

Введение в C++

Лекция 1

Искусство программирования

- ▶ **Законы Лопатина о программировании**
 - ▶ Если ты наконец-то выучил язык программирования, то он никому уже не нужен.
 - ▶ Если ты думаешь, что знаешь язык программирования, то ошибаешься - твои знания безнадежно устарели.
 - ▶ Если язык программирования необычайно полезен и популярен в этой стране, то за ее пределами он никому не нужен.
- ▶ **Второй закон Вейнберга**
 - ▶ Если бы строители строили здания так же, как программисты пишут программы, первый залетевший дятел разрушил бы цивилизацию.

Формула Николауса Вирта

алгоритмы

+

структуры данных

=

программы

Основные парадигмы программирования:

- ▶ процедурное
- ▶ модульное
- ▶ объектно-ориентированное
- ▶ обобщенное

Процедурное программирование

Определите, какие **процедуры** вам нужны, используйте
лучшие из известных вам **алгоритмов**

Модульное программирование

Определите, какие **модули** нужны, **поделите программу**
так, чтобы **данные** были **скрыты** в этих модулях

Объектно-ориентированное программирование

Определите, какой **класс** вам необходим, предоставьте
полный набор операций для каждого класса, общность
классов выразите явно с помощью **наследования**

Обобщенное программирование

Заключается в таком **описании данных и алгоритмов**,
которое можно применять к **различным типам данных**,
не меняя само это описание.

Вместо описания отдельного типа в обобщённом
программировании применяется **описание семейства
типов**, имеющих общий интерфейс и семантическое
поведение

Краткая история языка Си

- ▶ 1969 - Кен Томпсон в лаборатории Bell разработал ОС Unix для PDP-7 и язык B
- ▶ 1972 - Деннис Ритчи создает язык Си
- ▶ 1973 - исходный код ОС Unix переписывается с ассемблера на Си
- ▶ 1978 - первое издание книги “The C Programming language”
- ▶ 1982 - ANSI утвердил комитет X3J11 для разработки стандарта языка Си
- ▶ 1989 - был утвержден стандарт ISO/IEC 9899-1990
- ▶ 1999 - стандарт ISO/IEC 9899 (C99)

Достоинства

- ▶ Гибкость
- ▶ Компактность
- ▶ Переносимость
- ▶ Эффективность
- ▶ Мощность
- ▶ Универсальность

Недостатки

- ▶ Трудность в освоении
- ▶ Отсутствие автоматического управления памятью
- ▶ Слабая типизация

Происхождение C++

- ▶ Разработан Бъярном Страуструпом (Bell Laboratories)
- ▶ Название «C++» появилось в 1983 году
- ▶ C++ - это расширение C
- ▶ 2011 год - стандарт ISO/IEC 14882:2011

