Национальный исследовательский университет «Московский авиационный институт»

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

**ПО КУРСУ “ПРАКТИКУМ НА ЭВМ”**

**1 СЕМЕСТР ЗАДАНИЕ №4**

**“РАЗРЕЖЕННЫЕ МАТРИЦЫ”**

|  |  |
| --- | --- |
| Выполнил студент | Мохляков Павел Александрович |
| Группа | М80-108Б-19 |
| Преподаватель: | Поповкин Александр Викторович |
| Дата |  |
| Оценка |  |

Москва

2020

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc1111452900)

[ЗАДАНИЕ 2](#_Toc1892869717)

[ОСНОВНОЙ МЕТОД РЕШЕНИЯ 3](#_Toc948572980)

[ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ 4](#_Toc1736437164)

[ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ 5](#_Toc1944776340)

[ОПИСАНИЕ ПРОГРАММЫ 6](#_Toc470322930)

[АЛГОРИТМЫ РАБОТЫ 6](#_Toc1815828153)

[ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ 6](#_Toc250974967)

[ИСПОЛЬЗУЕМЫЕ ПЕРЕМЕНЫЕ 7](#_Toc1126547967)

[ПРОТОКОЛ 8](#_Toc850685514)

[ЗАКЛЮЧЕНИЕ 16](#_Toc1947361098)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc1663083247)

# ЗАДАНИЕ

Составиь программу на языке Си с процедурами и функциями для обработки прямоугольных разряженных матриц с элиментами целого типа, которая:

1. Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате, с одновременным размещением ненулевых элиментов в разреженной матрице в соответсвенной заданной схеме.
2. Печатает введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном виде.
3. Выполняет необходимые преобразования разреженных матрицпутем обращения к соответсвующим процедурам и функциям.
4. Печатает результат преобразования согласно заданной схеме размещения и в обычном виде.

В процедура и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании примеров.

ВАРИАНТ 16

Схема размещения: 2 вектора. Отображение на динамическую труктуру. Преобразование: умножить вектор-строку на разреженную матрицу и вычислить количество ненулевых элиментов результата.

# ОСНОВНОЙ МЕТОД РЕШЕНИЯ

Программа считывает файл два раза,первый раз она находит рамерности матриц, считая количество символов, находящихся после чисел. Далее во второй раз мы считываем все числа,зная размеры наших матриц мы можем расчитать их положение.

Считывая данные мы записываем их в структуру матрицы, которая состоит из размеров матрицы и указателей на два списка, которые содержат даные не нулевых ячеек и их положение в матрице. Далее происходит умножение матриц, считываются размеры матриц, проверяется можно ли их умножить и получаем размер произведения. Умножаем матрицы по обычному алгоритму, получая значения в нужной ячейке, путем расчета ее положения и если в структуре нет ее данных, то ее значение приравнивается нулю. Полученное значение записывается в структуру.

Печатаем матрицу во внутреннем представлении и в стандартном виде.

# ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

|  |  |
| --- | --- |
| Аппаратное обеспечение | Ноутбук на базу Intel Core i5 |
| Операционная система | Manjaro 5.4.33 |
| Язык и система программирования | GNU C |
| Число строк | 160+180+30 |
| Компиляция программы в терминале | Zsh 5.8 |

Таблица А.1 - Общие сведение о программе

# ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программы предназначены для записи, чтения и поиска в структуре матрицы на языке Си. Программа поиска работает с временной сложностью алгоритма .

# ОПИСАНИЕ ПРОГРАММЫ

## АЛГОРИТМЫ РАБОТЫ

1. Подключаем необходимые библиотеки
2. Создаем служебные функции
3. Создаем структуру данных
4. Считываем данные из файла
5. Преобразование данных
6. Вывод данных

## ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Таблица А.2 - Функции файла main.c

|  |  |  |
| --- | --- | --- |
| Название | Аргументы и их тип | Описание функции |
| int main() |  | Создает указатели на матрицы, считывает данные из файла и формирует вывод |
| void size\_matr() | int \*i1,int \*j1,int \*i2,int \*j2 | Считывает из файла и формирует размера матриц |

