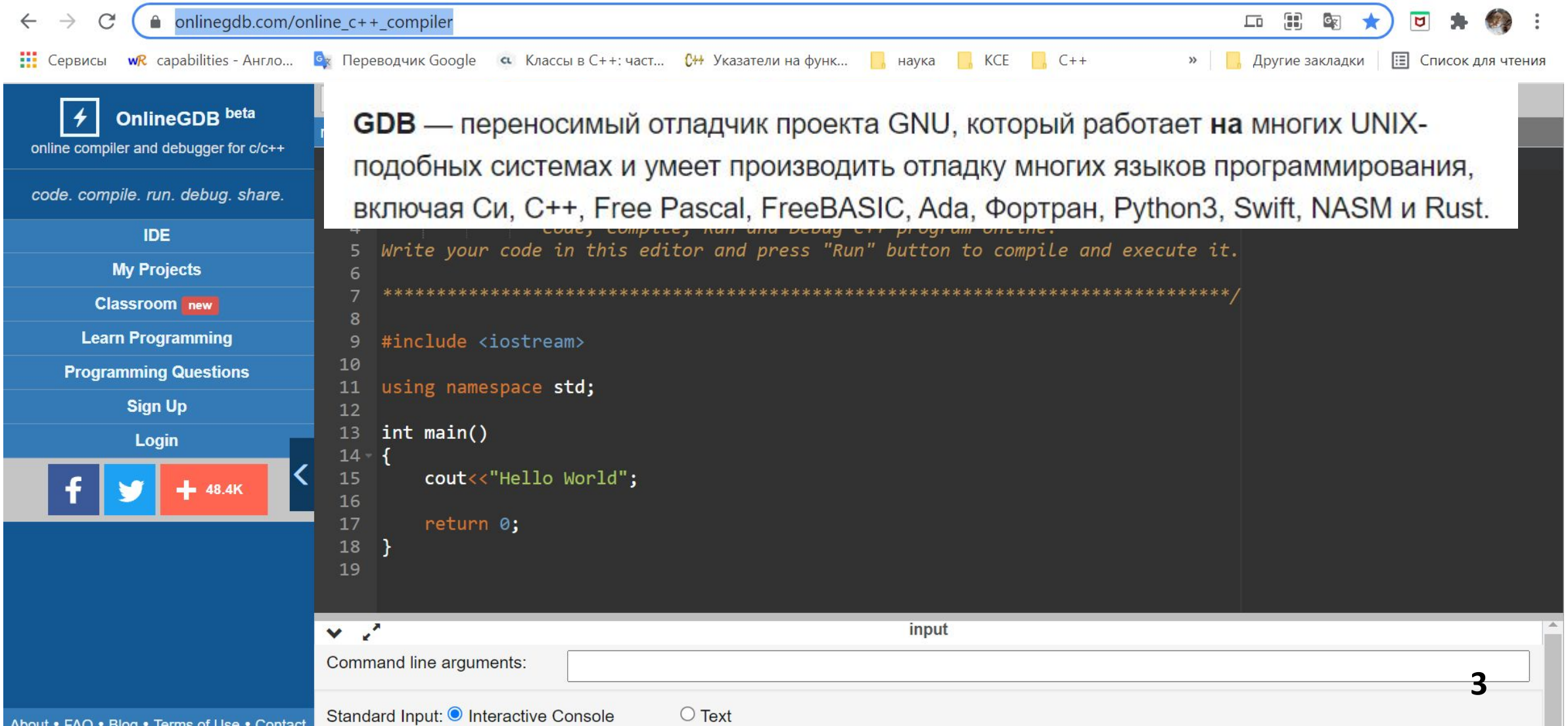
Лекция № 1. Вводная

Рекомендуемая литература

- Т.А. Павловская. *С/С++. Программирование на языке высокого уровня. СПб: Питер, 2013 - 461 стр.*
- Т.А. Павловская, Ю.А. Щупак. *С/С++. Программирование на языке высокого уровня. ПРАКТИКУМ. (Структурное программирование. СПб: Питер, 2013 - 264 стр.*

C++ online compiler:
https://www.onlinegdb.com/online_c++_compiler



Общие слова о языке, история

- Почему изучаем C++?
- Язык сложный, объемный, есть теория, что изучив язык можно легко перейти на другой язык
- Статистика C++ Не первый по популярности…Java,Питон и пр.
- Много чего написано на C++: Google, Яндекс, Facebook

(значительная часть), телеграмм и много чего другого.