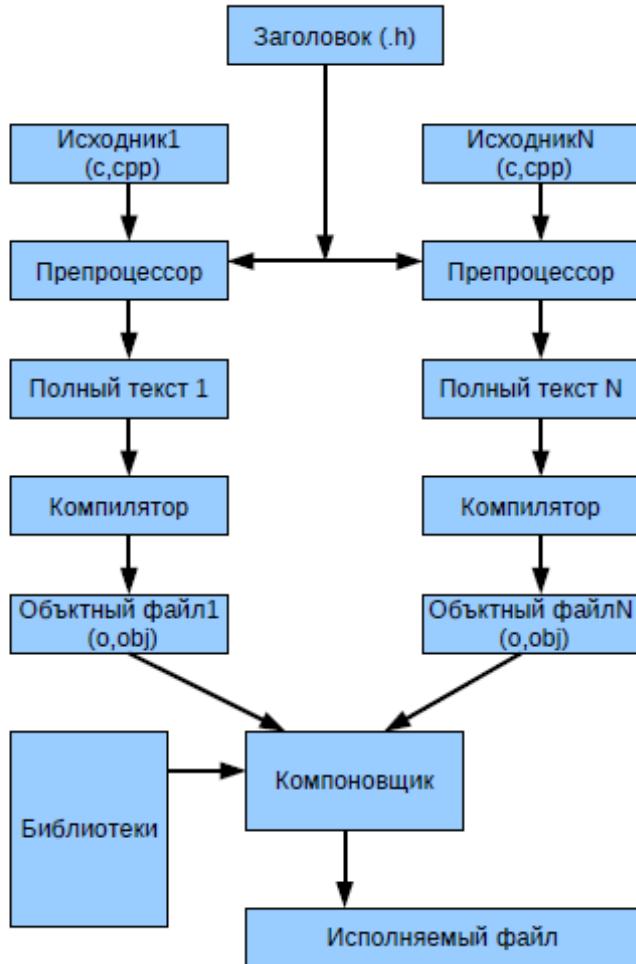
Что такое C++

- ▶ Расширение Си
- ▶ Поддерживает объектно-ориентированное программирование
- ▶ Поддерживает абстракцию данных
- ▶ Поддерживает обобщенное программирование

Среда разработки

- ▶ Текстовый редактор
- ▶ Препроцессор
- ▶ Компилятор
- ▶ Компоновщик
- ▶ Отладчик

Схема построения программы



build-time и run-time

- ▶ **build-time** - **до** выполнения программы. Выполняются директивы препроцессора, компиляция программы, сборка исполняемого файла.
- ▶ **run-time** - **во время** выполнения программы. Выделяется и инициализируется память, выполняется код программы и т.д.

Первая программа

```
#include <iostream>
```

Включение библиотеки
ввода/вывода

```
int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Основная функция программы

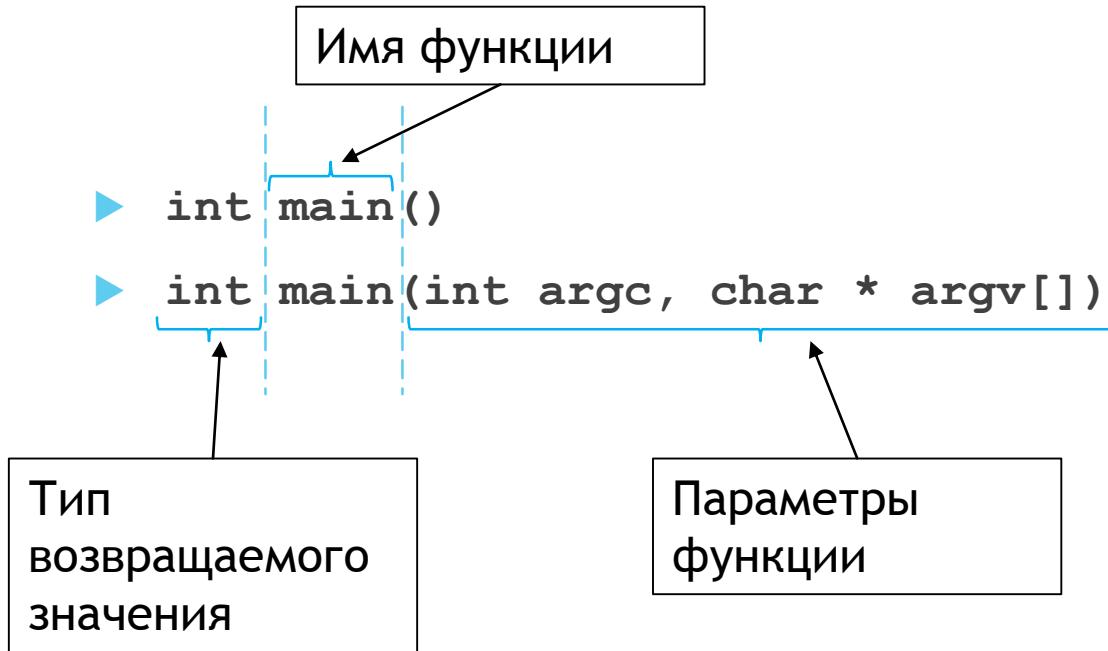
Hello, world!

Результат работы программы

main - главная функция программы

- ▶ Любая программа на C++ содержит функцию **main**
- ▶ Функция **main** может быть в любом месте программы
- ▶ Выполнение любой программы начинается с функции **main**
- ▶ Функция **main** должна быть только одна

Варианты объявления main



Тело функции

- ▶ Заключается в фигурные скобки
- ▶ Содержит инструкции для выполнения - операторы

Код для вывода
информации в консоль

```
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

Выводит строчку
«Hello, world!»

