Московский авиационный институт

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

КУРСОВОЙ ПРОЕКТ

по курсу “Практикум на ЭВМ”

II семестр

Тема “Разреженные матрицы”

Работу выполнила:

Студентка группы М8О-106Б-21

Деревянко Екатерина Андреевна

Работу проверил:

Старший преподаватель

Дубинин Алексей Владимирович

Москва 2022

Оглавление

**[Введение 3](#_Toc31672)**

**[Теоретическая часть 4](#_Toc21018)**

[Разреженные матрицы 4](#_Toc7332)

**[Практическая часть 7](#_Toc28518)**

[О программе 7](#_Toc19889)

[Протокол 7](#_Toc9742)

[Тесты 12](#_Toc7792)

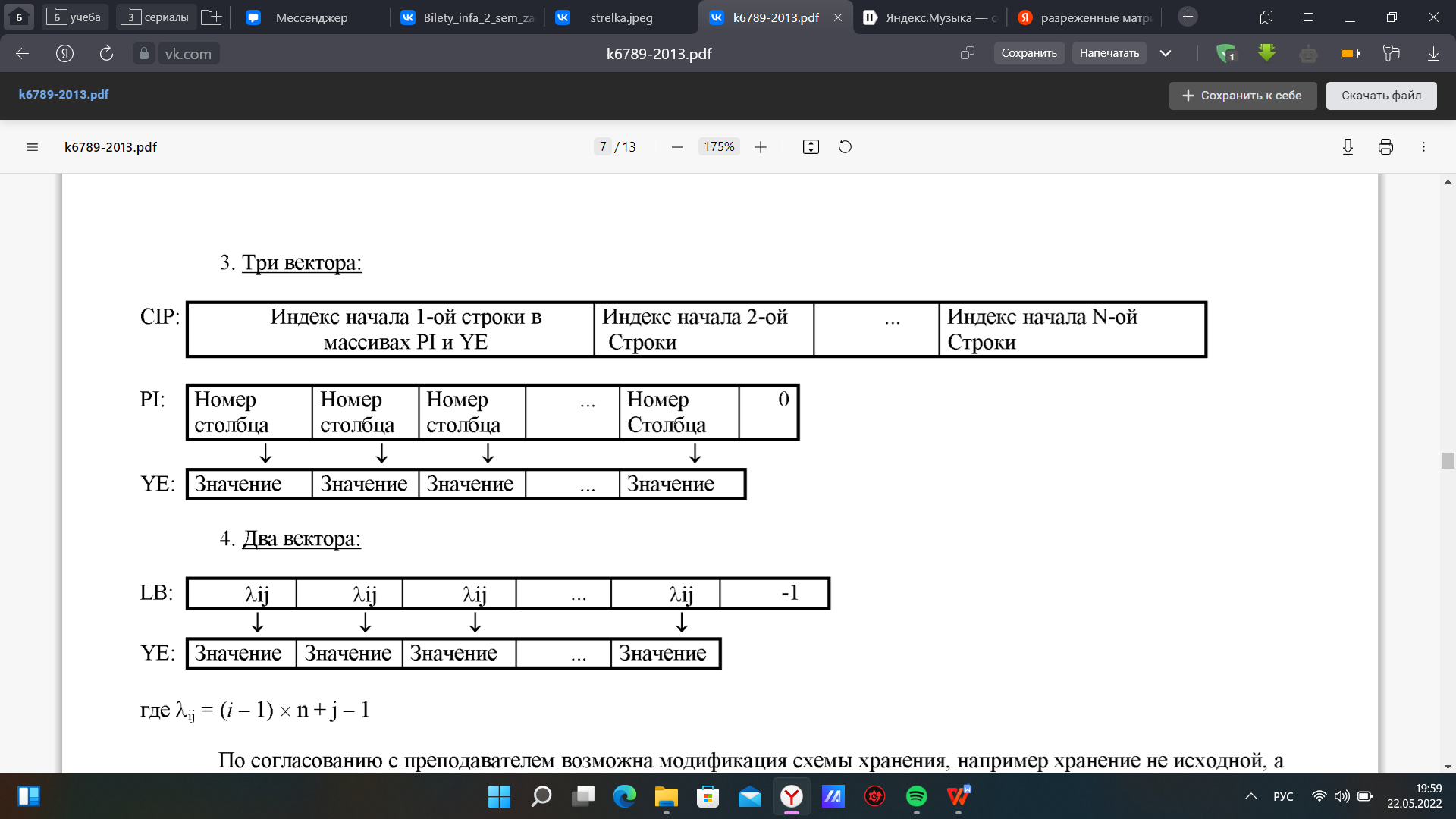
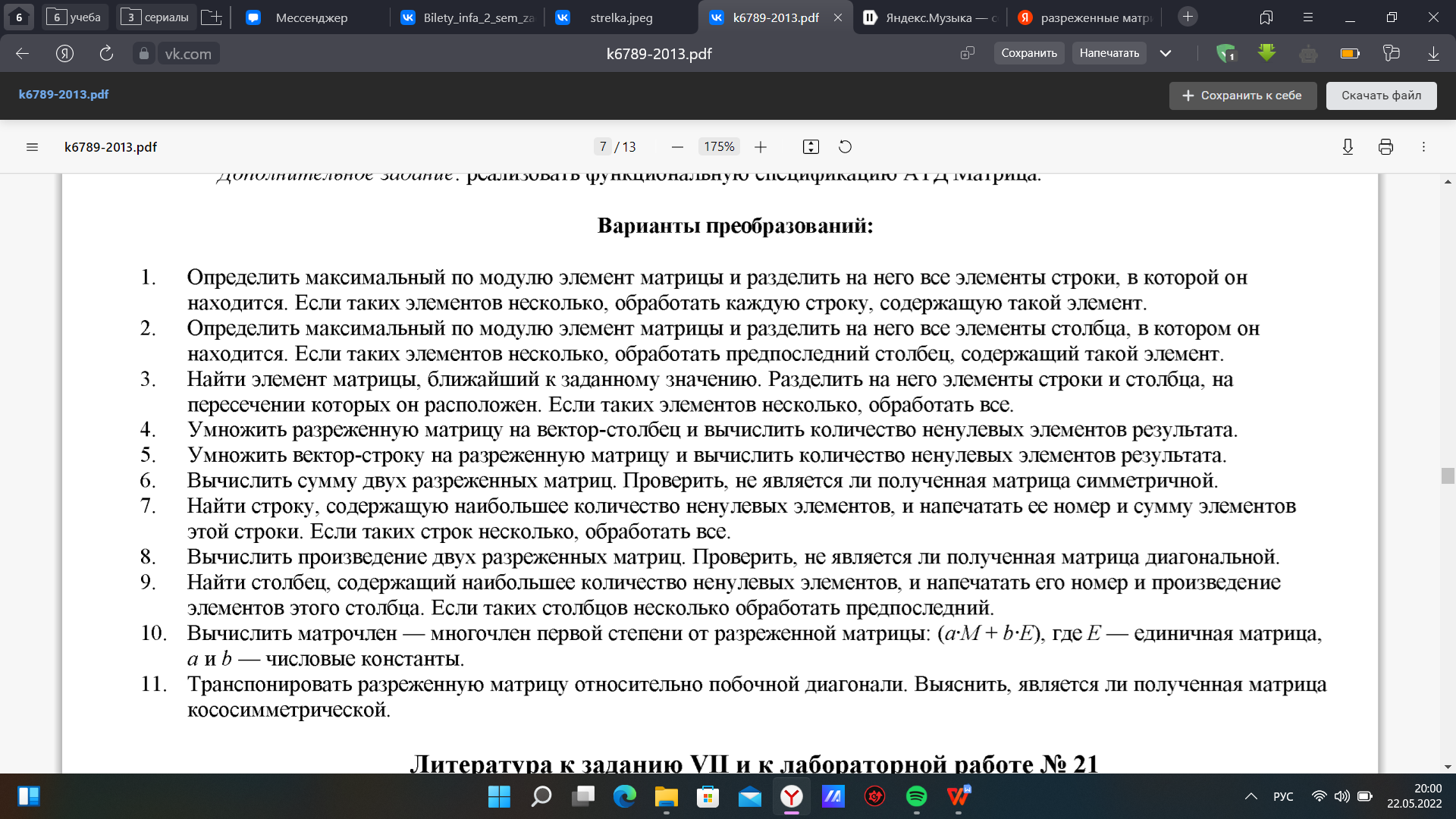
**[Заключение 14](#_Toc24399)**

**[Список источников 15](#_Toc21329)**

# Введение

Матрица может иметь сколь угодно большой размер, но при этом состоять преимущественно из нулей. Хранить в памяти компьютера все ячейки матрицы выйдет очень затратно, поэтому нужны методы, которые позволят сохранять только те значения, которые не равны нулю. Чтобы разобраться с этой темой будет необходимо составить программу для обработки прямоугольных разреженных матриц.

