Министерство образования РФ

Государственное образовательное учреждение

высшего профессионального образования

“Тверской государственный университет”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет прикладной математики и кибернетики

**ОТЧЕТ ПО УЧЕБНО-ВЫЧИСЛИТЕЛЬНОЙ ПРАКТИКЕ**

Направление: «*Прикладная математика и информатика»*

Специализация: *Бакалавриат*

**Выполнил:**

студент 14 группы

Доронин Виталий Евгеньевич

**Проверил:**

Доцент кафедры информатики

Дадеркин Дмитрий Ольгердович

*Тверь – 2015*

# **Задание**

**Вариант 17**

**1**. Разработать программу для работы с БД склада, обеспечить добавление, удаление и изменение элементов.

**2**. Создать класс "матрица n, m 2", где n, m - количество строк и столбцов в матрице. Каждый элемент, кроме элементов в крайних столбцах, должен быть связан с соседними (звено 4-связного списка). Найти сумму двух матриц.

# Оглавление

[**Задание** 2](#_Toc515557292)

[Оглавление 3](#_Toc515557293)

[**№1** 4](#_Toc515557294)

[Описание алгоритма 4](#_Toc515557295)

[Описание классов 4](#_Toc515557296)

[Класс Warehouse: 4](#_Toc515557297)

[Класс Demand: 4](#_Toc515557298)

[Реализация методов, конструкторов и деструкторов 5](#_Toc515557299)

[Класс Warehouse: 5](#_Toc515557300)

[Класс Demand: 5](#_Toc515557301)

[Использование классов 6](#_Toc515557302)

[Проверка 6](#_Toc515557303)

[**№2** 8](#_Toc515557304)

[Описание алгоритма 8](#_Toc515557305)

[Описание классов 8](#_Toc515557306)

[Класс Node: 8](#_Toc515557307)

[Класс Matrix: 8](#_Toc515557308)

[Реализация методов, конструкторов и деструкторов 9](#_Toc515557309)

[Класс Node: 9](#_Toc515557310)

[Класс Matrix: 9](#_Toc515557311)

[Использование классов 11](#_Toc515557312)

[Проверка работоспособности 11](#_Toc515557313)

[**Приложения** 12](#_Toc515557314)

[Исходные тексты программ 12](#_Toc515557315)

[№1: 12](#_Toc515557316)

[№2: 20](#_Toc515557317)

[Контрольные примеры 23](#_Toc515557318)

[№1: 23](#_Toc515557319)

[№2: 23](#_Toc515557320)

[**Список использованной литературы** 23](#_Toc515557321)

# **№1**

## Описание алгоритма

Программа работает в бесконечном цикле, выводя на экран базу данных склада и расхода со всей необходимой информацией, а так же командную строку, позволяющую производить удаление, добавление, изменение и поиск элементов по цене, сумме, количеству или названию.

## Описание классов

### Класс Warehouse:

Класс «склад». Включает переменные *ware\_price, ware\_amount, ware\_sum* типа *integer, ware\_name* типа *string*,содержащие цену, количество, сумму и название товара. Класс включает в себя конструкторы и деструкторы с параметрами и без. Тип доступа – *public*.

**class** Warehouse

{

    **public**:

 string ware\_name;

 **int** ware\_price;

 **int** ware\_amount;

 **int** ware\_sum;

 Warehouse();

 Warehouse(string ware\_name, **int** ware\_price, **int** ware\_amount);

 ~Warehouse();

};

### Класс Demand:

Класс «потребление». Включает переменные *good\_date, ware\_price, ware\_amount, ware\_sum* типа *integer, good\_name, cust\_name* типа *string*,содержащие год отпуска, цену, количество, сумму, название товара и имя покупателя. Класс включает в себя конструкторы и деструкторы с параметрами и без. Тип доступа – *public*.