Правила оформления кода

```
//обязательны комментарии
```

```
#include <iostream>
```

Одна строка - одна инструкция

```
int main()
```

```
{
```

Фигурные скобки -отдельная строка

```
    std::cout << "Hello, world!" << std::endl;
```

```
    return 0;
```

```
}
```

Отступы слева

Правила оформления программы

```
AsciiFile* file;  
int nPoints;  
float x, y;
```

Выравнивание улучшает читабельность кода

```
value = (potential * oilDensity) / constant1 +  
        (depth * waterDensity) / constant2 +  
        (zCoordinateValue * gasDensity) / constant3;
```

```
minPosition = computeDistance(min, x, y, z);  
averagePosition = computeDistance(average, x, y, z);
```

Виды комментариев

- ▶ Однострочные //комментарий
- ▶ Многострочные /*комментарий на 1 строке
комментарий на 2 строке*/

Однострочные комментарии предпочтительнее

Оформление комментариев

- ▶ **Заглавный комментарий.** Размещайте заглавный комментарий, который описывает назначение файла, вверху каждого файла.
- ▶ **Заголовок функции / конструктора.** Разместите заголовочный комментарий на каждом конструкторе и функции вашего файла. Заголовок должен описывать поведение и / или цель функции.
- ▶ **Параметры / возврат.** Если ваша функцию принимает параметры, то кратко опишите их цель и смысл. Если ваша функция возвращает значение — кратко опишите, что она возвращает.
- ▶ **Исключения.** Если ваша функция намеренно выдает какие-то исключения для определенных ошибочных случаев, то это требует упоминания.
- ▶ **Комментарии на одной строке.** Если внутри функции имеется секция кода, которая длинна, сложна или непонятна, то кратко опишите её назначение.

Комментарии

- ▶ Сложный код, написанный с использованием хитрых ходов, следует не комментировать, а переписывать!
- ▶ Следует делать как можно меньше комментариев, делая код **самодокументируемым** путём выбора правильных имён и создания ясной логической структуры.

Переменные

Переменная - это объект данных, который явным образом определен и именован в программе.

Переменные характеризуются с помощью следующих атрибутов:

- ▶ имя;
- ▶ тип;
- ▶ значение;
- ▶ адрес;
- ▶ время жизни;
- ▶ область видимости.

Имя переменной

- ▶ Это **идентификатор**, позволяющий обращаться к значению переменной.
- ▶ Идентификатор – это **последовательность символов**, используемая для обозначения одного из следующих элементов:
 - ▶ Имени объекта или переменной
 - ▶ Имени класса, структуры или объединения
 - ▶ Имени перечисленного типа
 - ▶ Члена класса, структуры, объединения или перечисления
 - ▶ Функции или функции члена класса
 - ▶ Имени определения типа (typedef)
 - ▶ Имени метки
 - ▶ Имени макроса
 - ▶ Параметра макроса

Правила именования идентификаторов

- ▶ Идентификатор содержит только латинские буквы, арабские цифры и знак подчеркивания
- ▶ Идентификатор не может начинаться с цифры
- ▶ Регистр букв имеет значение
- ▶ Имя идентификатора должно быть уникальным в его области видимости

Соглашения по именованию переменных

- ▶ Имена переменных записываются в смешанном регистре, начиная с нижнего (**camelStyle**)
- ▶ Имена следует записывать по-английски
- ▶ Переменные, имеющие большую область видимости, следует называть длинными именами, имеющие небольшую область видимости – короткими
- ▶ Префикс **n** следует использовать для представления числа объектов
- ▶ Суффикс **No** следует использовать для обозначения номера сущности
- ▶ Префикс **is** следует использовать только для логических переменных и методов (допускается замена на **can**, **has**, **should**)

Тип данных

Тип данных определяет:

- ▶ внутреннее представление данных в памяти компьютера;
- ▶ объем памяти, выделяемый под данные;
- ▶ множество (диапазон) значений, которые могут принимать величины этого типа;
- ▶ операции и функции, которые можно применять к данным этого типа.

