Лабораторная работа №5

Динамическое программирование

Задание:

Составить программы, решающие следующие задачи. Входные данные находятся в текстовом файле input.txt. В первой строке файла указаны через пробел размеры массива, каждый из которых не превосходит 20. В последующих строках перечислены элементы массива - целые числа. Результатом работы программы должны быть оптимальное значение целевой функции и путь, при котором оно достигается.

Для указания пути (кроме задачи 6) вывести массив, отметив элементы пути либо другим цветом, либо некоторым другим образом.

Дан двумерный числовой массив размером N1xN2. Найти такой путь из клетки [i1, j1] в клетку [i2, j2], чтобы сумма чисел по данному пути была минимальной. Из каждой клетки массива допустимо двигаться вправо, влево, вверх или вниз. В клетки с нулями заходить нельзя. Числа i1, i2, j1, j2 вводятся с клавиатуры.

Алгоритм:

Открытие файла input.txt для чтения. Если файл не удается открыть, выводится сообщение об ошибке, и программа завершает работу.

Чтение координат: запрашивает у пользователя начальные (i1, j1) и конечные (i2, j2) координаты для начала и конца пути, который нужно найти.

Чтение размеров массива: считываем размеры двумерного массива (N1, N2) из первой строки файла. Если размеры массива больше 20, выводится сообщение и программа завершается.

Инициализация двумерного массива data[N1][N2], который заполняется значениями из файла. Также происходит расчет общей суммы элементов массива.

Инициализация массивов для пути и сумм:

- Создается массив sums[N1][N2], который изначально заполняется очень большими значениями для хранения минимальных сумм.

- Массив way[N1][N2] инициализируется символами для обозначения направления движения по массиву ('<', '>', '^', 'v').

Запускается алгоритм, использующий флаг IsChanged, который указывает, была ли выполнена модификация массива сумм.

Программа проходит по каждой ячейке массива и проверяет соседей (верхнюю, нижнюю, левую, правую). Если найден более короткий путь, обновляется соответствующая ячейка в массиве sums и фиксируется направление в массиве way.

Поиск пути:

- Создается двумерный массив color[N1][N2], который помечает клетки, которые входят в минимальный путь.

- Начинается с конечной точки (i2, j2) и двигается обратно, следуя указаниям из массива way, пока не достигнет начальной координаты (i1, j1).

Происходит вывод всех элементов массива. Ячейки, которые входят в минимальный путь, отображаются другим цветом, используя функции Windows API для изменения цвета текста.

На экран выводится сумма элементов пути.

Текст программы:

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

int main() {

ifstream input("input.txt");

if (!input.is\_open()) {

cout << "Не удается открыть файл input.txt" << endl;

return 1;

}

// Чтение начальных и конечных координат

int i1, j1, i2, j2;

cout << "i1 = ";

cin >> i1;

cout << "j1 = ";

cin >> j1;

cout << "i2 = ";

cin >> i2;

cout << "j2 = ";

cin >> j2;

int N1, N2; //инициализация размера массива

input >> N1 >> N2;

if (N1 > 20 || N2 > 20) {

cout << "Размеры массива не должны превышать 20!" << endl;

return 1;

}

int sum = 1;

int\*\* data = new int\* [N1];

for (int i = 0; i < N1; i++) {

data[i] = new int[N2];

for (int j = 0; j < N2; j++) {

input >> data[i][j];

sum += data[i][j];

}

}

int\*\* sums = new int\* [N1];

for (int i = 0; i < N1; i++) {

sums[i] = new int[N2];

for (int j = 0; j < N2; j++) {

sums[i][j] = sum;

}

}

sums[i1][j1] = data[i1][j1]; // Начальная сумма

char\*\* way = new char\* [N1];

for (int i = 0; i < N1; i++) {

way[i] = new char[N2];

for (int j = 0; j < N2; j++) {

way[i][j] = '0';

}

}

bool IsChanged = true; //Флаг изменения (прошли ли вершину)

while (IsChanged == true) {

IsChanged = false;

for (int i = 0; i < N1; i++) {

for (int j = 0; j < N2; j++) {

if (!(i == i1 && j == j1) && data[i][j] != 0) {

if (j - 1 >= 0) {

if (data[i][j] + sums[i][j - 1] < sums[i][j]) { // Если соседняя левая клетка меньше sums

sums[i][j] = data[i][j] + sums[i][j - 1];

way[i][j] = '<';

IsChanged = true;

}

}

if (i - 1 >= 0) {

if (data[i][j] + sums[i - 1][j] < sums[i][j]) { // Если соседняя верхняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i - 1][j];

way[i][j] = '^';

IsChanged = true;

}

}

if (j + 1 < N2) {

if (data[i][j] + sums[i][j + 1] < sums[i][j]) { //если соседняя правая клетка меньше sums

sums[i][j] = data[i][j] + sums[i][j + 1];

way[i][j] = '>';

IsChanged = true;

}

}

if (i + 1 < N1) {

if (data[i][j] + sums[i + 1][j] < sums[i][j]) { // Если соседняя нижняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i + 1][j];

way[i][j] = 'v';

IsChanged = true;

}

}

}

}

}

}

bool\*\* color = new bool\* [N1]; // Массив цветов

for (int i = 0; i < N1; i++) {

color[i] = new bool[N2];

for (int j = 0; j < N2; j++) {

color[i][j] = false; // Пройден

}

}

int x = i2; // Элементы матрицы стрелочек (меток)

int y = j2;

while (x != i1 || y != j1) { // Идем обратно - ищем путь

color[x][y] = true;

if (way[x][y] == '<') {

y--;

}

else if (way[x][y] == '^') {

x--;

}

else if (way[x][y] == '>') {

y++;

}

else if (way[x][y] == 'v') {

x++;

}

}

color[x][y] = true;

int ko = 0;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

cout << endl;

for (int i = 0; i < N1; i++) {

for (int j = 0; j < N2; j++) {

if (color[i][j] == true) { // Если элемент тот, что нам нужен, то закрашиваем

SetConsoleTextAttribute(hConsole, 12);// Изменение цвета

cout << data[i][j] << " "; // Выводим закрашенный элемент

ko += data[i][j];

SetConsoleTextAttribute(hConsole, 7);// Изменение цвета в белый

}

else {//иначе просто выводим элемент клетки

cout << data[i][j] << " ";

}

}

cout << endl;

}

cout << "Sum: " << ko << endl << endl;

system("pause");

return 0;

}

Результат работы программы:

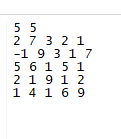
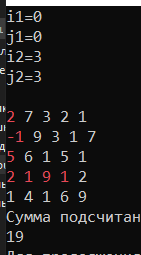


Рис. 1.

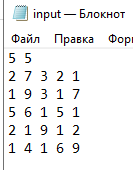
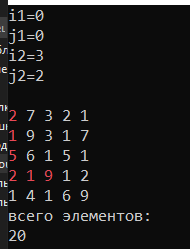


Рис. 2.

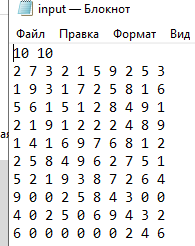
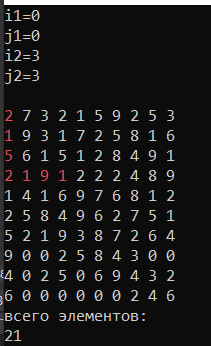


Рис. 3.