ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ (ФГБОУ ВПО ИрГУПС)»

Ф

Факультет: ФТС

Кафедра: Информационные системы и защита информации

**Лабораторная работа №7**

***Подпрограммы***

Выполнил: Проверил:

студент группы МР-14-1 старший преподаватель

Лучников В.А.

Баканов М.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016г

Иркутск 2016

***Лабораторная работа № 7***

Вариант 3

***Тема:*** Программирование алгоритмов работы с массивами.

***Цель работы:*** Освоить основные алгоритмы работы с одномерными и многомерными массивами.Освоить методы тестирования программ.

**Задание 1**

Создать функцию, определяющую среднее арифметическое значение элементов вектора ***vectorn***, находящихся между вторыми по значению максимальным и минимальным его элементами. Предусмотреть случай, когда таких элементов нет.

**Задание 2**.

Создать процедуру, определяющую в целочисленном векторе ***vectorn*** два элемента, наиболее близкие к среднему арифметическому всех элементов вектора. Элементы, расположенные между ними, отсортировать по возрастанию.

1. Код программы Задание 1

#include <conio.h>

#include <math.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

float function(int n, int \*vector);

int main() {

int i, n, m, vector\_max, vector\_min, l;

tme\_t t; // Текущее время для инициализации генератора случайных чисел(рандомайзера)

srand((unsigned) time(&t)); // Инициализация рандомайзера

printf("\nLaboratornaya rabota nomer 7");

printf("\nZadanie nomer 1");

printf("\nVariant 3");

printf("\nMR-14-1");

printf("\nBakanov Maxim\n");

printf("\nVvedite min znachenie diapazona:"); // Ввод данных

scanf("%d", &vector\_min);

printf("\nVvedite max znachenie diapazona:");

scanf("%d", &vector\_max);

printf("\nVvedite razmer vectora:");

scanf("%d", &n);

int vector[n];

printf("\nIshodniy Vector:\n");

m=vector\_max-vector\_min+1;

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

{

vector[i]=rand()%m+vector\_min; // Получение случайного числа в диапазоне от vector\_min до vector\_max

printf("%5d", vector[i]);

}

printf("\n");

l=function(n, vector);

printf("Srednee arifmeticheskoe znachenie majdu vtorim max i vtorim min= %d",l);

}

float function(int n, int \*vector)

{

int max, min, max\_2, min\_2, i, n\_max\_2=0, n\_min\_2=0, temp, k=0, summ=0;

max=min=\*vector;

for(i=0;i<n;i++) //находим первые мин. и макс. эл-ты

{

if (\*(vector+i)>max)

max=\*(vector+i);

if (\*(vector+i)<min)

min=\*(vector+i);

}

min\_2=max;

max\_2=min;

for(i=0;i<n;i++) //находим вторые по знач. мax и min

{

if ((\*(vector+i)<min\_2) && (\*(vector+i)!=min))

{

min\_2=\*(vector+i);

n\_min\_2=i;

}

if ((\*(vector+i)>max\_2) && (\*(vector+i)!=max))

{

max\_2=\*(vector+i);

n\_max\_2=i;

}

}

if (n\_min\_2>n\_max\_2)

{

temp=n\_min\_2;

n\_min\_2=n\_max\_2;

n\_max\_2=temp;

}

if(max\_2==min\_2)

{

printf("\nMAX i MIN odno i to zhe chislo\n");

}

else

{

printf("\nmax\_2=%d n\_max\_2=%d", max\_2, n\_max\_2);

printf("\nmin\_2=%d n\_min\_2=%d\n", min\_2, n\_min\_2);

for(i=((n\_min\_2)+1); i<n\_max\_2; i++)

{

summ+=\*(vector+i);

k++; //считаем кол-во элементов которые мы суммируем для получения ср. арифметич. знач.

}

printf("\nk=%d\n", k);

}

if (k==0) return 0;

else return summ/k;

}

**Задание 2**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <math.h>

void FillArray( int\* array, const int size, int min, int max ); //функция заполняет одномерный целочисленный массив псевдослучайными данными в диапазоне [ min, max ]

void PrintLnArray( int\* array, const int size ); //функция печатает одномерный целочисленный массив

double GetAverageArray( int\* array, const int size ); // функция возвращает среднее арифметическое одномерного целочисленного массива

void FindIndexes( int\* array, const int size, int\* index1, int\* index2 ); // функция находит и записывает в аргументы индексы двух элементов наиболее близких к среднему арифметическому одномерного целочисленного массива

void SortPartArray( int\* array, int start, int finish ); // функция сортирует часть одномерного целочисленного массива между указанными индексами, первый индекс должен быть меньше второго

int main(void )

