Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

Типовой расчет на тему

**«Моделирование типов. Ссылочные структуры»**

Вариант 21

Задание выполнил: Павелко Серегй

Студент группы A-17-20

Проверил: Воробьёва И. А.

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Замечания:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Москва 2021 НИУ «МЭИ»

**Задание Д – «МАТРИЦА»**

**Часть Д1.**

**Смоделировать двумя способами** новый тип «Матрица»:

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

**Описание базовых операций**

**bool Check(matr\*& M); // проверка на пустоту (TRUE - пусто)**

**void CreateEmpty(matr\*& M, int n, int m); // создание пустого массива размера NxM (выделение памяти)**

**int FreeMem(matr\*& M); // очистка (освобождение памяти)**

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

**void NextElemStolb1(matr\*& M); // сделать текущим следующим элемент в столбце**

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

**void NewStolb(matr\*& M); // создание нового столбца**

**int ReadCurrent(matr\*& M); // считать значение текущего элемента**

**void IznpozCurrent(matr\*& M, int a, int b); //помять позицию текущего на a u b только для 1 способа**

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

**void ChangeCar(matr\*& M, int b); // поменять значение текущего элемента**

**void MakeItI(matr\*& M, int a); //сделать текущим первый элемент строки под номером a**

**void MakeItJ(matr\*& M, int b); // сделать текущим первый элемент столбца под номером a**

|  |  |
| --- | --- |
| **Описание структуры нового типа «Матрица»** | |
| **Способ 1**  на основе ссылочного типа «Нелинейный двусвязный список» | **Способ 2**  на основе типизированного/двоичного файла |
| struct TElem{  int a;  TElem\* next;  TElem\* down;  };  struct matr {  TElem\* first; // первый элемент  TElem\* suk; // текущий элемент  int n, m, i, j;  }; | struct matr {  char name[20];  FILE\* f; // файл  int m, n;  }; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| **bool Check(matr\*& M) {**  if (M->first == NULL) {  return (TRUE); // матрица пуста  }  else {  return (FALSE); // матрица не пуста  }  } | **bool Check(matr& M) {**  {  if (M.f == NULL)  {  return TRUE;  }  int end1;  fseek(M.f, 0, SEEK\_END);  end1 = ftell(M.f);  fseek(M.f, 0, SEEK\_SET);  int end2 = ftell(M.f);  if (end1 == end2)  {  return (TRUE); // файл пуст => нет элементов  }  else  {return (FALSE);}  } |
| **void CreateEmpty(matr\*& M, int n, int m)**  **{**  TElem \* NewElem; //TElem для создания нового элемента  TElem\* UpperString = NULL; // верхняя строка для заполнения ссылок на нижние элементы матрицы  TElem\* UpperStringDop = NULL; // дополнительный указатель на верхнюю строку  for (int i = 0; i < n; i++) {  for (int j = 0; j < m; j++) {  NewElem = new TElem;// выделение памяти  NewElem->next = NULL;  NewElem->down = NULL;  if (i == 0 && j == 0) //запоминание первого элемнта  {  M -> first = NewElem;  }  if (j == 0)// создание первого элемента в строке и запоминание первого элемента в строке  {  UpperStringDop = NewElem; // запоминание первого элемента в строке  M->suk = NewElem; // тепрь это текущий элмент типо  }  else // создание связей в строке  { M->suk->next = NewElem;//создаёмс связь со след  M->suk = M->suk->next;// говорим что тек теперь следующий  }  if (i != 0) // создание нижних связей в верхней строке  { UpperString->down = 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;  M-> suk = M->first;  M->j = 0;  M->i = 0;} | **int CreateEmpty**(matr& M, int n, int m) {  M.f=fopen( "MATR", "w+b");  if (M.f == NULL)  {  printf("Невозможно открыть файл\n");  return 0;  }  int a = 0;  for (int i = 0; i < n; i++)  {  fwrite(&a, sizeof(int), m, M.f);  }  M.n = n;  M.m = m;  return 0;  } |
| void NextElemStr(matr\*& M) {  M->suk = M->suk->next;  M->j++;  } | void NextElemStr(matr& M)  {  fseek(M.f, sizeof(int), SEEK\_CUR);  } |
| void NextElemStolb(matr\*& M) {  M->suk = M->suk->down;  M->i++;  } | void NextElemStolb(matr& M)  {  fseek(M.f, M.m \* sizeof(int), SEEK\_CUR);  } |