**Вариант:**



# Теоретическая часть

## Разреженные матрицы

Сначала разберемся, что такое разреженные матрицы и как их можно хранить в памяти компьютера.

Разрежённая матрица — матрица с преимущественно нулевыми элементами. В противном случае, если большая часть элементов матрицы ненулевые, матрица считается плотной. Интерес к разреженности возникает потому, что его использование может привести к огромной вычислительной экономии и потому, что многие крупные матричные проблемы, возникающие на практике, редки.

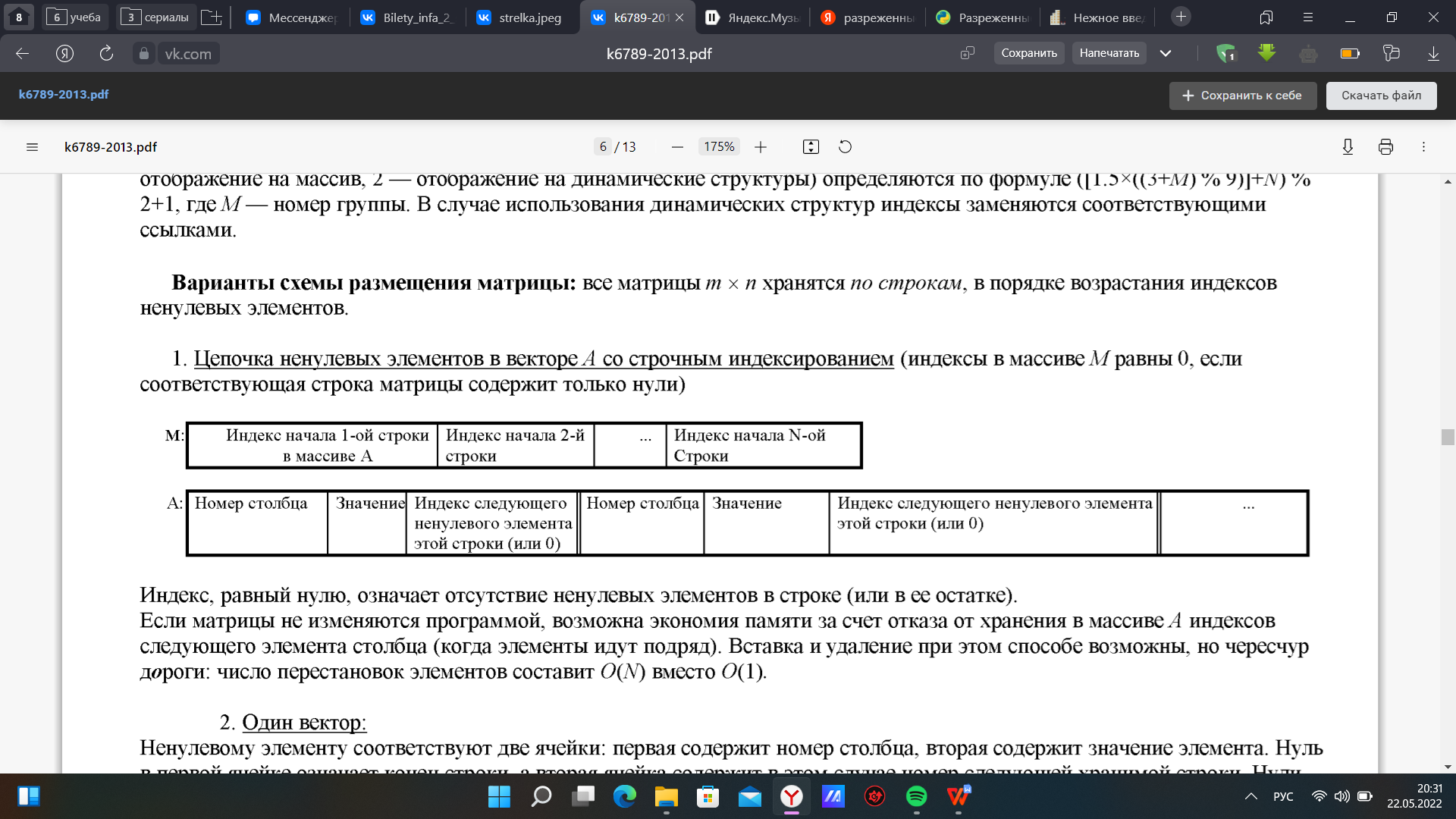
Разреженность количественно можно оценить с помощью показателя разреженности, равному числу нулевых значений в матрице, деленное на общее количество элементов в матрице.

