МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт искусственного интеллекта
Кафедра математического
обеспечения и стандартизации
информационных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2

Реализация структуры данных задачи на двумерном массиве
по дисциплине
«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

Выполнил студень группы ИНБО-02-21 Юдов С.А.

Принял старший преподаватель
кафедры МОСИТ Скворцова Л.А.

Практическая работа выполнена «__»_____2022г. _____

«Зачтено» «__»_____2022г. _____

Москва 2022

Содержание

Задание 1

1. Условие задачи и варианта

1.1. Дан статический массив из целых элементов.

1.1.1 Дана матрица размером n*m элементы которой заполнены цифрами от 0 до 9. Требуется найти такой путь из клетки (1,1) в клетку (n, m), чтобы сумма цифр в клетках, через которые он пролегает, была минимальной; из любой клетки ходить можно только вниз или вправо.

2. Разработка задачи

2.1. Постановка задачи.

2.1.1. Дано. Дан двумерный статический массив из n*m элементов целого типа int от 0 до 9

2.1.2. Результат. Путь из клетки (1,1) в клетку (n, m), чтобы сумма цифр в клетках, через которые он пролегает, была минимальной.

2.1.3. Ограничения. Из любой клетки ходить можно только вниз или вправо.

2.2. Описание модели решения

Модель решения заключается в создании нового массива такой же размерности, с последующим его наполнением. Сначала идёт проверка элемента с индексами от 1 до n, где n – это количество элементов в строке, при этом характеристику ответственную за номер строки оставляем константным. Такой же алгоритм проделываем со столбцами. Теперь нужно постепенно заполнять вторую матрицу. Для каждого элемента справедлив принцип: из какого (из левого или верхнего) нужно пойти, чтобы была минимальна конечная сумма. Так мы поступаем со всеми элементами и в конце имеем матрицу с обновлёнными значениями. В правом левом углу будет содержаться минимальная сумма, которую можно собрать.

2.3. Декомпозиция – список алгоритмов, которые требуются разработать в соответствии исследованной моделью

2.3.1. Список подзадач:

1. Отдельный алгоритм для первой строчки и левого столбца
2. Проверить, в какой ячейке минимальное число
3. «Движение» по массиву с помощью цикла
4. Вывод пути в консоль

2.3.2. Определение прототипов функций:

1) Заполнение массива значениями с помощью случайных значений

Предусловие. int x[n][m] – статичный массив, n > 0, m > 0

Постусловие. Заполненный массив из n элементов
void randomgenerate(int x[n][m], int n, int m)

2) Вывод значений массива

Предусловие. int x[n][m] – статичный массив, n > 0, m > 0
Постусловие. Вывод значений массива
void outputmass(int x[n][m], int n, int m)

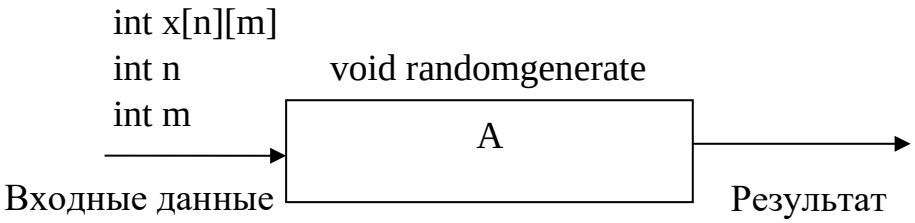
3. Декомпозиция

Задачу следует разбить на следующие подзадачи:

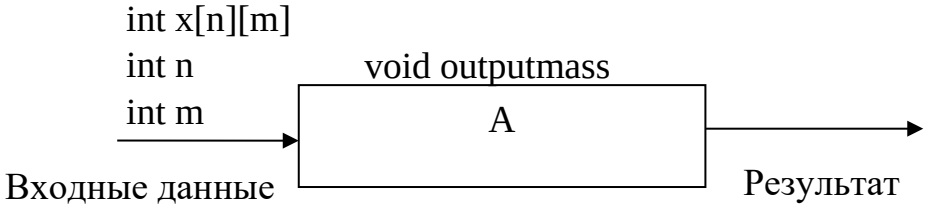
1. Создание нового двумерного массива для подсчёта значений оптимального пути
2. Подсчёт сначала верхней строки и левого столбца
3. Реализация оптимального прохода по новому массиву с прибавлением значений из старого
4. Поиск оптимального пути проходом через второй массив
5. Вывод пути в консоль.