- C++ развивается и раз в три года обновляется (C++ 03, C++11, C++14, C++ 17, C++20)
- Возникают проблемы в языке - они устраняются и опять добавляются новые. Существует комитет по стандартизации, который собирает информацию о проблемах, недочетах и решает что нужно добавить, а что устранить

Связи с другими языками

- Язык возник в начале 1980-х годов, когда сотрудник фирмы Bell Labs Бьёрн Страуструп придумал ряд усовершенствований к языку **C** под собственные нужды
- В некотором смысле C++ это улучшенный **C н**астолько что C маленькая часть от C++
- Почему C++ (++инкремент, т.е.+1)следовательно выше чес **C**
- Есть C#, тоже улучшение но в другую сторону
- Java, много общего C#

Структура программы и описание переменных

1. Структура программы

Программа состоит из нескольких разделов:

1. Раздел заголовков программы.

С помощью директивы #include подключаются необходимые заголовки, обеспечивающие включение в программу соответствующей информации.

С помощью директивы using namespace – объявляются используемые пространства имен.

Пространство имен это некоторая объявляемая область, необходимая для исключения конфликтов имен идентификаторов.

2. Раздел с объявлениями классов, прототипами и объявлениями функций.

Содержит прототипы функций.

3. Главная функция программы.

Эта функция имеет стандартное название main( )

4. Раздел с описанием функций.

Содержит описания функций.

7

Программа для отображения приветствия

```
#include <iostream>          // подключение заголовка
                             //  для поддержки ввода-вывода
using namespace std; // использовать стандартное пространство имен
int main( )          // главная функция программы
{    // начало блока
        cout  <<  "Hello, World!\n" ;      // вывод сообщения
        system("pause");
        return 0;       // функция возвращает значение 0
}    // завершение блока
```

Результат выполнения программы:

Hello, World!

2. Порядок работы в среде Visual Studio при создании проекта на C++

1. **Загружаем среду разработки**

   Microsoft Visual C++ 2008 Express Edition.lnk

или среду VisualStudio 2017

2. Меню Файл – *Создать – Проект*

3. **В диалоговом окне**:

- *Общие – Пустой проект – это важно! иначе проект будет создан по другому сценарию и креативной работы не получится*
- Вводим *Имя*
- Задаем – Расположение – Это необязательно для временных проектов: по умолчанию файлы пишутся в папки с самой средой VS.
- Кнопка ОК

