**МИНОБОРОНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра РЭС**

ОТЧЕТ

по лабораторной работе № 3

по дисциплине «Информатика»

Тема: Программа, переводящая неотрицательные десятичные целые числа в двоичные

Студент гр.: 3181 \_ \_ Долгих К. А.

Студент гр.: \_ \_ Ситников И. Ю.

Санкт-Петербург

2023

Содержание

1. Спецификация задания
2. Формализованное описание алгоритма решения задачи
3. Блок схема главного алгоритма
4. Выбор и обоснование типов переменных
5. Вводимые и выводимые параметры и их типы
6. Структура проекта, перечисление нужных файлов
7. Результаты лабораторной работы

Код программы

1. Спецификация задания

1. Необходимо создать консольное приложение Win32, осуществляющее

перевод десятичного числа со знаком в двоичное с выводом всех 32-х разрядов в прямом порядке.

2. Перевод должен быть осуществлен методом поразрядной проверки бита и логического сдвига маски.

3. На вход поступает строка, содержащая натуральное десятичное число

«a» в диапазоне -2 147 483 648…2 147 483 647. В строке возможны ошибки, такие как лишние символы, буквы, превышение длинны строки. Их необходимо исправить. Всего на вход может поступить до 100 строк.

4. Выход из программы происходит после ввода пользователем 0, перед выходом запрашивается подтверждение выхода.

5. Вывод сделать в виде таблицы со столбцами: номер, число, двоичный код числа.

1. Формализованное описание алгоритма решения задачи

На вход строки поступает строка символов. Необходимо определить какая эта строка по счету и не представляет ли собой строка символ 0 (это условие выхода из программы). Затем необходимо отфильтровать строку от лишних символов оставив символы цифр и символы «+» или «-» если они стоят первым символом. Если после от фильтровки осталась пустая строка необходимо сообщить об этом пользователю. Перед перевод строки в число необходимо проверить его на допустимый диапазон, количество символов в строке не должно превышать 11 (10 цифр и 1 знак «+» или «-»), затем определяем число со знаком или нет для это проверяем первый знак если он пустой значит у нас беззнаковое число и оно не должно превышать 4294967295, если знак «+» то 2147483647, если «-» то 2147483648 если входная строка по значению превышает то необходимо сообщить об этом пользователю иначе перевести строку в число. Затем необходимо перевести число в двоичную форму с разделением на тетрады. В консоль необходимо вывести 1 строку таблицы: 1 столбец порядковый номер введенного числа, 2 столбец строка после фильтра, 3 столбец число в двоичной форме, 4 столбец строка которую ввел пользователь до фильтра.

Для реализации необходимо

1. 3 строковых переменных
2. 3 переменные типа char
3. 2 целочисленных переменных

Строковые переменные:

1. Входная строка
2. Отфильтрованная строка
3. Строка представляющая собой число в двоичной форме

Переменные типа char

1. Счетчик для хранения порядкового номера введенного числа
2. Переменная хранящая номер ошибки
3. Переменная хранящая тип введенного числа (положительно, отрицательное или беззнаковое)

Целочисленные переменные

1. Число полученное после преобразования введенной строки
2. Число хранящие количество чисел в строке
3. Блок схема главного алгоритма
4. Выбор и обоснование типов переменных
5. Вводимые и выводимые данные
6. Структура проекта перечисление нужных файлов
7. Результаты лабораторной работы

Код программы

**Файл laba\_3.cpp**

#include <stdio.h>

#include <Windows.h>

#include "TranslateString.h"

const char\* int\_max = "+2147483647";

const char\* int\_min = "-2147483648";

const char\* uint\_max = " 4294967295";

int main()

{

#pragma region Объявление необходимых переменных

// Для работы с консолью

COORD pos;

HANDLE hConsole;

// Для работы программы

char input[13];

char count;

char errorFlag;

int inputNum;

int numCount;

char numType;

char\* filtrStr;

char\* tetra;

