**Лабораторна робота №3**

**Розробка та реалізація програм для роботи з вказівниками та одновимірними динамічними масивами**

**Ціль роботи**: оволодіння навичками складання програм з використанням багатовимірних динамічних масивів

**Завдання**

**Завдання 3.1.** Нехай є наступний фрагмент програми. Поясніть, яку задачу реалізує вказаний фрагмент

*Розв’язання*

#include <iostream>

#include <stdlib.h>

#include <iomanip>

const int N = 5;

using namespace std;

//Формирует и возвращает пустую квадратную матрицу n x n

int\*\* form\_matr(int n) {

int \*\*matr = new int\*[n]; //n рядков

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

matr[i] = new int[n]; //n столбцов

}

return matr;

}

//Заполняет матрицу matr размером n x n

void zapolnen(int \*\*matr, int n) {

int p = N; //p = 5

//Заполняет матрицу последовательными числами с 5

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

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

matr[i][j] = p++;

}

}

}

//Выводим матрицу в консоль

void printArray(int \*\*matr, int n) {

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

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

cout << setw(5) << matr[i][j];

cout << "\n";

}

}

int main() {

int n;

n = 10 + N % 10; //10 + последняя цыфра варианта -> 15

int \*\*matr = form\_matr(n); //пустая матрица n x n

zapolnen(matr, n); //Заполнение матрицы

printArray(matr, n); //Вывод матрицы

int x = 0;

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

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

//x += matr[i][j];

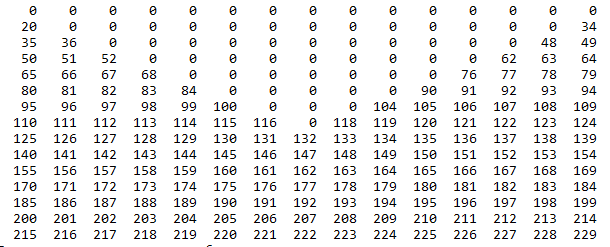
matr[i][j] = NULL;

}

}

. . .

Даний фрагмент програми знаходить сумму наступних елементів матриці:



Після виконання зміна x = 2868

**Завдання 3.2.** Скласти програму знаходження мінімального додатнього елементу в кожному стовпці матриці А.

*Розв’язання*

1. Постановка задачі:

Скласти програму знаходження мінімального додатнього елементу в кожному стовпці матриці А.

1. Алгоритм розв’язання задачі:

Алгоритм розв’язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Виділити динамічну пам'ять для елементів масиву.

Дія 2. Заповнити масив випадковими числами.

Дія 3. Вивести елементи масиву.

Дія 4. Знайти мінімальні елементі у кожному стовпці.

Дія 5. Вивести мінімальні елементи

Дія 6. Звільнити пам'ять, що була виділена під масив.

1. **Текст програми**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <cmath>

#include <ctime>

using namespace std;

int\*\* getArr(int k, int l);

void writeArr(int \*\*arr, int k, int l);

void printArr(int \*arr, int k, ofstream &fout);

void printArr(int \*\*arr, int k, int l, ofstream &fout);

int\* getMins(int \*\*arr, int k, int l);

void removeArr(int \*\*arr, int k, int l);

int main() {

ifstream fin("input.txt");

ofstream fout("output.txt");

srand(time(0));

int k, l; //k x l

fin >> k >> l;

fout << "Размерность матрицы: " << k << " x " << l << endl;

int \*\*matr = getArr(k, l);

writeArr(matr, k, l);

fout << endl << "Былa сгенерированa матрица: " << endl;

printArr(matr, k, l, fout);

int \*mins = getMins(matr, k, l);

fout << "Минимальные элементы на столбцах: " << endl;

printArr(mins, l, fout);

fin.close();

fout.close();

removeArr(matr, k, l);

return 0;

