Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект VII

по курсу «Практикум на ЭВМ»

II семестр

«Разряженные матрицы»

Студент	**Сыроежкин Кирилл Геннадьевич**
Группа	**М8О-104Б-18**
Руководитель	**Доцент кафедры 806 Никулин С.П**
Оценка	
Дата	

Москва 2019.

1.Постановка задачи.

Составить программу на языке Си с процедурами и функциями для обработки прямоугольных разряженных матриц с элементами вещественного типа, которая:

1. Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате, с одновременным размещением ненулевых элементов согласно заданной схеме размещения и в обычном виде;
2. Печатает введенные матрице во внутреннем представлении согласно заданной схеме размещения и в обычном виде;
3. Выполняет необходимые преобразования разряженных матриц путем обращения к соответствующим процедурам и функциям;
4. Печатает результат преобразования согласно заданной схеме размещения и в обычном виде.

1.1. Схема размещения матрицы (Один вектор).

Ненулевому элементу соответствуют две ячейки, первая содержит номер столбца, вторая содержит значение элемента. Нуль в первой ячейке означает конец строки, а вторая ячейка содержит в этом случае номер следующей хранимой строки. Нули в обеих ячейках являются признаком конца перечня ненулевых элементов разряженной матрицы.

1.2. Преобразование (7).

Найти строку, содержащую наибольшее количество ненулевых элементов, и напечатать её номер и сумму элементов этой строки. Если таких строк несколько, обработать все.

2. Общий метод решения

В программу передаётся в качестве аргумента название текстового файла, в который записана матрица. Далее вызывается функция, отвечающая за работу меню, которая, в свою очередь, вызывает одну из 4-х функций, выполняющих требуемую работу с матрицей.

3. Общие сведения о программе

Имена файлов:

1) kurs7.c - программа;

2) file1.txt - матрица для тестов;

3) file2.txt - матрица для тестов;

4) file3.txt - матрица для тестов;

5) file4.txt - матрица для тестов;

6) file5.txt - матрица для тестов;

7) file6.txt - матрица для тестов;

8) file7.txt - матрица для тестов;

9) file8.txt - матрица для тестов;

10) file9.txt - матрица для тестов;

11) file10.txt - матрица для тестов.

Программное и аппаратное обеспечения для запуска данной программы на ПК не ограничено в выборе. Операционная система семейства Unix - Ubuntu. Язык программирования Си. Строк в программе prog.c: 97, dump.c: 28. В заголовочном файле pass.h:15.

4. Описание логической структуры программы

Функция **main** должна принимать из терминала 1 параметр (**argv[1]**) - название файла с матрицей для работы. Здесь происходит открытие файла и вызывается функция **action_menu**.

Функция **action_menu** принимает в качестве параметра файл. Она печатает в терминал меню, вызывает функцию **matrix_to_vecror,** которую мы рассмотрим в самом конце, и просит выбрать один из пунктов меню:

1) вызывает функцию **print_array**(параметры: массив, файл), которая печатает матрицу в векторном представлении (простой вывод массива)

2) вызывает функцию **print_matrix** (параметры: массив, файл), которая печатает матрицу основываясь на векторном представлении (несколько вложенных циклов и ветвлений)

3) вызывает функцию **vector_trans** (параметры: массив, файл), которая находит строки, содержащие наибольшее количество ненулевых элементов, используя только векторное представление

4) вызывает функцию **matrix_trans** (параметры: файл), которая находит строки, содержащие наибольшее количество ненулевых элементов, используя только матричное представление.

5) вызывает функцию **exit** (Выход из программы)

Функция **matrix_to_vector** по сути является основной функций, так как переносит матрицу в векторное представление. Она принимает в качестве параметра заранее заданную матрицу и файл. Затем с помощью цикла и функции **fscanf** поэлементно заносится содержание файла в массив (с указанием номеров столбцов, строк и пропуском пустых элементов).