4. Определение функций

4.1. Заполнение массива случайными цифрами:



4.2. Вывод массива в консоль



5. Реализация функций

```
void outputmass(int x[n][m], int n, int m) {
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < m; j++) {
         cout << setw(3) << x[i][j] << ' ';
      }
```

```
        cout << endl;
    }
    cout << endl;
}
void randomgenerate(int x[n][m], int n, int m) {
    srand(time(0));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            x[i][j] = rand() % 10;
        }
    }
```

6. Кодирование алгоритма программы

```
#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

const int n = 4, m = 4;

int main() {
    setlocale(LC_ALL, "ru_RU.UTF-8");
    int A[n][m];
    int B[n][m];
    string track;


    randomgenerate(A, n, m);
    outputmass(A, n, m);
    B[0][0] = A[0][0];

    for (int i = 1; i < n; i++) B[i][0] = B[i - 1][0] + A[i][0];
    for (int j = 1; j < m; j++) B[0][j] = B[0][j - 1] + A[0][j];

    for (int i = 1; i < n; i++) {
        for (int j = 1; j < m; j++) {
            B[i][j] = min(B[i - 1][j], B[i][j - 1]) + A[i][j];
        }
    }


    int i = n - 1;
    int j = m - 1;

    while (i != 0 || j != 0) {
```

```
        track = " -> A[" + to_string(i) + "][" + to_string(j) + "]" + track;
        if (i == 0) j--;
        else if (j == 0) i--;
        else if (min(B[i - 1][j], B[i][j - 1]) >= B[i - 1][j]) i--;
        else j--;
    }

    track = "A[0][0]" + track;
    cout << track;
    return 0;
}
```

7. Таблица тестов программы

Номер задачи	Исходные данные	Ожидаемый результат	Результат программы	Тест пройден/не пройден
1	n = 5 m = 5 int A[n][m]	A[0][0] -> A[0][1] -> A[1][1] -> A[2][1] -> A[3][1] -> A[3][2] -> A[3][3]	A[0][0] -> A[0][1] -> A[1][1] -> A[2][1] -> A[3][1] -> A[3][2] -> A[3][3]	Тест пройден
2	n = 4 m = 4 int A[n][m]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[1][3] -> A[2][3] -> A[3][3]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[1][3] -> A[2][3] -> A[3][3]	Тест пройден
3	n = 3 m = 3 int A[n][m]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[2][2]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[2][2]	Тест пройден

Задание 2

1. Условие задачи и варианта

1.1. Дан динамический массив из целых элементов.

1.1.1 Дана матрица размером n*m элементы которой заполнены цифрами от 0 до 9. Требуется найти такой путь из клетки (1,1) в

клетку (n, m), чтобы сумма цифр в клетках, через которые он пролегает, была минимальной; из любой клетки ходить можно только вниз или вправо.

2. Декомпозиция

Задачу следует разбить на следующие подзадачи:

2.1 Ввод значений количества строк и столбцов от пользователя

2.2 Создание нового динамического двумерного массива для подсчёта значений оптимального пути

2.3 Подсчёт сначала верхней строки и левого столбца

2.4 Реализация оптимального прохода по новому массиву с прибавлением значений из старого

2.5 Поиск оптимального пути проходом через второй массив

2.6 Вывод пути в консоль

2.7 Удаление динамических массивов из памяти

3. Реализация функций

```
void outputmass(int **x, int n, int m) {
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < m; j++) {
         cout << x[i][j] << ' ';
      }
      cout << endl;
   }
   cout << endl;
}

int **createmas(int n, int m) {
   int **A;
   A = new int *[n];
   for (int i = 0; i < n; i++) {
      A[i] = new int[m];
   }
   return A;
}

void deletemas(int **A, int n) {
   for (int i = 0; i < n; i++) {
      delete[] A[i];
   }
   delete[] A; // Освобождение памяти
}
```

```
 void randomgenerate(int **x, int n, int m) {
    srand(time(0));
    for (int i = 0; i < n; i++) {
       for (int j = 0; j < m; j++) {
          x[i][j] = rand() % 10;
       }
    }
 }
```