**class** Demand

{

    **public**:

 string cust\_name;

 string good\_name;

 **int** good\_date;

 **int** ware\_price;

 **int** ware\_amount;

 **int** ware\_sum;

 Demand();

 Demand(string cust\_name, string goos\_name, **int** good\_date,**int** ware\_price, **int** ware\_amount);

 ~Demand();

};

## Реализация методов, конструкторов и деструкторов

### Класс Warehouse:

Warehouse::Warehouse()

{

 ware\_name = "";

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = ware\_price \* ware\_price;

}

Warehouse::Warehouse(string name, **int** price, **int** amount)

{

 ware\_name = name;

 ware\_price = price;

 ware\_amount = amount;

 ware\_sum = amount \* price;

}

Warehouse::~Warehouse()

{

 ware\_name = "";

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = 0;

}

### Класс Demand:

Demand::Demand()

{

 cust\_name = "";

 good\_name = "";

 good\_date = 0;

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = ware\_price \* ware\_price;

}

Demand::Demand(string name, string g\_name, **int** date, **int** price, **int** amount)

{

 cust\_name = name;

 good\_name = g\_name;

 good\_date = date;

 ware\_price = price;

 ware\_amount = amount;

 ware\_sum = amount \* price;

}

Demand::~Demand()

{

 cust\_name = "";

 good\_name = "";

 good\_date = 0;

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = 0;

}

## Использование классов

В функции *main* для примера задается 3 экземпляра каждого класса.

std::vector <Warehouse> wares;

std::vector <Demand> goods;

 Warehouse example("Kamen", 335, 10);

 Warehouse example2("Hleb", 20, 100);

 Warehouse example3("Klei", 4, 20);

 Demand exmpl("Vitaly", "Hleb", 2015, 20, 5);

 Demand exmpl2("Vlad", "Kamen", 2013, 335, 1);

 Demand exmpl3("Alan", "Klei", 2012, 4, 3);

 wares.push\_back(example);

 wares.push\_back(example2);

 wares.push\_back(example3);

 goods.push\_back(exmpl);

 goods.push\_back(exmpl2);

 goods.push\_back(exmpl3);

Далее эти экземпляры обрабатываются в цикле.

## Проверка

При запуске программы с контрольным примером появляется окно консоли, где выводится база данных в табличном виде, а так же командная строка с инструкциями, позволяющая производить поиск, изменение, удаление и добавление элементов.



# **№2**

## Описание алгоритма

В программе имеется два класса: «элемент матрицы» и «матрица». Матрица формируется в соответствие с данными, заданными в файле (первые два значения – N и M, далее следуют числа-элементы), связывая каждый элемент с соседями посредством 4-х указателей, при поддержке двух вспомогательных.

Функция «сложение» находит сумму двух матриц посредством последовательного прохода по их элементам в цикле.

## Описание классов

### Класс Node:

Узел (элемент) матрицы. Включает в себя переменную *data* типа *integer*, указатели класса *Node* – *pRight*, *pLeft*, *pUp*, *pDown*. Класс включает в себя конструктор без/с параметрами и деструктор. Тип доступа – *public*.

**class** Node

{

**public**:

 **int** data;

 Node \* pRight;

 Node \* pLeft;

 Node \* pUp;

 Node \* pDown;

 Node();

 Node(**int** number);

 ~Node();

 };

### Класс Matrix:

Матрица. Включает в себя переменные *N* и *M* типа *integer* (строки и столбцы), указатель класса *Node* – *pFirst*, указывающий на верхний левый элемент матрицы. Класс включает в себя конструктор без/с параметрами, деструктор. Для выведения на экран готовой матрицы используется метод *ShowAll*. Тип доступа – *public*.

