**Типовой расчет (второй семестр) на тему  
«Моделирование типов. Ссылочные структуры»**

**Часть 1**

Смоделировать двумя способами новый тип «Матрица»:

1. на основе ссылочного типа «Нелинейный двусвязный список»
2. на основе типизированного/двоичного файла.

**Определение моделируемой структуры данных:**

**Матрица -** массивы элементов, представленные в виде прямоугольных таблиц.

**Для работы с матрицей надо реализовать несколько элементарных операций:**

- проверка на пустоту массива;

- создание пустого массива размера NxM (выделение памяти);

- добавление строки после последней;

- добавление столбца после последнего;

- очистить матрицу (освободить память);

- сделать текущим элементом первый элемент i-й строки;

- сделать текущим элементом первый элемент j-го столбца;

- сделать текущим следующий элемент в строке/столбце;

- чтение значения текущего элемента (без изменения текущей позиции);

- изменение значения текущего элемента (без изменения текущей позиции);

-\* изменение элемента по индексам (строки и столбца) (без изменения текущей позиции);

-\* чтение элемента по индексам (строки и столбца) (без изменения текущей позиции);

- вывод (дописывание) содержимого матрицы в текстовый лог-файл для контроля за состоянием матрицы после каждой операции во время работы программы.

А) на основе нелинейного двусвязного списка:

Матрица

(адрес)

А

Б

В

NIL

Г

Д

Е

NIL

Ё

Ж

З

NIL

NIL

NIL

NIL

//Модуль:

#pragma once

#include <stdio.h>

#include <Windows.h>

struct matr1

{

int a; // значение

matr1\* next; // следующий элемент

matr1\* under; // нижний элемент

int n, m;

};

bool checker1(matr1\*& M); // проверка на пустоту (TRUE - пусто)

void CreateEmpty1(matr1\*& M, int n, int m); // создание пустой матрицы размера nXm

void NewStroka1(matr1\*& M); // создание новой строки (выделение памяти под новую строку)

void NewStolb1(matr1\*& M); // создание нового столбца

int out1(matr1\*& M, FILE\*& log); // вывод в log файл

int cleaner1(matr1\*& M); // очистка (освобождение памяти)

void changeCur1(matr1\*& Cur, int a); // поменять значение текущего элемента

void MakeItI1(matr1\*& M, matr1\*& CurM, int a); // сделать текущим первый элемент строки под номером a

void MakeItJ1(matr1\*& M, matr1\*& CurM, int a); // сделать текущим первый элемент столбца под номером a

void NextElemStr1(matr1\*& M); // сделать текущим следующий элемент в строке

void NextElemStolb1(matr1\*& M); // сделать текущим первый элемент в столбце

int read1(matr1\*& M); // считать значение текущего элемента

int Index1(matr1\*& M, int n, int m, int a); // изменить значение элемента с индексами i и j на заданное значение а

int ReadMatr1(matr1\*& M, int n, int m); // считать значение элемента с индексами i и j

//реализация:

#include "Unit1.h"

bool checker1(matr1\*& M)

{

if (M == NULL)

{

return (TRUE); // матрица пуста

}

else

{

return (FALSE); // матрица не пуста

}

}

void CreateEmpty1(matr1\*& M, int n, int m)

{

matr1\* NewElem; // для создания нового элемента

matr1\* dop; // будет использоваться вместо M, чтобы не потерять первый элемент

dop = M;

matr1\* UpperString = NULL; // верхняя строка для заполнения ссылок на нижние элементы матрицы

matr1\* UpperStringDop = NULL; // дополнительный указатель на верхнюю строку

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

{

for (int j = 0; j < m; j++)

{

NewElem = new matr1;

NewElem->next = NULL;

NewElem->under = NULL;

if (i == 0 && j == 0) //создание самого самого первого элемента

{

M = NewElem;

}

if (j == 0)// создание первого элемента в строке и запоминание первого элемента в строке

{

UpperStringDop = NewElem; // запоминание первого элемента в строке

dop = NewElem; // типо создание первого элемента в строке

}

else // создание связей в строке

{

dop->next = NewElem;

dop = dop->next;