Порядок ввода текста программы

4. **Меню Проект** – *Добавить новый элемент*
5. **В диалоговом окне**

   Visual C++ - здесь важно выбрать именно этот вид среды

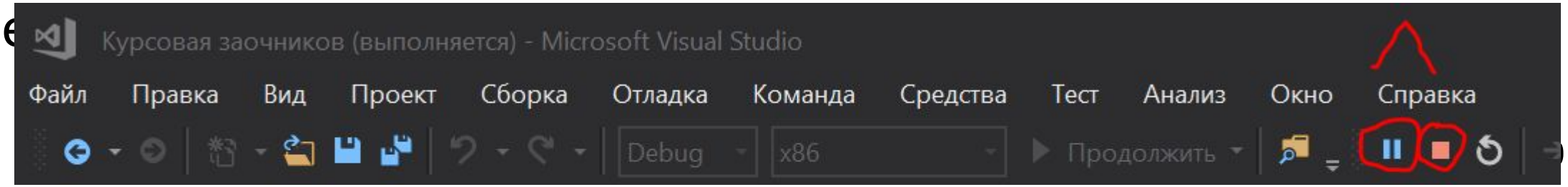
   ФайлС++(.cpp) – и этот тип файла

   Набираем Имя – имя тоже будет создаваться по умолчанию SourceN.cpp

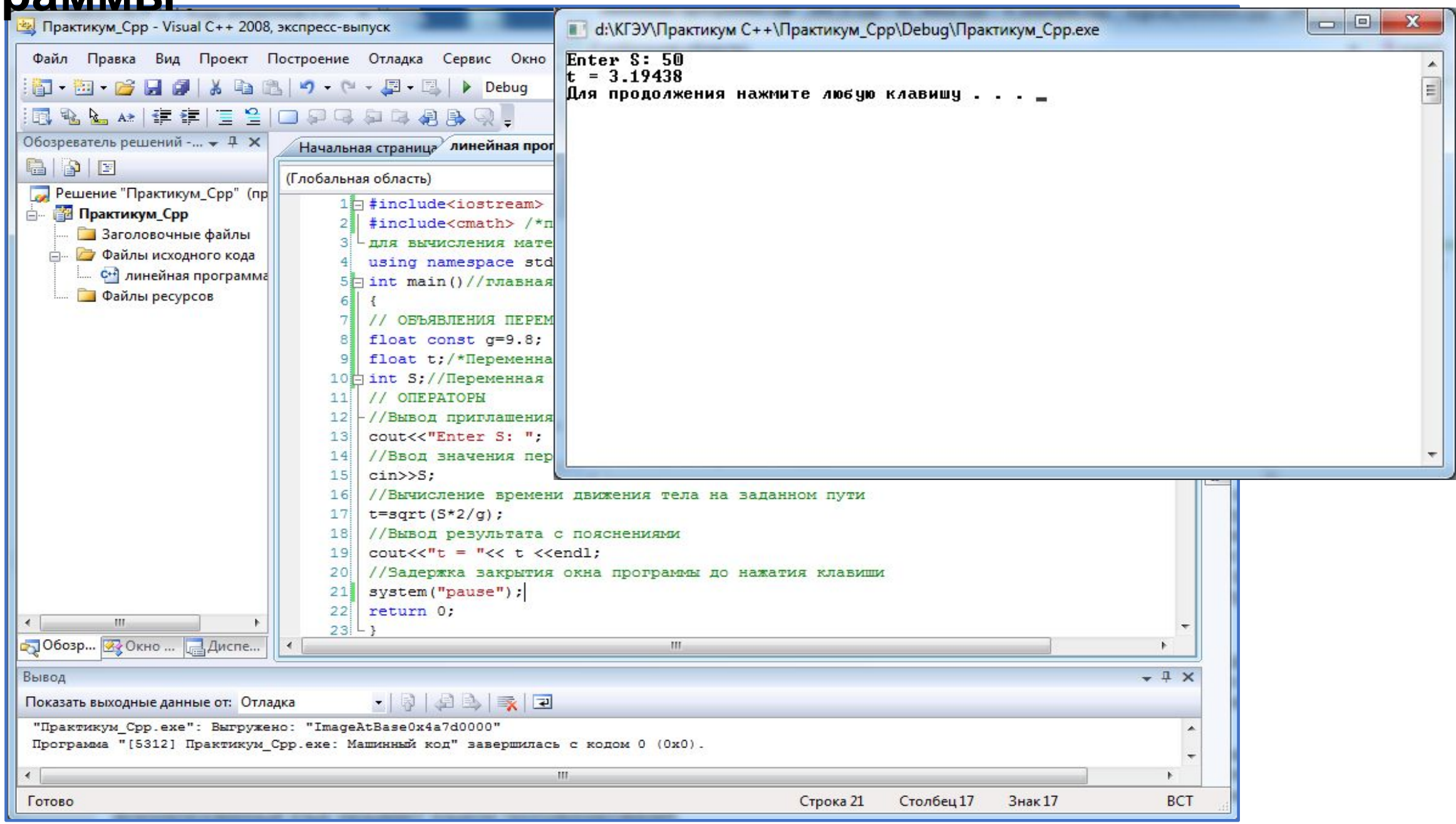
   Кнопка ДОБАВИТЬ

6. **В открывшемся окне набираем текст программы**
7. **Запускаем программу клавишей F5 или значко**

Рекомендация: В случае зависания программы или санкционированного прерывания её работы можно воспользоваться сле



Вид созданного проекта и результатов трансляции и работы программы



Структура программы

- Программа - это последовательность обьявлений. Одно из объявлений – это обявление функции main, она должна быть обязательна и выглядит:

```
Int main ()
{
Тело функции
}
```

  - Можно сказать что точка входа в программы находится здесь (мэйн)
  - Подключить заголовочные файлы. Что это?

- C++ не содержит в явном виде, например средств ввода-вывода. Чтобы они заработали нужно подключить некоторую библиотеку (стандартную, что описано в стандарте) в которой описано как они работают.
- Для этого нужно записать include и потом что вы хотите использовать, например, <iostream> (там реализованы средства ввода-вывода)
- Что содержит программа (что содержит тело функции). Тело функции состоит из инструкций, которые отделяются ; (точка с запятой) и их можно группировать фигурными скобками.

- Какие бывают инструкции. Упрощенно три вида: либо объявление, либо выражение, либо управляющая конструкция.

