**Лабораторная работа №4**

**«ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ ОДНОМЕРНЫХ СТАТИЧЕСКИХ МАССИВОВ»**

**Цель работы**: Изучить способы представления массивов в памяти ЭВМ, получить практические навыки реализации алгоритмов обработки одномерных массивов.

**1. Вариант задания**

Вариант 30: В одномерном массиве, состоящем из N целых элементов, вычислить:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после второго элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие на нечетных позициях, а во второй половине – элементы, стоявшие на четных позициях. Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена.

**2. Алгоритм программы**

Программа состоит из нескольких циклов , пару циклов вложенных друг в друга для сортировки, условий необходимых для выполнения правильных действий.

Рисунок 1 – Структурная схема программы для решения данной функции(1-я часть)

Рисунок 2 – Структурная схема программы для решения данной функции(2-я часть)

**3. Код программы**

#include<iostream>

#include<windows.h> // Подключаем три препроцессора . Один для ввода/вывода , второй для вычисления , третий для подключения консоли

#include<math.h>

using namespace std;

const int n = 50;

int main()

{

int minMod, suMod,count = 0,sum=0,temp;

system("chcp 1251"); // Задаем таблицу символов для работы с консолью

int massiv[n]; int temps[n];

srand( time(0) ); //задает постоянное рандомное значение

for (int i = 0; i < n;i++) // объявление массива , вывод его в консоль и поиск суммы элементов после двух 0

{

massiv[i] = rand()%200 - 100 ; //задаем случайное значение для элемента масива в рамках от -100 до 99

cout << massiv[i] << "; ";

if (massiv[i] == 0) {

count++;

}

if (count >= 2) // если два нулевых элемента было , сумируем остальные

{

sum += abs(massiv[i]);

}

}

minMod = abs(massiv[0]);

for (int i = 1; i < n; i++) //поиск минимального по модулю элемента

{

if (minMod > abs(massiv[i]))

{

minMod = abs(massiv[i]);

}

}

cout << "Минимальный по модулю элемнет массива равен = " << minMod<<endl;

cout << "Сумма элементов по модулю после 2-х нулевых = " << sum<< endl;

int center = n / 2 ; //определяем центр массива

int counter = 0; // переменная хранящая индекс числа

for (int i = 0; i < n; i++) //перенос данных согласно условию, слева то что стояло на нечетном месте , справа то что стояло на четной позиции

{

if (i % 2 == 1) // если позиция не кратна двум , то ставим его в начало

{

temps[counter] = massiv[i];

counter++;

}

else // если позиция кратна двум , то элементы ставим от центра

{

temps[center] = massiv[i];

center++;

}

}

cout << "Массив преобрпзованый так что в первой его половине располагаются элементы, стоявшие на нечетных позициях, а во второй половине – элементы, стоявшие на четных позициях " << endl;

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

{

cout << temps[i] << "; ";

}

cout << endl << "Упорядочить каждую часть массива по возрастанию, используя алгоритм сортировки методом прямого обмена : "<< endl;

//сортировка методом пузырька

center = n / 2; int j,i,tema;

for (j = center ; j > 1; j--) //сортировка левой части

for (i = 0; i < j - 1; i++)// сравнение двух соседних элементов

if (temps[i] > temps[i + 1]) {

tema = temps[i]; // перестановка элементов

temps[i] = temps[i + 1];

temps[i + 1] = tema;

}

for (j = n; j > 1; j--) //сортировка правой части

for (i = center; i < j - 1; i++)// сравнение двух соседних элементов

if (temps[i] > temps[i + 1]) {

tema = temps[i]; // перестановка элементов

temps[i] = temps[i + 1];

temps[i + 1] = tema;

}

// конец сортировки

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

{

cout << temps[i] << "; ";

}

return 0;

}

**4.Результаты тестирования программы**

Для проверки правильности работы программы , для начала , отключим полностью случайные значения. //srand( time(0) );

Выполним первичную генерацию массива и посмотрим что выведет программа.

