Московский Авиационный Институт

(Национальный Исследовательский Университет)

**Факультет прикладной математики и физики**

Кафедра вычислительной математики и программирования

**КУРСОВАЯ РАБОТА**

**по курсу**

**“Практикум на ЭВМ”**

**II семестр**

**«Разреженные матрицы»**

Студент: Бокоч С.М.

Группа: 08-104, № по списку 2

Руководитель: Никулин С.П.,  
 доцент каф.806

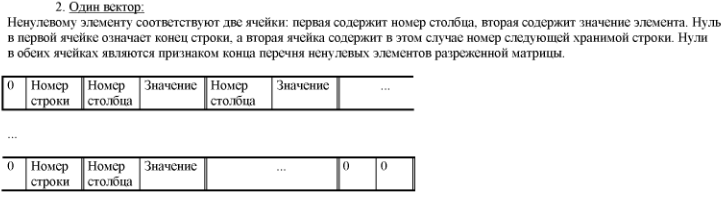
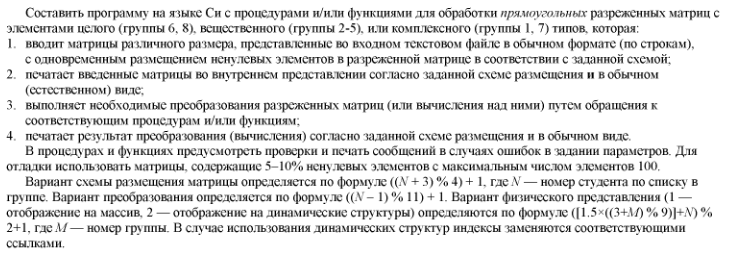
Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва, 2017**

**1.Задание**





**2. Общий метод решения**

* Прежде всего необходимо, считать матрицу n\*m и «загнать» ее в структуру vector. Вначале массива типа, обозначим за EMPTY — пустой элемент. EMPTY также добавляем в конце каждой строки. COMP — элемент матрицы. Нулевые элементы пропускаем. После последнего элемента признак конца, элемент END.
* Проходим по нашей структуре от начала до конца и ищем максимальный по модулю элемент (если это ноль, т. е. матрица нулевая, то выводим ошибку и завершаем программу)
* Проходим еще раз по структуре и ищем номер столбца в котором есть максимальный по модулю элемент. Если столбец нашелся, то отмечаем в массиве столбцов true. Чтобы потом не потерять т.к нам нужен предпоследний столбец, если их несколько.
* Если столбцов несколько (cnt>1), то идем по массиву столбцов от lastInd -1, чтобы не включать последний максимальный столбец.
* Проходим послений раз по структуре и находим элементы, принадлежащие lastInd столбцу. Эти элементы делим на максимальный элемент.
* Выводим результат
* Освобождаем память.

**3. Общие сведения о программe**

Необходимое программное и аппаратное обеспечение: компилятор gcc

Операционная система: любая операционная система с поддержкой Си

Язык: Си

Система программирования: Си

Число строк программ:

* vector.h - 33
* vector.c - 112
* MatrixSourse.c - 243

Местонахождение и имена файлов с исходными текстами и данными: serega@serega-Inspiron-3537:~/course\_project/kp7

Способ вызова и загрузки: в директории с файлом в bash ./main

**4. Функциональное назначение**

Программа расчитана на преобразование матрицы, размеры которой принадлежат диапазону от 1 до 100

С файла считываются размеры матрицы, затем элементы самой матрицы

Вообще говоря, размер матрицы можно увеличить, но это зависит от мощности используемого оборудования, т. к. кол-во необходимой выделяемой памяти может не хватить.

