МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №9

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Динамические массивы»

Работу выполнил

студент гр. 4941 Н.С. Горбунов

Санкт-Петербург

2020

***Цель лабораторной работы:*** *изучение структурной организации динамических массивов и способов доступа к их элементам с использованием указателей; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки динамических массивов.*

***Задание на программирование:*** *используя технологию процедурного программирования, разработать программу обработки одномерных и двумерных (матриц) динамических массивов в соответствии с индивидуальным заданием.*

***Порядок выполнения работы:***

1. Получить у преподавателя индивидуальное задание и выполнить постановку для каждой из задач: сформулировать условие, определить входные и выходные данные, их ограничения.

2. Разработать математическую модель: описать с помощью формул и рисунков структуру массивов и процесс их преобразования.

3. Построить схемы алгоритмов решения задач и основных функций.

4. Составить программу на языке *C*/*C*++.

5. Входные данные на этапах тестирования и демонстрации работы преподавателю должны задаваться либо с использованием специально подобранных арифметических формул, либо вводиться с клавиатуры по запросу. **Датчики псевдослучайных чисел использовать запрещается**.

Если это не оговорено в конкретном варианте задания, значения элементов в каждом исходном массиве **должны быть разными**.

6. Выходные данные должны выводиться на экран с пояснениями. Операторы вывода результатов работы должны находиться либо в функции *main*(), либо в специальной функции вывода (например, преобразованного массива), вызов которой осуществляется из функции *main*().

7. Проверить и продемонстрировать преподавателю работу программы на полном наборе тестов, в том числе с ошибочными входными данными. Входные и выходные массивы должны выводиться в одном и том же формате.

8. Использовать стандартные потоковые объекты ввода/вывода ***cin*** и ***cout***.

9. Оформить отчет о лабораторной работе в составе: постановка задачи, математическая модель, схема алгоритма решения, текст программы, контрольные примеры (скриншоты).

Вариант 13:

А. Дан массив a0, a1, a2,…, a2n-1. Определить произведение положительных значений элементов массива, лежащих между элементом с минимальным по абсолютной величине значением и элементом с номером n.

Б. В заданной квадратной матрице размера 2n\*2n поменять местами значения элементов строк с одинаковыми номерами областей 3 и 8 (см. рисунок).

Математическая модель:

А) Решение задачи начинается с ввода исходных данных. Прежде всего, необходимо ввести значение (*m*) размера массива.

После этого необходимо задать значения всех элементов массива. Доступ к элементам массива осуществляется по их индексу (номеру данного элемента массива). Поэтому перебираем все индексы элементов массива от 0 до *m*-1 (нумерация элементов массива начинается с 0) и задаём значения элементов с текущими номерами.

Теперь приступаем к решению задачи. Для начала необходимо найти индекс минимального по модулю элемента. В цикле необходимо перебрать все элементы, сравнивая их модулю с переменной, хранящей минимальное значение, если элемент меньше, то записать его в переменную и «запомнить» его индекс.

Следующим шагом необходимо определить границы произведения, чтобы меньшее было границей слева, а большее (по индексу) справа. Если это не так, то меняем значения местами. В цикле перебираем от левой границы до правой элементы массива, и если они больше нуля, то умножаем на переменную, хранящую произведение и в неё же записываем полученное значение. В итоге получим искомое. Задача решена.

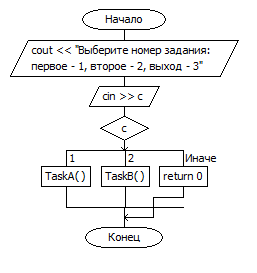
Б) Решение задачи начинается с ввода исходных данных. Из формулировки задачи понятно, что исходная матрица квадратная с четным числом элементов. Следовательно, прежде всего, необходимо ввести значение половины (*n*) размера матрицы.

После этого необходимо задать значения всех элементов матрицы. Доступ к элементам двумерного массива (матрицы) осуществляется по двум индексам: номеру строки и номеру столбца, на пересечении которых находится данный элемент массива. Поэтому перебираем все строки (с 0-ой до *n*-1). Внутри каждой строки перебираем все столбцы (с 0-ого до *n*-1) и задаём значения элементов, лежащих на пересечении строки и столбца с текущими номерами.

