**1. Введение**

Перед C++ был C

C (произносится как «Си»)

Был разработан в 1972 году Денисом Ритчи [Dennis Ritchie]

в Bell Telephone Laboratories как системный язык программирования, т. е. язык для написания операционных систем. Основной задачей Ритчи было создание легко компилируемого минималистического языка, который предоставлял бы эффективный доступ к памяти, относительно быстро выполнялся, и на котором можно было бы писать эффективный код.

Таким образом, при разработке высокоуровневого языка был создан язык Си, который во многом относился к языкам низкого уровня, оставаясь при этом независимым от платформ, для которых мог быть написан код.

C++

(произносится как «Си плюс плюс»)

Был разработан Бьёрном Страуструпом [Bjarne Stroustrup] в Bell Labs в качестве дополнения к Cи в 1979 г. Он добавил множество новых фич в язык Си. Его популярность была вызвана объектно-ориентированностью языка. Об объектно-ориентированном программировании (ООП) и его отличиях от традиционных методов программирования мы поговорим несколько позже.

Философия языков семейства Си.

Смысл философии языков С и C++ можно определить выражением «доверять программисту». Например, компилятор не будет вам мешать сделать что-то новое, что имеет смысл, но также не будет мешать вам сделать что-то такое, что может привести к сбою. Это одна из главных причин, почему так важно знать то, что вы не должны делать, как и то, что вы должны делать, создавая программы на языках С/С++.

**2. Hello World**

Сейчас мы по традиции любого языка программирования напишем программу «привет мир». Для этого вам понадобится любой компилятор, можете найти онлайн компилятор в интернете или скачать Visual Studio для полноценной разработки на языку С++.

Для Visual Studio

Откройте меню «Файл → Создать → Проект». Перейдите на вкладку «Общие» и выберите «Пустой проект». Придумайте проекту любое название, например «lesson1» и нажмите «OK».

В окне обозревателя решений (обычно он находится в левом верхнем углу) щелкните правой кнопкой на папке «файлы исходного кода». В диалоговом окне выберите пункт меню «Добавить → Создать элемент». Введите название для нового файла — main.cpp и нажмите кнопку «Добавить».

Код первой программы

Наберите следующий код:

**#include <iostream>**

**#include <cstdlib> // для system**

**int** **main**()

{

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

std::system("pause");

*// Только для тех, у кого MS Visual Studio*

**return** 0;

}

Описание синтаксиса

Директива #include используется для подключения других файлов в код. Строка #include <iostream>, будет заменена содержимым файла «iostream.h», который находится в стандартной библиотеке языка и отвечает за ввод и вывод данных на экран.

#include <cstdlib> подключает стандартную библиотеку языка С. Это подключение необходимо для работы функции system.

Все то, что находится внутри фигурных скобок функции int main() {} будет автоматически выполняться после запуска программы.

Строка cout << "Hello, world!" << endl; говорит программе выводить сообщение с текстом **«Hello, world»** на экран.

Оператор cout предназначен для вывода текста на экран командной строки. После него ставятся две угловые кавычки (<<). Далее идет текст, который должен выводиться. Он помещается в двойные кавычки. Оператор endl переводит строку на уровень ниже.

Если в процессе выполнения произойдет какой-либо сбой, то будет сгенерирован код ошибки, отличный от нуля. Если же работа программы завершилась без сбоев, то код ошибки будет равен нулю. Команда return 0 необходима для того, чтобы передать операционной системе сообщение об удачном завершении программы.

!В конце каждой команды ставится **точка с запятой!**

3. Переменные

На ум сразу приходят слова «переменять» - «поменять»

Объявление переменной в C++ происходит таким образом: сначала указывается тип данных для этой переменной а затем название этой переменной.

**int** a; *// объявление переменной a целого типа.*

**float** b; *// объявление переменной b типа данных с плавающей запятой.*

**double** c = 14.2; *// инициализация переменной типа double.* **char** d = 's'; *// инициализация переменной типа char.*

**bool** k = true; *// инициализация логической переменной k.*

Программируя на языке C++, мы создаем, обрабатываем и уничтожаем объекты.

**Объект** — это часть памяти, которая может хранить значение. В качестве аналогии мы можем использовать почтовый ящик, куда мы помещаем информацию и откуда её извлекаем. Все компьютеры имеют **оперативную память**, которую используют программы.

При создании объекта, часть оперативной памяти выделяется для этого объекта. Большинство объектов, с которыми мы будем работать в языке C++, являются переменными.

**4.** **Инициализация vs. Присваивание**

В языке C++ есть две похожие концепции, которые новички часто путают: присваивание и инициализация.