Объявление (declaration)

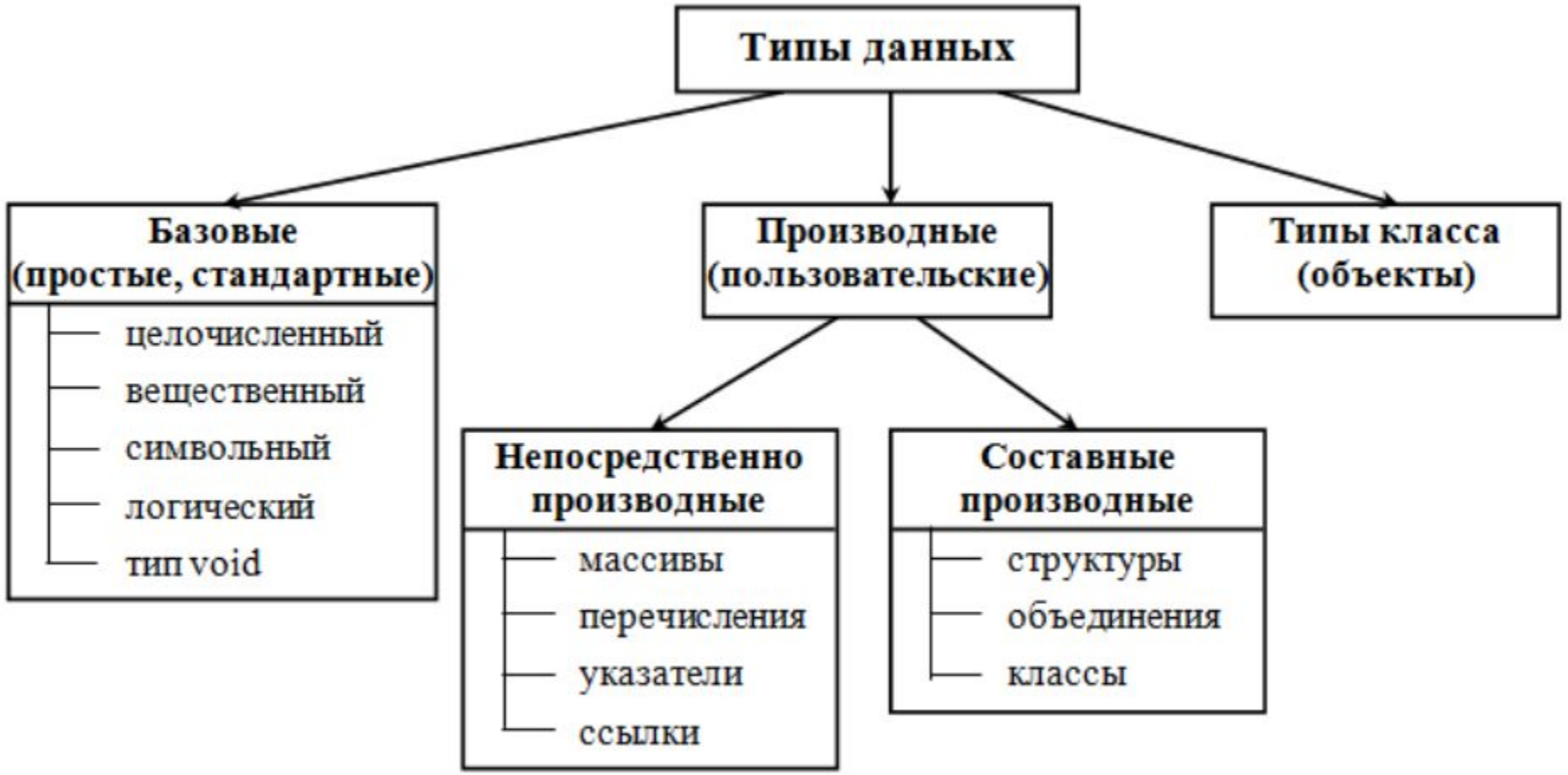
- Обьявление это когда вы вводите новую сущность, например переменную или новое название типа

Type identifier [=] ;

Type - тип

Identifier – идентификатор, имя переменной

Типы данных на C++



Целые типы данных (для 32-разрядного процессора)

Тип8	*Диапазон*	*Память, бит*	*Память, байт*
short int	от -32 768 (-2^{15}) до 32 767 ($+2^{15}$)	16	2
unsigned short int	от 0 до 65 535 ($+2^{16}$)	16	2
int (long)	от -2 147 483 648 (-2^{31}) до 2 147 483 647 ($+2^{31}$)	32	4
unsigned int	от 0 до 4 294 967 295 ($+2^{32}$)	32	4
long long	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 (?)	64	8
unsigned long long	от 0 до 18 446 744 073 709 551 615 (?)	64	8