| void MakeItI(matr\*& M, int a) {// ; текущий - 1 элемент строки а  M-> suk = M-> first;  for (int i = 0; i < a - 1; i++) // спуск до нудной строки  { M-> suk = M-> suk-> down;  }  M->i = a - 1;  M->j = 0;  } | void MakeItI(matr& M, int a)  {  fseek(M.f, (a - 1) \* M.m \* sizeof(int), SEEK\_SET);  } |
| void MakeItJ(matr\*& M, int b) {// ;текущий - 1 элемент столба б  M-> suk = M-> first;  for (int i = 0; i < b - 1; i++) // a заменить на b до нужного столбца  { M-> suk = M-> suk-> next;  }  M->j = b - 1;  M->i= 0;  } | void MakeItJ(matr& M, int a)  {  fseek(M.f, (a - 1) \* sizeof(int), SEEK\_SET);  } |
| void NewStroka(matr\*& M)  {  TElem\* NewElem = NULL;  MakeItI(M, M->n);// спуск до последней строки  TElem\* UpperElem = M->suk;//указатель на первый элемент верхней строки длля создания связей внзи  for (int j = 0; j < M->m; j++){  NewElem = new TElem;  NewElem->next = NULL;  NewElem->down = NULL;  UpperElem->down = NewElem;  if (j == 0)  {  M->suk = NewElem;  M->i++;  }else  {  M->suk->next = NewElem;  M->suk = M->suk->next;  M->j++;  }  UpperElem = UpperElem->next;  }  M->n++;} | void NewStroka(matr& M)  {  int a = 0;  fseek(M.f, 0, SEEK\_END);  fwrite(&a, sizeof(int), M.m, M.f);  M.n++;  fseek(M.f, 0, SEEK\_SET);  } |
| void NewStolb(matr\*& M) {  TElem\* NewElem; // для создания новых элементов  MakeItJ(M, M->m);  TElem\* LastStolb = M->suk; //последний существующий столбец шоб сделать связи next  for (int i = 0; i < M->n; i++){  NewElem = new TElem;  NewElem->next = NULL;  NewElem->down = NULL;  LastStolb->next = NewElem;  if (i == 0)  {  M->suk = NewElem;  M->j++;  }else  {  M->suk->down = NewElem;  M->suk = M->suk->down;  M->i++;  }  LastStolb = LastStolb->down;  }  M->m++;  } | void NewStolb(matr& M)  {  int a = 0;  int dop;  fseek(M.f, 0, SEEK\_END); // сдвигаем указатель в конец  fseek(M.f, (M.n - 1) \* sizeof(int), SEEK\_CUR); // сдвигаем указатель на сколько нужно элементов в права  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 иачально 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 FreeMem(matr\*& M) {  if (Check(M) == TRUE) {  printf("\nматрица уже пуста\n\n");  return 0;  }  TElem\* dop;  M->suk = M->first; // перемещаем текущий на 1  M->i = 0;  M->j = 0;  while (M->first != NULL) {  M->suk = M->first;  M->first = M->first->down;  while (M->suk != NULL) {  dop = M->suk;  M->suk = M->suk->next;  delete dop;  }  }  M->m = 0;  M->n = 0;  printf("\nОчищено\n\n");  return 0;  } | void FreeMem(matr& M)  {  if (checker(M) == TRUE)  {  printf("\nматрица уже пуста\n\n");  return ;  }  //remove("Matr");  M.f=fopen( "Matr", "w+b");  printf("\nочищено\n\n");  return ;  } |
| int ReadCurrent(matr\*& M) {  return( M-> suk -> a);  } | int ReadCurrent(matr& M)  {  int a;  fread(&a, sizeof(int), 1, M.f);  fseek(M.f, 0 - sizeof(int), SEEK\_CUR);  return a;  } |
| void ChangeCur(matr\*& M, int b) {  M->suk->a = b;  } | void ChangeCur(matr& M, int a)  {  fwrite(&a, sizeof(int), 1, M.f);  fseek(M.f, 0 - sizeof(int), SEEK\_CUR);  } |
| void IznpozCurrent(matr\*& M, int a, int b) // сделать текущим элемент [a - 1,b - 1 ]  {  MakeItI(M, a) ; // перемещаем текущий на [a -1, 0]  for (int i = 0; i < b - 1; i++) // идём построке до нужного элемента  { M->suk = M->suk->next;  M->j++;  }  } | --------------------------------- |
| int ReadMatrix(matr\*& M, int n, int m) {  IznpozCurrent (M, n,m);  return (ReadCurrent(M));  } | int ReadMatrix(matr& 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;  a = read(M);// считываем значение  fseek(M.f, CurPos, SEEK\_SET); // откат обратно  return a;  } |
| int Outlog(matr\*& M, FILE\* log) {  fprintf(log, "\nМатрица из списка:\n");  if (Check(M)) // проверка на пустоту  {  fprintf(log, "матрица пуста\n");  return 0;  }  for (int i = 0; i < M->n; i++){  for (int j = 0; j < M->m; j++)  fprintf(log, " %3d ", ReadMatrix(M, i + 1, j +1)); // вывод значений в лог файл  fprintf(log,"\n");  }  return 0;  } | int Outlog(matr& M, FILE\* log)  {  int a;  fprintf(log,"\n Матрица из файла \n");  if (M.f == NULL)  {  fprintf(log, "файл пуст\n");  return 0;  }  fseek(M.f, 0, SEEK\_END);  if (ftell(M.f) == 0)  {  fprintf(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(log, " %3d ", a);  }  fprintf(log, "\n");  }  return 0;  } |

