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 №10

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Методы сортировки»

Работу выполнил

студент гр. 4941 Н.С. Горбунов

Санкт-Петербург

2020

***Цель лабораторной работы:*** *изучение методов сортировки структур данных; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки статических и динамических массивов.*

***Задание на программирование:*** *используя технологию процедурного программирования реализовать заданный метод сортировки и применить его для указанных фрагментов числовой матрицы в соответствии с индивидуальным заданием.*

***Порядок выполнения работы:***

1. Получить у преподавателя индивидуальное задание: метод сортировки и вид сортируемых фрагментов матрицы. В зависимости от задания стрелка на рисунке указывает направление возрастания или убывания значений элементов фрагментов матрицы после сортировки.

2. Исходная матрица, содержащая 2\**n* строк и 2\**n* столбцов, не должна содержать элементов с одинаковыми значениями.

3. Разработать математическую модель: описать с помощью формул и рисунков структуру массива и процесс его преобразования. У результирующей матрицы должны быть отсортированы заданные фрагменты, а значения элементов не сортируемых фрагментов должны быть обнулены.

4. Построить схему алгоритма решения задачи.

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

6. 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**.

7. 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*().

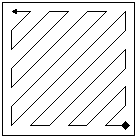
8. Проверить и продемонстрировать преподавателю работу программы на полном наборе тестов, в том числе с ошибочными входными данными. Входные и выходные массивы должны выводиться в одном и том же формате.

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

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

**Задание 7/39:**

Сортировка по возрастанию методом обмена с флагом перестановки.



**Математическая модель:**

Метод сортировки: идём по массиву чисел и проверяем порядок (следующее число должно быть больше и равно предыдущему), как только наткнулись на нарушение порядка, тут же обмениваем местами элементы, доходим до конца массива, после чего начинаем сначала. При этом не учитываем уже отсортированную часть массива. Введём переменную-флаг: он будет опущен (flag = 0), если массив отсортирован. Как только мы наткнёмся на нарушение порядка, то флаг будет поднят (flag = 1) и мы начнём сортировать массив как обычно.

Метод формирования ряда: Размер массива будет 4n2. Удобнее всего вести счет диагоналей от 0 до 2(2n – 1). Условия для первой половины матрицы (до побочной диагонали) отличаются от второй. Так же отличаются условия для четных и нечетных номеров диагоналей. Поэтому будет 4 условия:

1. До побочной диагонали, номер диагонали четный:

i меняется от 2 \* n - 1 – diag (где diag – номер диагонали) до 2 \* n (увеличивается)

j при этом от 2 \* n – 1 и уменьшается.

1. До побочной диагонали, номер диагонали нечетный:

i меняется от 2 \* n - 1 до 2 \* n - 1 – diag (уменьшается)

j при этом от 2 \* n - 1 – diag и увеличивается.

1. После побочной диагонали, номер диагонали четный:

i меняется от 0 до 2 \* (2 \* n - 1) - diag (увеличивается)

j при этом от 2 \* (2 \* n - 1) – diag и уменьшается.

1. После побочной диагонали, номер диагонали нечетный:

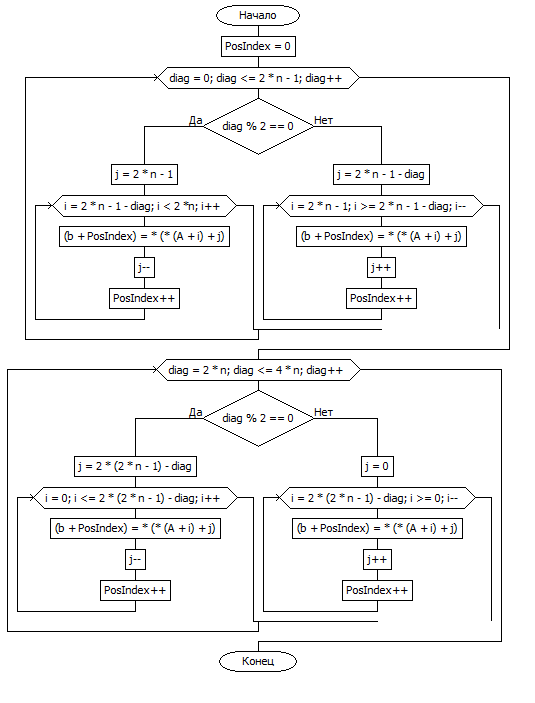
i меняется от 2 \* (2 \* n - 1) - diag до 0 (уменьшается)

j при этом от 0 и увеличивается.

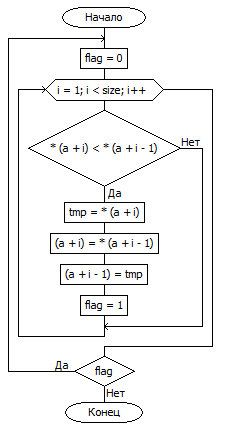
Сформированный ряд сортируется и по тому же алгоритму вставляется в изначальную матрицу. Задача решена.