5. Входные данные

Программа должна получать из терминала название текстового файла, который должен содержать разряженную матрицу.

6. Выходные данные

Результат работы программы может меняться в зависимости от выбранного пунктов меню:

1) Вывод матрицы в векторном представлении;

2) Вывод матрицы, используя векторное представление;

3) Нахождение строк с наибольшим числом элементов, используя; векторное представление.

4) Нахождение строк с наибольшим числом элементов, используя обычное представление.

7. Программа на СИ

```
#include <stdio.h>
#include <stdlib.h>
double zero=0;
int count_all_elem(FILE *input) // Количество элементов в матрице
{
    fseek(input, 0L, SEEK_SET);
    double sym;
```

```
    int i=0;
    while(fscanf(input,"%lf", &sym)!=EOF)
        i++;
    return i;
}
int count_line(FILE *input) //Количество строк в матрице
{
    fseek(input, 0L, SEEK_SET);
    int ch, count=0;
    while ( (ch=fgetc(input)) != EOF )
        if ( ch == '\n' )
            ++count;
    return count+1;
}
int count_column(FILE *input) //Количество столбцов в матрице
{
    fseek(input, 0L, SEEK_SET);
    return count_all_elem(input)/count_line(input);
}
void matrix_to_vector(double *matrix, FILE *input) //перевод матрицу в векторное представление
{
    for (int i=0; i<302; i++)
        matrix[i]=0;
    int column=count_column(input);
    fseek(input, 0L, SEEK_SET);
    int num_elem_in_string=1, i=1, num_string=1;
    double sym;
    while(fscanf(input,"%lf", &sym)!=EOF)
    {
        if (num_elem_in_string>=column)
```

```
{
     if (sym!=0)
     {
          matrix[i]=num_elem_in_string;
          matrix[i+1]=sym;
          i+=2;
          num_elem_in_string=1;
     }
     else
          num_elem_in_string=1;
}
else if (num_elem_in_string==1)
{
     if (sym!=0)
     {
          matrix[i]=0;
          matrix[i+1]=num_string;
          matrix[i+2]=num_elem_in_string;
          matrix[i+3]=sym;
          i+=4;
          num_elem_in_string++;
          num_string++;
     }
     else
     {
          matrix[i]=0;
          matrix[i+1]=num_string;
          i+=2;
          num_elem_in_string++;
          num_string++;
     }
```

```
            }
            else if (num_elem_in_string!=1)
            {
                if (sym!=0)
                {
                    matrix[i]=num_elem_in_string;
                    matrix[i+1]=sym;
                    i+=2;
                    num_elem_in_string++;
                }
                else
                    num_elem_in_string++;
            }
        }
}
int size(double *matrix, FILE *input) // размер матрицы, записанной в векторе
{
    int size_matrix=1;
    while(matrix[size_matrix]!=0||matrix[size_matrix+1]!=0
)
        size_matrix++;
    return size_matrix+2;
}
void print_matrix(double *matrix, FILE *input) //Печать матрицы в обычном виде, пользуясь данными только из векторного представления
{
    int size_matrix=size(matrix, input);
    int max_line=1, i=1, max_column=1, res=0;
    for (int i=1; i<size_matrix; i++)
        if (matrix[i]==0 && matrix[i+1]>max_line)
            max_line=matrix[i+1];
```

```
    for (int i=1; i<size_matrix; i+=2)
        if (matrix[i]>max_column)
            max_column=matrix[i];
    for (int f=1; f<=max_line; f++)
    {
        printf("\n");
        do
        {
            i+=2;
            if (matrix[i]-matrix[i-2]==1)
            {
                printf(" %4.1lf ", matrix[i+1]);
            }
            else if (matrix[i]-matrix[i-2]>1)
            {
                double count=matrix[i]-matrix[i-2];
                for(int k=1; k<count; k++)
                    printf(" %4.1lf ", zero);
                printf(" %4.1lf ", matrix[i+1]);
            }
            else   if   (matrix[i]==0   &&   matrix[i-
        2]!=max_column)
            {
                res=max_column-matrix[i-2]+1;