- **Вещественные типы**

форма с фиксированной точкой

$$\left\{\begin{matrix}[+]\\ -\end{matrix}\right\}\ \langle\text{целая часть числа}\rangle.\langle\text{дробная часть числа}\rangle$$

Примеры: +45.6 45.6 -283.7 -0.19

форма с плавающей точкой

$$\left\{\begin{matrix}\langle\text{вещественное с ф. т.}\rangle\\ \langle\text{целое число}\rangle\end{matrix}\right\}\ \mathbf{e}\left\{\begin{matrix}[+]\\ -\end{matrix}\right\}\langle\text{целое число без знака}\rangle$$

мантисса числа *e* **порядок числа**

Десятичная экспонента

число: $0{,}12 \cdot 10^{-2}$

+0.12e-2

+0.12E-002

1.2e-3 нормализованный вид

12e-4

число: - 471

-0.471e3

-0.471E+003

-4.71e2 нормализованный вид

-471e0

Вещественные типы данных

Тип	*Диапазон*	*Количество цифр*	*Память, бит*	*Память, байт*
float	от **3.4E-38** до **3.4E+38**	7	32	4
double	от **1.7e-308** до **1.7e+308**	15	64	8
long double	**от 3.4.e-4932 до 3.4e+4932**	**???**	**64**	**8/10/16**

Количество цифр в типе *long double* зависит от типа ПК и процессора. Должно быть меньше 15 цифр.

Различные виды целых и вещественных типов, различающихся диапазоном и точностью представления данных, введены для того, чтобы дать программисту ***возможность наиболее эффективно использовать возможности конкретной аппаратуры, поскольку от выбора типа зависит скорость вычислений и объем памяти.***

Логический тип

Тип	*Значения*	*Память, бит*	*Память, байт*
bool	**true** - истина или **false** - ложь	8	1

Символьный тип

Тип	*Значения*	*Память , бит*
char	символ (-128 до 127)	8
unsigned char	символ (0 до 255)	8

'Э' 'ю' ' F ' ' z' '□'

Умолчания

- При программировании на C++ недопустимы умолчания как на типы данных, так и на их значения: любая переменная или константа ДОЛЖНА быть описаны (им должен быть присвоен тип) перед её использованием.
- Отсутствие *начальных значений* для переменных считается признаком «плохого тона», но , если значение переменной не определено, компилятор не выдает ошибок. *Однако,* если в программе есть обращение к неинициализированной переменной, то появляется соответственное предупреждение при трансляции.

Идентификаторы

Идентификатор – это наименование, присвоенное программистом объекту программы: переменной, массиву, функции.

Идентификатор представляет собой последовательность букв, цифр, знака подчеркивания, начинающуюся с буквы или ***знака подчеркивания***.

Идентификаторы не должны совпадать с зарезервированными словами.

Прописные и строчные буквы рассматриваются компилятором как различные символы.

Идентификатор должен (но не обязан) отражать существо обозначаемого им объекта.

Примеры правильных идентификаторов:

x

Z45

MyProgram

Примеры неправильных идентификаторов:

45d (наименование начинается с цифры)

int (это служебное слово в программе)

sin a (недопустим пробел)

Литералы

- **Литеральные константы** (или просто ***«литералы»***) — это значения, которые вставляются непосредственно в код. Поскольку они являются константами, то их значения изменить нельзя. Например:

```
1 bool myNameIsAlex = true; // true - это литеральная константа типа bool
2 int x = 5; // 5 - это литеральная константа типа int
3 int y = 2 * 3; // 2 и 3 - это литеральные константы типа int
```

- С литералами типов **bool** и **int** всё понятно, а вот для литералов **типа с плавающей точкой** есть два способа определения:

```
1 double pi = 3.14159; // 3.14159 - это литерал типа double
2 double avogadro = 6.02e23; // число avogadro - 6.02 x 10^23
```

- Во втором способе определения, число после экспонента может быть и отрицательным:

```
1 double electron = 1.6e-19; // заряд электрона - 1.6 x 10^-19
```

- Числовые литералы могут иметь суффиксы, которые определяют их типы. Эти суффиксы не являются обязательными, так как компилятор понимает из контекста, константу какого типа данных вы хотите использовать.