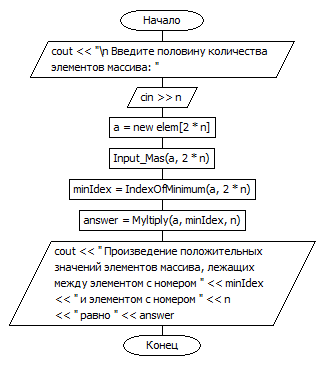
Теперь приступаем к решению задачи. Для начала определим изменение номера столбца в области 8 и номера строки . Заметно, что при переносе элементов из области 8 в область 3 номер строки может не меняется (j = const), при этом номер столбца будет подчиняться такой зависимости . То есть нам нужно перенести элемент с координат (i,j) в координаты (2\*n-1-i, j). Перебираю в двух циклах и меняю элементы в соответствии с заданием. Задача решена.

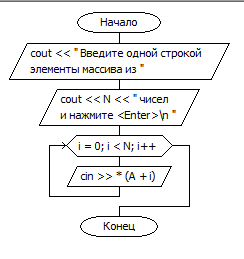
Блок-схема алгоритма

main():

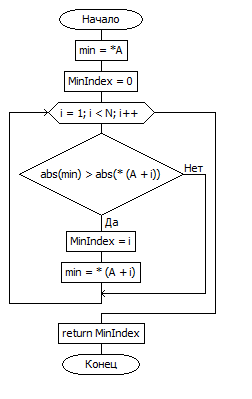


TaskA():

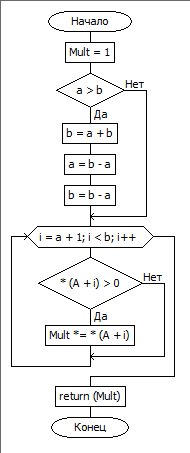


Input\_Mas():

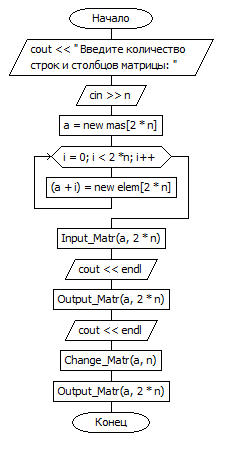
IndexOfMinimum():



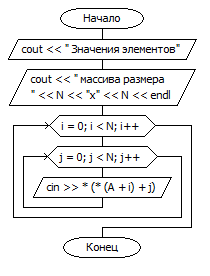
Myltiply():



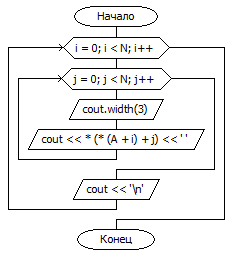
TaskB():



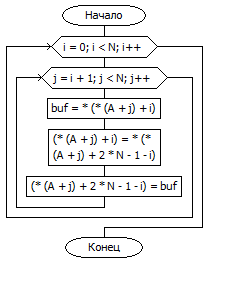
Input\_Matr():



Output\_Matr():



Change\_Matr():



Текст программы:

#include <iostream>

#include <math.h>

using namespace std;

typedef int elem; // тип элемент

typedef elem \*mas; //тип "указатель на элемент"

typedef mas\* matr; // тип "указатель на массив"

void Input\_Mas(mas A, int n); // функфия ввода

int IndexOfMinimum(mas A, int N);

elem Myltiply(mas A, int a, int b);

void TaskA();

void Input\_Matr(matr A, int N);

void Change\_Matr(matr A, int N);

void Output\_Matr(matr A, int N);

void TaskB();

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Выберите номер задания: первое - 1, второе - 2, выход - 3 ";

int c;

cin >> c;

switch (c)

{

case 1: TaskA(); break;

case 2: TaskB(); break;

default: return 0;

}

}

void TaskB()