                for(int o=1; o<res; o++)
                    printf(" %4.1lf ", zero);
            }
        }
        while (matrix[i]!=0);
    }
    printf("\n");
}
```

```
void print_array(double *matrix, FILE *input) //Печать матрицы в векторном представлении
{
    int size_matrix=size(matrix, input);
    printf("\n");
    for (int l=1; l<size_matrix; l++)
        printf("%4.1lf ", matrix[l]);
    printf("\n");
}
void vector_trans(double *matrix, FILE *input) //нахождение строк с наибольшим числом элементов относительно вектора.
{
    double summ[100];
    int count[100];
    for(int i = 0; i < 100; i++)
    {
        count[i]=0;
        summ[i]=0;
    }
    int size_matrix=size(matrix, input), max_line=1;
    for (int i=1; i<size_matrix; i++)
        if (matrix[i]==0 && matrix[i+1]>max_line)
            max_line=matrix[i+1];
    int l=1,string=0;
    for(int a = 1; a < max_line+1; a++)
    {
        string++;
        do
        {
            l+=2;
            if (matrix[l]!=0)
            {
```

```
                count[string]++;
                summ[string]+=matrix[l+1];
            }
        }
        while (matrix[l]!=0);
    }
    int max=0;
    for(int i = 1; i < 100; i++)
        if (count[i]>max)
            max=count[i];
    for(int i = 1; i < 100; i++)
        if (count[i]==max && max!=0)
            printf("\nНомер строки: %d; Сумма элементов:
        %4.1lf", i, summ[i]);
        else if (max==0)
        {
            printf("\nОшибка: Нулевая матрица!");
            break;
        }
    printf("\n");
}
void matrix_trans(FILE *input) //нахождение строк с
наибольшим числом элементов относительно матрицы
{
    double sym, summ[100];
    int count[100];
    for(int i = 0; i < 100; i++)
    {
        count[i]=0;
        summ[i]=0;
    }
    int max=0, k=0, line = 1, column=count_column(input);
```

```
    fseek(input, 0L, SEEK_SET);
    while(fscanf(input,"%lf", &sym)!=EOF)
    {
        k++;
        if (k>column)
        {
            k=1;
            line++;
        }
        if (sym!=0)
        {
            count[line]++;
            summ[line]+=sym;
        }
    }
    for(int i = 1; i < 100; i++)
        if (count[i]>max)
            max=count[i];
    for(int i = 1; i < 100; i++)
        if (count[i]==max && max!=0)
            printf("\nНомер строки: %d; Сумма элементов:
            %4.1lf", i, summ[i]);
        else if (max==0)
        {
            printf("\nОшибка: Нулевая матрица!");
            break;
        }
    printf("\n");
}
void print_menu()
{
```

```
    printf("\n 1. Вывести матрицу в векторном
    представлении. \n 2. Вывести матрицу, используя
    векторное представление. \n 3. Нахождение строк с
    наибольшим числом элементов, используя векторное
    представление. \n 4. Нахождение строк с наибольшим
    числом элементов, используя обычное представление\n 5.
    Выйти из программы. \n Выберите действие: ");
}
void action_menu(FILE *input)
{
    double matrix[320];
    matrix_to_vector(matrix, input);
    print_menu();
    int act;
    while(scanf("%d", &act)!=EOF)
    {
        switch (act)
        {
            case 1:
                print_array(matrix, input);
                break;
            case 2:
                print_matrix(matrix, input);
                break;
            case 3:
                vector_trans(matrix, input);
                break;
            case 4:
                matrix_trans(input);
                break;
            case 5:
                exit(0);
                break;
```

```
            default:
                printf("\nПожалуйста, выберите один из
                представленных пунктов меню!\n");
                break;
        }
        printf("\n Выберите действие:");
    }
}
int main(int argc, char* *argv)
{
    FILE *input=fopen(argv[1], "r");
    action_menu(input);
    fclose(input);
}
```