Типы данных

вещественные

целочисленные

	Тип данных	Байт	Диапазон
вещественные	long double	?	3.4e-4932..3.4e+4932
	double	8	1.7e-308..1.7e+308
	float	4	3.4e-38..3.4e+38
целочисленные	unsigned long long	8	0..18 446 744 073 709 551 615
	long long int	8	-9 223 372 036 854 775 808 .. 9 223 372 036 854 775 807
	unsigned long int	4	0..4 294 967 295
	long int	4	-2 147 483 648 .. 2 147 483 647
	int	?	?
	unsigned short int	2	0..65535
	short int	2	-32 768 .. 32 767
	unsigned char	1	0..255
	char	1	-128 .. 127
	bool	1	0 1

Целочисленные типы

- ▶ Описание переменной, имеющей тип `int`, сообщает компилятору, что он должен связать с идентификатором (именем) переменной количество памяти, достаточное для хранения целого числа во время выполнения программы.
- ▶ Границы диапазона целых чисел, которые можно хранить в переменных типа `int`, зависят от конкретного компьютера, компилятора и операционной системы (от реализации).
- ▶ Для внутреннего представления знаковых целых чисел характерно **определение знака по старшему биту** (0 - для положительных, 1 - для отрицательных). Поэтому число 0 во внутреннем представлении относится к положительным значениям. Следовательно, наблюдается асимметрия границ целых промежутков.
- ▶ В **целочисленных типах для всех значений определены следующий и предыдущий элементы**. Для максимального следующим значением будет являться минимальное в этом же типе, предыдущее для минимального определяется как максимальное значение. То есть **целочисленный диапазон условно можно представить сомкнутым в кольцо**.

Вещественные типы

- ▶ Вещественные числа хранятся в формате с плавающей точкой (экспоненциальной форме):

мантиssa E/e порядок

Например:

`5.235e+02 (5.235 * 102 = 523.5) ;`

`-3.4E-03 (-3.4 * 10-3 = - 0.0034)`

- ▶ Величина с модификатором типа `float` занимает 4 байта. Из них 1 бит отводится для знака, 8 бит для экспоненты и 23 бита для мантийсы. Старший бит мантийсы всегда равен 1, поэтому он не заполняется.
- ▶ Величина типа `double` занимает 8 байтов в памяти. Ее формат аналогичен формату `float`. Биты памяти распределяются следующим образом: 1 бит для знака, 11 бит для экспоненты и 52 бита для мантийсы. Старший бит мантийсы также пропускается.

Символьный тип

- ▶ Переменная типа `char` рассчитана на хранение только одного символа (например, буквы цифры или пробела). В памяти компьютера **символы хранятся в виде целых чисел**. Соответствие между символами и их кодами определяется **таблицей кодировки**, которая зависит от компьютера и операционной системы. Почти во всех таблицах кодировки есть прописные и строчные буквы латинского алфавита, цифры 0, ..., 9, и некоторые специальные символы. Самой распространенной таблицей кодировки является таблица символов **ASCII** (American Standard Code for Information Interchange - Американский стандартный код для обмена информацией).
- ▶ Так как в памяти компьютера символы хранятся в виде целых чисел, то тип `char` на самом деле **является подмножеством типа int**

Специальные (непечатные) символы

Описание символа	Специальная последовательность
Символ новой строки	\n
Горизонтальная табуляция	\t
Вертикальная табуляция	\v
Возврат на шаг	\b
Возврат каретки	\r
Обратная косая	\\"
Одиночная кавычка (апостроф)	\'
Двойные кавычки	\”
Звонок	\a

Строчный тип

- ▶ В C++ нет встроенного строчного типа
- ▶ Для работы со строками разработан класс **string** стандартной библиотеки. Необходимо подключить заголовочный файл **<string>**
- ▶ Класс **string** находится в стандартном пространстве имен **std**
- ▶ Упрощает работу со строками

Логический тип

- ▶ В языке C++ используется **двоичная логика** (истина, ложь). Лжи соответствует нулевое значение, истине - единица. величины данного типа могут также принимать значения `true` и `false`.
- ▶ **Внутренняя форма** представления значения `false` соответствует 0, **любое другое значение** интерпретируется как `true`. В некоторых компиляторах языка C++ нет данного типа, в этом случае используют тип `int`, который при истинных значениях выдает 1, а при ложных - 0.
- ▶ Под данные логического типа отводится 1 байт.