**Часть Д2. Создать приложение**

(консольное, на C или Delphi), позволяющее выбрать и выполнить следующие действия над матрицей

- очистить матрицу (освободить память);

- создать матрицу, считав размеры и значения элементов из текстового файла;

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

- изменить размер матрицы, добавив строку или столбец из 0(нулей);

- поиск: решить задачу

**int MainInput(matr& M);** // вывод матицы из текстового файла

**int MainChangeIndex(matr& M);** // изменить один элемент по индексам, на основе данных введенных в интерактивном режиме

**int MainStroka(matr& M);** // изменить размер матрицы, добавив строку из 0(нулей)

**int MainStolb(matr& M);**  // изменить размер матрицы, добавив столбец из 0(нулей)

**int Solution(matr& M);** // решение задачи

**Условие задачи**

|  |
| --- |
| Задан целочисленный двухмерный массив a из n строк и m столбцов. Найти номер первого максимального значения среди отрицательных элементов, расположенных после первого элемента, равного Т. Матрицу рассматривать по столбцам. В случае отсутствия отрицательных или равных Т элементов или невозможности поиска вывести соответствующие поясняющие сообщения. |

Порядок обхода элементов матрицы: **по строкам**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Формат входного текстового файла**

|  |
| --- |
| 4 4  2 -4 9 0  5 5 4 0  -8 7 1 4  10 7 2 -4  Т = 5 |

**Описание интерфейса (меню) созданного приложения**

Для взаимодействия с пользователем используется интерактивное 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  Ваш выбор? |