**class** Matrix

{

**public**:

 **int** N, M;

 Node \* pFirst;

 Matrix();

 Matrix(**char** \* file);

 ~Matrix();

 **void** ShowAll();

};

## Реализация методов, конструкторов и деструкторов

### Класс Node:

Node::Node()

{

 data = 1;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

Node::Node(**int** number)

{

 data = number;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

Node::~Node()

{

 data = 0;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

### Класс Matrix:

Matrix::Matrix()

{

 N = 0;

 M = 0;

 pFirst = NULL;

}

Matrix::Matrix(**char** \* file)

{

 Node \* curr = NULL, \* aux1 = NULL, \* aux2 = NULL, \* tempHead = NULL;

 curr = **new** Node;

 pFirst = NULL;

 ifstream inputFile(file);

 inputFile >> N >> M;

 **for**(**int** i = 0; i < M; i++)

 **for**(**int** j = 0; j < N; j++)

 {

 curr = **new** Node;

 inputFile >> curr->data;

 **if**(i == 0)

 {

 **if**(j == 0)

 {

 pFirst = curr;

 tempHead = curr;

 aux1 = curr;

 aux2 = curr;

 }

 **else**

 aux2->pRight = curr;

 curr->pLeft = aux2;

 aux2 = curr;

 }

 **else**

 {

 **if**(j == 0)

 {

 aux1->pDown = curr;

 curr->pUp = aux1;

 tempHead = curr;

 aux2 = curr;

 **if**(N == 1)

 aux1 = curr;

 **else**

 aux1 = aux1->pRight;

 }

 **else**

 {

 aux2->pRight = curr;

 aux1->pDown = curr;

 curr->pLeft = aux2;

 curr->pUp = aux1;

 aux2 = curr;

 **if**(j == N-1)

 aux1 = tempHead;

 **else**

 aux1 = aux1->pRight;

 }

 }

 }

}

Matrix::~Matrix()

{

 pFirst = NULL;

 N = 0;

 M = 0;

}

**void** Matrix::ShowAll()

{

 Node \* pCurr = pFirst, \* pHead = pFirst;

 **for**(**int** i = 0; i < M; i++)

 {

 **for**(**int** j = 0; j < N; j++)

 {

 std::cout << pCurr->data << " ";

 **if**(j == N-1)

 {

 pCurr = pHead->pDown;

 pHead = pCurr;

 std::cout << std::endl;

 }

 **else**

 pCurr = pCurr->pRight;

 }

 }

}

## Использование классов

В функции *main* инициализируется три экземпляра класса: две изначальные матрицы и их сумма.

**int** main()

{

 **char** \* input1 = "input1.txt";

 **char** \* input2 = "input2.txt";

 **char** \* add = "add.txt";

 Matrix matrix1(input1);

 Matrix matrix2(input2);

 matrix1.ShowAll();

 std::cout << endl;

 matrix2.ShowAll();

 std::cout << endl;

 Addition(matrix1, matrix2);

 Matrix Add(add);

 Add.ShowAll();

 cin.get();

 **return** 0;

}

## Проверка работоспособности

Работа программы на примере:



# **Приложения**

## Исходные тексты программ

### №1:

#include <iostream>

#include <vector>

#include <iomanip>

#include <string.h>

**using** **namespace** std;

**class** Warehouse

{

    **public**:

 string ware\_name;

 **int** ware\_price;

 **int** ware\_amount;

 **int** ware\_sum;

 Warehouse();

 Warehouse(string ware\_name, **int** ware\_price, **int** ware\_amount);

 ~Warehouse();

};

Warehouse::Warehouse()

{

 ware\_name = "";

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = ware\_price \* ware\_price;

}

Warehouse::Warehouse(string name, **int** price, **int** amount)

{

 ware\_name = name;

 ware\_price = price;

 ware\_amount = amount;

 ware\_sum = amount \* price;

}

Warehouse::~Warehouse()

{

 ware\_name = "";

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = 0;

}

**class** Demand

{

    **public**:

 string cust\_name;

 string good\_name;

 **int** good\_date;

 **int** ware\_price;

 **int** ware\_amount;

 **int** ware\_sum;

 Demand();

 Demand(string cust\_name, string goos\_name, **int** good\_date,**int** ware\_price, **int** ware\_amount);

 ~Demand();

};

Demand::Demand()

{

 cust\_name = "";

 good\_name = "";

 good\_date = 0;

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = ware\_price \* ware\_price;

}

Demand::Demand(string name, string g\_name, **int** date, **int** price, **int** amount)

{

 cust\_name = name;

 good\_name = g\_name;

 good\_date = date;

 ware\_price = price;

 ware\_amount = amount;

 ware\_sum = amount \* price;