**6. Описание переменных и констант**

**Файл vector.h —** включает описания струтуры vector, и заголовки всех используемых методов.

|  |  |
| --- | --- |
| #ifndef VECTOR\_H  #define VECTOR\_H  #include <stdlib.h> | Заголовочные файлы |
| typedef double \_template; | Тип определяемый по условию, вещественный |
| typedef struct \_Item  {  int ind;  \_template c;  } VECTOR\_TYPE; | Структура элемента вектора  Статус, описываемый в основной программе  Само значение |
| typedef struct \_Vector  {  VECTOR\_TYPE \*\_data;  int \_size;  int \_capacity;  } Vector; | Структура вектора  Массив, который задан по условию  Кол-во элементов  Общая размерность массива |
| void vectorCreate(Vector \*v, const int size);  int vectorSize(const Vector \*v);  int vectorCapacity(const Vector \*v);  VECTOR\_TYPE vectorLoad(const Vector \*v, const int index);  void vectorSave(Vector \*v, const int index, const VECTOR\_TYPE value);  int vectorPushBack(Vector \*v, const VECTOR\_TYPE value);  void vectorDestroy(Vector \*v); | Методы для работы с вектором  Создать вектор размерности size  Узнать размер вектора  Узнать вместимость вектора  Узнать значение index элемента массива data  Присвоить значение value index элемента массива data  Добавить в конец вектора элемент value  Удалить вектор, освободив динамическую память |

**Файл vector.c**— методы, для работы с вектором

|  |  |
| --- | --- |
| #include "vector.h" | Подключаем заголовочнй файл vector.h |
| void vectorCreate(Vector \*v, const int size)  {  if (size > 0)  {  v->\_data = (VECTOR\_TYPE \*)malloc(sizeof(VECTOR\_TYPE) \* size);  v->\_capacity = size;  } else {  v->\_data = (VECTOR\_TYPE \*)malloc(sizeof(VECTOR\_TYPE));  v->\_capacity = 1;  }  v->\_size = 0;  } | Если мы создаем пустой вектор, то он должен содержать как минимум один элемент, который в дальнейшем будет объявлен как EMPTY |
| int vectorSize(const Vector \*v)  {  return v->\_size;  } |  |
| int vectorCapacity(const Vector \*v)  {  return v->\_capacity;  } |  |
| VECTOR\_TYPE vectorLoad(const Vector \*v, const int index)  {  return v->\_data[index];  } |  |
| void vectorSave(Vector \*v, const int index, const VECTOR\_TYPE value)  {  v->\_data[index] = value;  } |  |
| int vectorPushBack(Vector \*v, const VECTOR\_TYPE value)  {  VECTOR\_TYPE \*ptr = NULL;  if (v->\_size == v->\_capacity)  {  ptr = (VECTOR\_TYPE \*)realloc(v->\_data, sizeof(VECTOR\_TYPE) \* v->\_capacity\*2);  if (ptr != NULL)  {  v->\_data = ptr;  v->\_capacity \*= 2;  }  else  return 0;  }  v->\_data[v->\_size++] = value;  return 1;  } | Если массив забит, то выделяем дополнительный блок памяти с помощью функции realloc. Если память выделить нельзя, то возвращаем 0.  Добавляем элемент в конец вектора |
| void vectorDestroy(Vector \*v)  {  if (v->\_data != NULL)  {  free(v->\_data);  v->\_data = NULL;  }  v->\_size = 0;  v->\_capacity = 0;  } |  |

