Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

**ДВОИЧНАЯ АРИФМЕТИКА**

Отчет по лабораторной работе №1 по дисциплине

«Вычислительная техника»

Студент гр. 431-3

\_\_\_\_\_\_\_\_\_ Гурулёв А.В

«11» сентября 2022 г.

Руководитель

\_\_\_\_\_\_\_\_\_\_\_ Алфёров С.М.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

Томск 2022

**Оглавление**

[1 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 4](#__RefHeading___Toc1136_256242134)

[4](#__RefHeading___Toc2045_952702260)

[ВЫВОД 4](#__RefHeading___Toc1890_3482136002)

[Приложение 1 5](#__RefHeading___Toc58981_1216390172)

[Приложение 2 8](#__RefHeading___Toc58981_12163901721)

**ВВЕДЕНИЕ**

Цель работы – Получить представление о способах хранения числовых данных в памяти ЭВМ.

Задание:

* Написать программу по выводу битовых данных (битов), числовых значений, хранящихся в переменных. Задать значения и ввести на экран биты: целых знаковых и беззнаковых чисел длиной 1,2 и 4 байта; вещественных чисел с плавающей запятой длиной 4 и 8 байт. Объяснить результат.

## **1** **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

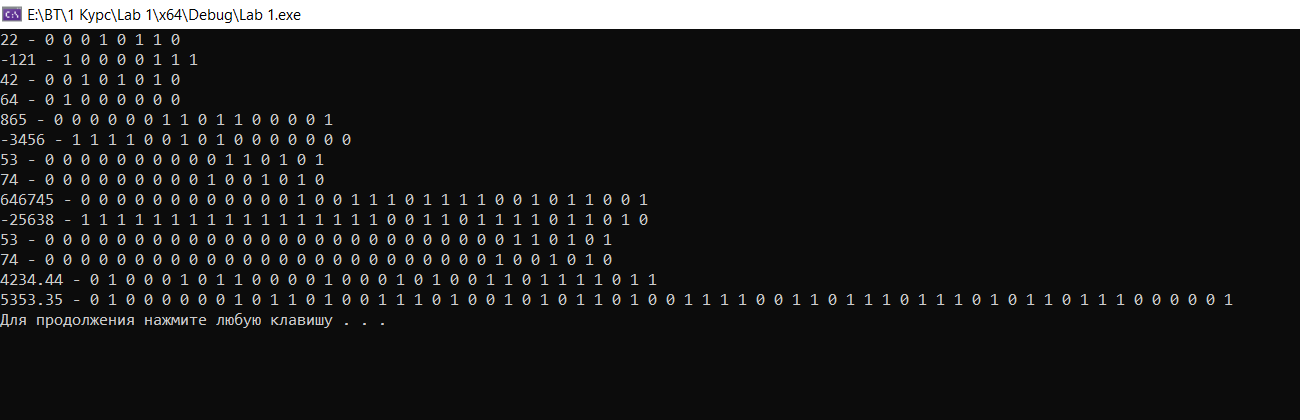


Рисунок 1.1 - Результат работы программы с 1 вариантом данных

## 

Рисунок 1.2 - Результат работы программы с 2 вариантом данных

## **ВЫВОД**

Я получил представление о способах хранения числовых данных в памяти ЭВМ.

## **Приложение 1**

**Листинг кода**

#include <iostream>

#include <ctime>

using namespace std;

int main()

{

setlocale(LC\_ALL, "russian");

//Переменные для работы с битами

unsigned char\* indOfNum;

unsigned char numInBites;

//Числа 1 байт

char nums\_OneBite[2] = { 22, -121 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&nums\_OneBite[j];

cout << (int)nums\_OneBite[j] << " - ";

for (int i = 0; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Числа только положительные 1 байт

unsigned char numsOneBite[2] = { 42, 64 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&numsOneBite[j];

cout << (int)numsOneBite[j] << " - ";

for (int i = 0; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Числа 2 байта

short int nums\_TwoBites[2] = { 865, -3456 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&nums\_TwoBites[j];

cout << nums\_TwoBites[j] << " - ";

for (int i = 1; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Числа только положительные 2 байт

unsigned short int numsTwoBite[2] = {53, 74 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&numsTwoBite[j];

cout << (int)numsTwoBite[j] << " - ";

for (int i = 1; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Числа 4 байта

int nums\_FourBites[2] = { 646745, -25638 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&nums\_FourBites[j];

cout << nums\_FourBites[j] << " - ";

for (int i = 3; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Числа только положительные 4 байт

unsigned int numsFourBite[2] = { 53, 74 };

for (int j = 0; j < 2; j++)

{

indOfNum = (unsigned char\*)&numsFourBite[j];

cout << (int)numsFourBite[j] << " - ";

for (int i = 3; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

}

//Число с точкой 4 байта

float num\_Poitn\_FourBites = 4234.4353;

indOfNum = (unsigned char\*)&num\_Poitn\_FourBites;

cout << num\_Poitn\_FourBites << " - ";

for (int i = 3; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

//Число с точкой 8 байт

double num\_Poitn\_EightBites = 5353.35342;

indOfNum = (unsigned char\*)&num\_Poitn\_EightBites;

cout << num\_Poitn\_EightBites << " - ";

for (int i = 7; i >= 0; i--)

{

for (int k = 7; k >= 0; k--)

{

numInBites = (indOfNum[i] >> k) % 2;

cout << (numInBites & 1) << " ";

}

}

cout << "\n";

system("pause");

return 0;

}

## **Приложение 2**

**Пояснение, почему вывод такой, каким является**

В случаях с положительными целочисленными значениями отображается результат перевода десятичного числа в двоичное.

С отрицательными же значениями, нужно проделать дополнительные операции, а именно произвести инверсию(1 заменить на 0 и наоборот), а так же добавить 1 к младшему разряд, это называется дополнительным кодом.

Двоичные числа сложнее всего представить.

Шаг 1: Перевести число в двоичный вид.

Шаг 2: Переместить точку так, чтобы слева была только одна «1»

Шаг 3: Найти порядок - p(количество цифр, которые прошла точка при сдвиге)

Шаг 4: Перевести результат данного вычисления p+127 в двоичную форму записи

Шаг 5: Записать число в следующей форме:

Если выделяется 4 байта:

1 бит — Знаковый

2 - 9 биты — записывается результат шага 4(экспонента).

10 - 32 биты — записывается результат шага 1(мантиса) со сдвигом в лево(удаляется первый бит, справа добавляется 0)

Если выделяется 8 байт:

Всё аналогично, но под экспоненту выделяются 2 - 12 биты, а под мантису 13 — 64 биты, а так же в шане 4, к p нужно прибавить не 127, а 1023