}

if (i != 0) // создание нижних связей в верхней строке

{

UpperString->under = NewElem; // создание самой связи

UpperString= UpperString->next; // переход к некст элементу верхней строки

}

}

UpperString = UpperStringDop; // присвоение первого элемента только что заполненной строки

}

M->n = n;

M->m = m;

}

int out1(matr1\*& M, FILE\*& log)

{

fprintf\_s(log, "/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\\nМатрица из списка:\n");

if (M == NULL) // проверка на пустоту

{

fprintf\_s(log, "матрица пуста\n");

return 0;

}

matr1\* elem; //дополнительный укзаатель для хождения по матрице

matr1\* FirstElem; // запоминает первый элемент в строке, чтобы при достижении конца строки перейти к началу следующей

elem = M;

FirstElem = elem;

while (elem!=NULL)

{

while (elem!= NULL)

{

if (elem->a == -842150451) // если была выделена память, то самих значений не будет (вывод вместо значений NULL)

{

fprintf\_s(log, " NULL ");

}

else

{

fprintf\_s(log, " %3d ", elem->a); // вывод значений в лог файл

}

elem = elem->next;

}

fprintf\_s(log, "\n");

elem = FirstElem->under; // переход к первому элементу некст строки

if (elem!= NULL) // если есть элемент некст строки, то снова запоминаем его

FirstElem = elem;

}

return 0;

}

void NewStroka1(matr1\*& M)

{

matr1\* UpperElem = M; // указатель на первый элемент верхней строки длля создания связей внзи

matr1\* elem = NULL; // указатель, который будет идти по последней строке

matr1\* NewElem = NULL;

while (UpperElem ->under != NULL) // спуск до последней строки

{

UpperElem = UpperElem->under;

}

while (UpperElem!= NULL) // содание нового строки

{

NewElem = new matr1;

NewElem->next = NULL;

NewElem->under = NULL;

UpperElem->under = NewElem;

if (elem == NULL) // создание первого элемента на новой строке

elem = NewElem;

else // если элемент не первый, то продолжаем связь

{

elem->next = NewElem;

elem = elem->next;

}

UpperElem = UpperElem->next;

}

M->n++;

}

void NewStolb1(matr1\*& M)

{

matr1\* elem = NULL; // идет по новому столбцу и делает связи вниз

matr1\* NewElem; // для создания новых элементов

matr1\* LastStolb = M; // последний существующий столбец шоб сделать связи next

while (LastStolb->next != NULL) // доходим до последнего столбца

{

LastStolb = LastStolb->next;

}

while (LastStolb != NULL)

{

NewElem = new matr1;

NewElem->next = NULL;

NewElem->under = NULL;

if (elem == NULL)

elem = NewElem;

else

{

elem->under = NewElem;

elem = elem->under;

}

LastStolb->next = NewElem;

LastStolb = LastStolb->under;

}

M->m++;

}

int cleaner1(matr1\*& M)

{

if (checker1(M) == TRUE)

{

printf\_s("\nматрица уже пуста\n\n");

return 0;

}

matr1\* NextStroka;

matr1\* dop;

while (M != NULL)

{

if (M->under != NULL)

NextStroka = M->under;

else

NextStroka = NULL;

while (M != NULL)

{

dop = M;

M = M->next;

delete dop;

}

M = NextStroka;

}

printf\_s("\nОчищено\n\n");

return 0;

}

void MakeItI1(matr1\*& M, matr1\*& CurM, int a)

{

matr1\* dop;

dop = M;

for (int i = 0; i < a - 1; i++) // спуск до нудной строки

{

dop = dop->under;

}

CurM = dop;

}

void MakeItJ1(matr1\*& M, matr1\*& CurM, int a)

{

matr1\* dop;

dop = M;

for (int i = 0; i < a - 1; i++) // до нужного столбца

{

dop = dop->next;

}

CurM = dop;

}

void NextElemStr1(matr1\*& M)

{

M = M->next;

}

void NextElemStolb1(matr1\*& M)

{

M = M->under;