}

//Cоздает и возвращает двумерный массив размером k x l

//Массив изначально заполняется нулями

int\*\* getArr(int k, int l) {

int \*\*arr = new int \*[k];

for (int i = 0; i < k; i++) {

arr[i] = new int[l];

for (int j = 0; j < l; j++) {

arr[i][j] = 0;

}

}

return arr;

}

//Заполняет двумерный массив размером k x l случайными числами

void writeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

arr[i][j] = 10 + rand() % 90;

}

}

}

//Выводит одномерный массив в файл

void printArr(int \*arr, int k, ofstream &fout) {

for (int i = 0; i < k; i++) {

fout << setw(4) << arr[i];

}

fout << endl;

}

//Выводит двумерный массив в файнл

void printArr(int \*\*arr, int k, int l, ofstream &fout) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

fout << setw(4) << arr[i][j];

}

fout << endl;

}

fout << endl;

}

//Возвращает массив минимальных элементов каждого столбца

int\* getMins(int \*\*arr, int k, int l) {

int \*mins = new int[l];

for (int i = 0; i < l; i++) {

int min = arr[0][i];

for (int j = 1; j < k; j++) {

if (arr[j][i] < min) min = arr[j][i];

}

mins[i] = min;

}

return mins;

}

void removeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

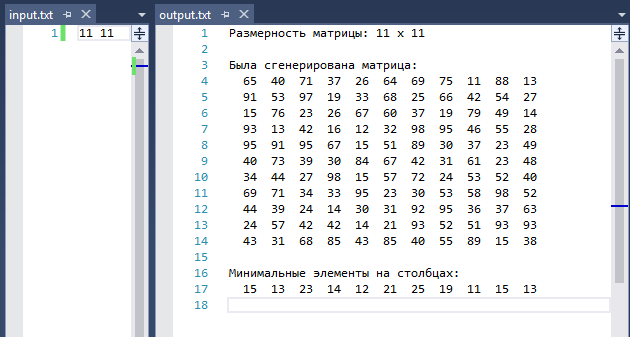
delete[] arr[i];

}

delete[] arr;

}

1. Результат виконання програми:



**Завдання 3.3:** Скласти програму, яка модифікує двовимірні динамічні масиви

1. Постановка задачі:

Скласти програму, яка модифікує двовимірні динамічн масиви (Додати стовпці в початок матриці).

1. Алгоритм розв’язання задачи

Дія 1. Виділити динамічну пам'ять для основного масиву.

Дія 2. Заповнити масив випадковими числами.

Дія 3. Вивести елементи масиву.

Дія 4. Повторити дії 1-3 для стровення додаткого масиву

Дія 5. Сформувати новый массив, утворенний з додаткового та основного масивів.

Дія 6. Звільнити пам'ять, що була виділена під основний та додатковий масиви

Дія 7. Вивести новий масив.

Дія 8. Звільнити пам’ять, що була виділена під новий масив.

1. Текст програми

#include <iostream>

#include <iomanip>

#include <fstream>

#include <cmath>

#include <ctime>

using namespace std;

int\*\* getArr(int k, int l);

void writeArr(int \*\*arr, int k, int l);

void printArr(int \*arr, int k, ofstream &fout);

void printArr(int \*\*arr, int k, int l, ofstream &fout);

void removeArr(int \*\*arr, int k, int l);

int\*\* concatArr(int \*\*matr, int \*\*other, int k, int l, int otherL, int &newL);

int main() {

ifstream fin("input.txt");

ofstream fout("output.txt");

srand(time(0));