{

srand( time( 0 ) ); // засеять генератор случайных чисел

//\*\*ВВОД ДАННЫХ\*\*

int size;

printf("\nLaboratornaya rabota nomer 7");

printf("\nZadanie nomer 2");

printf("\nVariant 3");

printf("\nMR-14-1");

printf("\nBakanov Maxim\n");

printf( "Vvedite razmer massiva " );

printf( "Vvedite razmer massiva " );

scanf( "%u", &size );

int minRange, maxRange;

printf( "vvedite cherez probel granitsi min i max " );

scanf( "%d%d", &minRange, &maxRange );

//\*\*ЗАПОЛНЕНИЕ МАССИВА\*\*

int\* vec = ( int\* ) malloc( size \* sizeof(int) ); // объявляем наш массив как указатель на int и выделяем необходимую память для него

FillArray( vec, size, minRange, maxRange ); // заполняем массив случайными данными в диапазоне [min, max]

puts( "\nIshodniy massiv:" );

PrintLnArray( vec, size ); // выодим массив на экран

//\*\*РАСЧЁТЫ И ВЫВОД\*\*

double avg = GetAverageArray( vec, size ); // находим среднее арифметическое массива и выводим его на экран

printf( "\nSrednee arifmeticheskoe = %.3f\n", avg );

int index1, index2; // находим индексы элементов

FindIndexes( vec, size, &index1, &index2 );

printf( "\nNaidennie elementi %d [ %d ]\t%d [ %d ]\n", index1, vec[ index1 ], index2, vec[ index2 ] );;

SortPartArray( vec, index1, index2 ); // сортируем элементы между найденными индексами

puts( "\nRezultiruiushiy massiv" );

PrintLnArray( vec, size ); // выводим результирующий массив на экран

free( vec ); // освобождаем память занятую массивом

return EXIT\_SUCCESS; // выходим из программы

}

void FillArray( int\* array, const int size, int min, int max )

{

int i;

++max; // если не увеличить max, то диапазон будет [min, max ), а нжно [min, max ]

for ( i = 0; i < size; ++i )

array[ i ] = min + rand() % ( max - min ); // каждый элемент массива заполняется значением

}

void PrintLnArray( int\* array, const int size ) // в цикле выводим все значения массива по одному

{

int i;

for ( i = 0; i < size; ++i )

printf( "%4d", array[ i ] );

puts( "" ); // обрыв строки

}

double GetAverageArray( int\* array, const int size )

{

int i;

double sum = 0;

for ( i = 0; i < size; ++i ) // в цикле считаем сумму всех элементов массива

sum += array[ i ]; // находим среднее арифметическое значение

return sum / size;

}

void FindIndexes( int\* array, const int size, int\* index1, int\* index2 )

{

double avg = GetAverageArray( array, size ); // рассчитываем и запоминаем среднее арифметическое

int i;

\*index1 = 0, \*index2 = 1;

for ( i = 1; i < size; ++i ) // в цикле находим индексы двух элементов значения которых наиболее близки к среднему арифметическому

{ // поочерёдно сравниваем разницу элемента с индексом index1 и среднего арифметического с разницей очередного элемента и среднего арифметического

if ( fabs( avg - array[ i ] ) < fabs( avg - array[ \*index1 ] ) )

// если нашли разницу меньше, запоминаем это

\*index2 = \*index1;

\*index1 = i;

// аналогично здесь

} else if ( fabs( avg - array[ i ] ) < fabs( avg - array[ \*index2 ] ) )

\*index2 = i;

}

// Может быть ситуация, когда index2 будет меньше чем index1

// Предусмотрим обмен значений индексов - чтобы index1 ВСЕГДА был НЕ БОЛЬШЕ чем index2

int temp;

if ( \*index1 > \*index2 ) {

temp = \*index1;

\*index1 = \*index2;

\*index2 = temp;

}

}

void SortPartArray( int\* array, int start, int finish )

{

int i, exit = 0;

// цикл ПОКА не закончили сортировку

while ( !exit )

{

exit = 1; // предполагаем, что на данном шаге можем закончить сортировку

for ( i = start + 1; i < finish - 1; ++i )

if ( array[ i ] > array[ i + 1 ] ) // в цикле поочерёдно сравниваем каждый элемент с последующим элементом

{ // если текущий элемент больше чем следующий за ним, меняем их местами

int temp = array[ i ];

array[ i ] = array[ i + 1 ];

array[ i + 1 ] = temp;

exit = 0; // если была перестановка, то ещё раз пробегаемся по массиву(пробуем найти ещё элементы для сортировки)

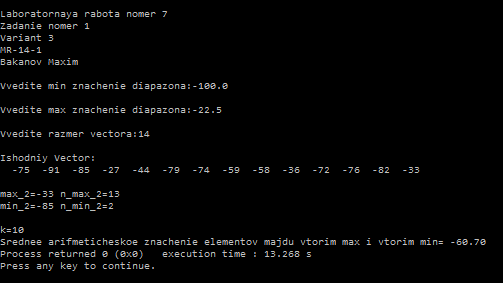
}

}

}

**4.** Компиляция, отладка, тестирование и получение результата:

Задание 1



Задание 2