Таблица А.3 - Функции файла matrix.c

|  |  |  |
| --- | --- | --- |
| Название | Аргументы и их тип | Описание функции |
| struct Matrix\* End\_Matrix() | struct Matrix \*mat | Добавляет позицию конца. |
| struct Matrix\* Mult\_Matrix() | struct Matrix \*am,  struct Matrix \*bm | Умножаит матрицы |
| void Print\_Matrix\_WR() | struct Matrix \*mat | Выводит матрицы во внитреннием представлении |
| struct Matrix\* Create\_Matrix() | struct Matrix \*mat,int n,int m | Создает матрицу |
| struct Matrix\* Push\_Matrix() | struct Matrix \*mat,int i,int j,  int data | Записывает значение матрицы в структуру |
| int Peek\_Matrix() | struct Matrix \*mat,int i,int j | Возвращает значение матрицы |
| void Print\_Matrix\_Fis() | struct Matrix \*mat | Выводит матрицы в обычном виде |
| struct Node\* Create\_Node() | struct Node \*top | Создает список |
| struct Node\* Push\_Node() | struct Node \*top,int data | Добавляет элимент в список |
| void Peek\_Node() | struct Node \*top,int ind,int \*dat | Возвращает значение из списка |
| int Size\_Node() | struct Node \*top | Возвращает размер списка |
| void Print\_Node() | struct Node \*top | Выводит список |

# ИСПОЛЬЗУЕМЫЕ ПЕРЕМЕНЫЕ

Таблица А.4 - Общие переменные

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной | Начальное значение | Тип | Назначение |
| fl |  | \*FILE | Файл |
| m,n |  | int | Размеры матрицы |
| name |  | Char\* | Има файла |
| size |  | int | Длина объекта |
| flag |  | int | Переключатель |
| data |  | int | Данные |
| pos |  | int | Позиция по формуле |
| i,j,k |  | int | Положение в матрице |
| mat,am,bm,cm |  | struct Matrix\* | Матрицы |
| top |  | struct Node\* | Вершина списка |

Таблица А.5 - Переменные Peek\_Matrix() matrix.c

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной | Начальное значение | Тип | Назначение |
| index | -1 | int | Номер ячейки |

# ПРОТОКОЛ

pavel@lenovo  ~/Programs/C/kp7   master ●  cat main.c    
#include <stdio.h>   
#include "matrix.h"   
  
void size\_matr(int \*i1,int \*j1,int \*i2,int \*j2)   
{   
 int n1=0;   
 int m1=0;   
 int n2=0;   
 int m2=0;   
 FILE \*fl;   
 char name[]="martix";   
 if ((fl = fopen(name, "r")) == NULL)   
 {   
   printf("Не удалось открыть файл");   
 }   
 int sim;   
 sim=fgetc(fl);   
 int flag=0;   
 int k=0;   
 while(sim!=EOF)   
 {   
   if((sim>=48)&&(sim<=57)) flag=1;   
   else   
   {   
     if(flag==1)   
     {   
       m1++;   
       flag=0;   
     }   
   }   
   if(sim==10) break;   
   sim=fgetc(fl);   
 }   
 flag=1;   
 while(sim!=EOF)   
 {   
   if((sim>=48)&&(sim<=57))   
   {   
     flag=1;   
     k=0;   
   }   
   else   
   {   
     if(sim==10)   
     {   
       if(flag==1)   
       {   
         n1++;   
       }   
       k++;   
     }   
     else k=0;   
     flag=0;   
     if(k==2) break;   
   }   
   //printf("%d,%d\n",sim,k);   
   sim=fgetc(fl);   
 }   
 sim=fgetc(fl);   
 flag=0;   
 k=0;   
 while(sim!=EOF)   
 {   
   if((sim>=48)&&(sim<=57)) flag=1;   
   else   
   {   
     if(flag==1)   
     {   
       m2++;   
       flag=0;   
     }   
   }   
   if(sim==10) break;   
   sim=fgetc(fl);   
 }   
 flag =1;   
 while(sim!=EOF)   
 {   
   if((sim>=48)&&(sim<=57))   
   {   
     flag=1;   
     k=0;   
   }   
   else   
   {   
     if(sim==10)   
     {   
       if(flag==1)   
       {   
         n2++;   
       }   
       k++;   
     }   
     else k=0;   
     flag=0;   
     if(k==2) break;   
   }   
   sim=fgetc(fl);   
 }   
 \*i1=n1;   
 \*j1=m1;   
 \*i2=n2;   
 \*j2=m2;   
 fclose(fl);   
}   
  
