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. 4141 В.С. Сыворотнев

Санкт-Петербург

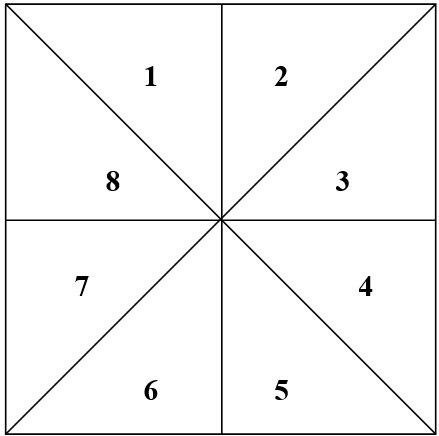
2022

***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.*

**31.**

А. Дан массив *b*0, *b*1, *b*2,…, *b*2*n-*1. Определить произведение значений элементов массива с чётными номерами, лежащих между первым элементом с наибольшим по абсолютной величине значением и последним элементом с отрицательным значением, имеющим номер меньше, чем *n*.

Б. В заданной квадратной матрице размера 2*n×*2*n* поменять местами значения элементов строк области [5+6+7] и элементов столбцов области [2+3+4] (см. рисунок) с одинаковыми номерами.

***A)***

***Математическая модель решения***

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

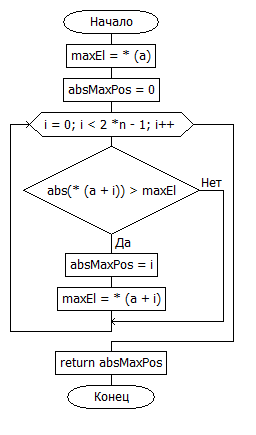
После этого необходимо задать значения всех элементов массива. Доступ к элементам массива осуществляется по их индексу (номеру данного элемента массива). Поэтому перебираем все индексы элементов массива от 0 до 2*n* (нумерация элементов массива начинается с 0) и задаём значения элементов с текущими номерами.

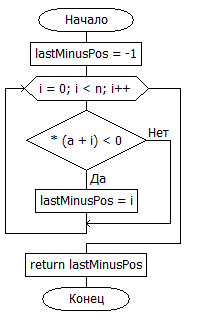
Пусть исходный массив имеет вид (*n*=6). Теперь можно приступать к решению задачи. Вначале находим первый элемент с наибольшим по абсолютной величине значением, затем последний элемент с отрицательным значением.

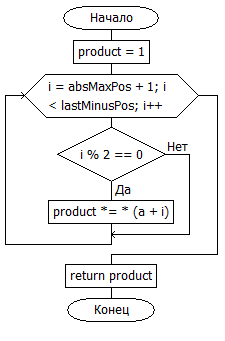
В результате мы имеем индексы элементов массива, которыми мы воспользуемся для поиска произведения. Примем absMaxPos(позиция первого элемента с наибольшим по абсолютной величине значением) и lastMinusPos(позиция последнего элемента с отрицательным значением. Затем запускаем цикл for, перебирая все элементы промежутка и записывая результат перемножения четных элементов в переменную product.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **n** | **7** | **8** | **9** | **10** | **2n -1** |
| **10** | **-30** | **10** | **2** | **5** | **-6** | **-1** | **-2** | **3** | **5** | **4** | **3** |

Задача решена.

***Схема алгоритма поиска индекса первого элемента массива с наибольшим по абсолютной величине значением (функция findAbsMaxPos)***

***Схема алгоритма поиска индекса последнего элемента массива с отрицательным значением (функция findMinusPos)***

***Схема алгоритма нахождения конечного произведения (функция findProduct)***

***Текст программы:***

#include<iostream>

#include<locale.h>

using namespace std;

typedef int telem;    //объявление типа элемента массива

typedef telem\* tmas;    //объявление типа "указатель на telem"

void inputArr(tmas& a, int n);

void outputArr(const tmas& a, int n);

int findAbsMaxPos(const tmas& a, int n);

int findMinusPos(const tmas& a, int n);

telem findProduct(const tmas& a, int absMaxPos, int lastMinusPos);

int main() {

  int n;                //размер массива

  int lastMinusPos;

  int absMaxPos;

  telem product;

  setlocale(LC\_ALL, "Russian");