Тип данных	Суффикс	Значение
int	u или U	unsigned int
int	l или L	long
int	ul, uL, Ul, UL, lu, lU, Lu или LU	unsigned long
int	ll или LL	long long
int	ull, uLL, Ull, ULL, llu, llU, LLu или LLU	unsigned long long
double	f или F	float
double	l или L	long double

- Суффиксы есть даже для целочисленных типов (но они почти не используются):

```
1 unsigned int nValue = 5u; // тип int unsigned
2 long nValue2 = 5L; // тип long
```

- По умолчанию литеральные константы типа с плавающей точкой являются типа double. Для конвертации литеральных констант в тип float можно использовать суффикс f или F:

```
1 float fValue = 5.0f; // тип float
2 double d = 6.02e23; // тип double (по умолчанию)
```

- Язык C++ также поддерживает литералы типов **string** и **char**:

```
1 char c = 'A'; // 'A' - это литерал типа char
2 std::cout << "Hello, world!"; // "Hello, world!" - это литерал строки C-style
3 std::cout << "Hello," " world!"; // C++ связывает последовательные литералы типа string
```

Математические операции и стандартные функции

1. Унарные операции по убыванию приоритетов

Знак операции	*Название*	*Краткое описание*
++ <переменная>	Инкремент (префикс)	Увеличение на 1
- - <переменная>	Декремент (префикс)	Уменьшение на 1
+ <переменная>	Унарный плюс	+x-2
- <переменная>	Унарный минус	Изменение знака на противоположный -x+3
! <переменная>	Логическое отрицание	Not (инверсия)
<переменная> ++	Инкремент (постфикс)	Увеличение на 1 после выполнения операции
<переменная> - -	Декремент (постфикс)	Уменьшение на 1 после выполнения операции
(<тип>) переменная	Преобразование типа	(int) x

Пусть x = 0

10 + ++x равно 11, x равен 1

a) x=0+1=1; b) 10+1=11

Пусть x = 0

10 + x++ равно 10, x равен 1

a) 10+0=10: b)x=0+1=1

2. Арифметические операции

Знак операции	*Название*	*Примечание*
+	Сложение	
-	Вычитание	
*	Умножение	
/	Деление	Если оба операнда целочисленные, то работает как целочисленное деление, в противном случае тип результата определяется правилами преобразования
%	Остаток от деления	Применяется только к целочисленным операндам

10 / 3 при целочисленном делении равно 3

10./3 равно 3.333…(в числителе вещественная константа)

10 % 3 равно 1 (это остаток от деления)

Если требуется, чтобы результат выражения i / 2 имел тип float необходимо записать
(float) i/ 2
(здесь i целого типа).

3. Математические функции

Обращение к функции: <имя функции> (<фактический параметр>)

В качестве фактического параметра может быть записано:
число, переменная, выражение.

Матем. функция	*Имя функции*	*Название*		*Заголовочный файл*
|x|	**abs(x)**	абсолютное значение **x**	**abs(-3.0)= 3.0** **abs(5.0)= 5.0**	<cmath>
	sqrt(x)	квадратный корень **x**	**sqrt(9.0)=3.0**	
ln x	**log(x)**	натуральный логарифм **x**	**log(1.0)=0.0**	
lg x	**log10(x)**	десятичный логарифм **x**	**log10(100)=2**	
e^x	**exp(x)**	**e** в степени **x**	**exp(0)=1**	
a^x	**pow(a,x)**	**a** в степени **x**	**pow(2,3)=8**	

4) Тригонометрические функции

Матем. функция	*Имя функции*	*Название* ***(аргумент задается в радианах)***	*Заголовочный файл*
sin X	**sin(x)**	синус **X**	<cmath>
cos X	**cos(x)**	косинус **X**	
tg X	**tan(x)**	тангенс **X**	
arcsin X	**asin(x)**	Возвращает угол, синус которого равен **X**	
arccos X	**acos(x)**	Возвращает угол, косинус которого равен **X**	
arctg X	**atan(x)**	Возвращает угол, тангенс которого равен **X**	

5) Дополнительные функции

Имя функции	*Название*	*Примеры*	*Заголовочный файл*
floor(x)	Округляет до целого в меньшую сторону	**floor(12.4)=12** **floor(-2.9)=-3**	<cmath>
ceil(x)	Округляет до целого в большую сторону	**ceil(2.3)=3.0** **ceil(-2.3)=-2.0**	
fmod(x, y)	Возвращает остаток от деления x/y	**fmod(4.4, 7.5) = 4.4** **fmod(7.5, 4.4) = 3.1**	
srand (x)	Используется для вычисления стартовой точки при генерации последовательности псевдослучайных чисел, где x типа unsigned int		<cstdlib>
rand ( )	Генерирует последовательность		

5. Арифметические выражения

Выражение – это синтаксическая единица языка, определяющая способ вычисления некоторого значения.

