**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: “Типы данных и их внутреннее представление в памяти”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Лицкалов М. Ю. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение внутреннего строения памяти, понимания, как хранятся типы данных.

**Основные теоретические положения.**

Память состоит из непрерывной последовательности байтов (8 бит). Каждый байт памяти имеет адрес, используя который можно записать данные в байт или прочитать их.

Но простой доступ к памяти может создавать коллизии, т.к. нет определённых границ между байтами, т.е. изменение первой переменной может привести к изменению второй.

Чтобы решить эту проблему, в языках программирования были созданы типы данных.

Каждый тип данных определяет:

* Характер данных (как обращаться с байтами в памяти)
* Объём памяти (сколько байтов отведено под тип)
* Множество значений (какие значения может принимать тип)
* Возможные операции с типом

Например, в языке С существуют следующие основные типы данных:

* int (целый)
* float (вещественный)
* double (вещественный, увеличенная точность)
* bool (логический)
* char (символьный)

Так же существуют модификаторы типов данных, которые влияют на их хранение в памяти (применяется к стандартным типам):

* short (короткий)
* long (длинный)
* signed (знаковый)
* unsigned (беззнаковый)

В языке С переменную можно определить так:

<тип данных> <имя> [инициализатор]

Например, конструкция определения переменной var1 с типом int и значением 42:

int var1 = 42;

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

Разберём как хранится целочисленный тип данных в памяти (для 32-битного процессора):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | 31 | 30 | 29 | … | 5 | 4 | 3 | 2 | 1 |
| 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 0 |

* 32й бит отвечает за знак (если signed)
* Остальные биты представляют значение числа в двоичной системе счисления.

Если целое число используется с модификатором unsigned, то знаковый бит используется для представления числа в двоичной системе счисления.

Теперь разберём как хранятся вещественные типы данных в памяти:

Каждое вещественное число можно представить в виде мантиссы, умноженной на экспоненту (экспоненциальный формат). Например:

**3.14** = **0.314**e**+1**

Любое вещественное число в памяти представляется так:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | 31 | 30 | 29 | … | 5 | 4 | 3 | 2 | 1 |
| 0 | 1 | 0 | 0 |  | 1 | 1 | 0 | 0 | 1 |

* 32й бит отвечает за знак
* 31-24 биты отвечают за экспоненту числа
* Остальные биты отвечают за мантиссу числа

**Постановка задачи.**

Разработать алгоритм и написать программу, которая позволяет:

1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

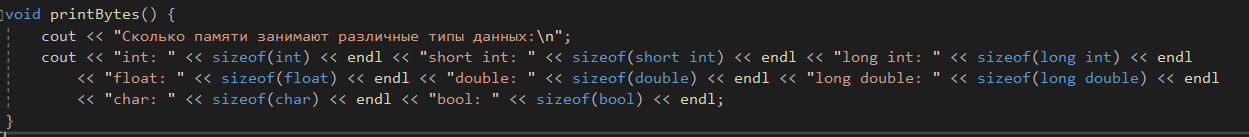
2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

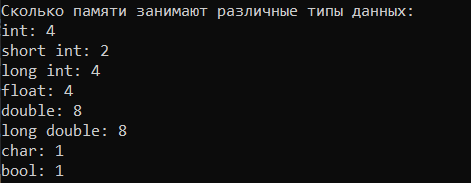
4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. (\*)

**Выполнение работы.**

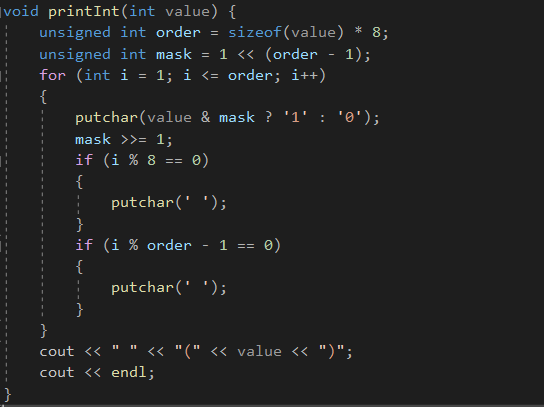
Функция, которая выводит на экран, сколько памяти отводится под различные типы данных

****

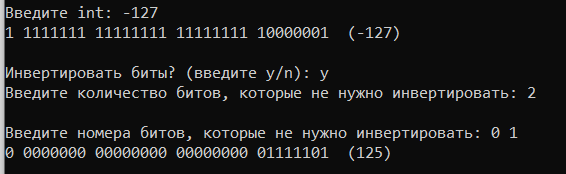
Результат выполнения:



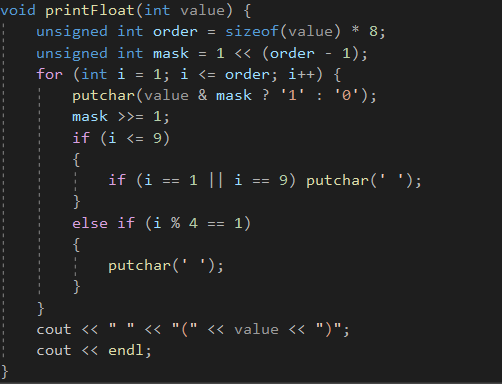
Функция для вывода на экран двоичного представления целого числа



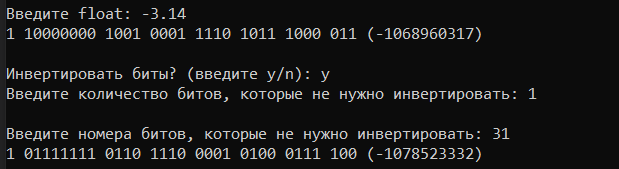
Результат выполнения:



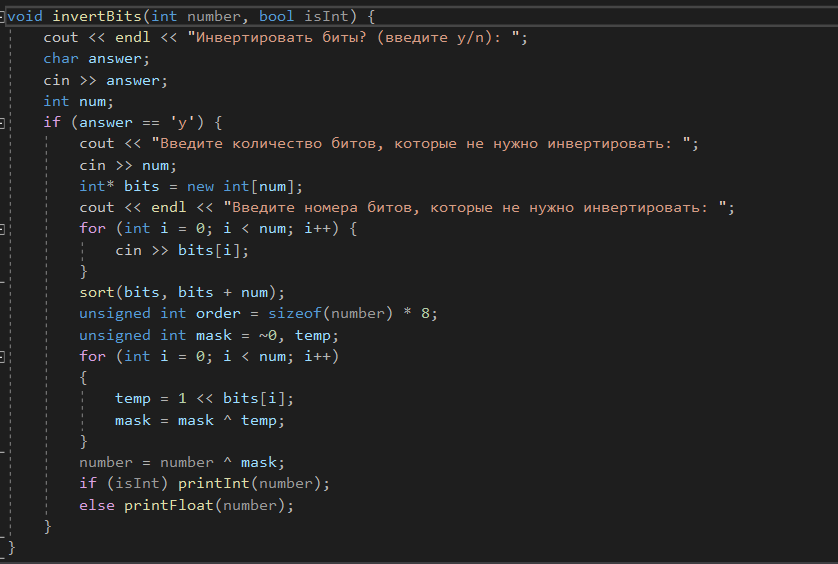
Функция для вывода на экран двоичного представления типа float



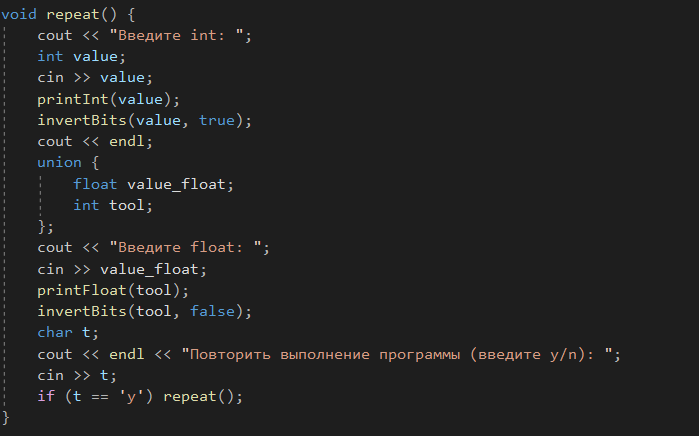
Результат выполнения:

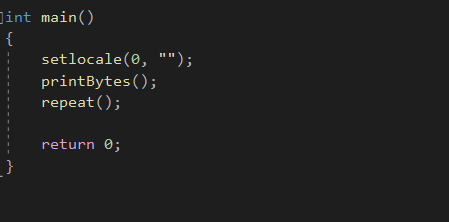


Функция, которая инвертирует все биты числа, кроме тех, номера которых ввел пользователь



Основная функция программы:





**Выводы.**

Я научился выводить двоичное представление типов данных (значения битов, которые хранятся в памяти), изучил внутреннее строение целых и вещественных чисел.