}

int read1(matr1\*& M)

{

return M->a;

}

void changeCur1(matr1\*& Cur, int a)

{

Cur->a = a;

}

int Index1(matr1\*& M, int n, int m, int a)

{

matr1\* dop;

dop = M;

for (int i = 0; i < n - 1; i++) // спуск до нужной строки

{

dop = dop->under;

}

for (int i = 0; i < m - 1; i++) // переход до нужного столбца

{

dop = dop->next;

}

dop->a = a; // изменение значения

return 0;

}

int ReadMatr1(matr1\*& M, int n, int m)

{

matr1\* dop;

dop = M;

for (int i = 0; i < n - 1; i++) // спуск до нужной строки

{

dop = dop->under;

}

for (int i = 0; i < m - 1; i++) // переход до нужного столбца

{

dop = dop->next;

}

return dop->a;

}

Б) на основе типизированного/двоичного файла:

Б

А

…

В

Г

Д

…

Е

Ё

Ж

…

Элементы располагаются построчно:

З

Первая строка

Вторая строка

Третья строка

//модуль:

#pragma once

#include <stdio.h>

#include <Windows.h>

struct matr2 {

char name[20];

FILE\* f; // файл

int m, n;

};

bool checker2(matr2& M); // проверка на пустоту (TRUE - пусто)

int CreateEmpty2(matr2& M, int n, int m); // создание пустой матрицы размера nXm

void NewStroka2(matr2& M); // создание новой строки (выделение памяти под новую строку)

void NewStolb2(matr2& M); // создание нового столбца

int out2(matr2& M, FILE\*& log); // вывод в log файл

int cleaner2(matr2& M); // очистка (освобождение памяти)

void MakeItI2(matr2& M, int a); // сделать текущим первый элемент строки под номером a

void MakeItJ2(matr2& M, int a); // сделать текущим первый элемент столбца под номером a

void NextElemStr2(matr2& M); // сделать текущим следующий элемент в строке

void NextElemStolb2(matr2& M); // сделать текущим первый элемент в столбце

int read2(matr2& M); // считать значение текущего элемента

void ChangeCur2(matr2& M, int a); // поменять значение текущего элемента

void Index2(matr2& M, int n, int m, int a); // изменить значение элемента с индексами i и j на заданное значение а

int ReadMatr2(matr2& M, int n, int m); // считать значение элемента с индексами i и j //реализация:

#include "Unit2.h"

bool checker2(matr2& M)

{

if (M.f == NULL)

{

return TRUE;

}

int end;

fseek(M.f, 0, SEEK\_END);

end = ftell(M.f);

fseek(M.f, 0, SEEK\_SET);

if (end == 0)

{

return (TRUE); // файл пуст => нет элементов

}

else

{

return (FALSE);

}

}

int CreateEmpty2(matr2& M, int n, int m)

{

fopen\_s(&M.f, "MATR", "w+b");

if (&M.f == NULL)

{

printf\_s("Невозможно открыть файл\n");

return 0;

}

int a;

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

{

fwrite(&a, sizeof(int), m, M.f);

}

M.n = n;

M.m = m;

return 0;

}

void NewStroka2(matr2& M)

{

int a;

fseek(M.f, 0, SEEK\_END);

fwrite(&a, sizeof(int), M.m, M.f);

M.n++;

fseek(M.f, 0, SEEK\_SET);

}

void NewStolb2(matr2& M)

{

int a;

int dop;

fseek(M.f, 0, SEEK\_END); // выделяем место под новый столбец (например было 3 строки => нужно место под 3 новых переменных)

fseek(M.f, (M.n - 1) \* sizeof(int), SEEK\_CUR); // сдвигаем к последнему доп элементу

fwrite(&a, sizeof(int), 1, M.f); // записываем последний доп элемент (значение последнего столбца в последней строке)

fseek(M.f, 0 - (M.n + 1) \* sizeof(int), SEEK\_CUR); // сдвигаем до последнего значения, которое было значально для сдвига к последнему доп элементу (для матрицы 3Х3 это будет (3,3) элемент)

