Отчет по Курсовой работе №7 по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Трофимов Владислав Олегович, № по списку 19

Контакты e-mail:trofimov.vlad-1234@yandex.ru
Github: https://github.com/ant1hype1488/labs

Работа выполнена: «14» мая 2022г.

Преподаватель: каф. 806 Севастьянов Виктор Сергеевич

Отчет сдан « » _____20__ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема**: Разреженные матрицы

1. **Цель работы:** Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементами вещественного
2. **Задание** (*вариант №* **3 8 2)** :
   Вариант размещения: 3 вектора .
   Вариант преобразования: Вычислить произведение двух разреженных матриц. Проверить ,не является полученная матрица диагональной.
   Вариант физического представления: отображение на динамической структуре
3. **Оборудование** (студента):
   Процессор *Intel Core i5-8265U @ 8x 3.9GH* с ОП *7851* Мб, НМД *1024* Гб. Монитор *1920x1080*

4. **Программное обеспечение (**студента**):**
   Операционная система семейства: *linux*, наименование: *ubuntu*, версия *18.10 cosmic*
   интерпретатор команд: *bash* версия *4.4.19*.
   Система программирования -- версия --, редактор текстов *emacs* версия *25.2.2*
   Утилиты операционной системы --
   Прикладные системы и программы --
   Местонахождение и имена файлов программ и данных на домашнем компьютере --

6. Идея, метод, алгоритм

chechDiag(Matrix *matrix):
Проверяет ,является ли матрица диагональной. Если элемент по индексу из массива CIP не равен элементу такого же индекса PI ,то матрица не диагональная
Matrix *matrixMultipy(Matrix *matrix1,Matrix *matrix2,m1,n1,m2,n2):
Умножает 2 матрицы
Состоит из 4 вложеннных циклов. Первый цикл (i) отвечает за номер строки первой матрицы. Второй цикл (k) перебивает столбцы второй матрицы. Третий цикл(j) пробегается по элементам текущий строки первой матрицы. Четвертый цикл (h) находит элемент из текущего столбца второй матрицы, на который нужно умножить текущий элемент строки левой матрицы.
void inputMatrix(Matrix *matrix1,Matrix *matrix2,int *m1,int *n1,int *m2,int *n2):
Считывает 2 матрицы из файла
void printMatrix(Matrix *matrix,int m,int n):
Выводит матрицу

7. Сценарий выполнения работы

Main.c

```
#include <stdio.h>

#include "stdlib.h"
#include "stdio.h"
#include "math.h"

#include "matrix.h"
int main() {
   Matrix * matrix1 = malloc(sizeof(Matrix));
   Matrix * matrix2 = malloc(sizeof(Matrix));

   Matrix * mult = malloc(sizeof(Matrix));


   int n1,m1;
   int n2,m2;
   inputMatrix(matrix1,matrix2,&m1,&n1,&m2,&n2);

   printf("Матрица 1 \n");

   printMatrix(matrix1,m1,n1);
   printf("\n");
   printf("Матрица 2 \n");


   printMatrix(matrix2,m2,n2);
   printf("\n");
   printf("Умножение матриц \n");

   mult = matrixMultipy(matrix1,matrix2,m1,n1,m2,n2);
   printMatrix(mult,m1,n2);
   printf("\n");

   printf("Проверка диагональности матрицы 1 :\n");
   chechDiag(matrix1);
   printf("\n");
   printf("\n");

   printf("Проверка диагональности умножения  :\n");
   chechDiag(mult);
   printf("\n");


   return 0;
}
```

