**Отчёт по лабораторной работе №2 Представление данных в ЭВМ**

**Группа 8 Лазу Игорь, Меркулов Сергей**

**ОС: MS Windows**

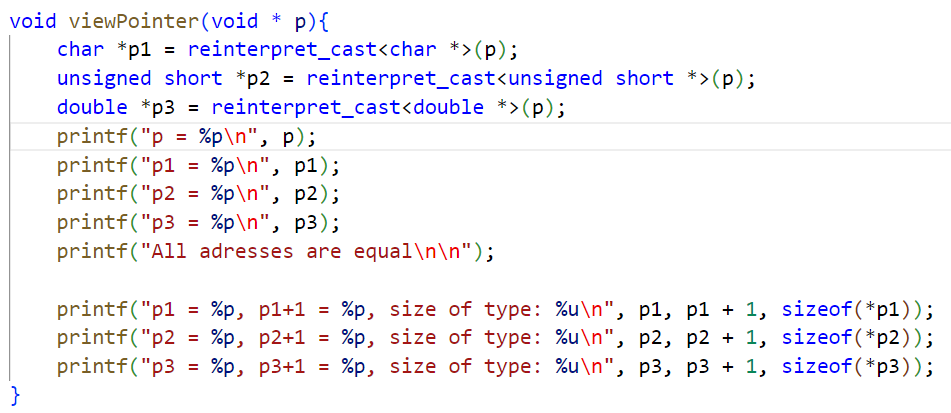
**Компилятор: gcc version \*Игорь, вставь свою версию компилятора сюда, пожалуйста\***

**Разрядность сборки: 64 бит**

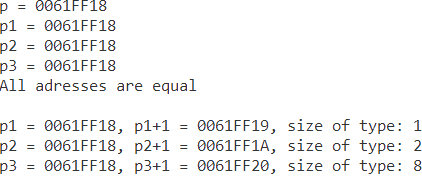
**Архитектура процессора: 64 бит**

**Назначение платформы: общее**

**Задание Л2.з1.** Разработайте функцию void 𝑣𝑖𝑒𝑤𝑃 𝑜𝑖𝑛𝑡𝑒𝑟(void \* 𝑝), которая принимает нетипизированный указатель 𝑝, преобразует его в типизированные: а) char \*p1 = reinterpret\_cast(p); б) unsigned short \*p2 = reinterpret\_cast(p); в) double \*p3 = reinterpret\_cast(p); и печатает 𝑝, 𝑝1, 𝑝2, 𝑝3 (не значения по этим адресам, а сами адреса). Убедитесь, что 𝑝, 𝑝1, 𝑝2, 𝑝3 — один и тот же адрес, то есть что оператор reinterpret\_cast не меняет преобразуемого указателя и, следовательно, может быть использован для интерпретации одной и той же области памяти как значений различных типов. Дополните 𝑣𝑖𝑒𝑤𝑃 𝑜𝑖𝑛𝑡𝑒𝑟() печатью смежных с 𝑝 адресов: 𝑝1 + 1, 𝑝2 + 1, 𝑝3 + 1. Сопоставьте разницу между 𝑝𝑖 и 𝑝𝑖 + 1 в байтах для типизированного указателя 𝑇 \* 𝑝𝑖 с размером типа 𝑇. Проверьте, позволяют ли текущие настройки компилятора рассчитать 𝑝 + 1. Если да — какова разница между 𝑝 и 𝑝 + 1 в байтах?