  
int main()   
{   
 FILE \*fl;   
 char name[]="martix";   
 //------------------------   
 struct Matrix \*am=NULL;   
 struct Matrix \*bm=NULL;   
 struct Matrix \*cm=NULL;   
 int n1,m1,n2,m2;   
 size\_matr(&n1,&m1,&n2,&m2);   
 am=Create\_Matrix(am,n1,m1);   
 bm=Create\_Matrix(bm,n2,m2);   
 if ((fl = fopen(name, "r")) == NULL)   
 {   
   printf("Не удалось открыть файл");   
   return 0;   
 }   
 for(int i=1;i<=n1;i++)   
 {   
   for(int j=1;j<=m1;j++)   
   {   
     int l;   
     fscanf(fl,"%d",&l);   
     Push\_Matrix(am,i,j,l);   
   }   
 }   
 for(int i=1;i<=n2;i++)   
 {   
   for(int j=1;j<=m2;j++)   
   {   
     int l;   
     fscanf(fl,"%d",&l);   
     Push\_Matrix(bm,i,j,l);   
   }   
 }   
 fclose(fl);   
 End\_Matrix(am);   
 End\_Matrix(bm);   
 printf("Матрица А по схеме размещения:\n");   
 Print\_Matrix\_WR(am);   
 printf("Матрица А в еслественном виде:\n");   
 Print\_Matrix\_Fis(am);   
  
 printf("Матрица Б по схеме размещения:\n");   
 Print\_Matrix\_WR(bm);   
 printf("Матрица Б в еслественном виде:\n");   
 Print\_Matrix\_Fis(bm);   
  
 cm=Mult\_Matrix(am,bm);   
 End\_Matrix(cm);   
 printf("Матрица С (ответ) по схеме размещения:\n");   
 Print\_Matrix\_WR(cm);   
 printf("Матрица С (ответ) в еслественном виде:\n");   
 Print\_Matrix\_Fis(cm);   
  
 printf("В матрице С %d ненулевых элиментов",Size\_Node(cm->Data));   
 return 0;   
}   
pavel@lenovo  ~/Programs/C/kp7   master ●  cat matrix.h    
#ifndef \_MATRIX\_   
#define \_MATRIX\_   
  
struct Matrix{   
 int m;   
 int n;   
 struct Node \*Data;   
 struct Node \*Pos;   
};   
  
struct Node{   
 int index;   
 int data;   
 struct Node \*next;   
 struct Node \*previous;   
};   
  
  
struct Matrix\* End\_Matrix(struct Matrix \*mat);   
struct Matrix\* Mult\_Matrix(struct Matrix \*am,struct Matrix \*bm);   
void Print\_Matrix\_WR(struct Matrix \*mat);   
struct Matrix\* Create\_Matrix(struct Matrix \*mat,int n,int m);   
struct Matrix\* Push\_Matrix(struct Matrix \*mat,int i,int j,int data);   
int Peek\_Matrix(struct Matrix \*mat,int i,int j);   
void Print\_Matrix\_Fis(struct Matrix \*mat);   
//--------------------------------------------------------------------------   
struct Node\* Create\_Node(struct Node \*top);   
struct Node\* Push\_Node(struct Node \*top,int data);   
void Peek\_Node(struct Node \*top,int ind,int \*dat);   
int Size\_Node(struct Node \*top);   
void Print\_Node(struct Node \*top);   
  
#endif   
pavel@lenovo  ~/Programs/C/kp7   master ●  cat matrix.c   
#include <stdio.h>   
#include <stdlib.h>   
#include "matrix.h"   
  