//Ввод исходных данных

  cout << "Массив вида 2\*n" << endl;

  cout << "**\n**Введите количество элементов массива: ";

  cin >> n;

  tmas a = new telem[2 \* n];    */\*создание переменной-указателя на telem,*

*выделение динамической памяти под массив*

*адрес начала области заносится в a\*/*

  inputArr(a, n);

  absMaxPos = findAbsMaxPos(a, n);

  lastMinusPos = findMinusPos(a, n);

  if (lastMinusPos == -1) {

    cout << "Ошибка. Элементов с отрицательным значением, имеющих номер меньше n не найдено.";

    return 1;

  }

  if (abs(absMaxPos - lastMinusPos) == 1) {

    cout << "Промежутка нет, элементы являются соседними";

    return 1;

  }

  cout << "исходный массив:" << endl;

  outputArr(a, n);

  product = findProduct(a, absMaxPos, lastMinusPos);

  cout << "произведение: " << product;

  delete[] a;            //освобождение динамической памяти

  return 0;

}

void inputArr(tmas& a, int n) {

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

  cout << 2 \* n<< " чисел и нажмите <Enter>" << endl;

  for (int i = 0; i < 2 \* n; i++)

    cin >> \*(a + i);

}

int findAbsMaxPos(const tmas& a, int n) {

  telem maxEl = \*(a);

  int absMaxPos = 0;

  for (int i = 1; i < 2 \* n; i++)

    if (abs(\*(a + i)) > abs(maxEl)) {

      absMaxPos = i;

      maxEl = \*(a + i);

    }

  return absMaxPos;

}

int findMinusPos(const tmas& a, int n) {

  int lastMinusPos = -1;

  for (int i = n - 1; i > 0; i--) {

    if (\*(a + i) < 0) {

      lastMinusPos = i;

      break;

    }

  }

  return lastMinusPos;

}

telem findProduct(const tmas& a, int absMaxPos, int lastMinusPos) {

  telem product = 1;

  if (absMaxPos < lastMinusPos) {

    if (absMaxPos % 2 == 0) {

      for (int i = absMaxPos + 2; i < lastMinusPos; i += 2) {

        product \*= \*(a + i);

      }

    } else {

      for (int i = absMaxPos + 1; i < lastMinusPos; i += 2) {

        product \*= \*(a + i);

      }

    }

  } else if (lastMinusPos < absMaxPos) {

    if (lastMinusPos % 2 == 0) {

      for (int i = lastMinusPos + 2; i < absMaxPos; i += 2) {

        product \*= \*(a + i);

      }

    } else {

      for (int i = lastMinusPos + 1; i < absMaxPos; i += 2) {

        product \*= \*(a + i);

      }

    }

  }

  return product;

}

void outputArr(const tmas& a, int n) {

  for (int i = 0; i < 2 \* n; i++) {

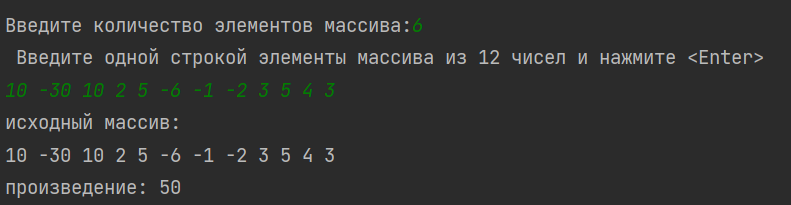
    cout << (\*(a + i)) << " ";

  }

  cout << endl;

}

***Скриншот результатов выполнения программы:***

******

***Б)***

*В заданной квадратной матрице размера 2n×2n поменять местами значения элементов строк области [5+6+7]  
 и элементов столбцов области [2+3+4] (см. рисунок) с одинаковыми номерами.*

***Математическая модель решения***

Решение задачи начинается с ввода исходных данных. Из формулировки задачи понятно, что исходная матрица квадратная с **чётным** числом (2*n*) строк и столбцов. Следовательно, прежде всего, необходимо ввести значение (*n*) половины размера квадратной 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 (нумерация строк и столбцов начинается с 0). Поэтому перебираем все строки (с 0-ой до 2*n*-1). Внутри каждой строки перебираем все столбцы (с 0-ого до 2*n*-1) и задаём значения элементов, лежащих на пересечении строки и столбца с текущими номерами, проверяя, что j(столбец) >= n – 1, чтобы не затрагивать области 1 и 8.