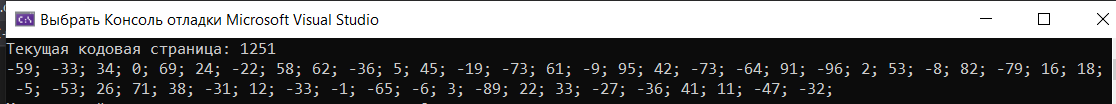


Рисунок 3 – Сгенерированный программой массив из целых рандомных значений.

Программа создала массив с такими переменными: -59; -33; 34; 0; 69; 24; -22; 58; 62; -36; 5; 45; -19; -73; 61; -9; 95; 42; -73; -64; 91; -96; 2; 53; -8; 82; -79; 16; 18; -5; -53; 26; 71; 38; -31; 12; -33; -1; -65; -6; 3; -89; 22; 33; -27; -36; 41; 11; -47; -32;

В данном массиве минимальный по модулю элемент = 0, сумма элементов по модулю после двух 0 равна 0 , так как второго ноля в массиве нет.

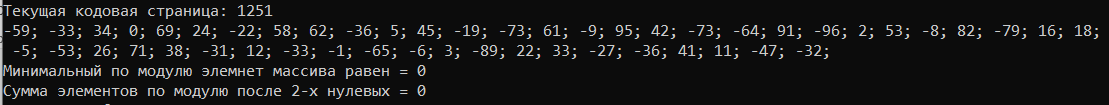


Рисунок 4 – Найденные минимальный по модулю элемент и сумма элементов по модулю после двух элементов равных 0.

Преобразованный по правилу массив должен быть таким : -33; 0; 24; 58; -36; 45; -73; -9; 42; -64; -96; 53; 82; 16; -5; 26; 38; 12; -1; -6; -89; 33; -36; 11; -32; -59; 34; 69; -22; 62; 5; -19; 61; 95; -73; 91; 2; -8; -79; 18; -53; 71; -31; -33; -65; 3; 22; -27; 41; -47;

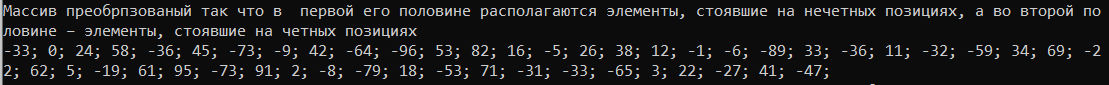


Рисунок 5 – Массив , полученные в результате перестановки элементов по правилу из задания

Данный массив надо отсортировать методом пузырька, при этом левая и правая сторона сортируются отдельно. В результате получим такой массив:

-96; -89; -73; -64; -36; -36; -33; -32; -9; -6; -5; -1; 0; 11; 12; 16; 24; 26; 33; 38; 42; 45; 53; 58; 82; -79; -73; -65; -59; -53; -47; -33; -31; -27; -22; -19; -8; 2; 3; 5; 18; 22; 34; 41; 61; 62; 69; 71; 91; 95;

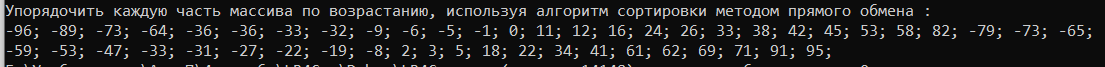


Рисунок 6 – Две половинки массива отсортированные по возрастанию элементов.

В результате тестирования, видно, что программа выполняет действия правильно , верно находит сумму элементов , минимальный по модулю элемент. Правильно переставляет элементы местами и сортирует массив методом прямого обмена(пузырька).

**Вывод**

В результате выполнения данной лабораторной работы были изучены принципы работы с массивами в C/C++, задание рандомных значения для элементов массива, вывод элементов массива в цикле в консоль. Так же были получены знания по применению на практике различных методы сортировки одномерных массивов.