Правила записи арифметических выражений на алгоритмическом языке очень близки к обычным алгебраическим записям.

Последовательность (приоритет) выполнения арифметических операций следующая:

1) ( ) – операции в скобках;

2) вычисление стандартных функций;

3) * и / - умножение и деление;

4) % - вычисление остатка от целочисленного деления;

5) + и – сложение и вычитание.

Примеры арифметических выражений:

tg x	**tan(x)**
$\sin^2 x + \cos x^2$	**pow (sin(x), 2) + cos (pow (x, 2))**
$\frac{a \cdot e^{4.51 \cdot b}}{1 + c}$	**A*exp(4.51 * b)/(1+c)**
$ln\left\|a - \sqrt{b^2 - c^3}\right\|$	**log(abs(a-sqrt(b*b-pow(c, 3))))**

Поскольку в Си нет операции возведения в степень, то возведение в квадрат, а часто и в куб заменяют умножением: **arctg x^2** **atan(x*x)**

6. Операции присваивания

Знак операции	*Название*
=	Присваивание
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Остаток от деления с присваиванием

Операторы составных (кратких) присваиваний упрощают написание конструкций присваивания и обеспечивают более эффективный программный код.

Вместо `x = x + 10;` можно записать `x += 10;`

Операторная пара `+=` указывает компилятору, что переменной `x` следует присвоить значение `x` плюс `10`.

7. Операции отношения

Знак операции	*Название*
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
==	Равно
!=	Не равно

Слово «отношение» обозначает взаимоотношение двух величин, т.е. результат их сравнения: k < n.

Результатом операции отношения является логическое значение true или false.

Операнды, участвующие в операции отношения, могут принадлежать почти любому типу; необходимо только, чтобы их сравнение имело смысл.

Логическое отношение в общем случае имеет следующий вид:

<Выражение> <Знак отношения> < Выражение>

Примеры логических отношений:

a > b + g **f - k != 85**

a + c < d * sin(e) **s / t >= p**

p + q == r * 7 **w <= x + y – z**

Логические отношения используются в условных операторах.

8. Логические операции

Знак операции	*Название*
&&	И (AND)
||	ИЛИ (OR)
!	НЕ - Логическое отрицание

Результатом логической операции является логическое значение true или false.

Поскольку ненулевое значение истинно, а нулевое ложно, это означает, что логические операторы допустимо использовать с любым выражением, дающим нулевой или ненулевой результат.

Компилятор C++ автоматически преобразует true в 1, а false в 0 и обратно.

На Си всё, что не ноль, есть истина:

bool a=-2.e-3;→a=true

int*p=NULL; if (p) →условие ложное

Таблица истинности для логических операций

p	q	p && q	p || q	! p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Ключевые термины

- **Базовые типы** – это типы данных, которые предопределены *стандартом языка*, указываются зарезервированными ключевыми словами, характеризуются одним значением и внутренним представлением.
- **Вещественный тип** – это *базовый тип данных*, который применяется для хранения дробных чисел в формате с плавающей точкой.
- **Логический (булевый) тип** – это *базовый тип данных*, который применяется для хранения значений двузначной логики.
- **Неявное приведение типа** – это преобразование значения переменной к новому типу, которое происходит автоматически, по правилам, заложенным в языке программирования.
- **Перечисляемый тип** – это *производный тип данных*, он определяется как набор идентификаторов, являющихся именованными целыми константами, которым приписаны уникальные обозначения
- **Преобразование типов** – это приведение значения переменной одного типа в *значение* другого типа.
- **Производные типы** – это типы данных, которые задаются пользователем.
- **Символьный тип** – это *базовый тип данных*, который применяется для хранения символов или *управляющих последовательностей* в виде кода.
- **Тип данных** – это множество допустимых значений, которые может принимать тот или иной *объект*, а также множество допустимых операций, которые применимы к нему.
- **Типы класса** – это типы данных, экземплярами которых являются объекты.
- **Целочисленный тип** – это *базовый тип данных*, который применяется для хранения целых чисел.
- **Явное приведение типа** – это преобразование значения переменной к новому типу, при котором указывается *тип переменной*, к которому необходимо привести исходную переменную.