Пример объявления переменных

```
double total = 0.0;  
  
double speed = 3.0e8;  
  
int nLines = 15, columnNo = 25;  
  
bool isEmpty = false;
```

Хорошо!!!
Инициализация
в момент
объявления

```
double total;  
  
double speed;  
  
int nLines, columnNo;  
  
bool isEmpty;
```

Плохо!!!
Объявление без
инициализации

Пример объявления переменных

```
char c='c'; //хорошо  
char a, b; //плохо
```

```
char r[]={ 'A', 'B', 'C', 'D', 'E', 'F', '\0' };  
char s[] = "ABCDEF";      Объявление строк в формате Си
```

Объявления `r` и `s` одинаковы,
но `s` - короче

Пример работы со строками

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

Включение
заголовочного
файла

Объявление строковой переменной

```
int main()
{
    std::string myString = "Hello, world!";
    std::cout << myString << std::endl;
    std::cout << myString.c_str() << std::endl;
    return 0;
}
```

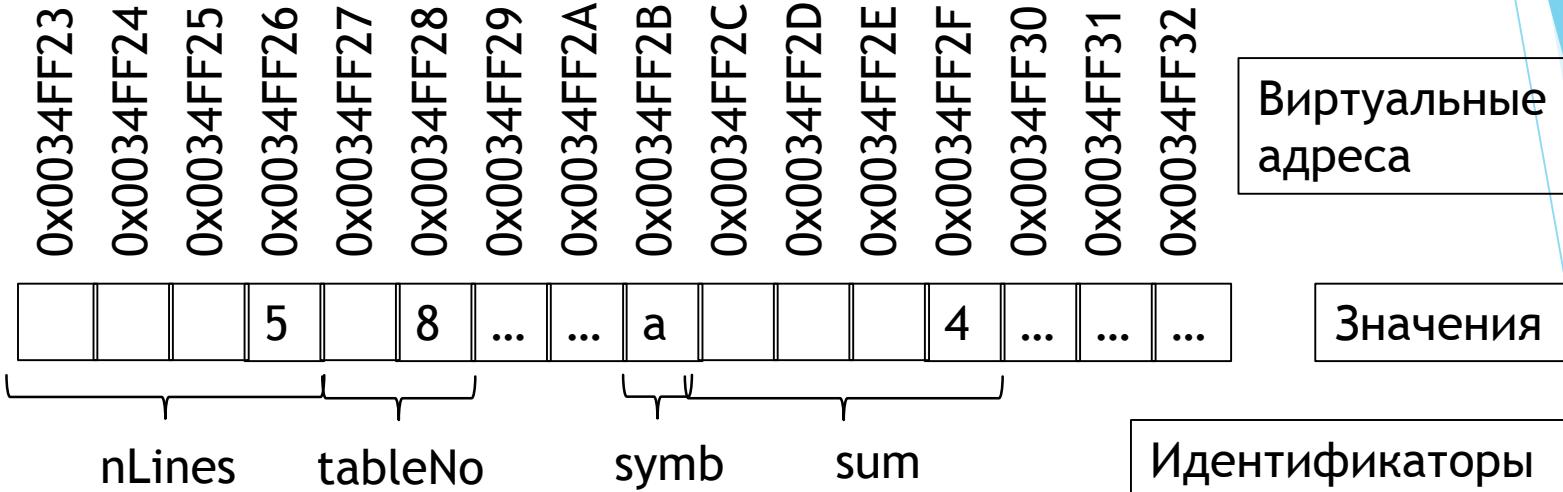
Вывод строки в
новом
компиляторе

Вывод строки в
старом
компиляторе

Модель памяти компьютера

- ▶ Память дискретна, состоит из отдельных байтов.
- ▶ Каждый байт пронумерован, номер байта называется адресом.
- ▶ Минимально доступный программисту участок памяти - один байт.
- ▶ Переменная может занимать больше одного байта.
- ▶ Обычная переменная не может занимать меньше одного байта.
- ▶ Все байты, занимаемые переменной, идут подряд.
- ▶ Адрес переменной - адрес старшего байта (или младшего).