**Файл MatrixSource.c** — основная программа

|  |  |
| --- | --- |
| #include <stdio.h>  #include <math.h>  #include "vector.c" | Заголовочные файлы |
| const int N = 100; | Максимальный размер матрицы |
| typedef enum \_kInd  {  END = -3,  COMP,  EMPTY  } kInd; | Перечисление значений  конец матрицы  значение в ячейке матрицы  пустое значение |
| typedef struct \_Cell  {  Vector \*v;  int ind;  int row;  int col;  \_template data;  } Cell; | Вспомогательная структура для обработки vectorа. Содержит элементы VECTOR\_TYPE c дополнением  номера строки  номера столбца |
| void InputMatrix(Vector \*v, FILE \*in, int \*m, int \*n)  {  int isRowBegin;  VECTOR\_TYPE tmpItem;  fscanf(in, "%d", m);  fscanf(in, "%d", n);  if (\*m < 1 || \*m > N || \*n < 1 || \*n > N)  {  printf("Количество строк должно быть в диапозоне от 1 до %d\n", N);  exit(EXIT\_FAILURE);  }  vectorCreate(v, 1);  tmpItem.ind = EMPTY;  \_template tmp\_template;  vectorPushBack(v, tmpItem);  for (int i = 0; i < \*m; i++)  {  isRowBegin = 0;  for (int j = 0; j < \*n; j++)  {  fscanf(in, "%lf", &tmp\_template);  if (tmp\_template == 0.0)  continue;  if (!isRowBegin)  {  isRowBegin = 1;  tmpItem.ind = i;  vectorPushBack(v, tmpItem);  }  tmpItem.ind = j; //  vectorPushBack(v, tmpItem);  tmpItem.c = tmp\_template;  tmpItem.ind = COMP;  vectorPushBack(v, tmpItem);  }  if (isRowBegin)  {  tmpItem.ind = EMPTY;  vectorPushBack(v, tmpItem);  }  }  tmpItem.ind = END;  vectorPushBack(v, tmpItem);  return;  } | Ввод матрицы  Признак начала строки  Вспомогательная переменная для добавления в вектор  Проверка правильности ввода  Создание вектора  Добавление пустого значения в вектор(см.условие)  Нулевые элементы пропускаем  Если это начало строки, то добавляем номер строки в вектор  Добавляем номер столбца в вектор  Добаляем сам элемент матрицы i j  После прохода по строке добавляем в вектор признак конца строки, EMPTY элемент  добавляем признак конца матрицы |
| Cell cellFirst(Vector \*v)  {  Cell res;  res.v = v;  res.ind = 2;  res.row = END;  res.col = EMPTY;  res.data = 0.0;  if (vectorLoad(v, res.ind - 1).ind != END)  {  res.row = vectorLoad(v, res.ind - 1).ind;  res.col = vectorLoad(v, res.ind).ind;  res.data = vectorLoad(v, res.ind + 1).c;  }  return res;  } | Возвращает структуру cell первого элемента вектора  По заданию нам необходима работа со столбцом, поэтому номер столбца будет ведущим.  ind = 2 потому что элемент 1 1 хранится на 3 позиции в векторе, т. к. сначала создался сам вектор с 1 элементом, затем добавилось два элемента: номер строки и столбца  Если это не конец вектора, т. е. Значения не выходят за пределы существующих элементов  Добавляем данные в переменную res  ЕСЛИ ЭТО НЕ КОНЕЦ СТРУКТУРЫ, ТО ЗАПИСЫВАЕМ КООРДИНАТЫ |
| void cellNext(Cell \*cell)  {  int c1, c2;  if (cell->row == END)  return;  cell->ind += 2;  c1 = vectorLoad(cell->v, cell->ind).ind;  c2 = vectorLoad(cell->v, cell->ind + 1).ind;  if (c1 > EMPTY && c2 == COMP)  {  cell->col = vectorLoad(cell->v, cell->ind).ind;  cell->data = vectorLoad(cell->v, cell->ind + 1).c;  }  else if (c1 == EMPTY && c2 > EMPTY)  {  cell->row = vectorLoad(cell->v, cell->ind + 1).ind;  cell->col = vectorLoad(cell->v, cell->ind + 2).ind;  cell->data = vectorLoad(cell->v, cell->ind + 3).c;  cell->ind += 2;  }  else  {  cell->row = END;  cell->col = EMPTY;  }  } | Если это последний элемент, то завершаем программу  Делаем прыжок через 1 элемент: элемента матрицы. Cell→ind указывает на номер строки, пустой элемент или на конец вектора END  Теперь происходят сдвиги с помощью if-ов, находим номер столбца следующего элемента  если конец строки, то перепрыгиваем элементы  Если ничего не подходит |
| void printSourceMatrix(Vector \*v, const int m, const int n)  {  int i, j;  Cell cell = cellFirst(v);  for (i = 0; i < m; i++)  {  for (j = 0; j < n; j++)  {  if (i == cell.row && j == cell.col)  {  printf("%.2lf ", cell.data);  cellNext(&cell);  }  else  printf("%.2lf ", 0.0);  }  printf("\n");  }  } | Печать матрицы в нормальном виде  Если следующий элемент является i j,  то печатаем этот элемент, иначе 0 |
| void printInnerMatrix(const Vector \*v)  {  int i;  VECTOR\_TYPE item;  for (i = 0; i < vectorSize(v); i++)  {  item = vectorLoad(v, i);  if (item.ind == COMP)  printf("%.2lf ", item.c);  else  printf("%d ", item.ind);  }  printf("\n");  } | Печать матрицы в структурном виде |
| int main(void)  {  int lastInd, cnt;  int maxCols[N];  for (int i = 0; i < N; i++)  maxCols[i] = 0;  FILE \*in = fopen("test", "r");  if (in == NULL)  {  printf("Неудается открыть файл");  return 1;  }  Vector v;  int n, m;  Cell cell;  InputMatrix(&v, in, &m, &n);  /\* Задание \*/  printf("Обычное представление:\n");  printSourceMatrix(&v, m, n);  printf("Внутреннее представление\n");  printInnerMatrix(&v);  \_template tmp\_template, max\_template;  cell = cellFirst(&v);  /\* Пока не конец матрицы \*/  while (cell.row != END)  {  if (abs(cell.data) > max\_template)  max\_template = abs(cell.data);  cellNext(&cell);  }  printf("Максимальное вещественное число по модулю: %.2lf\n", max\_template);  if (max\_template == 0.0)  {  printf("Делить на него нельзя, так как его модуль равен нулю\n");  return 0;  }  lastInd = 0;  cnt = 0;  cell = cellFirst(&v);  while (cell.row != END)  {  if (abs(cell.data) == max\_template)  {  maxCols[cell.col] = 1;  lastInd = cell.col;  cnt++;  }  cellNext(&cell);  }  VECTOR\_TYPE tmpItem;  if (cnt > 1)  for (int i = lastInd - 1; i >= 0; i--)  if (maxCols[i])  {  lastInd = i;  break;  }  cell = cellFirst(&v);  while (cell.row != END)  {  if (cell.col == lastInd)  {  tmpItem = vectorLoad(&v, cell.ind + 1);  tmpItem.c = cell.data/max\_template;  vectorSave(&v, cell.ind + 1, tmpItem);  }  cellNext(&cell);  }  printf("Обычное представление после преобразования:\n");  printSourceMatrix(&v, m, n);  printf("Внутреннее представление после преобразования:\n");  printInnerMatrix(&v);  vectorDestroy(&v);  return 0;  } | Вспомогательная переменная и кол-во столбцов с максимальным значением  Массив столбцов, можно булевский. Определяет столбцы с максимальным значением элемента  Ищем максимальный элемент  столбец с максимальным значением помечаем  последний такой столбец  от предпоследнего столбца с максимальным значением , если несколько.  Если текущий столбец равен «максимальному»  Делим элементы на максимальное значение  Удаляем структуру |

