**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(ка) гр. | Тимошенков И.М. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Изучение и работы с памятью и представлением чисел в памяти.

**Основные теоретические положения.**

Внутреннее представление величин целого типа – целое число в двоичном коде. При использовании спецификатора signed старший бит числа интерпретируется как знаковый (0 – положительное число, 1 – отрицательное). Для кодирования целых чисел со знаком применяется прямой, обратный и дополнительный коды.

Представление положительных и отрицательных чисел в прямом, обратном и дополнительном кодах отличается. В прямом коде в знаковый разряд помещается цифра 1, а в разряды цифровой части числа – двоичный код его абсолютной величины. Прямой код числа −3 (для 16- разрядного процессора):



Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака: нули заменяются единицами, единицы – нулями. Прямой код можно преобразовать в обратный, инвертировав все значения всех битов (кроме знакового). Обратный код числа −3:



Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Дополнительный код числа −3:



Увидеть, каким образом тип данных представляется на компьютере, можно при помощи логических операций: побитового сдвига (<<) и поразрядной конъюнкции (&).

putchar(value & mask ? '1' : '0'); // если 1, то возвращается 1, иначе 0

value <<= 1; // побитовый сдвиг влево на 1 бит

Putchar возвращает один символ в консоль. Альтернатива - cout. В представленном способе, маска - то, с чем сравнивается значение. И побитовый сдвиг применяется для value. Таким образом 1 бит будет сравниваться с каждым битом числа. Альтернатива - побитовый сдвиг вправо, но при этом нужно проводить данную операцию не над значением(единицей), а над маской (исходым числом, битовое представление которого нужно получить).

При сдвиге вправо для чисел без знака позиции битов, освобожденные при операции сдвига, заполняются нулями. Для чисел со знаком бит знака используется для заполнения освобожденных позиций битов. Другими словами, если число 25 является положительным, используется 0, если число является отрицательным, используется 1. При сдвиге влево позиции битов, освобожденных при операции сдвига, заполняются нулями. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Вещественные типы данных хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка.

Для 32-разрядного процессора для float под мантиссу отводится 23 бита, под экспоненту – 8, под знак – 1. Для double под мантиссу отводится 52 бита, под экспоненту – 11, под знак – 1:



Увидеть, каким образом вещественные типы данных представляются в компьютере немного сложнее. Логические операции, которые использовались с int, для вещественных типов данных не подходят. Но это ограничение можно легко обойти, использовав объединения.

Объединения – это две или более переменных расположенных по одному адресу (они разделяют одну и ту же память). Объединения определяются с использованием ключевого слова union. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

С объединениями нужно быть острожным. Вся работа с памятью требует грамотного подхода. Более подробно с объединениями можно будет ознакомиться при изучении структур. Пока что объедения будут служить инструментом для работы с float и double.

**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.**

Код программы представлен в приложении А.

#include <iostream>

using namespace std;

int main() {

setlocale(0, "Rus");

cout << "int: " << sizeof(int) << "\n";

cout << "short: " << sizeof(short int) << "\n";

cout << "long: " << sizeof(long int) << "\n";

cout << "float: " << sizeof(float) << "\n";

cout << "double: " << sizeof(double) << "\n";

cout << "long double: " << sizeof(long double) << "\n";

cout << "char: " << sizeof(char) << "\n";

cout << "bool: " << sizeof(bool) << "\n";

int mask;

mask = 1 << (sizeof(int) \* 8) - 1; // создаем маску 1000000... и переводим байты в биты

// сдвигаем 1 на последний бит

int number; // инициализайия для введенного числа

cout << "Введите целое число \n";

cin >> number;

for (int i = 0; i < sizeof(int) \* 8; i++) // условие цикла

{

if (i == 1) // разделение разрядов на знак под символ и под значение

cout << ' ';

if (number & mask) // сравнение с маской, если значение после конъюкции не 0, то 1 на экран

cout << '1';

else

cout << '0';

number <<= 1; // сдвиг числа влево

}

cout << "\nВведите вещественное число \n";

union {

float floatValue; // запись двух переменных в один адрес

int numberForFloat;

};

cin >> floatValue;

for (int i = 0; i < sizeof(int) \* 8; i++) // условие цикла

{

if (i == 1 || i == 23) // выделение мантисы и порядка 0 1000011100111010000000 000000000

cout << ' ';

if (numberForFloat & mask) // сравнение с маской, если значение после конъюкции не 0, то 1 на экран

cout << '1';

else

cout << '0';

numberForFloat <<= 1; // сдвиг числа влево

}

cout << "\nВведите вещественное число \n";

union {

double doubleValue; // запись двух переменных в один адрес

int listForDouble[2]; // выделения массива, отводится 8 байт на 2 целочисленных числа