Адрес переменной



```
int nLines      = 5;      //0x0034FF23
short tableNo   = 8;      //0x0034FF27
char symb       = 'a';    //0x0034FF2B
long sum        = 8;      //0x0034FF2C
```

Адрес переменной

Computer		Programmers		
Address	Content	Name	Type	Value
90000000	00			
90000001	00			
90000002	00	sum	int (4 bytes)	000000FF (255 ₁₀)
90000003	FF			
90000004	FF	age	short (2 bytes)	FFFF (-1 ₁₀)
90000005	FF			
90000006	1F			
90000007	FF			
90000008	FF	averge	double (8 bytes)	1FFFFFFFFFFFFFFF (4.45015E-308 ₁₀)
90000009	FF			
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			
9000000E	90	ptrSum	int* (4 bytes)	90000000
9000000F	00			
90000010	00			
90000011	00			

Note: All numbers in hexadecimal

Время жизни и область видимости

- ▶ **Время жизни** - это время, в течение которого переменная связана с определенной областью памяти. Определяется классом памяти. Может быть:
 - ▶ локальным;
 - ▶ глобальным.
- ▶ **Область видимости** - это блок программы, из которого можно обратиться к переменной. Может быть:
 - ▶ блок;
 - ▶ функция;
 - ▶ файл;
 - ▶ класс;
 - ▶ пространство имен;
 - ▶ вся программа.

Константные переменные

- ▶ Константные переменные - это переменные, объявленные с модификатором `const` и не изменяющие своего значения
- ▶ Именованные константы (включая значения перечислений) должны быть записаны в верхнем регистре с нижним подчёркиванием в качестве разделителя
- ▶ Только константы могут быть глобальными переменными
- ▶ Все выражения с константами вычисляются в момент build-time

Примеры объявления констант

```
const int MY_FIRST_CONST = 17;  
  
const double PI = 3.1415926535;  
  
const float E = 2.71828;  
  
const char* GREETING = "Hello, world!";
```

Хорошо!

```
MY_FIRST_CONST = 29;
```

Плохо!
Попытка изменить
значение константы

```
const int MY_FIRST_CONST;
```

Плохо!
Объявление константы
без инициализации

Перечисления

- ▶ **Перечисление** - набор именованных констант.
- ▶ Именованные константы списка имеют тип `int`.
- ▶ Количество памяти, выделяемой под переменную перечисления, - это количество памяти, необходимой для размещения значения типа `int`.
- ▶ Константы в перечислениях могут иметь префикс – общее имя типа. Это даёт дополнительную информацию о том, где находится объявление, какие константы описаны в одном перечислении, а также какую концепцию являются собой константы

```
enum Color {  
    COLOR_RED,  
    COLOR_GREEN,  
    COLOR_BLUE  
};
```

Пример объявления перечислений

```
enum DayTime { MORNING, DAY, EVENING, NIGHT };

DayTime current;

current = DAY;

if (current != NIGHT)

{

    // выполнить работу

}
```

Значения элементов перечисления

//каждому атрибуту задается число

```
enum { MORNING = 4, DAY = 3, EVENING = 2,  
NIGHT = 1 };
```

// последовательные числа начиная с 1

```
enum { MORNING = 1, DAY, EVENING, NIGHT };
```

// используются числа 0, 2, 3 и 4

```
enum { MORNING, DAY = 2, EVENING, NIGHT };
```

// значения могут повторяться

```
enum { MORNING, DAY = 2, EVENING, NIGHT = 2 };
```

Пример работы с перечислением

```
int main()
{
    enum season{ SPRING = 1, SUMMER, AUTUMN, WINTER };

    season currentSeason;

    currentSeason = season(2);

    std::cout << currentSeason << std::endl; //2

    currentSeason = WINTER;

    std::cout << currentSeason << std::endl; //4

    if( currentSeason == WINTER )
        std::cout << "Winter!" << std::endl; //Winter

    return 0;
}
```

Получение данных от пользователя

```
int a = 0, b = 0;  
float f = 0.0;  
double d = 0.0;  
std::cin >> a >> d >> b >> f;
```

Ввод числовых
данных

```
const int SIZE = 256;  
char str[SIZE] = {0};  
std::getline(str, SIZE);
```

Ввод символьного
массива

```
std::string myString;  
std::getline(std::cin, myString);
```

Ввод строки

Форматированный вывод

```
int nColumn = 15;  
  
const double PI = 3.1415926535;  
  
std::cout.setf(std::ios::fixed); //формат вывода  
//количество знаков после запятой  
  
std::cout.precision(2);  
  
std::cout << nColumn << std::endl;  
  
std::cout << PI << std::endl;
```

Вывод:

```
15  
3.14  
—
```

Конец