После объявления переменной ей можно **присвоить** значение с помощью оператора присваивания (знак равенства =):

|  |  |
| --- | --- |
| 1  2 | int a; // это объявление переменной  a = 8; // а это присваивание переменной a значения 8 |

В языке C++ вы можете объявить переменную и присвоить ей значение одновременно. Это называется **инициализацией**(или ***«определением»***).

int a = 8; // инициализируем переменную a значением 8

Переменная может быть инициализирована только после операции объявления.

Хотя эти два понятия близки по своей сути и часто могут использоваться для достижения одних и тех же целей, все же в некоторых случаях следует использовать инициализацию, вместо присваивания, а в некоторых — присваивание вместо инициализации.

**Правило: если у вас изначально имеется значение для переменной, то используйте инициализацию, вместо присваивания.**

В отличие от других языков программирования, языки Cи и C++ не инициализируют переменные определенными значениями (например, нулем) по умолчанию. Поэтому, при создании переменной, ей присваивается ячейка в памяти, в которой уже может находиться какой-нибудь мусор! Переменная без значения (со стороны программиста или пользователя) называется **неинициализированной переменной**.

Использование неинициализированных переменных может привести к ошибкам, например:

|  |  |
| --- | --- |
|  | #include <iostream>    int main()  {      // Объявляем целочисленную переменную a      int a;        // Выводим значение переменной a на экран (a - это неинициализированная переменная)      std::cout << a;        return 0;  } |

**5.** **Типы данных. Целочисленный тип данных**

Начнем с того, что может иметь тип данных

Переменная типа void — это самый простой тип данных, который означает «отсутствие любого типа данных».

Следовательно, переменные не могут быть типа void:

Тип void, как правило, используется в 2-х случаях:

Использование №1: Указать, что функция не возвращает значение:

Использование №2: указать, что функция не имеет никаких параметров (перешло из языка Cи):

Указание типа void как «никаких параметров» является пережитком, сохранившимся еще со времен языка Cи.

Целочисленный тип данных — это тип, переменные которого могут содержать только целые числа

(без дробной части, например: -2, -1, 0, 1, 2).

В языке C++ есть 5 основных целочисленных типов, доступных для использования:

Размер Диапозон:

short int 2 -32 768    /    32 767

unsigned short int 2 0  /  65 535

int 4 -2 147 483 648   /   2 147 483 647

unsigned int 4 0     /     4 294 967 295

long int 4 -2 147 483 648    /    2 147 483 647

unsigned long int 4 0     /     4 294 967 295

Диапазон — это значения от и до, которые может хранить определенный тип данных. Диапазон целочисленной

переменной определяется двумя факторами:

её размером (измеряется в битах)

и её знаком (может быть signed или unsigned).

Целочисленный тип signed(со знаком) означает, что переменная может содержать как положительные, так и отрицательные числа.

По умолчанию, ключевое слово signed пишется перед типом данных.

Целочисленный тип unsigned(без знака) может содержать только положительные числа.

**6.** **Действительные числа.**

или типы данных с плавающей точкой!

В С++ существуют два типа данных с плавающей точкой: float и double.

Размер Диапозон:

float 4 -2 147 483 648.0 / 2 147 483 647.0

double 8

-9 223 372 036 854 775 808.0 / 9 223 372 036 854 775 807.0

Типы данных с плавающей точкой предназначены для хранения чисел с плавающей точкой. Типы данных float и double могут хранить как положительные, так и отрицательные числа с плавающей точкой.

У типа данных float размер занимаемой памяти в два раза меньше, чем у типа данных double, а значит и диапазон принимаемых значений тоже меньше. Если тип данных float объявить с приставкой long, то диапазон принимаемых значений станет равен диапазону принимаемых значений типа данных double.

!В основном, типы данных с плавающей точкой нужны для решения задач с высокой точностью вычислений, например, операции с деньгами.

**7.** **Логический тип данных. Bool**

В реальной жизни перед нами очень часто возникают вопросы, на которые можно ответить однозначно:

«Да» или «Нет».

Яблоко является фруктом?

Да!

Вам нравится спаржа?

Нет!

Рассмотрим утверждение:

«Яблоко — это фрукт».

Является ли это правдой? Да! Яблоко действительно является фруктом.

Или как насчет:

«Я люблю спаржу».

Нет!

Подобные стейтменты, которые имеют только два возможных исхода – да-правда или нет-ложь,

настолько распространены, что многие языки программирования добавили специальный тип для работы с ними — логический тип данных.

В языке С++ это логический тип данных bool(от англ. «boolean»).