Пусть в качестве примера исходная матрица имеет вид (*n*=5):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n-1** | **n** |  |  | **2n-2** | **2n-1** |
| **0** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** |
|  | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** |
|  | **30** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** |
| **n-1** | **40** | **41** | **42** | **43** | **44** | **45** | **46** | **47** | **48** | **49** |
| **n** | **50** | **51** | **52** | **53** | **54** | **55** | **56** | **57** | **58** | **59** |
|  | **60** | **61** | **62** | **63** | **64** | **65** | **66** | **67** | **68** | **69** |
|  | **70** | **71** | **72** | **73** | **74** | **75** | **76** | **77** | **78** | **79** |
| **2n-2** | **80** | **81** | **82** | **83** | **84** | **85** | **86** | **87** | **88** | **89** |
| **2n-1** | **90** | **91** | **92** | **93** | **94** | **95** | **96** | **97** | **98** | **99** |

Теперь можно приступать к решению задачи. Требуется поменять местами значения элементов строк области [2+3+4] и значения элементов столбцов области [7+6+5] с одинаковыми номерами. Значения элементов 0-ой строки меняются со значениями 0-го столбца и т.д. При таком обмене значения элементов, принадлежащих главной диагонали, останутся на своём месте.

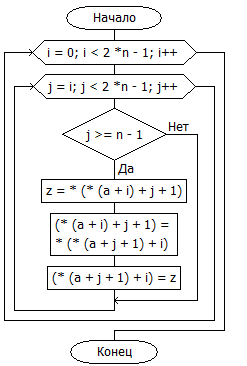
Поэтому в обмене участвуют значения элементов строк с 0 до 2*n*-2 и элементы столбцов с 0 до 2*n*-2, при этом мы выделяем область [1 + 8] и ее пропускаем. Используем метод «трёх стаканов». Перебираем номера строк с 0 до 2*n*-2. По очереди засылаем значения элементов текущей *i*-ой строки с индексами *a*[*i*][*j*+1] (где *j* – текущий номер элемента в строке *i*, который меняется с номера *i* до 2*n*-2) во вспомогательную переменную *z*. На их место записываем значения элементов столбца с индексами *a*[*j*+1][*i*], а затем на их место записываем значения из переменной *z*.

В результате такого обмена получаем преобразованную матрицу:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** |  |  | **n-1** | **n** |  |  | **2n-2** | **2n-1** |
| **0** | **0** | **1** | **2** | **3** | **4** | **50** | **60** | **70** | **80** | **90** |
| **1** | **10** | **11** | **12** | **13** | **14** | **51** | **61** | **71** | **81** | **91** |
|  | **20** | **21** | **22** | **23** | **24** | **52** | **62** | **72** | **82** | **92** |
|  | **30** | **31** | **32** | **33** | **34** | **53** | **63** | **73** | **83** | **93** |
| **n-1** | **40** | **41** | **42** | **43** | **44** | **54** | **64** | **74** | **84** | **94** |
| **n** | **5** | **15** | **25** | **35** | **45** | **55** | **65** | **75** | **85** | **95** |
|  | **6** | **16** | **26** | **36** | **46** | **56** | **66** | **76** | **86** | **96** |
|  | **7** | **17** | **27** | **37** | **47** | **57** | **67** | **77** | **87** | **97** |
| **2n-2** | **8** | **18** | **28** | **38** | **48** | **58** | **68** | **78** | **88** | **98** |
| **2n-1** | **9** | **19** | **29** | **39** | **49** | **59** | **69** | **79** | **89** | **99** |

Задача решена.

***Схема алгоритма обмена значений элементов строк области [2+3+4] и элементов столбцов области [7+6+5] (функция swapEl)***

******

***Текст программы:***

*/\**

*Б. В заданной квадратной матрице размера 2n×2n поменять местами значения элементов строк области [5+6+7]*

*и элементов столбцов области [2+3+4] (см. рисунок) с одинаковыми номерами.*

*\*/*

#include<iostream>

#include<locale.h>

using namespace std;

typedef int telem;        //определение типа элементов массива

typedef telem\* tstr;        //определение типа "указатель на telem"

typedef tstr\* tmatr;        //тип "указатель на указатель на telem"

void input\_matrix(tmatr a, int n);

void output\_matrix(tmatr a, int n);

void swapEl(tmatr a, int n);

int main() {

  int n;                  //число строк (столбцов) матрицы

  tmatr a;                //переменная-указатель на указатель на telem

  setlocale(LC\_ALL, "Russian");

  cout << "Введите размерность матрциы (матрица вида 2n\*2n)" << endl;

  cin >> n;

  a = new tstr[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 telem[2 \* n];    */\*каждому элементу массива указателей*