При подходе к работе с такими матрицами «в лоб» появляются две проблемы. Первая - хранение каждого элемента приведет к огромной затрате памяти, потому что для каждого нуля, который не несет никакой информации, будет выделена отдельная память. И вторая проблема - большое время, затраченное на операции с разреженными матрицами, большая часть выполненных вычислений будет включать сложение или умножение нулевых значений вместе.

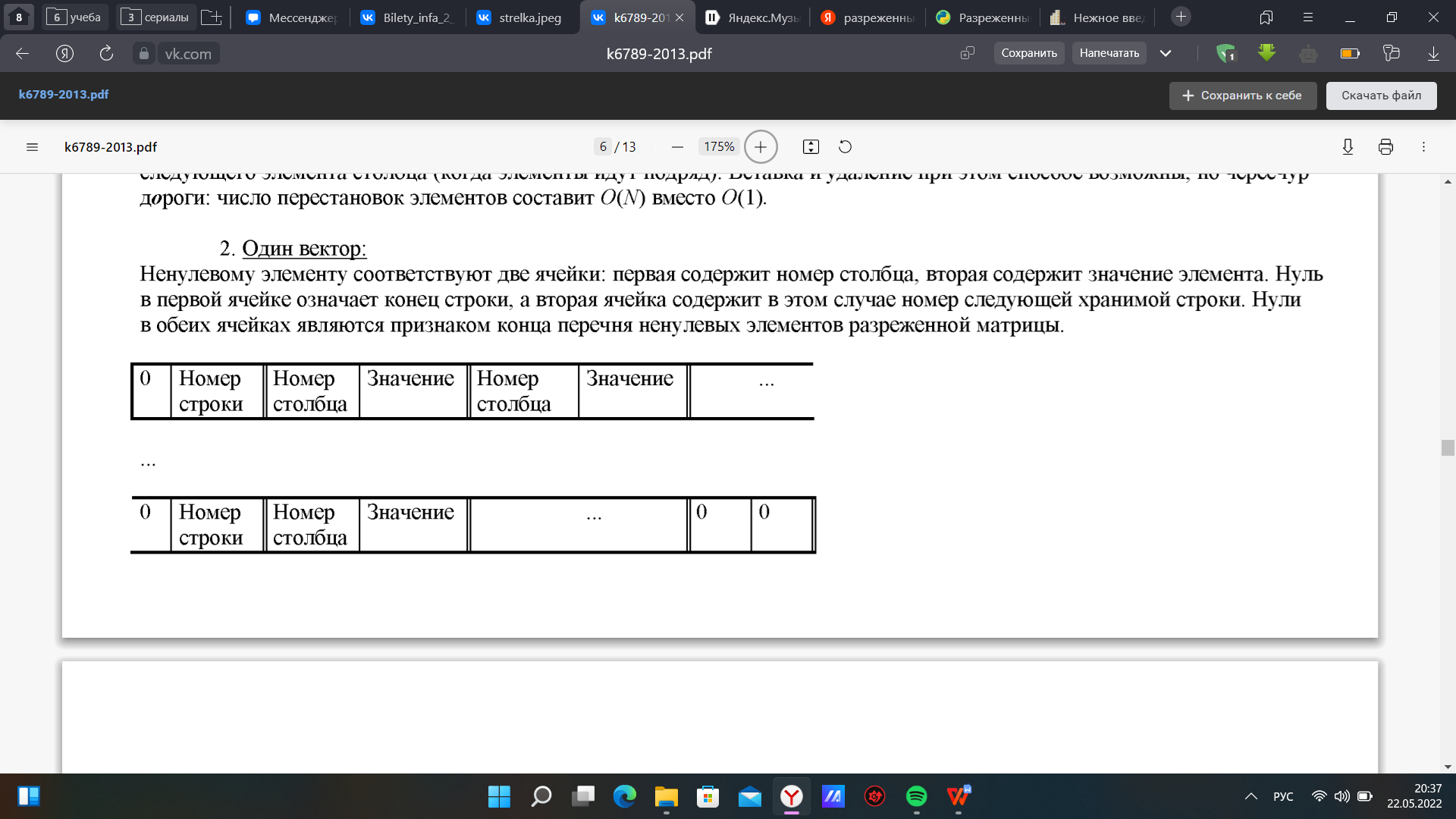
Решение для представления и работы с разреженными матрицами заключается в использовании альтернативной структуры данных для представления разреженных данных. Нулевые значения можно игнорировать и записывать только ненулевые.