Переменные логического типа — это переменные,

диапазон которых состоит только из двух возможный значений: true(1) и false(0).

Для объявления логической переменной используется ключевое слово bool:

bool b;

Инициализировать логическую переменную или выполнить операцию присваивания можно с помощью ключевых слов true или false:

На самом деле, логические значения не сохраняются как true или false. Они обрабатываются в виде целых чисел:

вместо true — единица, вместо false — ноль.

**8.** **Символьный тип данных. Char**

Переменная типа char занимает 1 байт.

Однако вместо конвертации значения типа char в целое число, оно интерпретируется как ASCII-символ. ASCII(сокр.от «American Standard Code for Information Interchange») — это американский стандартный код для обмена информацией, который определяет способ представления символов английского языка(+несколько других) в виде чисел от 0 до 127.

Например: код буквы 'а' — 97,

код буквы 'b' — 98.

Символы всегда помещаются в одинарные кавычки

Следующие два стейтмента выполняют одно и то же (присваивают переменным типа char целое число 97):

char ch1(97); // инициализация переменной типа char целым числом 97

char ch2('a'); // инициализация переменной типа char символом 'a' (97)

Оператор static\_cast

Если вы хотите вывести символы в виде цифр, а не в виде букв, то вам нужно сообщить cout выводить переменные типа char в виде целочисленных значений.

!!Не очень хороший способ это сделать — присвоить переменной типа int переменную типа char и вывести её!!

Следующая программа просит пользователя ввести символ. Затем она выводит этот символ и его ASCII-код:

"\n#include <iostream>" +

"\nint main()" +

"\n{" +

"\nstd::cout << \"nput a keyboard character:\";" +

"\nchar ch;" +

"\nstd::cin >> ch;" +

"\nstd::cout << ch << \" has ASCII code \" << static\_cast<int>(ch) << std::endl;" +

"\nreturn 0;" +

"\n}" +

"\nРезультат выполнения программы:" +

"\nInput a keyboard character: q" +

"\nq has ASCII code 113\n";

"}"

**9.** **Cтроки - массив «чаров»**

Char string[] = "QuizLove"; // объявление символьного массива (строки)

Текст в кавычках и есть строковая константа.  Кавычки используются для определения начала и конца строковой константы и её частью не являются.

Объявляется строка таким образом – создаем массив типа **char**, размер в квадратных скобках указывать не обязательно (его подсчитает компилятор), оператор **=** и в двойных кавычках пишем необходимый текст. То есть инициализируем массив строковой константой:

Программа:

include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

char str[] = "Am I a great progger?";

// '\0' присутствует неявно

cout << str << endl;

return 0;

}

Прописывать нулевой символ не надо. Он присутствует неявно и добавляется в каждую такую строковую константу автоматически. **Таким образом, при том, что мы видим 21 символов в строке, размер массива будет 22, так как \0 тоже символ и занимает один байт памяти**.Займет онпоследнюю ячейку этого символьного массива.

Чтобы выводить русский текст пропишите перед объявлением массива символов функцию:

setlocale(LC\_ALL, "Russian");

Пока не заморачивайтесь , над этой функцией!

**10.** **Литералы**

Когда в программе встречается некоторое число, например 1, то это число называется литералом, или литеральной константой. Константой, потому что мы не можем изменить его значение, и литералом, потому что его значение фигурирует в тексте программы.

Виды литералов:

1. целочисленный литерал

Литералы целых типов можно записать в десятичном, восьмеричном и шестнадцатеричном виде:

Например, число 20 в разных СС:

20 //десятичный  
024 // восьмеричный  
0х14 // шестнадцатеричный

2. действительные числа

Их можно записывать с десятичной точкой, так и в научной (экспоненциальной) нотации.

Например:

3.14

1.0E-3E

3. слова true и false

Слова true и false являются литералами типа bool.

4. специальные знаки

Их довольно много, самые часто встречающиеся — это

новая строка \n

горизонтальная табуляция \t

одиночная кавычка \'

двойная кавычка \"

5. символьный литерал

Представимые литеральные символьные константы записываются как символы в одинарных кавычках. Например:

'a' '2' ',' ' '

6. строковый литерал

Строковый литерал – строка символов, заключенная в двойные кавычки. Такой литерал может занимать и несколько строк, в этом случае в конце строки ставится обратная косая черта.

Например:

adfjrngviewgvgrfrfqwe \

1234utuutuuuas"

Фактически строковый литерал представляет собой массив символьных констант, где по соглашению языков С и С++ последним элементом всегда является специальный символ с кодом 0 (\0) – пустой символ!