*на строки присваивается адрес начала*

*области памяти, выделяемой под строку\*/*

  input\_matrix(a, n);

  cout << " Исходная матрица:**\n**";

  output\_matrix(a, n);

  swapEl(a, n);

  cout << " Результирующая матрица:**\n**";

  output\_matrix(a, n);

  cout << endl << " Для завершения нажмите <Enter>";

  //Освобождение динамической памяти

  for (int i = 0; i < 2 \* n; i++)

    delete \*(a + i);

  delete[]a;

  return 0;

}

void input\_matrix(tmatr a, int n) {

  cout << " Значения элементов";

  cout << " массива размера " << n << "x" << n;

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

  for (int i = 0; i < 2 \* n; i++)

    for (int j = 0; j < 2 \* n; j++)

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

}

void output\_matrix(tmatr a, int n) {

  for (int i = 0; i < 2 \* n; i++) {

    for (int j = 0; j < 2 \* n; j++) {

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

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

    }

    cout << '**\n**';

  }

}

void swapEl(tmatr a, int n) {

  telem z;

  for (int i = 0; i < 2 \* n - 1; i++)

    for (int j = i; j < 2 \* n - 1; j++) {

      if (j >= n - 1) {

        z = \*(\*(a + i) + j + 1);

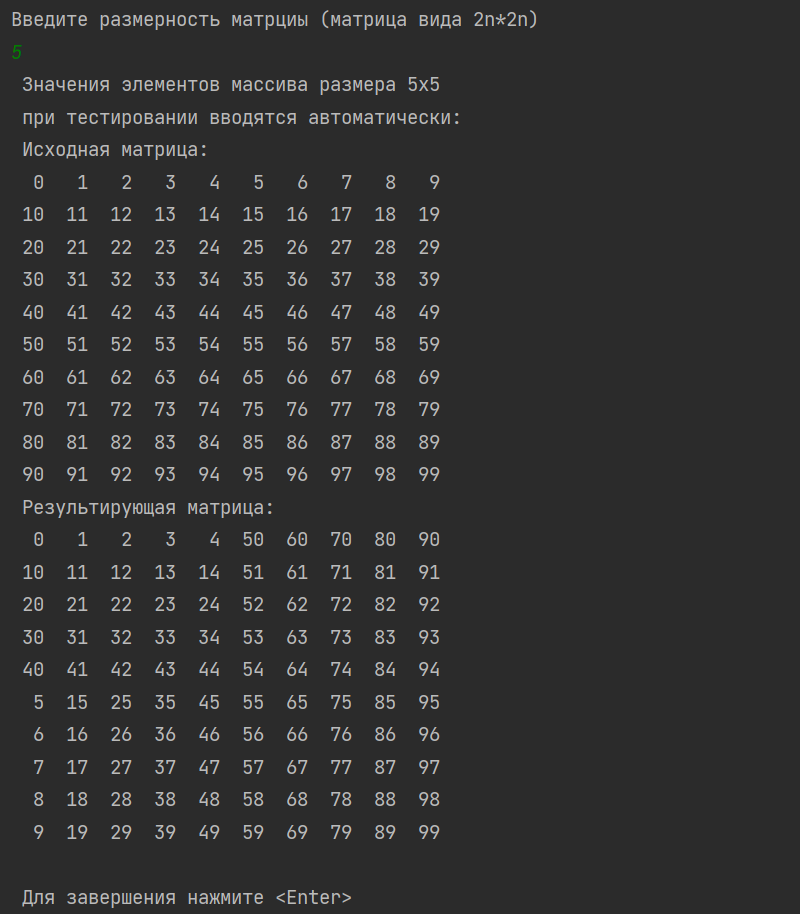
        \*(\*(a + i) + j + 1) = \*(\*(a + j + 1) + i);

        \*(\*(a + j + 1) + i) = z;

      }

    }

}

***Скриншот результатов выполнения программы:*** ******

***Вывод:*** *я**изучил структурную организацию динамических массивов и способов доступа к их элементам с использованием указателей; усовершенствовал навыки процедурного программирования на языке C/C++ при решении задач обработки динамических массивов.*