8. Демонстрация работы программы

8.1. Файлы для тестирования

file1.txt:

1 0 0

0 0 2

0 22.2 0

file2.txt:

1.43 0.4 -2.1 0

0 0 0 1

4 0 0 0

0 1 0 0

0 3.1 0 0

file3.txt:

32.14 43.1 0 0 0 0

0 0 9 0 0 0.99

21 0 0 0 1 0

0 0 0 43.1 0 0

file4.txt:

0 0 0 0 0 1

0 0 0 0 1 0

0 0 0 1 0 0

0 0 1 0 0 0

0 1 0 0 0 0

1 0 0 0 0 0

file5.txt:

0 0 0 21 0 0 0

0 0 0 3.1 0 0 0

0 0 0 1.2 0 0 0

12 1.2 12.3 1.1 1 2 3

0 0 0 4 0 0 0

0 0 0 -2 0 0 0

0 0 0 3 0 0 0

file6.txt:

390.21 0 1

30 0 1

file7.txt:

890 0 0 0 0 0 0

0 0 0 0 0 0 -21.1

1 2 3 4 5 6 7

file8.txt:

1 845 22 2 1

0 2 -23 1.1 3

0 0 3 2 1

0 0 0 4 -1.5

0 0 0 0 -89.9

file9.txt:

0 1

2 0

2 0

0 3

3 0

0 -234.24

file10.txt:

-1 0 0 0 3

0 -1 0 0 -4

0 0 -1 0 -2.5

0 0 0 -1 -1.23

11 12 13 14 -1

8.2. Тестирование

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ gcc -o k.exe kurs7.c

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file1.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 1.0 0.0 2.0 3.0 2.0 0.0 3.0 2.0 22.2 0.0 0.0

Выберите действие:2

1.0 0.0 0.0
0.0 0.0 2.0
0.0 22.2 0.0

Выберите действие:3

Номер строки: 1; Сумма элементов: 1.0
Номер строки: 2; Сумма элементов: 2.0
Номер строки: 3; Сумма элементов: 22.2

Выберите действие:4

Номер строки: 1; Сумма элементов: 1.0
Номер строки: 2; Сумма элементов: 2.0
Номер строки: 3; Сумма элементов: 22.2

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file2.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 1.4 2.0 0.4 3.0 -2.1 0.0 2.0 4.0 1.0 0.0 3.0 1.0 4.0 0.0 4.0 2.0 1.0 0.0 5.0 2.0 3.1 0.0 0.0

Выберите действие:2

1.4 0.4 -2.1 0.0

0.0 0.0 0.0 1.0

4.0 0.0 0.0 0.0

0.0 1.0 0.0 0.0

0.0 3.1 0.0 0.0

Выберите действие:3

Номер строки: 1; Сумма элементов: -0.3

Выберите действие:4

Номер строки: 1; Сумма элементов: -0.3

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file3.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 32.1 2.0 43.1 0.0 2.0 3.0 9.0 6.0 1.0 0.0 3.0 1.0 21.0 5.0 1.0 0.0 4.0 4.0 43.1 0.0 0.0

Выберите действие:2

32.1 43.1 0.0 0.0 0.0 0.0

0.0 0.0 9.0 0.0 0.0 1.0

21.0 0.0 0.0 0.0 1.0 0.0

0.0 0.0 0.0 43.1 0.0 0.0

Выберите действие:3

Номер строки: 1; Сумма элементов: 75.2

Номер строки: 2; Сумма элементов: 10.0

Номер строки: 3; Сумма элементов: 22.0

Выберите действие:4

Номер строки: 1; Сумма элементов: 75.2

Номер строки: 2; Сумма элементов: 10.0

Номер строки: 3; Сумма элементов: 22.0

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file4.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 6.0 1.0 0.0 2.0 5.0 1.0 0.0 3.0 4.0 1.0 0.0 4.0 3.0 1.0 0.0 5.0 2.0 1.0 0.0 6.0 1.0 1.0 0.0 0.0