4. Кодирование алгоритма программы

```
#include <iostream>

using namespace std;

int main() {
   setlocale(LC_ALL, "ru_RU.UTF-8");
   int n, m;
   string track;
   cin >> n >> m;

   int **A = createmas(n, m);

   int **B = createmas(n, m);


   randomgenerate(A, n, m);
   outputmass(A, n, m);

   B[0][0] = A[0][0];

   for (int i = 1; i < n; i++) B[i][0] = B[i - 1][0] + A[i][0];
   for (int j = 1; j < m; j++) B[0][j] = B[0][j - 1] + A[0][j];

   for (int i = 1; i < n; i++) {
      for (int j = 1; j < m; j++) {
         B[i][j] = min(B[i - 1][j], B[i][j - 1]) + A[i][j];
      }
   }

   int i = n - 1;
   int j = m - 1;

   while (i != 0 || j != 0) {
```

```
        track = " -> A[" + to_string(i + 1) + "][" + to_string(j + 1) + "]" + track;
        if (i == 0) j--;
        else if (j == 0) i--;
        else if (min(B[i - 1][j], B[i][j - 1]) >= B[i - 1][j]) i--;
        else j--;
    }

    track = "A[1][1]" + track;

    cout << track;

    deletemas(A, n);
    deletemas(B, n);

    return 0;
}
```

5. Таблица тестов программы

Номер теста	Исходные данные	Ожидаемый результат	Результат программы	Тест пройден/не пройден
1	n = 5 m = 5 int A[n][m]	A[0][0] -> A[0][1] -> A[1][1] -> A[2][1] -> A[3][1] -> A[3][2] -> A[3][3]	A[0][0] -> A[0][1] -> A[1][1] -> A[2][1] -> A[3][1] -> A[3][2] -> A[3][3]	Тест пройден
2	n = 4 m = 4 int A[n][m]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[1][3] -> A[2][3] -> A[3][3]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[1][3] -> A[2][3] -> A[3][3]	Тест пройден
3	n = 3 m = 3 int A[n][m]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[2][2]	A[0][0] -> A[0][1] -> A[0][2] -> A[1][2] -> A[2][2]	Тест пройден

Задание 3

1. Условие задачи и варианта

1.1. Дан вектор из целых элементов.

1.1.1 Найти вертикальную медиану этого множества точек на плоскости в предположении, что никакие две точки не лежат на одной прямой.

2. Разработка задачи

2.1. Постановка задачи.

2.1.1. Дано. Множество точек на плоскости

2.1.2. Результат. Вертикальная медиана множества точек

2.1.3. Ограничения. Никакие две точки не лежат на одной прямой.

2.2. Описание модели решения

Модель решения заключается сначала в проверке двух любых точек на нахождение на одной прямой. Если таких точек не нашлось, то начинается поиск уравнения медианы и последующий вывод ответа.

3. Декомпозиция

Задачу следует разбить на следующие подзадачи:

2.1. Вывод вектора в консоль.

2.2. Заполнение вектора с указанием его длины с клавиатуры.

2.3. Перезапись координат m в новый одномерный вектор.

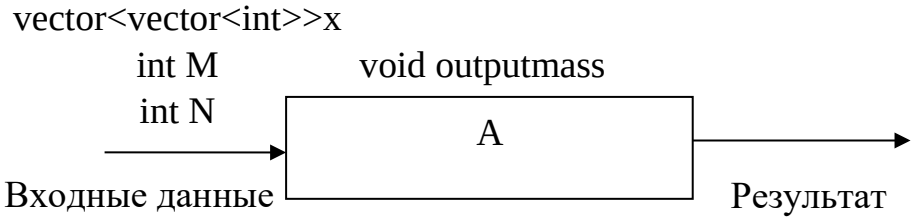
2.4. Перезапись координат n в новый одномерный вектор.

2.5. Проверка, лежат ли какие-либо 2 точки на одной прямой

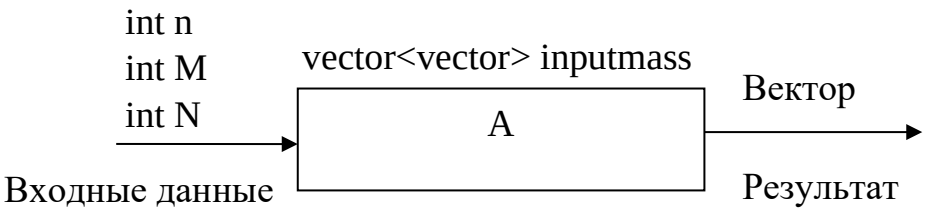
2.6. Поиск уравнения медианы.