for (int i = 0; i < M.n - 1; i++) // идем построчно, начиная с последней строки

{

for (int j = 0; j < M.m; j++) // сдвигаем строку строки

{

fread(&dop, sizeof(int), 1, M.f); // считываем значение

//printf\_s("dop: %d\n", dop);

fseek(M.f, (M.n - 2 - i) \* sizeof(int), SEEK\_CUR); // перемещаем указатель на нужную позицию для этого элемента (для каждой строки переносится на своё значение ибо с каждой строкой появляется новый элемент)

fwrite(&dop, sizeof(int), 1, M.f); // запись элемента

fseek(M.f, 0 - (M.n + 1 - i) \* sizeof(int), SEEK\_CUR); // сдвигаемся назад до предшествующего элемента

}

fseek(M.f, (M.n - i - 1) \* sizeof(int), SEEK\_CUR); // сдвиг для записи доп элемента ( нового элемента в столбце)

//printf\_s("pos1: %d\n", ftell(M.f));

fwrite(&a, sizeof(int), 1, M.f); // запись этого доп элемента

//printf\_s("pos2: %d\n", ftell(M.f));

fseek(M.f, 0 - (M.n - i) \* sizeof(int), SEEK\_CUR); // сдвиг назад до последнего элемента верхней строки

//printf\_s("pos3: %d\n", ftell(M.f));

}

M.m++;

}

int out2(matr2& M, FILE\*& log)

{

int a;

fprintf\_s(log, "/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\\nМатрица из файла:\n");

if (M.f == NULL)

{

fprintf\_s(log, "файл пуст\n");

return 0;

}

fseek(M.f, 0, SEEK\_END);

if (ftell(M.f) == 0)

{

fprintf\_s(log, "файл пуст\n");

return 0;

}

fseek(M.f, 0, SEEK\_SET);

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

{

for (int j = 0; j < M.m; j++)

{

fread(&a, sizeof(int), 1, M.f);

fprintf\_s(log, " %3d ", a);

}

fprintf\_s(log, "\n");

}

return 0;

}

int cleaner2(matr2& M)

{

if (checker2(M) == TRUE)

{

printf\_s("\nматрица уже пуста\n\n");

return 0;

}

remove("Matr");

fopen\_s(&M.f, "Matr", "w+b");

printf\_s("\nочищено\n\n");

return 0;

}

void MakeItI2(matr2& M, int a)

{

fseek(M.f, (a - 1) \* M.m \* sizeof(int), SEEK\_SET);

}

void MakeItJ2(matr2& M, int a)

{

fseek(M.f, (a - 1) \* sizeof(int), SEEK\_SET);

}

void NextElemStr2(matr2& M)

{

fseek(M.f, sizeof(int), SEEK\_CUR);

}

void NextElemStolb2(matr2& M)

{

fseek(M.f, M.m \* sizeof(int), SEEK\_CUR);

}

int read2(matr2& M)

{

int a;

fread(&a, sizeof(int), 1, M.f);

fseek(M.f, 0 - sizeof(int), SEEK\_CUR);

return a;

}

void ChangeCur2(matr2& M, int a)

{

fwrite(&a, sizeof(int), 1, M.f);

fseek(M.f, 0 - sizeof(int), SEEK\_CUR);

}

void Index2(matr2& M, int n, int m, int a)

{

int CurPos = ftell(M.f); // запоминаем текущую позицию

fseek(M.f, ((n - 1) \* M.m + m - 1) \* sizeof(int), SEEK\_SET); // откатываем до позиции по индексам

fwrite(&a, sizeof(int), 1, M.f); // записываем туды значения a

fseek(M.f, CurPos, SEEK\_SET); // откатывает обратно в текущую позицию

}

int ReadMatr2(matr2& M, int n, int m)

{

int CurPos = ftell(M.f); // запоминаем позицию

fseek(M.f, ((n - 1) \* M.m + m - 1) \* sizeof(int), SEEK\_SET); // перемещаемся в нужную заданую позицию

int a;

fread(&a, sizeof(int), 1, M.f); // считываем значение

fseek(M.f, CurPos, SEEK\_SET); // откат обратно

return a;