  
  
struct Matrix\* End\_Matrix(struct Matrix \*mat)   
{   
 if(mat!=NULL)Push\_Node(mat->Pos,-1);   
 return mat;   
}   
struct Matrix\* Mult\_Matrix(struct Matrix \*am,struct Matrix \*bm)   
{   
 if(am->m==bm->n)   
 {   
   struct Matrix \*cm = NULL;   
   cm = Create\_Matrix(cm,am->n,bm->m);   
   for(int i=1;i<=am->n;i++)   
   {   
     for(int j=1;j<=bm->m;j++)   
     {   
       int ch=0;   
       for(int k=1;k<=bm->n;k++)   
       {   
         int a=Peek\_Matrix(am,i,k);   
         int b=Peek\_Matrix(bm,k,j);   
         ch=ch+(a\*b);   
       }   
       Push\_Matrix(cm,i,j,ch);   
     }   
   }   
   return cm;   
 }   
 else   
 {   
   printf("Невозможно умножить\n");   
   return NULL;   
 }   
}   
  
void Print\_Matrix\_Fis(struct Matrix \*mat)   
{   
 if(mat!=NULL){   
 int n=mat->n;   
 int m=mat->m;   
 for(int i=1;i<=n;i++)   
 {   
   for(int j=1;j<=m;j++)   
   {   
     printf("%d\t",Peek\_Matrix(mat,i,j));   
   }   
   printf("\n");   
 }   
}   
}   
  
int Peek\_Matrix(struct Matrix \*mat,int i,int j)   
{   
 int data;   
 int pos=(i-1)\*(mat->m)+j-1;   
 int index=-1;   
 struct Node \*n=mat->Pos;   
 while(n->data!=-1)   
 {   
   if(n->data==pos)   
   {   
     index=n->index;   
     break;   
   }   
   n=n->next;   
 }   
 if(index==-1) return 0;   
 else   
 {   
   Peek\_Node(mat->Data,index,&data);   
   return data;   
 }   
}   
  
void Print\_Matrix\_WR(struct Matrix \*mat)   
{   
 if(mat!=NULL)   
 {   
   int size = Size\_Node(mat->Pos);   
   int data;   
   for(int i=0;i<size;i++) printf("%d\t",i);   
   printf("\n");   
   for(int i=0;i<size;i++)   
   {   
     Peek\_Node(mat->Pos,i,&data);   
     printf("%d\t",data);   
   }   
   printf("\n");   
   for(int i=0;i<size-1;i++)   
   {   
     Peek\_Node(mat->Data,i,&data);   
     printf("%d\t",data);   
   }   
   printf("\n");   
 }   
}   
  
struct Matrix\* Create\_Matrix(struct Matrix \*mat,int n,int m)   
{   
 if(mat==NULL)   
 {   
   mat=malloc(sizeof(struct Matrix));   
   mat->Data=NULL;   
   mat->Pos=NULL;   
   mat->m=m;   
   mat->n=n;   
 }   
 return mat;   
}   
  
struct Matrix\* Push\_Matrix(struct Matrix \*mat,int i,int j,int data)   
{   
 if((mat!=NULL)&&(data!=0))   
 {   
   mat->Data=Push\_Node(mat->Data,data);   
   int l=(i-1)\*(mat->m)+j-1;   
   mat->Pos=Push\_Node(mat->Pos,l);   
 }   
 return mat;   
}   
  
//------------------------------------------------------------------------------   
  
struct Node\* Create\_Node(struct Node \*top)   
{   
 if(top==NULL)   
 {   
   top=malloc(sizeof(struct Node));   
   top->index=-1;   
   top->data=0;   
   top->next=NULL;   
   top->previous=NULL;   
 }   
 return top;   
}   
  
struct Node\* Push\_Node(struct Node \*top,int data)   
{   
 if(top==NULL) top=Create\_Node(top);   
 if(top->index == -1)   
 {   
   top->data=data;   
   top->index=0;   
   return top;   
 }   
 else   
 {   
   int i=0;   
   struct Node \*q=top;   
   while(q->next!=NULL)   
   {   
     q=q->next;   
     i++;   
   }   
   q->next=Create\_Node(q->next);   
   q->next->data=data;   
   q->next->index=i+1;   
   q->next->previous=q;   
 }   
 return top;   
}   
  