}

Demand::~Demand()

{

 cust\_name = "";

 good\_name = "";

 good\_date = 0;

 ware\_price = 0;

 ware\_amount = 0;

 ware\_sum = 0;

}

**int** wares\_object\_search\_name(string name, std::vector <Warehouse> wares)

{

 **int** result = 0;

 **for**(std::vector<Warehouse>::iterator it = wares.begin(); it != wares.end(); ++it)

 {

 **if**((\*it).ware\_name == name)

 **return** result;

 **else**

 result++;

 }

 **return** -1;

}

**int** wares\_object\_search\_number(**int** number, std::vector <Warehouse> wares)

{

 **int** result = 0;

 **for**(std::vector<Warehouse>::iterator it = wares.begin(); it != wares.end(); ++it)

 {

 **if**((\*it).ware\_price == number **or** (\*it).ware\_amount == number **or** (\*it).ware\_sum == number)

 **return** result;

 **else**

 result++;

 }

 **return** -1;

}

**int** goods\_object\_search\_name(string name, std::vector <Demand> goods)

{

 **int** result = 0;

 **for**(std::vector<Demand>::iterator it = goods.begin(); it != goods.end(); ++it)

 {

 **if**((\*it).cust\_name == name **or** (\*it).good\_name == name)

 **return** result;

 **else**

 result++;

 }

 **return** -1;

}

**int** goods\_object\_search\_number(**int** number, std::vector <Demand> goods)

{

 **int** result = 0;

 **for**(std::vector<Demand>::iterator it = goods.begin(); it != goods.end(); ++it)

 {

 **if**((\*it).ware\_price == number **or** (\*it).ware\_amount == number **or** (\*it).ware\_sum == number **or** (\*it).good\_date == number)

 **return** result;

 **else**

 result++;

 }

 **return** -1;

}

**int** main()

{

 string temp;

 **int** number;

 std::vector <Warehouse> wares;

 std::vector <Demand> goods;

 Warehouse example("Kamen", 335, 10);

 Warehouse example2("Hleb", 20, 100);

 Warehouse example3("Klei", 4, 20);

 Demand exmpl("Vitaly", "Hleb", 2015, 20, 5);

 Demand exmpl2("Vlad", "Kamen", 2013, 335, 1);

 Demand exmpl3("Alan", "Klei", 2012, 4, 3);

 wares.push\_back(example);

 wares.push\_back(example2);

 wares.push\_back(example3);

 goods.push\_back(exmpl);

 goods.push\_back(exmpl2);

 goods.push\_back(exmpl3);

 **while**(**true**)

 {

 std::cout << "###########################################################################" << std::endl;

 std::cout << std::setw(20) << "Nazvanie" << std::setw(15) << "Cena" << std::setw(15) << "Kol-vo" << std::setw(15) << "Summa" << std::endl;

 std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << std::endl;

 **for**(std::vector<Warehouse>::iterator it = wares.begin(); it != wares.end(); ++it)

 {

 std::cout << std::setw(20) << (\*it).ware\_name << std::setw(15) << (\*it).ware\_price << std::setw(15) << (\*it).ware\_amount << std::setw(15) << (\*it).ware\_sum << std::endl;

 }

 std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

 std::cout << "D - udalit, C - izmenit, A - dobavit ";

 std::cin >> temp;

 **if**(temp == "D" **or** temp == "d")

 {

 string choice;

 std::cout << "N - po cene, summe ili kolichestvu , n - po nazvaniu ";

 std::cin >> choice;

 **if**(choice == "n")

 {

 string name;

 std::cout << "Kakoi element udalit? ";

 std::cin >> name;

 **if**(wares\_object\_search\_name(name, wares) == -1)

 {

 std::cout << "Ne naiden" << std::endl;

 std::cout << std::endl;

 }

 **else**

 wares.erase(wares.begin() + wares\_object\_search\_name(name, wares));

 }

 **else** **if**(choice == "N")

 {

 **int** code;

 std::cout << "Kakoi element udalit? ";

 std::cin >> code;