}

**Часть 2.**

На основе описанных типов и базовых операций, решить следующие подзадачи:

- очистить матрицу (освободить память); ( MainCleaner(1/2) )

- создать матрицу, считав размеры и значения элементов из текстового файла;( MainInput(1/2) )

- изменить один элемент по индексам, на основе данных введенных в интерактивном режиме (с клавиатуры или с формы);( MainChangeIndex(1/2) )

- изменить размер матрицы, добавив строку или столбец из 0(нулей); ( MainStolb(1/2) ) или ( MainStroka(1/2) )

- поиск: решить задачу (Задан целочисленный двухмерный массив a из n строк и m столбцов. Найти номер первого минимального элемента среди положительных элементов, расположенных до первого элемента, кратного пяти. Матрицу рассматривать по строкам. В случае отсутствия положительных или кратных пяти элементов или невозможности поиска вывести соответствующие поясняющие сообщения.) ( MainSolution(1/2) )

**Интерфейс**

Для интерактивного взаимодействия с пользователем используется меню:

1 – очистить матрицу (освободить память)

2 – создать матрицу, считав размеры и значения элементов из текстового файла

3 – изменить один элемент по индексам, на основе данных введенных в интерактивном режиме

4 – изменить размер матрицы, добавив строку или столбец из 0(нулей)

5 – поиск: решить задачу

0 - Выход

Программный код для первого способа (через двусвязный список):

int MainCleaner1(matr1\*& M)

{

cleaner1(M);

return 0;

}

int MainInput1(matr1\*& M)

{

int n, m;

char name[20];

printf\_s("Введите название файла: ");

gets\_s(name);

FILE\* f;

fopen\_s(&f, name, "r");

if (f == NULL)

{

printf\_s("\nневозможно открыть файл\n\n");

return 0;

}

fseek(f, 0, SEEK\_END);

int end = ftell(f);

fseek(f, 0, SEEK\_SET);

if (end == 0)

{

printf\_s("\nФайл пуст\n\n");

return 0;

}

fscanf\_s(f, "%d", &n);

fscanf\_s(f, "%d", &m);

CreateEmpty1(M, n, m);

int a;

for (int i = 0; i <= n - 1; i++)

{

for (int j = 0; j <= m - 1; j++)

{

fscanf\_s(f, "%d", &a);

Index1(M, i + 1, j + 1, a);

}

}

fclose(f);

printf\_s("\nсоздали\n\n");

return 0;

}

int MainChangeIndex1(matr1\*& M)

{

if (checker1(M) == TRUE)

{

printf\_s("\nМатрица пуста => нечего менять\n\n");

return 0;

}

int i, j;

int a;

printf\_s("Введите значение i: ");

scanf\_s("%d", &i);

getchar();

printf\_s("Введите значение j: ");

scanf\_s("%d", &j);

getchar();

printf\_s("Введите значение a: ");

scanf\_s("%d", &a);

Index1(M, i, j, a);

printf\_s("\nизменили\n\n");

return 0;

}

int MainStolb1(matr1\*& M)

{

if (checker1(M) == TRUE)

{

printf\_s("\nматрица пуста => нельзя создать новый столбец\n\n");

return 0;

}

NewStolb1(M);

for (int i = 1; i <= M->n; i++)

{

Index1(M, i, M->m, 0);

}

printf\_s("\nстолбец создан\n\n");

return 0;

}

int MainStroka1(matr1\*& M)

{

if (checker1(M) == TRUE)

{

printf\_s("\nматрица пуста => нельзя создать новую строку\n\n");

return 0;

}

NewStroka1(M);

for (int i = 1; i <= M->m; i++)

{

Index1(M, M->n, i, 0);

}

printf\_s("\nстрока создана\n\n");

return 0;

}

int MainSolution1(matr1\*& M)

{

if (checker1(M) == TRUE)

{

//printf\_s("\nматрица пуста => нельзя найти решение\n\n");

return 0;

}

int schet = 0;

int minschet = 0;

int min = 999;

int a;

for (int i = 1; i <= M->n; i++)