void Peek\_Node(struct Node \*top,int ind,int \*dat)   
{   
 while((top->index!=ind)&&(top->next!=NULL)) top=top->next;   
 if(top->index==ind) \*dat=top->data;   
}   
  
int Size\_Node(struct Node \*top)   
{   
 if(top!=NULL)   
 {   
   while(top->next!=NULL) top=top->next;   
   return (top->index)+1;   
 }   
 else return 0;   
}   
  
void Print\_Node(struct Node \*top)   
{   
   while(top!=NULL)   
   {   
     printf("%d\t",top->data);   
     top=top->next;   
   }   
   printf("\n");   
}   
pavel@lenovo  ~/Programs/C/kp7   master ●  cat Makefile    
CC=gcc   
  
CFLAGS=-g -c -Wall   
  
all: kp7   
  
kp7: main.o matrix.o   
       $(CC) -g main.o matrix.o -o prog   
  
main.o: main.c   
       $(CC) $(CFLAGS) main.c   
  
matrix.o: matrix.c   
       $(CC) $(CFLAGS) matrix.c   
  
clean:   
       rm -rf \*.o prog   
pavel@lenovo  ~/Programs/C/kp7   master ●  cat martix    
1 4 8 9 5 3 8 9   
  
1 0 0 0 1   
0 1 0 1 0   
0 1 0 0 0   
0 0 0 0 0   
0 0 0 1 0   
0 0 0 1 1   
0 0 0 0 0   
0 1 0 0 0   
pavel@lenovo  ~/Programs/C/kp7   master ●  ./prog    
Матрица А по схеме размещения:   
0       1       2       3       4       5       6       7       8   
0       1       2       3       4       5       6       7       -1   
1       4       8       9       5       3       8       9   
Матрица А в еслественном виде:   
1       4       8       9       5       3       8       9   
Матрица Б по схеме размещения:   
0       1       2       3       4       5       6       7       8       9   
0       4       6       8       11      23      28      29      36      -1   
1       1       1       1       1       1       1       1       1   
Матрица Б в еслественном виде:   
1       0       0       0       1   
0       1       0       1       0   
0       1       0       0       0   
0       0       0       0       0   
0       0       0       1       0   
0       0       0       1       1   
0       0       0       0       0   
0       1       0       0       0   
Матрица С (ответ) по схеме размещения:   
0       1       2       3       4   
0       1       3       4       -1   
1       21      12      4   
Матрица С (ответ) в еслественном виде:   
1       21      0       12      4   
В матрице С 4 ненулевых элиментов                                                                                                                                                                                                            
pavel@lenovo  ~/Programs/C/kp7   master ●  cat martix   
1 4 8 9 5 3 8 9   
  
1 0 0 0 1   
0 1 0 1 0   
0 1 0 0 0   
0 0 0 0 0   
0 0 0 1 0   
0 0 0 1 1   
0 0 0 0 0   
pavel@lenovo  ~/Programs/C/kp7   master ●  ./prog       
Матрица А по схеме размещения:   
0       1       2       3       4       5       6       7       8   
0       1       2       3       4       5       6       7       -1   
1       4       8       9       5       3       8       9   
Матрица А в еслественном виде:   
1       4       8       9       5       3       8       9   
Матрица Б по схеме размещения:   
0       1       2       3       4       5       6       7       8   
0       4       6       8       11      23      28      29      -1   
1       1       1       1       1       1       1       1   
Матрица Б в еслественном виде:   
1       0       0       0       1   
0       1       0       1       0   
0       1       0       0       0   
0       0       0       0       0   
0       0       0       1       0   
0       0       0       1       1   
0       0       0       0       0   
Невозможно умножить   
 pavel@lenovo  ~/Programs/C/kp7   master ●  cat martix    
1 4 8 9 5 3 8 9   
  