{

matr a;

int n;

cout << " Введите половину количества строк и столбцов матрицы: ";

cin >> n;

a = new mas[2\*n]; //выделение динамической памяти под массив

//указателей на строки массива

for (int i = 0; i < 2\*n; i++) //выделение памяти под каждую строку

\*(a + i) = new elem[2\*n]; //каждому элементу массива указателей

//на строки присваивается адрес начала

//области памяти, выделяемой под строку

Input\_Matr(a, 2\*n);

cout << endl;

Output\_Matr(a, 2\*n);

cout << endl;

Change\_Matr(a, n);

Output\_Matr(a, 2\*n);

}

void TaskA()

{

int n; //размер массива

cout << "\n Введите половину количества элементов массива: ";

cin >> n;

mas a = new elem[2\*n];

Input\_Mas(a, 2\*n);

int minIdex = IndexOfMinimum(a, 2 \* n);

elem answer = Myltiply(a, minIdex, n);

cout << " Произведение положительных значений элементов массива, лежащих между элементом с номером "<< minIdex <<" и элементом с номером " << n <<" равно "<< answer;

}

void Input\_Mas(mas A, int N)

{

cout << " Введите одной строкой элементы массива из ";

cout << N << " чисел и нажмите <Enter>\n ";

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

cin >> \*(A + i);

}

int IndexOfMinimum(mas A, int N)

{

elem min = \*A;

int MinIndex = 0;

for (int i = 1; i < N; i++)

{

if (abs(min) > abs(\* (A + i)))

{

MinIndex = i;

min = \*(A + i);

}

}

return MinIndex;

}

elem Myltiply(mas A, int a, int b)

{

elem Mult = 1;

if (a > b)

{

b = a + b;

a = b - a;

b = b - a;

}

for (int i = a + 1; i < b; i++)

{

if(\*(A + i) > 0)

Mult \*= \*(A + i);

}

return(Mult);

}

void Input\_Matr(matr A, int N)

{

cout << " Значения элементов";

cout << " массива размера " << N << "x" << N;

cout << "\n при тестировании вводятся автоматически:" << endl;

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

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

\*(\*(A + i) + j) = 10 \* i + j;

}

void Output\_Matr(matr A, int N)

{

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

{

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

{

cout.width(3); //ширина поля выводимого параметра

cout << \*(\*(A + i) + j) << ' ';

}

cout << '\n';

}

}

void Change\_Matr(matr A, int N)

{

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

{

for (int j = i + 1; j < N; j++)

{

elem buf = \*(\*(A + j) + i);

\*(\*(A + j) + i) = \*(\*(A + j) + 2 \* N - 1 - i);

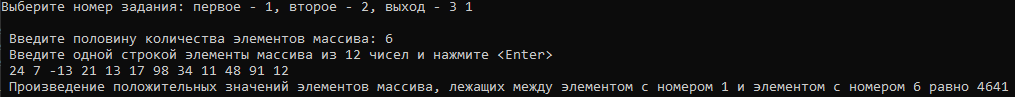
\*(\*(A + j) + 2 \* N - 1 - i) = buf;

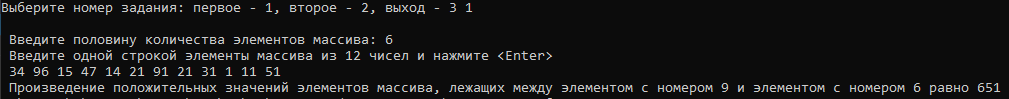
}

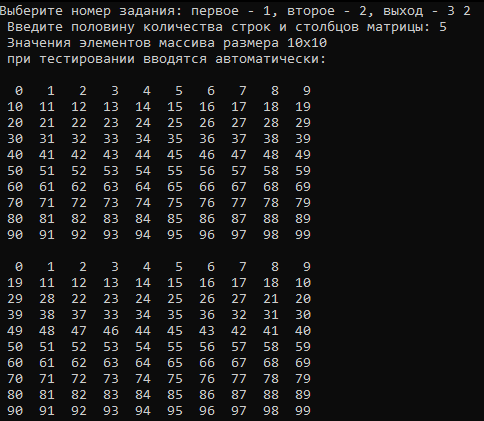
}

}

Скриншоты тестов:







Вывод: *используя технологию процедурного программирования, разработал программу обработки одномерных и двумерных (матриц) динамических массивов в соответствии с индивидуальным заданием.*