**Функциональные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тест** | **Ожидаемый результат** | **Смысл теста** |
| **1** | 1 1  1  Т = 2 | Невозможно найти ответ | Единственный элемент |
| **2** | 1 1  5  Т = 5 | Невозможно найти ответ | Единственный элемент равный т |
| **3** | 3 3  -3 -3 -3  -3 -3 -3  -3 -3 -3  Т = -3 | ответ 2 | Все элементы т |
| **4** | 3 4  1 2 3 4  5 8 9 8  4 9 7 6  Т = 6 | Невозможно найти ответ | Последний элемент т |
| **5** | 3 3  -2 -3 -5  -4 -5 – 7  -6 – 1 – 2  Т = -5 | Ответ 6 | Все элементы меньше 0 |
| **6** | 4 2  3 4 5 8  8 -9 -1 -2 | Ответ 4 | Первый элемент т |
| **7** | 4 4  2 -4 9 0  5 5 4 0  -8 7 1 4  10 7 2 -4  Т = 5 | Ответ 5 | Тест на средних |
| **8** | 3 4  -1 -2 -3 -1  -2 -4 -5 -1  -3 -1 -4 -9  Т = -2 | Ответ 6 | все числа отричательные несколько подходящих |
| **9** | 4 2  3 0  2 3  4 0  7 8  Т = 2 | Невозможно найти ответ | Все числа положительные  И есть 0 |

**Программный код процедуры для решения задачи (фильтр/поиск)**

|  |
| --- |
| int Solution(matr& M, int t) {  if (checker(M) == TRUE) {  printf("\nматрица пуста => нельзя найти решение\n\n");  return 0;  }  int nom = 0;  int minnom = 0;  int mx ;  int a;  bool flag, flag2 = true;  for (int j = 1; j <= M.m; j++) {  for (int i = 1; i <= M.n; i++) {  nom++;  if (ReadMatrix(M, i, j) == t && !flag) {  flag=true;  }else  if ((ReadMatrix(M, i, j) < 0) && (flag)) {  if (flag2){  mx = ReadMatrix(M, i, j);  flag2 = false;  }  if ((ReadMatrix(M, i, j) >= mx))  {  minnom = nom;  mx = ReadMatrix(M, i, j);  } } } }  return minnom;  } |

**Программный код процедуры добавление столба из 0**

|  |
| --- |
| int MainStolb(matr& M) {  if (checker(M) == TRUE) {  printf("\nматрица пуста => нельзя создать новый столбец\n\n");  return 0;  }  NewStolb(M);  for (int i = 1; i <= M.n; i++) {  Index(M, i, M.m, 0);  }  printf("\nстолбец создан\n");  getchar();  return 0;  } |

**Программный код процедуры добавление строки из 0**

|  |
| --- |
| int MainStroka(matr& M) {  if (checker(M) == TRUE) {  printf("\nматрица пуста => нельзя создать новую строку\n\n");  return 0;  }  NewStroka(M);  for (int i = 1; i <= M.m; i++) {  Index(M, M.n, i, 0);  }  printf("\nстрока создана\n");  getchar();  return 0;  } |

**Программный код процедуры считывание из файла**

|  |
| --- |
| int MainInput(matr& M) {  int n, m;  char name[20];  printf("Введите название файла: ");  gets(name);  FILE\* f;  f = fopen( name, "r");  if (f == NULL) {  printf("\nневозможно открыть файл\n\n");  return 0;  }  fseek(f, 0, SEEK\_END);  int end1 = ftell(f);  fseek(f, 0, SEEK\_SET);  int end2 = ftell(f);  if (end1 == end2) {  printf("\nФайл пуст\n\n");  return 0;  }  fscanf(f, "%d", &n);  fscanf(f, "%d", &m);    CreateEmpty(M, n, m);  int a;  for (int i = 0; i <= n - 1; i++) {  for (int j = 0; j <= m - 1; j++) {  fscanf(f, "%d", &a);  //printf("%d ", a);  Index(M, i + 1, j + 1, a);  }  }  fclose(f);  printf("\nсоздали\n\n");  return 0;  } |

**Программный код процедуры для изменения значения по индексу**

|  |
| --- |
| int MainChangeIndex(matr& M) {  if (checker(M) == TRUE) {  printf("\nМатрица пуста => нечего менять\n\n");  return 0;  }  int i, j;  int a;  printf(" Введите значение i: ");  scanf("%d", &i);  getchar();  printf(" Введите значение j: ");  scanf("%d", &j);  getchar();  printf(" Введите значение a: ");  scanf("%d", &a);  Index(M, i, j, a);  printf("\n Изменили\n");  getchar();  return 0;  } |