**1. Основные характеристики языка программирования С++**

**С++** - **компилируемый мультипарадигменный язык со строгой статистической типизацией**, поддерживающий решение как **«низкоуровневых»,** так и **«высокоуровневых»** задач.

**Компилируемый** (отличие от python)=> трансляция высокоуровневого исходного кода в низкоуровневый (машинный) код происходит путем компиляции, т.е. до выполнения программы файл с исходным текстом программы перед запуском анализируется и транслируется в машинный код без выполнения программы. Далее программа выполняется более «выгодным» образом. Отсюда возможен и выигрыш в скорости выполнения: мы сразу переводим программу выгодным образом, а не «придумываем» перевод походу – построчно- как в случае интерпретатора.

**Минусы компиляции:**

Отчасти к минусам компиляции можно отнести, что при каждом новом изменении приходится компилировать новый файл (сложнее исправлять и изменять программу).

**Мультипарадигменность**

Существует несколько парадигм (идеалогий) программирования, то есть различных идей, определяющих стиль написания компьютерных программ, например:

* Функционально программирование
* Объектно-ориентированное программирование
* Обобщённое программирование

На С++ можно писать программы, соответствующие разным парадигмам (писать те или иные операции как с помощью функций, так и, например, создавать классы, имеющие набор свойств, позволяющий совершить ту же операцию, не прибегая к функциям).

**Строгая статическая типизация** отличает C++ от python, где типизация динамическая. То есть в python, например, возможно изменить тип переменной, заданный исходно, данные возвращаемые функцией не привязаны к определенному типу (не задаются при создании функции) – в C и С++ так сделать нельзя.

«Высокоуровневые» задачи, решаемые с помощью C++ (задачи, выполняемые C++ как языком высокого уровня):

- разработка игр

- разработка настольных и кроссплатформенных приложений

- разработка библиотек, используемых для различных ПО

«Низкоуровневые» задачи, решаемые с помощью С++ (задачи, выполняемые как языком низкого уровня):

- вычислительные системы

- программирование встроенных систем

**2. Фундаментальные типы данных в С++:**

К фундаментальным типам данных в С++ относятся:

**- bool**

**- char**

**- int**

**- float**

**- void** (этот тип не обсуждался на семинаре)

Подробнее о каждом:

**bool – логический тип данных**

**Логические переменные —** это переменные, которые могут принимать только одно из двух возможных значений: **true (1) и false (0).** Логические значения не сохраняются как true или false, они обрабатываются в виде целых чисел: вместо true — 1, вместо false — 0. Именно эти значения и хранятся в памяти под этими переменными. Если нам необходимо выводить именно true/false, то нужно применять манипулятор форматирования **std::boolalpha: .**

**Для объявления логической переменной используется ключевое слово «bool»:**

**bool b; - объявление переменной b типа bool**

**bool b = true; - инициализация bool-евской переменной b**

**bool b(true); - иначе инициализация bool-евской переменной b**

Для типа bool применяются различные логические операции, результат которых мы можем предсказать, пользуясь таблицей истинности.

**Размер bool-евского типа (минимальный) – 1 байт.**

Язык C++ ограничивает только минимальный размер типов. Поэтому фактический размер переменных может отличаться на разных компьютерах. Для определения размера в конкретном случае используют оператор sizeof.

**Char – символьный тип данных**

Переменная типа char может хранить только 1 символ.

Переменная типа char занимает 1 байт. 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, он определяет способ представления символов английского языка (+ и широкоиспользуемых других) в виде чисел от 0 до 127. Эта часть ASCII универсальна для всех стран, однако char вмещает в себя ещё 128 – 255–ый символ, которые могут отличаться в зависимости от локализации (разные кодировки) => от них может быть проблема переносимости.

Char также может быть как signed, так и unsigned. По умолчанию может быть как signed, так и unsigned.