**7. Тестовые примеры**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| ***TEST\_1*** | |
| 5 5  1 2 3 4 5  6 7 8 9 25  1 4 5 8 9  1 23 0 1 4  0 0 0 5 6 |  |
| 4 4  0 0 0 0  0 0 0 0  0 0 0 0 |  |
| 1 1  25 |  |
| 3 4  1 1 -5 0  0 0 0 0  -1 2 3 4 |  |

|  |  |
| --- | --- |
| 4 5  15 14 12 15 15  0 0 0 0 0  1 1 1 1 1  -4 -4 2 4 5 |  |
| 3 3  -10 0 0  0 -10 0  0 0 10 |  |

**9. Дневник отладки**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | Дата | Время | Место | Наиболее характерные ошибки | Действия по исправлению | Внешние признаки | Сведения о степени самостоятель-ности |
| 1 | 14.05.2017 | 10:01. | Общежитие №5 МАИ | Проблемы, с выделением памяти при использовании realloc | Capacity \*=2 | Вывод в консоль имена ячеек с использованной памятью | Обнаружено самосто-ятельно |

**12. Выводы по задаче**

Проделав данную лабораторную работу, я познакомился с АТД(абстрактными типами данных), научился преставлять матрицу в виде вектора, стал применять перечисления(enum),

а также писать

***Протокол исходного кода***

serega@serega-Inspiron-3537:~/course\_project/kp7$ cat **vector.h**

#ifndef VECTOR\_H

#define VECTOR\_H

#include <stdlib.h>

typedef double \_template;

typedef struct \_Item

{

int ind;