Выберите действие:2

0.0 0.0 0.0 0.0 0.0 1.0

```
0.0  0.0  0.0  0.0  1.0  0.0
0.0  0.0  0.0  1.0  0.0  0.0
0.0  0.0  1.0  0.0  0.0  0.0
0.0  1.0  0.0  0.0  0.0  0.0
1.0  0.0  0.0  0.0  0.0  0.0
```

Выберите действие:3

Номер строки: 1; Сумма элементов: 1.0
Номер строки: 2; Сумма элементов: 1.0
Номер строки: 3; Сумма элементов: 1.0
Номер строки: 4; Сумма элементов: 1.0
Номер строки: 5; Сумма элементов: 1.0
Номер строки: 6; Сумма элементов: 1.0

Выберите действие:4

Номер строки: 1; Сумма элементов: 1.0
Номер строки: 2; Сумма элементов: 1.0
Номер строки: 3; Сумма элементов: 1.0
Номер строки: 4; Сумма элементов: 1.0
Номер строки: 5; Сумма элементов: 1.0
Номер строки: 6; Сумма элементов: 1.0

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file5.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 4.0 21.0 0.0 2.0 4.0 3.1 0.0 3.0 4.0 1.2 0.0 4.0 1.0 12.0
2.0 1.2 3.0 12.3 4.0 1.1 5.0 1.0 6.0 2.0 7.0 3.0 0.0 5.0 4.0 4.0
0.0 6.0 4.0 -2.0 0.0 7.0 4.0 3.0 0.0 0.0

Выберите действие:2

0.0 0.0 0.0 21.0 0.0 0.0 0.0
0.0 0.0 0.0 3.1 0.0 0.0 0.0
0.0 0.0 0.0 1.2 0.0 0.0 0.0
12.0 1.2 12.3 1.1 1.0 2.0 3.0
0.0 0.0 0.0 4.0 0.0 0.0 0.0
0.0 0.0 0.0 -2.0 0.0 0.0 0.0
0.0 0.0 0.0 3.0 0.0 0.0 0.0

Выберите действие:3

Номер строки: 4; Сумма элементов: 32.6

Выберите действие:4

Номер строки: 4; Сумма элементов: 32.6

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file6.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 390.2 3.0 1.0 0.0 2.0 1.0 30.0 3.0 1.0 0.0 0.0

Выберите действие:2

390.2 0.0 1.0

30.0 0.0 1.0

Выберите действие:3

Номер строки: 1; Сумма элементов: 391.2

Номер строки: 2; Сумма элементов: 31.0

Выберите действие:4

Номер строки: 1; Сумма элементов: 391.2

Номер строки: 2; Сумма элементов: 31.0

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file7.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 890.0 0.0 2.0 7.0 -21.1 0.0 3.0 1.0 1.0 2.0 2.0 3.0 3.0 4.0 4.0 5.0 5.0 6.0 6.0 7.0 7.0 0.0 0.0

Выберите действие:2

890.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0 0.0 -21.1

1.0 2.0 3.0 4.0 5.0 6.0 7.0

Выберите действие:3

Номер строки: 3; Сумма элементов: 28.0

Выберите действие:4

Номер строки: 3; Сумма элементов: 28.0

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file8.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 1.0 2.0 845.0 3.0 22.0 4.0 2.0 5.0 1.0 0.0 2.0 2.0 2.0 3.0 -23.0 4.0 1.1 5.0 3.0 0.0 3.0 3.0 3.0 4.0 2.0 5.0 1.0 0.0 4.0 4.0 4.0 5.0 -1.5 0.0