**Int – целочисленный тип данных**

Целочисленный тип данных — это тип, переменные которого могут содержать только целые числа (без дробной части, например: 1, 2, 3, 0). В языке C++ есть 4 основных целочисленных типов данных:

- short – минимальный размер 2 байт

- int – минимальный размер 2 или 4 байт

- long – минимальный размер 4 байт

- long-long – минимальный размер 8 байт

Тип char также может быть целочисленным типом данных. (char – 1 байт)

Основным различием между целочисленными типами является их размер, чем он больше, тем больше значений сможет хранить переменная этого типа.

Int бывает signed и unsigned. По умолчанию int знаковый, однако можно задать беззнаковость создав собственный тип данных или используя size\_t. Применять беззнаковость для расширения диапазона хранимых значений в положительную сторону не очень хорошая идея (т.к. объем данных зависит не только от типа данных, но и от архитектуры компьютера и от компилятора => работать на «краю» диапазона всё равно не удастся – при переходе на другой компьютер или компилятор может возникнуть ошибка - переполнение). Однако бывают задачи, где это удобно (где нет отрицательных чисел, например).

(Однако существуют целочисленные типы данных фиксированного размера, например, uint8\_t - 1 байт unsigned).

**Float, Double и long double – числа с плавающей точкой**

Переменная такого типа может хранить любые действительные дробные числа. Есть три типа данных с плавающей точкой: float, double и long double.

Минимальный размер:

float - 4 байта

double - 8 байт

long double - 8 байт

Типы данных с плавающей точкой всегда являются signed (т.е. могут хранить как положительные, так и отрицательные числа).

Переменные типа с плавающей запятой могут хранить только определенное количество значащих цифр, остальные — отбрасываются. Точность определяется количеством значащих цифр, которые представляют число без потери данных.

Точность зависит от размера типа данных (в типе float точность меньше, чем в типе double) и от присваиваемого значения:

точность float: от 6 до 9 цифр (в основном 7);

точность double: от 15 до 18 цифр (в основном 16);

точность long double: 15, 18 или 33 цифры (в зависимости от того, сколько байт занимает тип данных на компьютере).

В связи с этими ограничениями может происходить потеря/изменение точности (она характеризуется как увеличением числа значащих цифр (ошибка округления), так и уменьшением).

**3. Когда возникает проблема переносимости и как её предотвратить**

Проблема переносимости возникает вследствие 3х причин:

* Разный выбор диапазонов char
* Разные кодировки в unsigned char в символах 128 – 255
* Традиционно С++ не типизирует размер того или иного типа (есть ограничения только по минимальному размеру). => в зависимости от архитектуры компьютера тип int, например, разный размер.
* Кроме того, отсутствие установленного размера int неудобно при попытке совмещения с другими языками, где размер типов строго указан.

Решение вышеназванных проблем:

* Использовать только универсальные символы типа char (с 0 – 127)
* Использовать целочисленный тип данных не доходя до лимита памяти (придерживаться среднего числа).
* Использование псевдонимов из стандартной библиотеки с указанием размера int-а (например, uint32\_t – 4 байтовый беззнаковый int).

**4. Определение, объявление, инициализация, присваивание - …**

**Объявление переменной/ функции** – оно сообщает компилятору, что х является переменной, а также сообщает тип этой переменной.

**int х; — это объявление**

**int func() – это объявление**

**Определение переменной/ функции** – оно сообщает компилятору имя, тип и участок памяти, который будет занимать данная переменная.

Пример:

**Int main()**

**{**

**…**

**}**

Объявление функции + тело функции = определение функции

**Присваивание значения переменной** – присваиваем значение ранее объявленной переменной. Теперь, если позже мы вызовем эту переменную, то в выделенной для неё ячейки памяти будет «лежать» присвоенное ей значение.

**х = 5** — присваиваем значение 5 переменной х.

**Инициализация переменной** – объявление переменной и присвоение ей значения одновременно

**int x = 5** – это инициализация переменной

**5. Почему выравнивание данных в памяти увеличивает производительность?**

Выравнивание памяти увеличивает производительность, т.к. процессор за раз считывает определенное число «ячеек» памяти, определенное число байт. Следовательно, чтобы наши данные были считаны быстрее и корректнее, нам необходимо, чтобы они находились в одном блоке считывания (или в целом числе блоков, если данные больше блока).