//k x l - размерность первой матрицы

//otherL - количество столбцов второй

int k, l, otherL;

fin >> k >> l >> otherL;

fout << "Размерность основной матрицы: " << k << " x " << l << endl;

fout << "Размерность дополнительной матрицы: " << k << " x " << otherL << endl;

int \*\*matr = getArr(k, l);

writeArr(matr, k, l);

fout << endl << "Основная матрица: " << endl;

printArr(matr, k, l, fout);

int \*\*other = getArr(k, otherL);

writeArr(other, k, otherL);

fout << "Дополнительная матрица: " << endl;

printArr(other, k, otherL, fout);

int newL;

int \*\*result = concatArr(matr, other, k, l, otherL, newL);

removeArr(matr, k, l);

removeArr(other, k, otherL);

fout << "Итоговая матрица: " << endl;

printArr(result, k, newL, fout);

removeArr(result, k, newL);

fin.close();

fout.close();

return 0;

}

//Cоздает и возвращает двумерный массив размером k x l

//Массив изначально заполняется нулями

int\*\* getArr(int k, int l) {

int \*\*arr = new int \*[k];

for (int i = 0; i < k; i++) {

arr[i] = new int[l];

for (int j = 0; j < l; j++) {

arr[i][j] = 0;

}

}

return arr;

}

//Заполняет двумерный массив размером k x l случайными числами

void writeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

arr[i][j] = 10 + rand() % 90;

}

}

}

//Выводит одномерный массив в файл

void printArr(int \*arr, int k, ofstream &fout) {

for (int i = 0; i < k; i++) {

fout << setw(4) << arr[i];

}

fout << endl;

}

//Выводит двумерный массив в файнл

void printArr(int \*\*arr, int k, int l, ofstream &fout) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

fout << setw(4) << arr[i][j];

}

fout << endl;

}

fout << endl;

}

int\*\* concatArr(int \*\*matr, int \*\*other, int k, int l, int otherL, int &newL) {

int \*\*result = new int \*[k];

newL = l + otherL;

for (int i = 0; i < k; i++) {

result[i] = new int[newL];

for (int j = 0; j < otherL; j++) {

result[i][j] = other[i][j];

}

for (int j = 0; j < l; j++) {

result[i][otherL + j] = matr[i][j];

}

}

return result;

}

void removeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

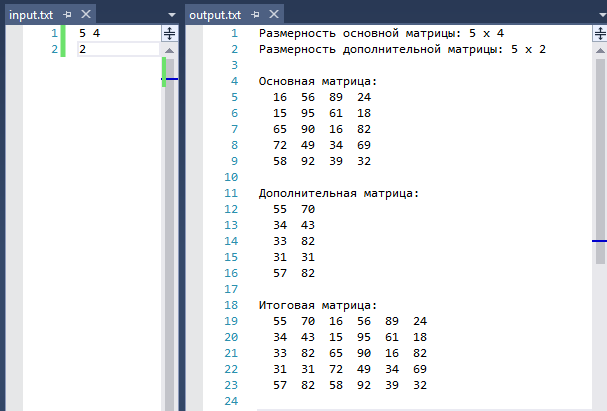
delete[] arr[i];

}

delete[] arr;

}

1. Результат виконання програми



**Завдання 3.4:** Розробити програму, яка заповнює двовимірний масив

наступним чином: елементи, що належать заштрихованій області, генеруються випадковим чином; всі інші дорівнюють N, де N – номер варіанту. Упорядкувати (за спаданням) всі випадкові елементи масиву, які належать заштрихованій області.

1. Постановка задачі:

Розробити програму, яка заповнює двовимірний масив

наступним чином: елементи, що належать заштрихованій області, генеруються випадковим чином; всі інші дорівнюють N, де N – номер варіанту. Упорядкувати (за спаданням) всі випадкові елементи масиву, які належать заштрихованій області.



1. Алгоритм розв’язання задачи

Дія 1. Виділити динамічну пам'ять для основного масиву.

Дія 2. Заповнити масив випадковими числами (на тих місцях, що входять в оласть, и номером варіанта в інших).

Дія 3. Вивести елементи масиву.

Дія 4. Отримати масив чисел, що знаходяться у області

Дія 5. Відсотрувати отриманний массив

Дія 6. Замінити числа в області на відсортовані числа з отриманого массиву

Дія 7. Звільнити пам’ять від отриманого массиву

Дія 8. Вивести кінцевий масив

Дія 9. Видалити кінцевий масив.

