**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.

**12 Вариант**

12. Дан двумерный числовой массив размером N1xN2, в котором имеются отрицательные числа. Найти такой путь от нижней строки массива к верхней, чтобы сумма чисел по данному пути была минимальной. Из каждой клетки массива допустимо двигаться либо через одну клетку вверх, либо на одну клетку влево-вверх или вправо-вверх.

Код программы:

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

using namespace std;

void setBlueTextAttribute() {

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

SetConsoleTextAttribute(handle, FOREGROUND\_INTENSITY | FOREGROUND\_BLUE);

}

int main() {

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

SetConsoleTextAttribute(handle, FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE);

setlocale(LC\_ALL, "rus");

ifstream file("input.txt");

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

cout << "Unable to open file!" << endl;

return 1;

}

int N1, N2;

file >> N1 >> N2;

if (N1 % 2 == 0 && N2 == 1) {

cout << "it is impossible to get to the top row of the array" << endl;

return 1;

}

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

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

floor[i] = new int[N2];

}

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

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

file >> floor[i][j];

}

}

file.close();

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

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

min[i] = new int[N2];

}

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

min[N1 - 1][j] = floor[N1 - 1][j];

}

for (int i = N1 - 2; i >= 0; --i) {

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

int down = (i < N1 - 2) ? min[i + 2][j] : INT\_MAX;

int left\_down = (i < N1 - 1 && j > 0) ? min[i + 1][j - 1] : INT\_MAX;

int right\_down = (i < N1 - 1 && j < N2 - 1) ? min[i + 1][j + 1] : INT\_MAX;

int min\_neighbor = min(down, min(left\_down, right\_down));

if (min\_neighbor == down) {

min[i][j] = down + floor[i][j];

}

else if (min\_neighbor == left\_down) {

min[i][j] = left\_down + floor[i][j];

}

else {

min[i][j] = right\_down + floor[i][j];

}

}

}

int minSum = min[0][0];

int endCol = 0;

for (int j = 1; j < N2; ++j) {

if (min[0][j] < minSum) {

minSum = min[0][j];

endCol = j;

}

}

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

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

dir[i] = new int[N2];

}

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

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

dir[i][j]=0;

}

}

int n1 = 0, n2 = endCol; // Начинаем снизу с финальной колонки

dir[n1][n2] = 1; // Помечаем начальную точку

while (n1 < N1 - 1) {

int down = (n1 < N1 - 2) ? min[n1 + 2][n2] : INT\_MAX;

int left\_down = (n2 > 0) ? min[n1 + 1][n2 - 1] : INT\_MAX;

int right\_down = (n2 < N2 - 1) ? min[n1 + 1][n2 + 1] : INT\_MAX;

if (down < left\_down && down < right\_down) {

n1+=2;

}

else if (left\_down < down && left\_down < right\_down) {

n1++;

n2--;

}

else if (right\_down < down && right\_down < left\_down) {

n1++;

n2++;

}

else {

n1+=2;

}

dir[n1][n2] = 1;

}

//вывод результатов с подсветочкой

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

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

setBlueTextAttribute();

if (dir[i][j] == 1) SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout.width(4);

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

}

cout << endl;

}

cout << endl;

setBlueTextAttribute();

cout << "Минимальная сумма пути: " << minSum << endl;

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

delete[] floor[i];

delete[] min[i];

delete[] dir[i];

}

delete[] floor;

delete[] min;

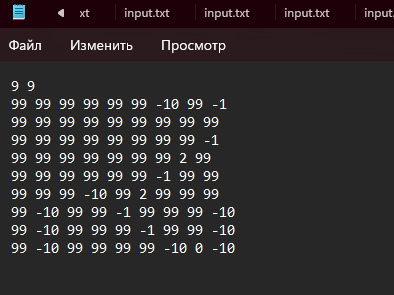
delete[] dir;

return 0;

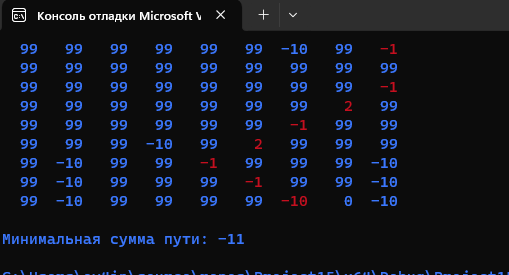
}

Тесты:

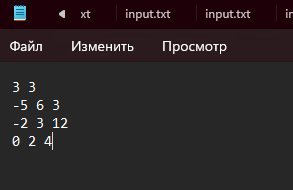
1. input.txt :



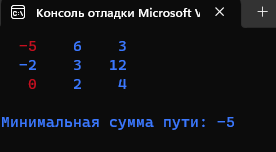
Результат:



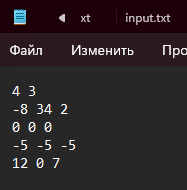
1. input.txt :



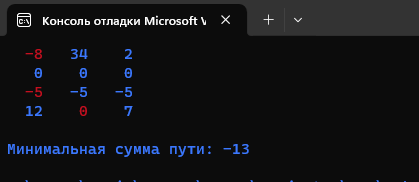
Результат:



1. input.txt :



Результат:



Алгоритм:

1. Инициализация и установка цветового атрибута:
   * Функция **setBlueTextAttribute()** используется для установки цвета текста в консоли на синий.
2. Открытие файла и чтение данных:
   * Открывается файл "input.txt" для чтения.
   * Если файл не открылся, выводится сообщение об ошибке, и программа завершает выполнение.
3. Чтение размеров и данных для матрицы:
   * Считываются значения **N1** и **N2** - размеры матрицы - из файла "input.txt". Если кол-во строк четное значение, а количество столбцов 1 программа завершается, и выводится оповещение, что в этом массиве невозможно добраться до верхней строки.
   * Создается двумерный массив **floor** размера **N1 x N2**.
   * Данные из файла "input.txt" считываются в созданный массив **floor**.
4. Инициализация и заполнение матрицы **min**:
   * Создается двумерный массив **min** такого же размера как **floor**.
   * Заполняется последняя строка массива **min** значениями из последней строки массива **floor**.
5. Нахождение минимальной суммы пути:

* Начиная с предпоследней строки и двигаясь вверх, программа вычисляет минимальные суммы путей для каждой клетки **(i, j)**:
  + Для текущей клетки **(i, j)** смотрит на три направления: вниз, вниз-влево и вниз-вправо.
  + Выбирает направление с минимальной суммой из предыдущих клеток и прибавляет значение текущей клетки **floor[i][j]**.
  + Эти минимальные суммы путей записываются в массив **min**.

1. Определение пути с минимальной суммой:

* Программа находит путь с минимальной суммой, двигаясь от нижней строки к верхней.
* Для каждой клетки **(i, j)** определяется направление движения:
  + Смотрит на три возможных соседа: вниз, вниз-влево и вниз-вправо.
  + Выбирает направление с наименьшей суммой пути и отмечает это направление в массиве **dir**.

7. Вывод пути с подсветкой:

* + Матрица **floor** выводится на экран.
  + Найденный путь с минимальной суммой подсвечивается красным цветом, используя функцию **SetConsoleTextAttribute()**.

8. Вывод минимальной суммы пути:

* + На экран выводится минимальная сумма пути.

9. Освобождение памяти:

* + Освобождается память, выделенная под динамические массивы **floor**, **min** и **dir**.

10. Завершение программы.