Краткие итоги

1. C++– компилятор, машиноориентированный, обладающий высокой производительностью программ, имеет сложный, но компактный синтаксис.
2. Для организации хранения данных и корректного выполнения операций над ними в языках программирования определены типы данных.
3. Типы характеризуются схожим внутренним представлением данных в памяти компьютера; объемом памяти, выделяемым под данные; множеством (диапазоном) принимаемых значений; допустимыми операциями и функциями.
4. В языке C++ типы классифицируются на базовые, производные и классы.
5. Для базовых типов определены их подмножества и расширения, что обеспечивает повышение точности расчетов или экономный расход памяти.
6. Над типами данных определена операция преобразования типов. Ее следует применять с осторожностью при переходе к типу, у которого меньше по модулю границы диапазонов.

Контрольные вопросы

1. Чем отличаются и в чём похожи языки Phyton и C++?
2. Почему в языке C++ определена строгая типизация данных, используемых в программе?
3. Как определяются границы диапазона базового типа в зависимости от выделяемой под этот тип памяти?
4. Почему наблюдается *асимметрия* значений границ диапазонов целочисленных типов?
5. Чему будет равно значение операции *инкремента* для максимального числа в целочисленном типе? А каков результат *декремента* для минимального значения в таком же типе?
6. Каким образом представлено число ноль в вещественных типах?
7. При преобразовании целого со знаком к целому без знака всегда ли будет получено исходное числовое значение? Ответ обоснуйте.

Проверьте себя: какое значение примут переменные z,y в приведённом фрагменте программы:

```
int a = 2, b = 3,z;
float c = 6.5, d = 12.4;
bool x = 1, y;
z = a / b + a * c - d;
y = x + b / a + c * d;
```

Операции ввода-вывода на консоль

Ввод-вывод данных

В C++ ввод-вывод данных осуществляется с использованием так называемых потоков.

Поток – это абстрактное понятие, связанное с передачей данных от источника к приемнику.

В программе объект- входной поток cin – используется для ввода данных с клавиатуры (console input – консольный ввод),

а объект- выходной поток cout – используется для вывода на экран (console output – консольный вывод).

cout(**c**onsole **out**put)

Для **ввода** из потока cin используется оператор >> ,

который задает направление потока данных: от клавиатуры cin в переменную.

cin(**c**onsole **in**put)

Для **вывода** в поток cout используется оператор << ,

который задает направление потока данных: от переменной или строки на экран cout .

5.1. Ввод данных

осуществляется из потока cin
оператором чтения из потока >>

Пример:

```
int n1 = 0, n2 = 0;  // описание и инициализация переменных
double f = 0.0; // описание и инициализация переменной
cin >> n1 >> f >> n2;     // ввод данных
```

Последняя строка читает вводимые с клавиатуры три числа, разделенный пробелами (например, **5 7.23 89**), и присваивает эти числа:

первое число **5** - переменной n1,

второе число **7.23** - переменной f,

третье число **89** - переменной n2.

5.2. Вывод данных

осуществляется в поток cout
оператором записи в поток <<

Пример:

```
int n1 = 5, n2 = -89;     // описание и инициализация переменных
double f = 7.23;     // описание и инициализация переменной
cout << " n1 = " << n1 << "   f = " << f << "   n2 = " << n2 << endl;  // вывод данных
```

Последняя строка последовательно выводит на экран:

"n1 = " - строка (пояснительный текст),
n1 - значение переменной ,
" f = " - строка (два пробела и текст),
f - значение переменной ,
" n2 = " - строка (два пробела и текст),
n2 - значение переменной ,
endl - перевод курсора на новую строку.

Результаты вывода на экран:

Управление процессом ввода/вывода

Управляющие символы (или как их ещё называют — escape-последовательность)

Символ	Описание
\r	возврат каретки в начало строки
\n	новая строка
\t	горизонтальная табуляция
\v	вертикальная табуляция
\»	двойные кавычки
\’	апостроф
\\	обратный слеш
\0	нулевой символ
\?	знак вопроса
\a	сигнал бипера (спикера) компьютера

Если необходимо вывести какое-то сообщение, то управляющие символы можно записывать сразу в сообщении, в любом его месте.

Примеры

```
cout<< "Vector: "<<'\t'<<a<<'\t'<<b<<'\t'<<c<<endl;
```

```
 Input 3 coordinates of the vector: 1 -2 0
Vector:         1       -2      0
 Input 3 coordinates of the vector: 2 7 -4
Vector:         2       7       -4
```

Пример задач Лабораторной работы №1

1.1 Вычислить функцию: $x = 4y^2/(\sqrt{(4z)}-2t^3)$ при $t = 1$; $z = 3$; $y = \sin t$.

1.2 Вычислить сопротивление проводника при заданном значении напряжения и тока по закону Ома.

Решение задачи 1.1