 **if**(wares\_object\_search\_number(code, wares) == -1)

 {

 std::cout << "Ne naiden" << std::endl;

 std::cout << std::endl;

 }

 **else**

 wares.erase(wares.begin() + wares\_object\_search\_number(code, wares));

 }

 }

 **if**(temp == "C" **or** temp == "c")

 {

 string choice, position;

 std::cout << "N - cena, kolichestvo ili summa, n - nazvanie ";

 std::cin >> choice;

 **if**(choice == "n")

 {

 **int** number;

 string name;

 std::cout << "Kakoi element izmenit? ";

 std::cin >> name;

 std::cout << std::endl;

 number = wares\_object\_search\_name(name, wares);

 **if**(number != -1)

 {

 std::cout << "P - cena, A - kolichestvo, N - nazvanie ";

 std::cin >> position;

 std::cout << std::endl;

 **if**(position == "p" **or** position == "P")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_price;

 }

 **if**(position == "N" **or** position == "n")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_name;

 }

 **if**(position == "a" **or** position == "A")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_amount;

 }

 }

 **else**

 std::cout << "Ne naiden" << std::endl;

 }

 **else** **if**(choice == "N")

 {

 **int** code;

 std::cout << "Kakoi element izmenit? ";

 std::cin >> code;

 std::cout << std::endl;

 number = wares\_object\_search\_number(code, wares);

 **if**(number != -1)

 {

 std::cout << "N - nazvanie, A - kolichestvo, P - cena ";

 std::cin >> position;

 std::cout << std::endl;

 **if**(position == "n" **or** position == "N")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_name;

 }

 **if**(position == "a" **or** position == "A")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_amount;

 }

 **if**(position == "p" **or** position == "P")

 {

 std::cout << "Izmenit na: ";

 std::cin >> wares[number].ware\_price;

 }

 }

 **else**

 std::cout << "Ne naiden" << std::endl;

 }

 }

 **if**(temp == "A" **or** temp == "a")

 {

 string new\_ware\_name;

 **int** new\_ware\_price;

 **int** new\_ware\_amount;

 std::cout << "Vvedite nazvanie tovara: ";

 std::cin >> new\_ware\_name;

 std::cout << "Vvedite cenu tovara: ";

 std::cin >> new\_ware\_price;

 std::cout << "Vvedite kol-vo tovara: ";

 std::cin >> new\_ware\_amount;

 Warehouse new\_ware(new\_ware\_name, new\_ware\_price, new\_ware\_amount);

 wares.push\_back(new\_ware);

 }

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 std::cout << "###########################################################################" << std::endl;

 std::cout << std::setw(15) << "Imya" << std::setw(15) << "Nazvanie" << std::setw(7) << "Data" << std::setw(10) << "Cena" << std::setw(15) << "Kol-vo" << std::setw(15) << "Summa" << std::endl;

 std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << std::endl;

 **for**(std::vector<Demand>::iterator it = goods.begin(); it != goods.end(); ++it)

 {

 std::cout << std::setw(15) << (\*it).cust\_name << std::setw(15) << (\*it).good\_name << std::setw(7) << (\*it).good\_date << std::setw(10) << (\*it).ware\_price << std::setw(15) << (\*it).ware\_amount << std::setw(15) << (\*it).ware\_sum << std::endl;

 }

 std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << std::endl;

 std::cout << "D - udalit, C - izmenit, A - dobavit ";

 std::cin >> temp;

 **if**(temp == "D" **or** temp == "d")

 {

 string choice;

 std::cout << "N - po cene, summe ili kolichestvu , n - po nazvaniu ";

 std::cin >> choice;

 **if**(choice == "n")

 {

 string name;

 std::cout << "Kakoi element udalit? ";

 std::cin >> name;

 **if**(goods\_object\_search\_name(name, goods) == -1)

 {

 std::cout << "Ne naiden" << std::endl;

 std::cout << std::endl;

 }

 **else**

 goods.erase(goods.begin() + goods\_object\_search\_name(name, goods));