};

cin >> doubleValue; // после ввода, память выделенная для массива заполняется

for (int i = 0; i < 8 \* 8; i++) // в условии содержиться 64 бита, что соответсвует 8 байтам для типа double

{

// так как массив заполняет ячейки по порядку, то сначала выведем алгоритм для 0 элемента

for (; i < 4 \* 8; i++) { // программа работает пока i < 32

if (i == 1)

cout << ' ';

if (listForDouble[0] & mask) // сравнение с маской, если значение после конъюкции не 0, то 1 на экран

cout << '1';

else

cout << '0';

listForDouble[0] <<= 1; // сдвиг числа влево

}

// после окончания работы первого цикла, начинается работа с индексом 1 массива

if (i == 52) // выделение мантисы

cout << ' ';

if (listForDouble[1] & mask) // сравнение с маской, если значение после конъюкции не 0, то 1 на экран

cout << '1';

else

cout << '0';

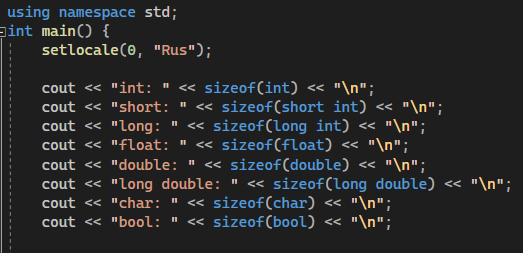
listForDouble[1] <<= 1; // сдвиг числа влево

}

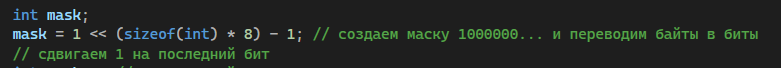
}

Блок описания кода и использованных алгоритмов

Сначала происходит вычисление размера памяти, который занимают разные типы данных, при помощи sizeof(), выводит на экран значения в байтах.

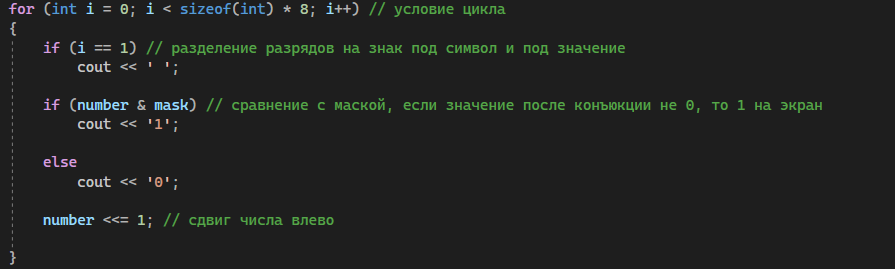


Далее, для выполнения задания, создается маска, она равна числено 1, при помощи битового сдвига происходит сдвиг битов числа 1, таким образом в памяти, цифра 1 стоит в 31 разряде.

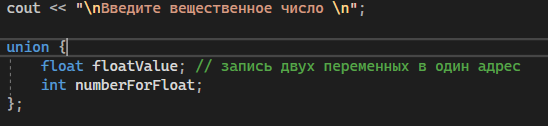


Пользователь вводит целое число.



Затем, алгоритм производит поразрядную конъюнкцию, введенное число сравнивается с маской, таким образом, если при конъюкции получилась правда, то выводится 1, введенное число сдвигается на 1 бит влево, алгоритм повторяется, таким образом получается копия представления числа в памяти.

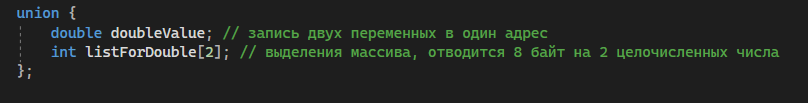
Программа предлагает пользователю ввести вещественное число, число имеет тип float, при помощи объединения 2 переменных располагаются в одном адресе, одна из них имеет типа int.



После ввода вещественного числа, разряды числа типа int тоже заполняются, таким образом, уже знакомый алгоритм выводит на экран двоичное представления вещественного числа в памяти. 

Так же, в алгоритме предусмотрено разделение выведенных символов на разряд, мантиссу и порядок.