**Код: **

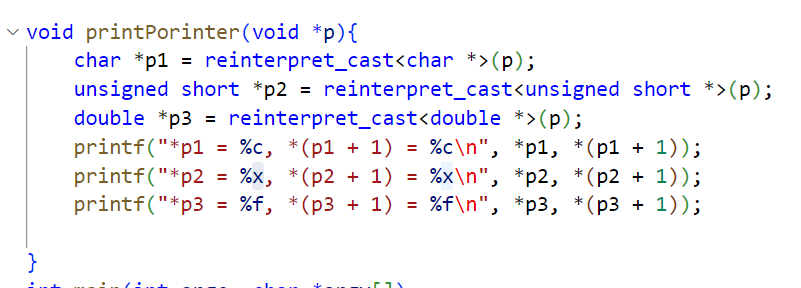
**Результат:**

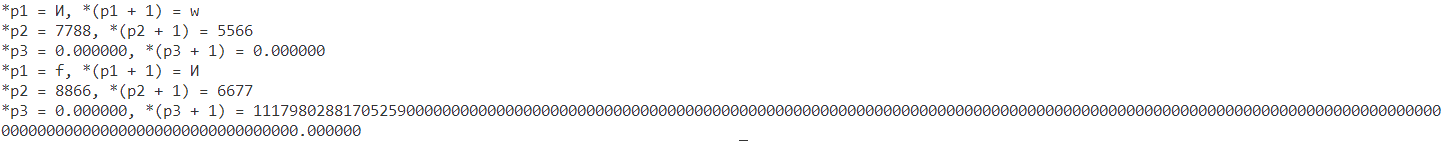
****

**Какова разница между p и p + 1 в байтах?**

- Разница между p и p+1 в байтах соответствует размеру типа данных самой переменной.

**Задание Л2.з2.** Разработайте функцию void 𝑝𝑟𝑖𝑛𝑡𝑃𝑜𝑖𝑛𝑡𝑒𝑟(void \*𝑝), которая принимает нетипизированный указатель 𝑝, преобразует его в типизированные 𝑝1, 𝑝2, 𝑝3 аналогично 𝑣𝑖𝑒𝑤𝑃 𝑜𝑖𝑛𝑡𝑒𝑟() и печатает значения соответствующих типов по адресу 𝑝: \*𝑝1, \*𝑝2, \*𝑝3. Дополните 𝑝𝑟𝑖𝑛𝑡𝑃 𝑜𝑖𝑛𝑡𝑒𝑟() печатью значений по смежным с 𝑝 адресам: \*(𝑝1 + 1), \*(𝑝2 + 1), \*(𝑝3 + 1). Все целые числа выводите в шестнадцатеричном виде. Проверьте работу функции 𝑝𝑟𝑖𝑛𝑡𝑃 𝑜𝑖𝑛𝑡𝑒𝑟() на значениях 0x1122334455667788 (𝑙𝑜𝑛𝑔 𝑙𝑜𝑛𝑔), "0123456789abcdef" (𝑐ℎ𝑎𝑟[])

**Код** 

**Результат (каво?)**

**Можно ли рассчитать (и, соответственно, напечатать) \*P?**

Что за вопрос? Какой-то бред? Да, ведь нам известен адрес переменной

**Задание Л2.з3.** Разработайте функцию void 𝑝𝑟𝑖𝑛𝑡𝐷𝑢𝑚𝑝(void \* 𝑝, size\_t 𝑁), которая принимает нетипизированный указатель 𝑝, преобразует его в типизированный указатель на байт 𝑢𝑛𝑠𝑖𝑔𝑛𝑒𝑑 𝑐ℎ𝑎𝑟 \* 𝑝1 и печатает шестнадцатеричные значения 𝑁 байтов, начиная с этого адреса: \*𝑝1, \*(𝑝1 + 1), ... \* (𝑝1 + (𝑁 − 1)) — шестнадцатеричный дамп памяти. Каждый байт должен выводиться в виде двух шестнадцатеричных цифр; байты разделяются пробелом

С помощью 𝑝𝑟𝑖𝑛𝑡𝐷𝑢𝑚𝑝() определите и выпишите в отчёт, как хранятся в памяти компьютера в программе на C/C++:

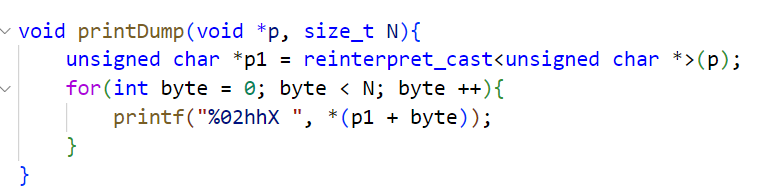
– целое число 𝑥 (типа 𝑖𝑛𝑡; таблица Л2.1); по результату исследования определите порядок следования байтов в словах для вашего процессора:

а) прямой (младший байт по младшему адресу, порядок Intel, Little-Endian, от младшего к старшему);

б) обратный (младший байт по старшему адресу, порядок Motorola, BigEndian, от старшего к младшему);

– массив из трёх целых чисел (статический или динамический, но не высокоуровневый контейнер) с элементами 𝑥, 𝑦, 𝑧; – число с плавающей запятой 𝑦 (типа 𝑑𝑜𝑢𝑏𝑙𝑒; таблица Л2.1).

**Вариант 2**

****

**Порядок прямой**