\_template c;

} VECTOR\_TYPE;

typedef struct \_Vector

{

VECTOR\_TYPE \*\_data;

int \_size;

int \_capacity;

} Vector;

void vectorCreate(Vector \*v, const int size);

int vectorSize(const Vector \*v);

int vectorCapacity(const Vector \*v);

VECTOR\_TYPE vectorLoad(const Vector \*v, const int index);

void vectorSave(Vector \*v, const int index, const VECTOR\_TYPE value);

int vectorPushBack(Vector \*v, const VECTOR\_TYPE value);

void vectorDestroy(Vector \*v);

#endif

serega@serega-Inspiron-3537:~/course\_project/kp7$ cat **vector.c**

#include "vector.h"

void vectorCreate(Vector \*v, const int size)

{

if (size > 0)

{

v->\_data = (VECTOR\_TYPE \*)malloc(sizeof(VECTOR\_TYPE) \* size);

v->\_capacity = size;

}

else

{

v->\_data = (VECTOR\_TYPE \*)malloc(sizeof(VECTOR\_TYPE));

v->\_capacity = 1;

}

v->\_size = 0;

}

int vectorSize(const Vector \*v)

{

return v->\_size;

}

int vectorCapacity(const Vector \*v)

{

return v->\_capacity;

}

VECTOR\_TYPE vectorLoad(const Vector \*v, const int index)

{

return v->\_data[index];

}

void vectorSave(Vector \*v, const int index, const VECTOR\_TYPE value)

{

v->\_data[index] = value;

}

int vectorPushBack(Vector \*v, const VECTOR\_TYPE value)

{

VECTOR\_TYPE \*ptr = NULL;

if (v->\_size == v->\_capacity)

{

ptr = (VECTOR\_TYPE \*)realloc(v->\_data, sizeof(VECTOR\_TYPE) \* v->\_capacity\*2);

if (ptr != NULL)

{

v->\_data = ptr;

v->\_capacity \*= 2;

}

else

return 0;

}

v->\_data[v->\_size++] = value;

return 1;

}

void vectorDestroy(Vector \*v)

{

if (v->\_data != NULL)

{

free(v->\_data);

v->\_data = NULL;

}

v->\_size = 0;

v->\_capacity = 0;

serega@serega-Inspiron-3537:~/course\_project/kp7$ cat **MatrixSourse.c**

#include <stdio.h>

#include <math.h>

#include "vector.c"

const int N = 100;

typedef enum \_kInd

{

END = -3,

COMP,

EMPTY

} kInd;

typedef struct \_Cell

{

Vector \*v;

int ind;

int row;

int col;

\_template data;

} Cell;

Cell cellFirst(Vector \*v);

void cellNext(Cell \*cell);

void printSourceMatrix(Vector \*v, const int m, const int n);

void printInnerMatrix(const Vector \*v);

void InputMatrix(Vector \*v, FILE \*in, int \*m, int \*n);

int main(void)

{

int lastInd, cnt, maxCols[N];

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

maxCols[i] = 0;

FILE \*in = fopen("test", "r");

if (in == NULL)

{

printf("Неудается открыть файл");

return 1;

}

Vector v;

int n, m;

Cell cell;

InputMatrix(&v, in, &m, &n);

printf("Обычное представление:\n");

printSourceMatrix(&v, m, n);

printf("Внутреннее представление\n");

printInnerMatrix(&v);

\_template tmp\_template, max\_template;

cell = cellFirst(&v);

while (cell.row != END)

{

if (abs(cell.data) > max\_template)

max\_template = abs(cell.data);

cellNext(&cell);

}

printf("Максимальное вещественное число по модулю: %.2lf\n", max\_template);

if (max\_template == 0.0)

{

printf("Делить на него нельзя, так как его модуль равен нулю\n");

return 0;

}

lastInd = 0;

cnt = 0;

cell = cellFirst(&v);

while (cell.row != END)

{

if (abs(cell.data) == max\_template)

{

maxCols[cell.col] = 1;

lastInd = cell.col;

cnt++;