#pragma endregion

#pragma region Выделение памяти и инициализирование переменных

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

pos.X = 0;

pos.Y = 0;

filtrStr = (char\*)malloc(12);

tetra = (char\*)malloc(40);

count = 0;

errorFlag = 0;

numCount = 0;

numType = 0;

inputNum = 0;

#pragma endregion

#pragma region Начальные настройки

SetConsoleTextAttribute(hConsole, FOREGROUND\_INTENSITY);

COORD bufSize = { 120, 120 };

SMALL\_RECT windowSize = { 0, 0, 119, 119 };

SetConsoleTitle(L"Ass Convertor 4000");

SetConsoleScreenBufferSize(hConsole, bufSize);

SetConsoleWindowInfo(hConsole, true, &windowSize);

#pragma endregion

while (true)

{

#pragma region Украшательства

printf\_s("---------------------------\n");

printf\_s("|Enter number: |\n");

printf\_s("---------------------------");

pos.X = 15;

pos.Y = count + 1;

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

SetConsoleCursorPosition(hConsole, pos); // перемещяем курсор в координаты

#pragma endregion

#pragma region Ввод данных

for (size\_t i = 0; i < 13; i++)

input[i] = ' '; /\* на всякий затираем массив пустыми символами

до этого могли появлятся проблеммы по типу программа выдавала ошибку когда ее быть не должно

скорее всего это из-за того что считывая мы заполняем не весь массив а только часть и в оставшихся ячейках непонятные данные \*/

fgets(input, sizeof(input), stdin); /\* читаем строку из консоли

fgets т.к. scanf\_s не хочет считывать пустую строку из-за чего программа работает некоректно

также из-за это массив входных данных не 11 символов а 13 \*/

for (size\_t i = 0; i < 12; i++)

{

// для коректного вывода в таблицу убираем символы окночания строки(так как он может стоять не там где нам нужно) и символ перееноса строки

if (input[i] == '\0' || input[i] == '\n')

input[i] = ' '; // fgets помимо прочего также считывает символ переноса строки и окончания наличие которых мешает нам коректно вывести это строку в таблицу

}

input[12] = '\0'; // последний символ символ переноса строки

#pragma endregion

#pragma region Проверка на выход

if (input[0] == 48 && input[1] == 32)// если первый символ '0' а последующий пустой значит мы ввели только 0, а по условию задачи это условие окончания

{

pos.X = 0;

pos.Y = count + 2;

SetConsoleCursorPosition(hConsole, pos);

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

printf\_s("\nDosvidyli\n");

SetConsoleTextAttribute(hConsole, FOREGROUND\_INTENSITY);

break;

}

count++;

if (count > 100)

{

pos.X = 0;

pos.Y = count + 2;

SetConsoleCursorPosition(hConsole, pos);

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

printf\_s("\npls stop\n");

SetConsoleTextAttribute(hConsole, FOREGROUND\_INTENSITY);

break;

}

#pragma endregion

#pragma region Подготовка данных к выводу

filtrStr = StrToRight(Filter(input, errorFlag), 12); // фильтруем входную строку и выравниваем ее по правому краю

#pragma region Прверка на диапозон

if (errorFlag != 1)

{

for (size\_t i = 0; i < 11; i++)

if (filtrStr[i] > 47 && filtrStr[i] < 58)

numCount++;

if (numCount == 11)

errorFlag = 2;

if (errorFlag != 2)

switch (filtrStr[0])

{

case '+':

if (0 != ChecStr(filtrStr, StrToCharArray(int\_max, 11)))

errorFlag = 2;

break;

case '-':

if (0 != ChecStr(filtrStr, StrToCharArray(int\_min, 11)))

errorFlag = 2;

break;

default:

if (0 != ChecStr(filtrStr, StrToCharArray(uint\_max, 11)))

errorFlag = 2;

break;

}

}

#pragma endregion

SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN);

switch (errorFlag)

{

case 3:

case 0:

// преобразуем входную строку в число и переводим его в двоичный код разделенный на тетрады

tetra = ToBinCode(atoll(filtrStr));

break;

case 1:

tetra = StrToCharArray(" Empty string", 39);

break;

case 2:

tetra = StrToCharArray(" So big", 39);

break;

}

if (errorFlag) // если фильтр обнаружил некоректые символы то окрышиваем строку таблицы в красный иначе она остается зеленной

{

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED);

errorFlag = 0;

}

#pragma endregion

#pragma region Вывод одной строки таблицы

// устанавливаем курсор в начало строки

pos.X = 0;

pos.Y = count - 1;

SetConsoleCursorPosition(hConsole, pos);

// вывод 1 строку в таблицы

printf\_s("| %s |", ToString(count, 3));

printf\_s(" %s |", filtrStr);

printf\_s(" %s |", tetra);

printf\_s("%s |\n", StrToRight(input, 13));

SetConsoleTextAttribute(hConsole, FOREGROUND\_INTENSITY);

#pragma endregion

// устанавливаем координаты курсора

pos.X = 0;

pos.Y = count;

}

}

**Файл TranslateString.h**

#pragma once

char ChecStr(char\*, char\*);

char\* StrToCharArray(const char\*, int);

char\* Filter(char\*, char&);

char\* ToBinCode(unsigned int);

char\* StrToRight(char\*, int);

char\* ToString(int, int);

**Файл TranslateString.cpp**

#include <stdio.h>

#include <stdlib.h>

#include "TranslateString.h"

char ChecStr(char\* one, char\* two)

{

for (int i = 0; i < 11; i++)

{

if (one[i] > two[i])

return 1;

else if (one[i] < two[i])

return 0;

}

return 0;

}

char\* StrToCharArray(const char\* str, int len)

{

char\* buf;

buf = (char\*)malloc(len + 1);

char f = 0;

for (int i = 0; i < len; i++)

{

char a = str[i];

buf[i] = ' ';

if (str[i] == '\0' && f != 1) {

f = 1;

continue;

}

if(!f)

buf[i] = str[i];

}

buf[len] = '\0';

return buf;

}

char\* Filter(char\* src, char& errFlag)

{

char i = 0, j = 0;

char\* buf;

buf = (char\*)malloc(12);

buf[11] = '\0';

if (src[0] == '+' || src[0] == '-')

{

buf[j] = src[i];

i++;

j++;

}

for (;i < 11; i++)

{

buf[i] = ' ';

char f = src[i];

if (!(src[i] > 47 && src[i] < 58))

{

if (src[i] != ' ' && src[i] != '\0' && src[i] != '\n' && src[i] != -2)

errFlag = 3;

continue;

}

buf[j] = src[i];

j++;

}

if (j == 0)

errFlag = 1;

return buf;

}

char\* ToBinCode(unsigned int num)

{

char\* str;

str = (char\*)malloc(40);

str[39] = '\0';

for (

char i = 1, j = 0;

i < 40; i++) // в объявлении цикла объявляем переменные i и j типа char (чтоб они меньше места занимали)

{

if (i % 5 == 0 && i != 0)

{

str[39 - i] = ' ';

continue;

}

str[39 - i] = '0' + ((num >> j) & 1);

j++;

}

return str;

}

char\* StrToRight(char\* src, int size)

{

char\* buf;

buf = (char\*)malloc(size);

buf[size - 1] = '\0';

char i = 0, j = 0;

for (; i < size - 1; i++)

{

if (src[size - i - 2] != ' ')

{

buf[size - j - 2] = src[size - i - 2];

j++;

}

}

for (; j < size - 1; j++)

{

buf[size - j - 2] = ' ';

}

return buf;

}

char\* ToString(int num, int len)

{

char\* str;

str = (char\*)malloc(len + 1);

str[len] = '\0';

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

{

if (num == 0)

{

str[i] = ' ';

continue;

}

str[i] = '0' + (num % 10);

num /= 10;

}

return str;

}