{

for (int j = 1; j <= M->m; j++)

{

if (ReadMatr1(M, i, j) % 5 == 0)

{

return minschet;

}

schet++;

if ((ReadMatr1(M, i, j) >= 0) && (ReadMatr1(M, i, j) < min))

{

minschet = schet;

min = ReadMatr1(M, i, j);

}

}

}

return 0;

}

Программный код для второго способа (через типизированный файл):

void MainCleaner2(matr2& M)

{

cleaner2(M);

}

int MainInput2(matr2& M)

{

{

char name[20];

printf\_s("Введите название файла: ");

gets\_s(name);

FILE\* f;

fopen\_s(&f, name, "r");

if (f == NULL)

{

printf\_s("\nневозможно открыть файл\n\n");

return 0;

}

fseek(f, 0, SEEK\_END);

int end = ftell(f);

fseek(f, 0, SEEK\_SET);

if (end == 0)

{

printf\_s("\nФайл пуст\n\n");

return 0;

}

fscanf\_s(f, "%d", &M.n);

fscanf\_s(f, "%d", &M.m);

CreateEmpty2(M,M.n,M.m);

int a;

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

{

for (int j = 0; j < M.m; j++)

{

fscanf\_s(f, "%d", &a);

Index2(M, i + 1, j + 1, a);

}

}

printf\_s("\nсоздали\n\n");

fclose(f);

return 0;

}

}

int MainChangeIndex2(matr2& M)

{

if (checker2(M) == TRUE)

{

printf\_s("\nМатрица пуста => нечего менять\n\n");

return 0;

}

int i, j;

int a;

printf\_s("Введите значение i: ");

scanf\_s("%d", &i);

getchar();

printf\_s("Введите значение j: ");

scanf\_s("%d", &j);

getchar();

printf\_s("Введите значение a: ");

scanf\_s("%d", &a);

Index2(M, i, j, a);

printf\_s("\nизменили\n\n");

return 0;

}

int MainStolb2(matr2& M)

{

if (checker2(M) == TRUE)

{

printf\_s("\nфайл пуст => нельзя создать новый столбец\n\n");

return 0;

}

NewStolb2(M);

for (int i = 1; i <= M.n; i++)

{

Index2(M, i, M.m, 0);

}

printf\_s("\nстолбец создан\n\n");

return 0;

}

int MainStroka2(matr2& M)

{

if (checker2(M) == TRUE)

{

printf\_s("\nфайл пуст => нельзя создать новую строку\n\n");

return 0;

}

NewStroka2(M);

for (int i = 1; i <= M.n; i++)

{

Index2(M, M.n, i, 0);

}

printf\_s("\nстрока создана\n\n");

return 0;

}

int MainSolution2(matr2& M)

{

if (checker2(M) == TRUE)

{

//printf\_s("\nфайл пуст => нельзя найти решение\n\n");

return 0;

}

int schet = 0;

int minschet = 0;

int min = 999;

int a;

for (int i = 1; i <= M.n; i++)

{

for (int j = 1; j <= M.m; j++)

{

if (ReadMatr2(M, i, j) % 5 == 0)

{

return minschet;

}

schet++;

if ((ReadMatr2(M, i, j) >= 0) && (ReadMatr2(M, i, j) < min))

{

minschet = schet;

min = ReadMatr2(M, i, j);

}

}

}

return 0;

}

**Заключение**

Смоделирован новый тип «Матрица» двумя способами: 1) на основе двусвязного списка и 2) на основе типизированного файла.

1. Скорость работы: безусловно хранение данных в оперативной памяти, что происходит в случае со списком будет намного эффективнее хранения данных в типизированном файле т.е. на жестком диске.
2. Количество памяти: Хранение данных в списке занимает больше памяти, ведь помимо общих данных (значения и размеров матрицы), там будут храниться и ссылки на следующий и нижний элементы.