 }

 **else** **if**(choice == "N")

 {

 **int** code;

 std::cout << "Kakoi element udalit? ";

 std::cin >> code;

 **if**(goods\_object\_search\_number(code, goods) == -1)

 {

 std::cout << "Ne naiden" << std::endl;

 std::cout << std::endl;

 }

 **else**

 {

 goods.erase(goods.begin() + goods\_object\_search\_number(code, goods));

 **break**;

 }

 }

 }

 **if**(temp == "C" **or** temp == "c")

 {

 string choice, position;

 std::cout << "N - cena, kolichestvo ili summa, n - nazvanie ";

 std::cin >> choice;

 **if**(choice == "n")

 {

 **int** number;

 string name;

 std::cout << "Kakoi element izmenit? ";

 std::cin >> name;

 std::cout << std::endl;

 number = goods\_object\_search\_name(name, goods);

 **if**(number != -1)

 {

 std::cout << "P - cena, A - kolichestvo, N - imya, n - nazvanie, D - data ";

 std::cin >> position;

 std::cout << std::endl;

 **if**(position == "p" **or** position == "P")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].ware\_price;

 }

 **if**(position == "N")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].cust\_name;

 }

 **if**(position == "a" **or** position == "A")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].ware\_amount;

 }

 **if**(position == "n")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].good\_name;

 }

 **if**(position == "D" **or** position == "d")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].good\_date;

 }

 }

 **else**

 std::cout << "Ne naiden" << std::endl;

 }

 **else** **if**(choice == "N")

 {

 **int** code;

 std::cout << "Kakoi element izmenit? ";

 std::cin >> code;

 std::cout << std::endl;

 number = goods\_object\_search\_number(code, goods);

 **if**(number != -1)

 {

 std::cout << "P - cena, A - kolichestvo, N - imya, n - nazvanie, D - data ";

 std::cin >> position;

 std::cout << std::endl;

 **if**(position == "p" **or** position == "P")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].ware\_price;

 }

 **if**(position == "N")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].cust\_name;

 }

 **if**(position == "a" **or** position == "A")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].ware\_amount;

 }

 **if**(position == "n")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].good\_name;

 }

 **if**(position == "D" **or** position == "d")

 {

 std::cout << "Izmenit na: ";

 std::cin >> goods[number].good\_date;

 }

 }

 **else**

 std::cout << "Ne naiden" << std::endl;

 }

 }

 **if**(temp == "A" **or** temp == "a")

 {

 string new\_good\_name;

 string new\_cust\_name;

 **int** new\_good\_date;

 **int** new\_ware\_price;

 **int** new\_ware\_amount;

 std::cout << "Vvedite imya: ";

 std::cin >> new\_cust\_name;

 std::cout << "Vvedite nazvanie tovara: ";

 std::cin >> new\_good\_name;

 std::cout << "Vvedite god: ";

 std::cin >> new\_good\_date;

 std::cout << "Vvedite cenu tovara: ";

 std::cin >> new\_ware\_price;

 std::cout << "Vvedite kol-vo tovara: ";

 std::cin >> new\_ware\_amount;

 Demand new\_good(new\_cust\_name, new\_cust\_name, new\_good\_date, new\_ware\_price, new\_ware\_amount);

 goods.push\_back(new\_good);

 }

 }

 **return** 0;

}

### №2:

#include <iostream>

#include <fstream>

**using** **namespace** std;

**class** Node

{

**public**:

 **int** data;

 Node \* pRight;

 Node \* pLeft;

 Node \* pUp;

 Node \* pDown;

 Node();

 Node(**int** number);

 ~Node();

};

Node::Node()

{

 data = 1;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

Node::Node(**int** number)

{

 data = number;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

Node::~Node()

{

 data = 0;

 pRight = NULL;

 pLeft = NULL;

 pUp = NULL;

 pDown = NULL;

}

**class** Matrix

{

**public**:

 **int** N, M;

 Node \* pFirst;

 Matrix();

 Matrix(**char** \* file);

 ~Matrix();

 **void** ShowAll();

};