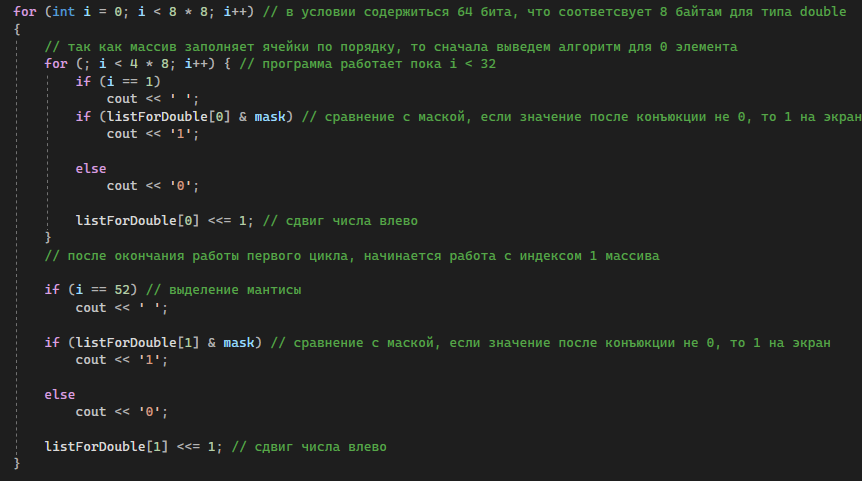
Далее пользователь должен ввести 1 вещественное число типа double. Производя объединение, программа записывает в 1 адрес число и массив 2х чисел типа int , под которой выделяется 8 байт, именно столько байт занимает тип double, такое объединение возможно так как массив занимает память компьютера последовательно, поэтому пол 2 int числа выделяется по 32 разряда.



После ввода числа типа double, память, выделенная под массив, тоже заполняется.

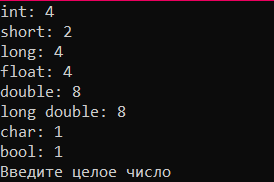


Затем начинается работа алгоритм, однако в этот раз, основной цикл содержит в себе вложенный цикл. Вложенный цикл работает для вывода слепка первого числа в массива, после окончания работы начинается работу основной алгоритм.



Исправная работа алгоритма возможна, из-за того, что массив расположен в памяти последовательно, поэтому сначала работает алгоритм для 0вого элемента, а потом для 1го. В циклах также предусмотрены условия для вывода мантиссы и порядка вещественного числа.

Блок скриншотов работы программы

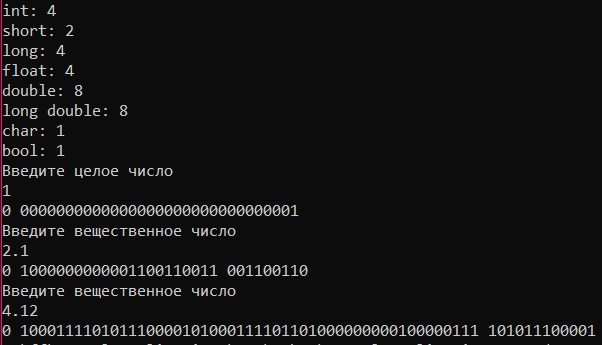


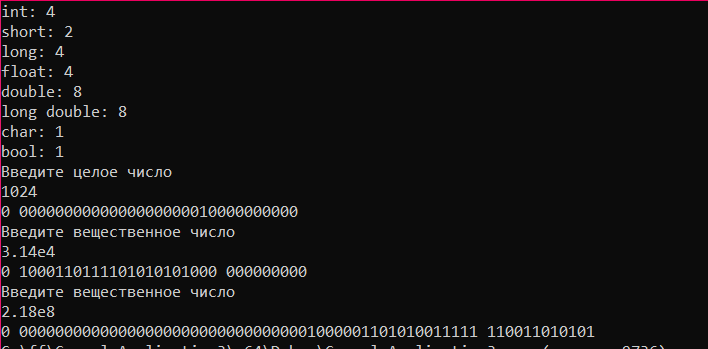


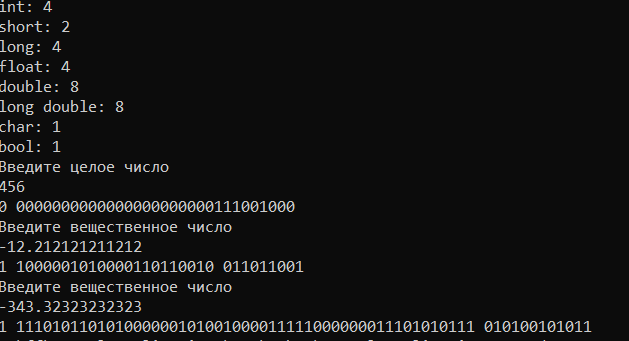




Блок таблицы с тестовыми данными







**Выводы:**

Я узнал о расположении переменных в памяти компьютера. Мною была написана программа, для вывода представления числа в памяти компьютера, так же я ознакомился с объединением переменных, ознакомился с функцией sizeof и размерами разных типов данных**.**