4. Определение функций

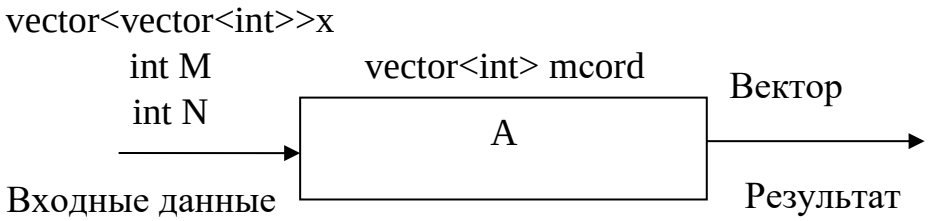
4.1. Вывод массива в консоль:



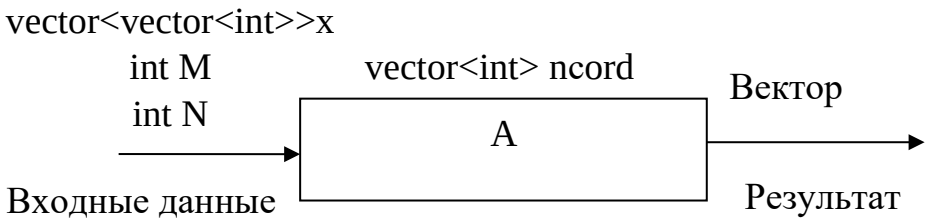
4.2. Заполнение вектора с указанием его длины с клавиатуры



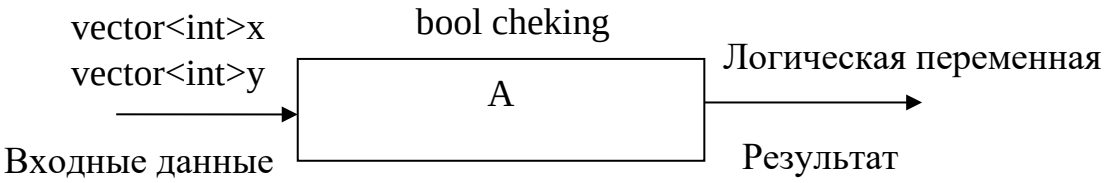
4.3. Перезапись координат m в новый одномерный вектор:



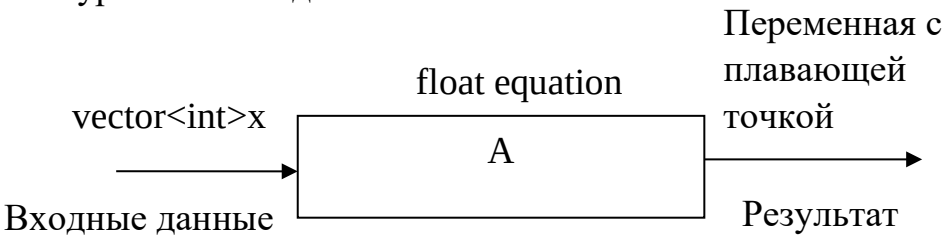
4.4. Перезапись координат n в новый одномерный вектор:



4.4. Проверка, лежат ли какие-либо 2 точки на одной прямой:



4.5. Поиск уравнения медианы:



5. Реализация функций

```
vector<vector<int>> inputmass(int n, int M, int N) {
   int y, x;
   vector<vector<int>> v(N, vector<int>(M, 0));
   for (int i = 0; i < n; i++) // заполнение поля точек
   {
      cout << "Введите координаты " << i + 1 << " точки\n ";
      cout << "Координата по x:";
      cin >> x;
      cout << "Координата по y:";
```

```
        cin >> y;
        vector<int> temp;
        v[x][y] = 1; // призваивание еденицы значению переменной на координате
    }
    return v;
}

void outputmass(vector<vector<int>> x, int M, int N) {
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            if (x[i][j] == 0) {
                cout << " ";
            } else {
                cout << "*";
            }
        }
        cout << endl;
    }
}

vector<int> mcord(vector<vector<int>> x, int M, int N) {
    vector<int> y;
    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            if (x[i][j] == 1) { // если текущий элемент это точка на координате
                y.push_back(i); // добавляем координату m в новый одномерный вектор
            }
    return y; // возврат нового одномерного вектора
}

vector<int> ncord(vector<vector<int>> x, int M, int N) {
    vector<int> y;
    for (int i = 0; i < M; i++) // цикл, идущий по строкам
        for (int j = 0; j < N; j++)
            if (x[i][j] == 1) { // если текущий элемент это точка на координате
                y.push_back(j); // добавляем координату n в новый одномерный вектор
            }
    return y; // возврат нового одномерного вектора
}

bool cheking(vector<int> x, vector<int> y) {
    bool flag = 0;
    for(int i = 0; i < (x.size() -1); i++)//Проверка, лежат ли 2 точки на одной прямой по координате X
        for (int j = i + 1; j < x.size(); j++) {
            if (x[i] == x[j]) {
                flag = 1;
                break;
            }
```

```
            if (flag == 1)
                break;
        }
    for (int i = 0; i < (y.size() - 1); i++) { // лежат ли 2 точки на одной прямой по координате Y
        for (int j = i + 1; j < y.size(); j++) {
            if (y[i] == y[j]) {
                flag = 1;
                break;
            }
            if (flag == 1)
                break;
        }
        return flag;
    }
}

float equation(vector<int> x) {
    float l = x.size() / 2.0;
    return (x[l - 1] + 1 + x[l] + 1) / 2.0; // вычисление медианы по заданной формуле
}
```

6. Кодирование алгоритма программы

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int main() {
    setlocale(LC_ALL, "ru_RU.UTF-8");

    int p_n, p_m;
    long n = 0;
    vector<vector<int>> vec;
    vector<int> vecm;
    vector<int> vecn;
    cout << "Введите количество точек в плоскости (кратное 2): ";
    cin >> n;
    if ((n <= 0 || n > 100) || (n%2==1)) // проверка ввода n
    {
        cout << "\nНеверный ввод n";
        return 1;
    }

    cout << "Введите размер поля(mxn), в пределах которого расположены точки\n";
    cout << "M = ";
    cin >> p_m;

    cout << "N = ";
```

```
    cin >> p_n;

    if (p_m <= 0 || p_m > 100 || p_n <= 0 || p_n > 100) // проверка ввода
    {
        cout << "\nНеверный ввод";
        return 1;
    }

    vec = inputmass(n, p_m, p_n);

    cout << "Вывод поля плоскости" << endl;
    outputmass(vec, p_m, p_n);

    vecm = mcord(vec, p_m, p_n);
    vecn = ncord(vec, p_m, p_n);

    if (cheking(vecn, vecm) == 0) {
        cout << "Вертикальная медиана имеет уравнение x = " << equation(vecm) << endl;
    } else {
        cout << "Какие-либо 2 точки лежат на одной горизонтальной или вертикальной прямой" <<
endl;
        cout << "Вертикальная медиана имеет уравнение x = " << equation(vecm) << endl;
    }

    return 0;
}
```

7. Таблица тестов программы

Номер теста	Исходные данные	Ожидаемый результат	Результат программы	Тест пройден/не пройден
1	problem=1 n = 2 p_m =2 p_n=2 x=0;1 y=0;1 problem=3	1.5	1.5	Тест пройден
2	p_n=6 x=0;3 y=4;1 problem=3	Какие либо 2 точки лежат на одной горизонтальной или вертикальной прямой	Какие либо 2 точки лежат на одной горизонтальной или вертикальной прямой	Тест пройден

3	n = 0	Неверный ввод	Неверный ввод	Тест пройден

Вывод

В ходе выполнения задания №2 мною были изучены на языке C++ такие структуры как статический двумерный массив, динамический двумерный массив. Наиболее важный опыт я приобрел при решении задачи 1 и 2, где пришлось изначально тщательно прорабатывать математическую модель решения задачи, а также структуру данных, с которой было бы эффективно и удобно работать.

Двумерный массив – мощный инструментарий, который при должном умении станет очень полезен в реализации многих математических задач

Список информационных источников

- Учебник по C++ http://www.lmpt.univ-tours.fr/~volkov/C++.pdf
- Документация по языку C++ https://docs.microsoft.com/ruru/cpp/?view=msvc-160