**Блок-схемы:**

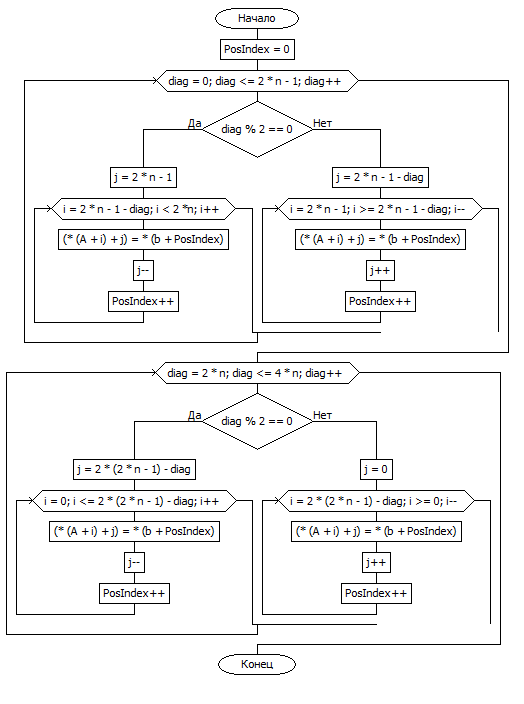
Функция CreateRange



Функция Sort



Функция PastRange



**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\_Matr(matr A, int N);

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

void CreateRange(matr A, int n, mas b);

void PasteRange(matr A, int n, mas b);

void Sort(mas a, int size);

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Программа сортировки матрицы по возрастанию методом обмена с флагом перестановки\n Введите n, учитывая, что размер матрицы 2n\*2n:\n";

int n;

cin >> n;

cout << "Введите матрицу построчно:";

matr A;

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); //ввод матрицы

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

//Создаем последовательность

mas pos = new elem[4 \*n \* n];

CreateRange(A, n, pos);

//

Sort(pos, 4 \* n \* n); //сортируем

PasteRange(A, n, pos); //вставляем

Output\_Matr(A, 2 \* n); //выводим матрицу.

}

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';

}

cout << '\n';

}

void Sort(mas a, int size) {

int i;

int tmp;

char flag;

do {

flag = 0;

for (i = 1; i < size; i++) {

if (\*(a + i) < \*(a+ i - 1)) {

tmp = \*(a + i);

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

\*(a + i - 1) = tmp;

flag = 1;

}

}

} while (flag);

}

void CreateRange(matr A, int n, mas b)

{

int PosIndex = 0;

//Обработка матрицы ДО побочной диагонали (включительно)

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

{

if (diag % 2 == 0)

{

int j = 2 \* n - 1;

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

{

\*(b + PosIndex) = \*(\*(A + i) + j);

j--;

PosIndex++;

}

}

else

{

int j = 2 \* n - 1 - diag;

for (int i = 2 \* n - 1; i >= 2 \* n - 1 - diag; i--)

{

\*(b + PosIndex) = \*(\*(A + i) + j);

j++;

PosIndex++;

}

}

}

//

//Обработка матрицы ПОСЛЕ побочной диагонали

for (int diag = 2 \* n; diag <= 4 \* n; diag++)

{

if (diag % 2 == 0)

{

int j = 2 \* (2 \* n - 1) - diag;

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

{

\*(b + PosIndex) = \*(\*(A + i) + j);

j--;

PosIndex++;

}

}

else

{

int j = 0;

for (int i = 2 \* (2 \* n - 1) - diag; i >= 0; i--)

{

\*(b + PosIndex) = \*(\*(A + i) + j);

j++;

PosIndex++;

}

}

}

//

}

void PasteRange(matr A, int n, mas b)

{

int PosIndex = 0;

//Обработка матрицы ДО побочной диагонали (включительно)

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

{

if (diag % 2 == 0)

{

int j = 2 \* n - 1;

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

{

\*(\*(A + i) + j) = \*(b + PosIndex);

j--;

PosIndex++;

}

}

else

{

int j = 2 \* n - 1 - diag;

for (int i = 2 \* n - 1; i >= 2 \* n - 1 - diag; i--)

{

\*(\*(A + i) + j) = \*(b + PosIndex);

j++;

PosIndex++;

}

}

}

//

//

for (int diag = 2 \* n; diag <= 4 \* n; diag++)

{

if (diag % 2 == 0)

{

int j = 2 \* (2 \* n - 1) - diag;

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

{

\*(\*(A + i) + j) = \*(b + PosIndex);

j--;

PosIndex++;

}

}

else

{

int j = 0;

for (int i = 2 \* (2 \* n - 1) - diag; i >= 0; i--)

{

\*(\*(A + i) + j) = \*(b + PosIndex);

j++;

PosIndex++;

}

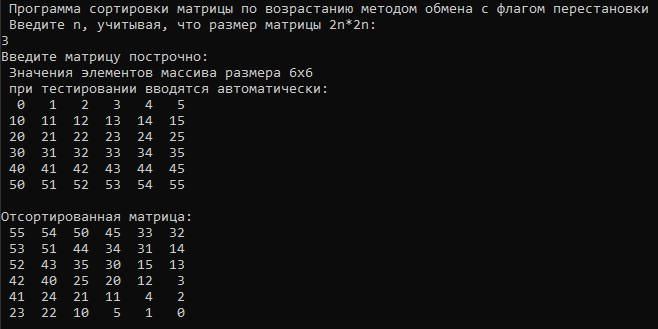
}

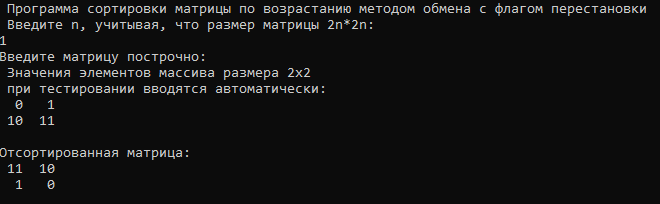
}

//

}

**Скриншоты работы программы**

****



**Вывод:** изучил методы сортировки структур данных; усовершенствовал навыков процедурного программирования на языке C/C++ при решении задач обработки статических и динамических массивов.