Matrix::Matrix()

{

 N = 0;

 M = 0;

 pFirst = NULL;

}

Matrix::Matrix(**char** \* file)

{

 Node \* curr = NULL, \* aux1 = NULL, \* aux2 = NULL, \* tempHead = NULL;

 curr = **new** Node;

 pFirst = NULL;

 ifstream inputFile(file);

 inputFile >> N >> M;

 **for**(**int** i = 0; i < M; i++)

 **for**(**int** j = 0; j < N; j++)

 {

 curr = **new** Node;

 inputFile >> curr->data;

 **if**(i == 0)

 {

 **if**(j == 0)

 {

 pFirst = curr;

 tempHead = curr;

 aux1 = curr;

 aux2 = curr;

 }

 **else**

 aux2->pRight = curr;

 curr->pLeft = aux2;

 aux2 = curr;

 }

 **else**

 {

 **if**(j == 0)

 {

 aux1->pDown = curr;

 curr->pUp = aux1;

 tempHead = curr;

 aux2 = curr;

 **if**(N == 1)

 aux1 = curr;

 **else**

 aux1 = aux1->pRight;

 }

 **else**

 {

 aux2->pRight = curr;

 aux1->pDown = curr;

 curr->pLeft = aux2;

 curr->pUp = aux1;

 aux2 = curr;

 **if**(j == N-1)

 aux1 = tempHead;

 **else**

 aux1 = aux1->pRight;

 }

 }

 }

}

Matrix::~Matrix()

{

 pFirst = NULL;

 N = 0;

 M = 0;

}

**void** Matrix::ShowAll()

{

 Node \* pCurr = pFirst, \* pHead = pFirst;

 **for**(**int** i = 0; i < M; i++)

 {

 **for**(**int** j = 0; j < N; j++)

 {

 std::cout << pCurr->data << " ";

 **if**(j == N-1)

 {

 pCurr = pHead->pDown;

 pHead = pCurr;

 std::cout << std::endl;

 }

 **else**

 pCurr = pCurr->pRight;

 }

 }

}

**void** Addition(Matrix matrix1, Matrix matrix2)

{

 Node \* pCurr1 = matrix1.pFirst, \* pCurr2 = matrix2.pFirst;

 Node \* pHead1 = matrix1.pFirst, \* pHead2 = matrix2.pFirst;

 ofstream outputFile("add.txt");

 outputFile << matrix1.N << std::endl;

 outputFile << matrix1.M << std::endl;

 **for**(**int** i = 0; i < matrix1.M; i++)

 {

 **for**(**int** j = 0; j < matrix1.N; j++)

 {

 outputFile << (pCurr1->data + pCurr2->data) << " ";

 **if**(j == matrix1.N-1)

 {

 pCurr1 = pHead1->pDown;

 pHead1 = pCurr1;

 pCurr2 = pHead2->pDown;

 pHead2 = pCurr2;

 }

 **else**

 {

 pCurr1 = pCurr1->pRight;

 pCurr2 = pCurr2->pRight;

 }

 }

 }

}

**int** main()

{

 **char** \* input1 = "input1.txt";

 **char** \* input2 = "input2.txt";

 **char** \* add = "add.txt";

 Matrix matrix1(input1);

 Matrix matrix2(input2);

 matrix1.ShowAll();

 std::cout << endl;

 matrix2.ShowAll();

 std::cout << endl;

 Addition(matrix1, matrix2);

 Matrix Add(add);

 Add.ShowAll();

 cin.get();

 **return** 0;

}

## Контрольные примеры

### №1:

("Kamen", 335, 10);

("Hleb", 20, 100);

("Klei", 4, 20);

("Vitaly", "Hleb", 2015, 20, 5);

("Vlad", "Kamen", 2013, 335, 1);

("Alan", "Klei", 2012, 4, 3);

### №2:

2 2 1 1 – input1.txt

2 2 1 1 – input2.txt

**Список использованной литературы**

**Cplusplus.com [Электронный ресурс]: “std::settw function” -** http://www.cplusplus.com/reference/iomanip/setw/