matrix.c

```
#include "stdlib.h"
#include "stdio.h"

#include "matrix.h"


void  inputMatrix(Matrix *matrix1,Matrix *matrix2,int *m1,int *n1,int *m2,int *n2){
   FILE *fptr;
   int num ;
   int count =0;
   if ((fptr = fopen("data.txt","r")) == NULL){
      printf("Error! opening file");
      exit(1);
   }
   fscanf(fptr, "%d %d", m1,n1);

   for (int i = 0; i <*m1 ; ++i) {
      if((count  == matrix1->CIP[i-1]) ){
         *(matrix1->CIP + i) = 0;
      } else{
         *(matrix1->CIP + i) = count;

      }
      for (int j = 0; j < *n1; ++j) {
         fscanf(fptr, "%d", &num);
         if(num !=0){
            *(matrix1->PI + count) = j;
            *(matrix1->YE + count) = num;
            count++;
         }

      }

   }

   fscanf(fptr, "%d %d", m2,n2);
   count = 0;
   for (int i = 0; i <*m2 ; ++i) {
      if((count  == matrix2->CIP[i-1]) ){
         *(matrix2->CIP + i) = 0;
      } else{
         *(matrix2->CIP + i) = count;

      }
      for (int j = 0; j < *n2; ++j) {
         fscanf(fptr, "%d", &num);
         if(num !=0){
            *(matrix2->PI + count) = j;
            *(matrix2->YE + count) = num;
            count++;
         }

      }

   }
   fclose(fptr);
}
void printMatrix(Matrix *matrix,int m,int n){
   int count = 0;
   for (int i = 0; i < m; ++i) {
```

```
        for (int j = 0; j < n ; ++j) {
            if (i !=m -1 ){
                if (matrix->CIP[i+1]!=count&& matrix->CIP[i+1]!=0 ){

                    if (matrix->PI[count] == j){
                        printf("%d ",matrix->YE[count]);
                        count ++;
                    }else {
                        printf("0 ");
                    }
                }else {
                    printf("0 ");
                }
            } else {
                if (matrix->PI[count] == j) {
                    printf("%d ", matrix->YE[count]);
                    count++;
                } else {
                    printf("0 ");
                }
            }
        }
        printf("\n");
    }
}
Matrix *matrixMultipy(Matrix *matrix1,Matrix *matrix2,m1,n1,m2,n2){
    int count = 0;
    int num = 0 ;
    int g = 0;
    Matrix * m3 = malloc(sizeof(Matrix));;
    if (n1 == m2){
        for (int i = 0; i < m1; ++i) {
            m3->CIP[i] = count;
            for (int k = 0; k < n2; ++k) {
                if (i<m1-1){
                    for (int j = matrix1->CIP[i]; j < matrix1->CIP[i+1] ; ++j) {
                        if (matrix1->PI[j] <n1 -1){
                            for (int h = matrix2->CIP[ matrix1->PI[j] ]; h < matrix2->CIP[ matrix1->PI[j]+1  ] ; ++h) {
                                if (matrix2->PI[h] == k){
                                    num += matrix2->YE[h] * matrix1->YE[j];
                                    g = 1;

                                }
                            }
                        } else{
                            int h = matrix2->CIP[ matrix1->PI[j] ];

                            while(matrix2->YE[h]!=0) {

                                if (matrix2->PI[h] == k) {
                                    num += matrix2->YE[h] * matrix1->YE[j];
                                    g = 1;
                                }
                                h++;
                            }
                        }

                    }
                }else{
                    int j = matrix1->CIP[i];
                    while(matrix1->YE[j]!=0){
                        if (matrix1->PI[j] <n1 -1){
```

```
                for (int h = matrix2->CIP[ matrix1->PI[j] ]; h < matrix2->CIP[ matrix1->PI[j]+1  ] ; ++h) {
                    if (matrix2->PI[h] == k){
                        num += matrix2->YE[h] * matrix1->YE[j];
                        g = 1;

                    }
                }
            } else{
                int h = matrix2->CIP[ matrix1->PI[j] ];

                while(matrix2->YE[h]!=0) {

                    if (matrix2->PI[h] == k) {
                        num += matrix2->YE[h] * matrix1->YE[j];
                        g = 1;
                    }
                    h++;
                }
            }

            j++;
        }
    }

    m3->PI[count] = k;
    m3->YE[count] = num;

    if (g == 1){
        count ++;
    }
    g = 0;
    num = 0;

  }


 }
 return m3;



}
void chechDiag(Matrix *matrix){
 int i =0;
 int diag = 0;
 while (matrix->YE[i]!=0){
    if (matrix->CIP[i]!=matrix->PI[i]){
        diag = 1;
        break;
    }
    i++;

 }
 if(diag==0)
    printf("матрица диагональная");
 else
    printf("матрица не диагональная");
```

Matrix.h

```
#ifndef KP7_MATRIX_H
#define KP7_MATRIX_H

typedef struct Matrix {
   int CIP[100];
   int PI[100];
   int YE[100];
} Matrix;

void  inputMatrix(Matrix *matrix1,Matrix *matrix2,int *m1,int *n1,int *m2,int *n2);
void printMatrix(Matrix *matrix,int m,int n);
void chechDiag(Matrix *matrix);
Matrix *matrixMultipy(Matrix *matrix1,Matrix *matrix2,m1,n1,m2,n2);
#endif //KP7_MATRIX_H
```

8. Распечатка протокола
Матрица 1
1 0 0
0 1 0
0 0 1
Матрица 2
1 3 1 1 3
1 3 1 1 3
1 3 1 1 3
Умножение матриц
1 3 1 1 3
1 3 1 1 3
1 3 1 1 3
Проверка диагональности матрицы 1 :
матрица диагональная

Проверка диагональности умножения :
матрица не диагональная

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора
Работу считаю актуальной :она помогает поближе познакомиться с разреженными матрицами
11. **Выводы**
Благодаря данный работе я научился работать с разреженными матрицами и их преобразовать матрицы

Подпись студента _____