}

cellNext(&cell);

}

VECTOR\_TYPE tmpItem;

if (cnt > 1)

for (int i = lastInd - 1; i >= 0; i--)

if (maxCols[i])

{

lastInd = i;

break;

}

cell = cellFirst(&v);

while (cell.row != END)

{

if (cell.col == lastInd)

{

tmpItem = vectorLoad(&v, cell.ind + 1);

tmpItem.c = cell.data/max\_template;

vectorSave(&v, cell.ind + 1, tmpItem);

}

cellNext(&cell);

}

printf("Обычное представление после преобразования:\n");

printSourceMatrix(&v, m, n);

printf("Внутреннее представление после преобразования:\n");

printInnerMatrix(&v);

vectorDestroy(&v);

return 0;

}

void InputMatrix(Vector \*v, FILE \*in, int \*m, int \*n)

{

int isRowBegin;

VECTOR\_TYPE tmpItem;

fscanf(in, "%d", m);

fscanf(in, "%d", n);

if (\*m < 1 || \*m > N || \*n < 1 || \*n > N)

{

printf("Количество строк должно быть в диапозоне от 1 до %d\n", N);

exit(EXIT\_FAILURE);

}

vectorCreate(v, 1);

tmpItem.ind = EMPTY;

\_template tmp\_template;

vectorPushBack(v, tmpItem);

for (int i = 0; i < \*m; i++)

{

isRowBegin = 0;

for (int j = 0; j < \*n; j++)

{

fscanf(in, "%lf", &tmp\_template);

if (tmp\_template == 0.0)

continue;

if (!isRowBegin)

{

isRowBegin = 1;

tmpItem.ind = i;

vectorPushBack(v, tmpItem);

}

tmpItem.ind = j;

vectorPushBack(v, tmpItem);

tmpItem.c = tmp\_template;

tmpItem.ind = COMP;

vectorPushBack(v, tmpItem);

}

if (isRowBegin)

{

tmpItem.ind = EMPTY;

vectorPushBack(v, tmpItem);

}

}

tmpItem.ind = END;

vectorPushBack(v, tmpItem);

return;

}

Cell cellFirst(Vector \*v)

{

Cell res;

res.v = v;

res.ind = 2;

res.row = END;

res.col = EMPTY;

res.data = 0.0;

if (vectorLoad(v, res.ind - 1).ind != END)

{

res.row = vectorLoad(v, res.ind - 1).ind;

res.col = vectorLoad(v, res.ind).ind;

res.data = vectorLoad(v, res.ind + 1).c;

}

return res;

}

void cellNext(Cell \*cell)

{

int c1, c2;

if (cell->row == END)

return;

cell->ind += 2;

c1 = vectorLoad(cell->v, cell->ind).ind;

c2 = vectorLoad(cell->v, cell->ind + 1).ind;

if (c1 > EMPTY && c2 == COMP)

{

cell->col = vectorLoad(cell->v, cell->ind).ind;

cell->data = vectorLoad(cell->v, cell->ind + 1).c;

}

else if (c1 == EMPTY && c2 > EMPTY)

{

cell->row = vectorLoad(cell->v, cell->ind + 1).ind;

cell->col = vectorLoad(cell->v, cell->ind + 2).ind;

cell->data = vectorLoad(cell->v, cell->ind + 3).c;

cell->ind += 2;

}

else

{

cell->row = END;

cell->col = EMPTY;

}

}

void printSourceMatrix(Vector \*v, const int m, const int n)

{

int i, j;

Cell cell = cellFirst(v);

for (i = 0; i < m; i++)

{

for (j = 0; j < n; j++)

{

if (i == cell.row && j == cell.col)

{

printf("%.2lf ", cell.data);

cellNext(&cell);

}

else

printf("%.2lf ", 0.0);

}

printf("\n");

}

}

void printInnerMatrix(const Vector \*v)

{

int i;

VECTOR\_TYPE item;

for (i = 0; i < vectorSize(v); i++)

{

item = vectorLoad(v, i);

if (item.ind == COMP)

printf("%.2lf ", item.c);

else

printf("%d ", item.ind);

}

printf("\n");

}