**Лабораторная работа №5**

**Динамическое программирование**

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.

1. Дан двумерный числовой массив размером N1xN2. Найти такой путь от левого столбца массива к правому, чтобы сумма чисел по данному пути была минимальной. Из каждой клетки массива допустимо двигаться вправо, вправо-вниз или вправо-вверх. Считать, что нижняя и верхняя строки массива «склеены», т.е из первой строки можно попасть в последнюю и наоборот, например, в массиве размером 8х8 из ячейки [1,1], двигаясь вправо-вверх, попадаем в ячейку [8,2] (нумерация строк и столбцов описана с единицы).

* Открыть файл «input.txt», прочитать размер массива – количество строк и столбцов.
* Создать 3 двумерных динамических массива: value – для хранения значений массива, bestValue – для хранения результатов сложения текущей суммы со стоимостью посещения ячейки, direction – для хранения пути, где «1» значит, что в ячейку попали переходом вправо-вверх, «0» – переходом вправо, «-1» – переходом вправо-вниз.
* Заполнить массив value, а массивы direction и bestValue – обнулить.
* Пока не последняя строка:
  + bestValue[i][0] += value[i][0]; – заполнение стартового столбца.
* Пока не последний столбец (i):
  + Пока не последняя строка: bestValue[i][j+1] = bestValue[i][j] + value[i][j+1];
  + Пока не последняя строка (j):
    - если i == 0
      * если bestValue[i][j] + value[n1-1][j+1] < bestValue[n1-1][j+1]
        + bestValue[n1-1][j+1]= bestValue[i][j]+ value[n1-1][j+1];
        + direction[n1-1][j+1] = 1;
    - если i != 0
      * если bestValue[i][j] + value[i-1][j+1] < bestValue[i-1][j+1]
        + bestValue[i-1][j+1] = bestValue[i][j] + value[i-1][j+1];
        + direction[i-1][j+1] = 1;
  + Пока не первая строка:
    - если i == n1 – 1
      * если bestValue[i][j] + value[0][j+1] < bestValue[0][j+1])
        + bestValue[0][j+1] = bestValue[i][j] + value[0][j+1];
        + direction[0][j+1] = -1;
    - если i != n1-1
      * если bestValue[i][j] + value[i+1][j+1] < bestValue[i+1][j+1])
        + bestValue[i+1][j+1] = bestValue[i][j] + value[i+1][j+1];
        + direction[i+1][j+1] = -1;
* Создать массив p размером в количество столбцов массива value, в последний элемент записать номер строки минимального значения последнего столбца массива bestValue.
* Пока не второй столбец
  + если direction[p[j]][j] == 1: p[j - 1] = p[j] + 1;
  + иначе если direction[p[j]][j] == 0: p[j - 1] = p[j];
  + иначе если direction[p[j]][j] == -1: p[j - 1] = p[j] - 1;
* Вывести массив value, выделив цветом ячейки, номер которых записан в p.

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

using namespace std;

void SetColor(int text, int bg) {

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

SetConsoleTextAttribute(hStdOut, (WORD)((bg << 4) | text));

}

int main() {

int n1, n2;

ifstream f("input.txt");

f >> n1 >> n2;//Чтение размера массива: строки-столбцы

cout << n1 << " " << n2 << endl;

int\*\* value = new int\* [n1];

int\*\* bestValue = new int\* [n1];

int\*\* direction = new int\* [n1];

for (int i = 0; i < n1; i++) {

bestValue[i] = new int[n2];

value[i] = new int[n2];

direction[i] = new int[n2];

for (int j = 0; j < n2; j++) {

bestValue[i][j] = 0;

f >> value[i][j];

direction[i][j] = 0;

}

}

for (int i = 0; i < n1; i++) bestValue[i][0] += value[i][0];//получили текущее значение столбца J при движении вправо

for (int j = 0; j < n2 - 1; j++) {//n2 столбцов

for (int i = 0; i < n1; i++)

bestValue[i][j + 1] = bestValue[i][j] + value[i][j + 1];//получили следующее значение столбца J при движении вправо

for (int i = 0; i < n1; i++) {//идём вправо-вверх

if (i == 0)

if (bestValue[i][j] + value[n1 - 1][j + 1] < bestValue[n1 - 1][j + 1]) {

bestValue[n1 - 1][j + 1] = bestValue[i][j] + value[n1 - 1][j + 1];

direction[n1 - 1][j + 1] = 1;

}

if (i != 0)

if (bestValue[i][j] + value[i - 1][j + 1] < bestValue[i - 1][j + 1]) {

bestValue[i - 1][j + 1] = bestValue[i][j] + value[i - 1][j + 1];

direction[i - 1][j + 1] = 1;

}

}

for (int i = n1 - 1; i >= 0; i--) {//идём вправо-вниз

if (i == n1 - 1)

if (bestValue[i][j] + value[0][j + 1] < bestValue[0][j + 1]) {

bestValue[0][j + 1] = bestValue[i][j] + value[0][j + 1];

direction[0][j + 1] = -1;

}

if (i != n1 - 1)

if (bestValue[i][j] + value[i + 1][j + 1] < bestValue[i + 1][j + 1]) {

bestValue[i + 1][j + 1] = bestValue[i][j] + value[i + 1][j + 1];

direction[i + 1][j + 1] = -1;

}

}

}

cout << endl;

int min;

int\* p = new int[n2];

min = bestValue[0][n2 - 1];

p[n2 - 1] = 0;

for (int i = 1; i < n1; i++) {

if (bestValue[i][n2 - 1] < min) {

min = bestValue[i][n2 - 1];

p[n2 - 1] = i;

}

}

for (int j = n2 - 1; j > 0; j--) {

p[j];

if (direction[p[j]][j] == 1) {

if (p[j] + 1 < n1)p[j - 1] = p[j] + 1;

else p[j - 1] = 0;

}

else if (direction[p[j]][j] == 0) {

p[j - 1] = p[j];

}

else if (direction[p[j]][j] == -1) {

if (p[j] - 1 > 0)p[j - 1] = p[j] - 1;

else p[j - 1] = n1-1;

}

}

int width = 5;

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n2; j++) {

cout.width(width);

if (p[j] == i) {

SetColor(4, 0);

cout << bestValue[i][j];

}

else {

SetColor(7, 0);

cout << bestValue[i][j];

}

}

cout << endl;

}

cout << endl;

SetColor(7, 0);

for (int i = 0; i < n1; i++) {

for (int j = 0; j < n2; j++) {

cout.width(width);

if (p[j] == i) {

SetColor(4, 0);

cout << value[i][j];

}

else {

SetColor(7, 0);

cout << value[i][j];

}

}

cout << endl;

}

cout << endl;

SetColor(7, 0);

f.close();

cout << endl;

system("pause");

return 0;

}