Размер 1 элемента списка: 20 байт

Размер 1 элемента типизированного файла:4 байт

///////////////////////////////////////////////

//Main:

#include "Unit1.h"

#include "Unit2.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

bool flag;

FILE\* log;

fopen\_s(&log, "log.txt", "w+");

matr1\* M1 = NULL; // первый элемент матрицы

matr2 M2;

M2.f = NULL;

int v = 9;

int k = 9;

int a;

char b[100];

while (v != 0)

{

printf\_s("1-очистить матрицу (освободить память)\n2-создать матрицу, считав размеры и значения элементов из текстового файла\n3-изменить один элемент по индексам, на основе данных введенных в интерактивном режиме\n4-изменить размер матрицы, добавив строку или столбец из 0(нулей)\n5-поиск: решить задачу\n0-выход");

printf\_s("\nвы выбираете: ");

scanf\_s("%d", &v);

if (v == 1)

{

k = 9;

printf\_s("\n1-список\n2-файл");

printf\_s("\nВы выбираете: ");

scanf\_s("%d", &k);

if ((k == 1))

{

MainCleaner1(M1);

out1(M1, log);

}

else if ((k == 2))

{

MainCleaner2(M2);

out2(M2, log);

}

else

{

gets\_s(b);

printf\_s("\nВы ввели неправильное значение\n\n");

}

}

else if (v == 2)

{

k = 9;

printf\_s("\n1-список\n2-файл");

printf\_s("\nВы выбираете: ");

scanf\_s("%d", &k);

if ((k == 1))

{

getchar();

MainInput1(M1);

out1(M1, log);

}

else if ((k == 2))

{

getchar();

MainInput2(M2);

out2(M2, log);

}

else

{

gets\_s(b);

printf\_s("\nВы ввели неправильное значение\n\n");

}

}

else if (v == 3)

{

k = 9;

printf\_s("\n1-список\n2-файл");

printf\_s("\nВы выбираете: ");

scanf\_s("%d", &k);

if ((k == 1))

{

getchar();

MainChangeIndex1(M1);

out1(M1, log);

}

else if ((k == 2))

{

getchar();

MainChangeIndex2(M2);

out2(M2, log);

}

else

{

gets\_s(b);

printf\_s("\nВы ввели неправильное значение\n\n");

}

}

else if (v == 4)

{

k = 9;

printf\_s("\n1-список столб\n2-файл столб\n3-список строка\n4-файл строка");

printf\_s("\nВы выбираете: ");

scanf\_s("%d", &k);

if ((k == 1))

{

getchar();

MainStolb1(M1);

out1(M1, log);

}

else if ((k == 2))

{

getchar();

MainStolb2(M2);

out2(M2, log);

}

else if ((k == 3))

{

getchar();

MainStroka1(M1);

out1(M1, log);

}

else if ((k == 4))

{

getchar();

MainStroka2(M2);

out2(M2, log);

}

else

{

gets\_s(b);

printf\_s("\nВы ввели неправильное значение\n\n");

}

}

else if (v == 5)

{

int j = 0;

k = 9;

printf\_s("\n1-список\n2-файл");

printf\_s("\nВы выбираете: ");

scanf\_s("%d", &k);

if ((k == 1))

{

getchar();

j = MainSolution1(M1);

if (j == 0)

{

printf\_s("\nНевозможно найти нужный элемент\n\n");

}

out1(M1, log);

if (j != 0)

{

printf\_s("\nзадача решена\n\n");

fprintf\_s(log, "номер элемента: %d\n", j);

}

}

else if ((k == 2))

{

getchar();

j = MainSolution2(M2);

if (j == 0)

{

printf\_s("\nНевозможно найти нужный элемент\n\n");

}

out2(M2, log);

if (j != 0)

{

printf\_s("\nзадача решена\n\n");

fprintf\_s(log, "номер элемента: %d\n", j);

}

}

else

{

gets\_s(b);

printf\_s("\nВы ввели неправильное значение\n\n");

}

}

else if (v == 0)

{

if (M2.f != NULL)

{

fclose(M2.f);

}

if (log != NULL)

{

fclose(log);

}

}

else

{

gets\_s(b);

printf\_s("\nвы ввели неправильное значение\n\n");

}

}

return 0;

}