1 0 0 0 1 0 0 0   
0 1 0 1 0 0 1 0   
0 1 0 0 0 1 0 0   
0 0 0 0 0 0 0 1   
0 0 0 1 0 0 0 0   
0 0 0 1 1 0 1 0   
0 0 0 0 0 0 0 0   
1 0 0 0 1 0 0 1   
pavel@lenovo  ~/Programs/C/kp7   master ●  ./prog       
Матрица А по схеме размещения:   
0       1       2       3       4       5       6       7       8   
0       1       2       3       4       5       6       7       -1   
1       4       8       9       5       3       8       9   
Матрица А в еслественном виде:   
1       4       8       9       5       3       8       9   
Матрица Б по схеме размещения:   
0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15   
0       4       9       11      14      17      21      31      35      43      44      46      56      60      63      -1   
1       1       1       1       1       1       1       1       1       1       1       1       1       1       1   
Матрица Б в еслественном виде:   
1       0       0       0       1       0       0       0   
0       1       0       1       0       0       1       0   
0       1       0       0       0       1       0       0   
0       0       0       0       0       0       0       1   
0       0       0       1       0       0       0       0   
0       0       0       1       1       0       1       0   
0       0       0       0       0       0       0       0   
1       0       0       0       1       0       0       1   
Матрица С (ответ) по схеме размещения:   
0       1       2       3       4       5       6       7   
0       1       3       4       5       6       7       -1   
10      12      12      13      8       7       18   
Матрица С (ответ) в еслественном виде:   
10      12      0       12      13      8       7       18   
В матрице С 7 ненулевых элиментов                                                                                                                                                                                                           
pavel@lenovo  ~/Programs/C/kp7   master ●  cat martix    
1 4 8 9 5   
  
5 0 0 0   
0 87 0 13   
0 9 0 0   
0 0 0 0   
0 0 0 100   
pavel@lenovo  ~/Programs/C/kp7   master ●  ./prog       
Матрица А по схеме размещения:   
0       1       2       3       4       5   
0       1       2       3       4       -1   
1       4       8       9       5   
Матрица А в еслественном виде:   
1       4       8       9       5   
Матрица Б по схеме размещения:   
0       1       2       3       4       5   
0       5       7       9       19      -1   
5       87      13      9       100   
Матрица Б в еслественном виде:   
5       0       0       0   
0       87      0       13   
0       9       0       0   
0       0       0       0   
0       0       0       100   
Матрица С (ответ) по схеме размещения:   
0       1       2       3   
0       1       3       -1   
5       420     552   
Матрица С (ответ) в еслественном виде:   
5       420     0       552   
В матрице С 3 ненулевых элиментов

ЗАКЛЮЧЕНИЕ

В данной работе я изучил работу с разряженными матрицами, структуры для их хранения и работы с ними, и реализовал знания на практике.

В программа ищет каждую ячейку в матрице при полном объходе структуры, что является ее недостатком, но для его решения придется изменять строение и принцип работы структуры, что идет в противоречие с заданием.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. РосДиплом, Оформление таблиц в дипломной работе, особенности и

требования ГОСТ / Электронный диплом / Режим доступа: <https://www.rosdiplom.ru/rd/pubdiplom/view.aspx?id=288>

1. Диплом Журнал, Оформление курсовой работы по ГОСТу

2019(образец) / Электронный диплом / Режим доступа: <https://journal.duplom.ru/kursovaya/oformlenie-kursov..>

1. Vyuchit.work – универсальная методичка / Электронный диплом / Режим доступа: <https://vyuchit.work/samorazvitie/sekretyi/oformlenie..>
2. Архив вопросов и ответов для программистов / Электронный диплом / Режим доступа: <https://qarchive.ru/320864_parametry_gcc__lm__lz__lrt..>
3. Компилятор GCC / Электронный диплом / Режим доступа: <http://parallel.uran.ru/book/export/html/25>
4. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования С, 2-е издание. :Пер. с англ. – М. : Издательский дом «Вильямс», 2009. – 304 с. : ил. –
5. Умножение разреженных матриц: <https://www.intuit.ru/studies/courses/4447/983/lecture/14931?page=5>