5.0 5.0 -89.9 0.0 0.0

Выберите действие:2

1.0 845.0 22.0 2.0 1.0

0.0 2.0 -23.0 1.1 3.0

0.0 0.0 3.0 2.0 1.0

0.0 0.0 0.0 4.0 -1.5

0.0 0.0 0.0 0.0 -89.9

Выберите действие:3

Номер строки: 1; Сумма элементов: 871.0

Выберите действие:4

Номер строки: 1; Сумма элементов: 871.0

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file9.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 2.0 1.0 0.0 2.0 1.0 2.0 0.0 3.0 1.0 2.0 0.0 4.0 2.0 3.0
0.0 5.0 1.0 3.0 0.0 6.0 2.0 -234.2 0.0 0.0

Выберите действие:2

0.0 1.0

2.0 0.0

2.0 0.0

0.0 3.0

3.0 0.0

0.0 -234.2

Выберите действие:3

Номер строки: 1; Сумма элементов: 1.0

Номер строки: 2; Сумма элементов: 2.0

Номер строки: 3; Сумма элементов: 2.0

Номер строки: 4; Сумма элементов: 3.0

Номер строки: 5; Сумма элементов: 3.0

Номер строки: 6; Сумма элементов: -234.2

Выберите действие:4

Номер строки: 1; Сумма элементов: 1.0

Номер строки: 2; Сумма элементов: 2.0

Номер строки: 3; Сумма элементов: 2.0

Номер строки: 4; Сумма элементов: 3.0

Номер строки: 5; Сумма элементов: 3.0

Номер строки: 6; Сумма элементов: -234.2

Выберите действие:5

kircatle@DESKTOP-70J5NO3:/mnt/c/Kirill/Dev/LabsMAI/SecondSem/course7$ ./k.exe file10.txt

1. Вывести матрицу в векторном представлении.

2. Вывести матрицу, используя векторное представление.

3. Нахождение строк с наибольшим числом элементов, используя векторное представление.

4. Нахождение строк с наибольшим числом элементов, используя обычное представление

5. Выйти из программы.

Выберите действие: 1

0.0 1.0 1.0 -1.0 5.0 3.0 0.0 2.0 2.0 -1.0 5.0 -4.0 0.0 3.0 3.0 -1.0 5.0 -2.5 0.0 4.0 4.0 -1.0 5.0 -1.2 0.0 5.0 1.0 11.0 2.0 12.0 3.0 13.0 4.0 14.0 5.0 -1.0 0.0 0.0

Выберите действие:2

-1.0 0.0 0.0 0.0 3.0
 0.0 -1.0 0.0 0.0 -4.0
 0.0 0.0 -1.0 0.0 -2.5
 0.0 0.0 0.0 -1.0 -1.2
11.0 12.0 13.0 14.0 -1.0

Выберите действие:3

Номер строки: 5; Сумма элементов: 49.0

Выберите действие:4

Номер строки: 5; Сумма элементов: 49.0

Выберите действие:5

Заключение

Разреженные матрицы используются для хранения сравнительно небольшого объема данных, которые располагаются в большой области данных. Реализация разреженных матриц связана со значительными издержками, и это делает их все более непрактичными по мере заполнении области данных значимыми величинами, и в тоже время при большей разреженности эти структуры становятся более эффективными.

Были разобраны 4 варианта представления разреженных матриц:

1. Цепочка ненулевых элементов в векторе со строчным индексированием.

2. Один вектор

3. Три вектора

4. Два вектора

Литература

Методические указания к выполнению курсовых работ. Зайцев В. Е.

https://learnc.info/c/text_files.html

http://www.sbp-program.ru/c/sbp-file-c.htm

http://files.mai.ru/site/priem/documents/orders/2017/111_03.08.2017.pdf