1. Текст програми

#include <iostream>

#include <iomanip>

#include <fstream>

#include <cmath>

#include <ctime>

using namespace std;

int\*\* getArr(int k, int l);

void writeArr(int \*\*arr, int k, int l);

void printArr(int \*\*arr, int k, int l, ofstream &fout);

void removeArr(int \*\*arr, int k, int l);

bool inFillArea(int k, int i, int j);

void quickSort(int \*arr, int l, int r);

void sortByTask(int \*\*matr, int k, int l);

int main() {

ifstream fin("input.txt");

ofstream fout("output.txt");

srand(time(0));

int k, l;

fin >> k;

l = k;

fout << "Размерность матрицы: " << k << " x " << l << endl;

int \*\*matr = getArr(k, l);

writeArr(matr, k, l);

sortByTask(matr, k, l);

fout << endl << "Была сгенерирована матрица: " << endl;

printArr(matr, k, l, fout);

removeArr(matr, k, l);

fin.close();

fout.close();

return 0;

}

void sortByTask(int \*\*matr, int k, int l) {

//Определение цифр, что находятся в fillArea

int \*nums = new int[k \* l / 2 + k];

int cur = 0;

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

if (inFillArea(k, i, j)) nums[cur++] = matr[i][j];

}

}

quickSort(nums, 0, cur - 1);

//Замена чисел отсортироваными

for (int i = 0, cur = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

if (inFillArea(k, i, j)) matr[i][j] = nums[cur++];

}

}

delete[] nums;

}

void quickSort(int \*arr, int l, int r) {

int i = l, j = r, middle = arr[(i + j) / 2];

do {

while (middle < arr[i]) i++;

while (middle > arr[j]) j--;

if (i <= j) {

int tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

i++; j--;

}

} while (i < j);

if (i < r) quickSort(arr, i, r);

if (j > l) quickSort(arr, l, j);

}

//Заполняет двумерный массив размером k x l случайными числами

void writeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

if (inFillArea(k, i, j)) {

arr[i][j] = 10 + rand() % 90;

}

else {

arr[i][j] = 5;

}

}

}

}

bool inFillArea(int k, int i, int j) {

return (i <= j && k - i - 1 >= j)

|| (i >= j && k - i - 1 <= j);

}

//Выводит двумерный массив в файнл

void printArr(int \*\*arr, int k, int l, ofstream &fout) {

for (int i = 0; i < k; i++) {

for (int j = 0; j < l; j++) {

fout << setw(3) << arr[i][j];

}

fout << endl;

}

fout << endl;

}

//Cоздает и возвращает двумерный массив размером k x l

//Массив изначально заполняется нулями

int\*\* getArr(int k, int l) {

int \*\*arr = new int \*[k];

for (int i = 0; i < k; i++) {

arr[i] = new int[l];

}

return arr;

}

void removeArr(int \*\*arr, int k, int l) {

for (int i = 0; i < k; i++) {

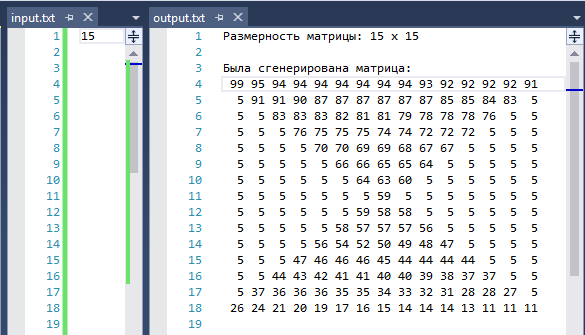
delete[] arr[i];

}

delete[] arr;

}

1. Результат виконання програми



**Висновок:** на цій лабораторній роботі я оволодів навичками складання програм з використанням багатовимірних динамічних масивів.