Рассмотрим несколько методов представления разреженных матриц.

*Цепочка ненулевых элементов в векторе со строчным индексированием.*

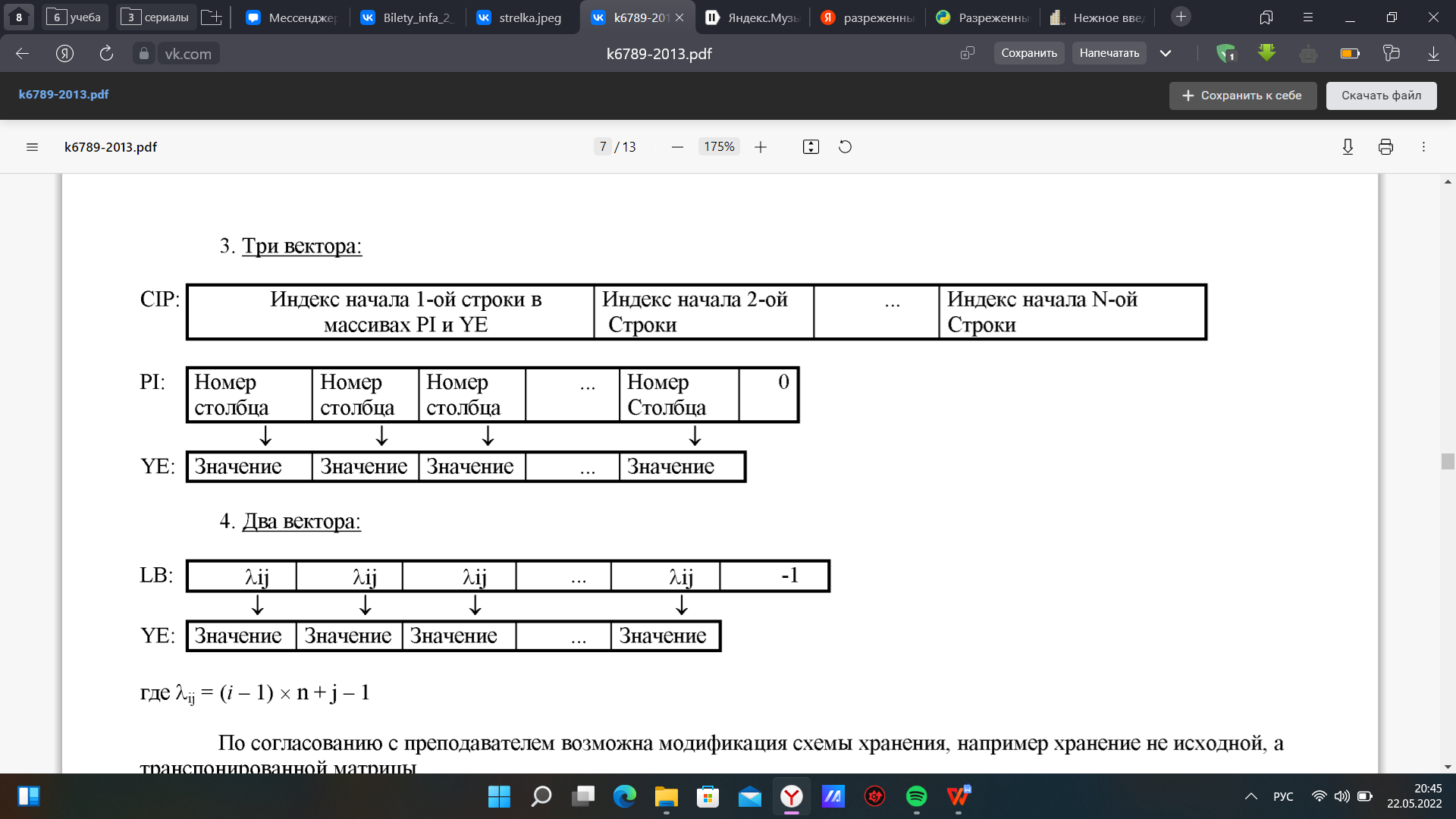


В массиве M хранятся индексы массива А, с которых начинается новая строка. Если в строке были только нулевые элементы, то в массив М записывается, например, -1.

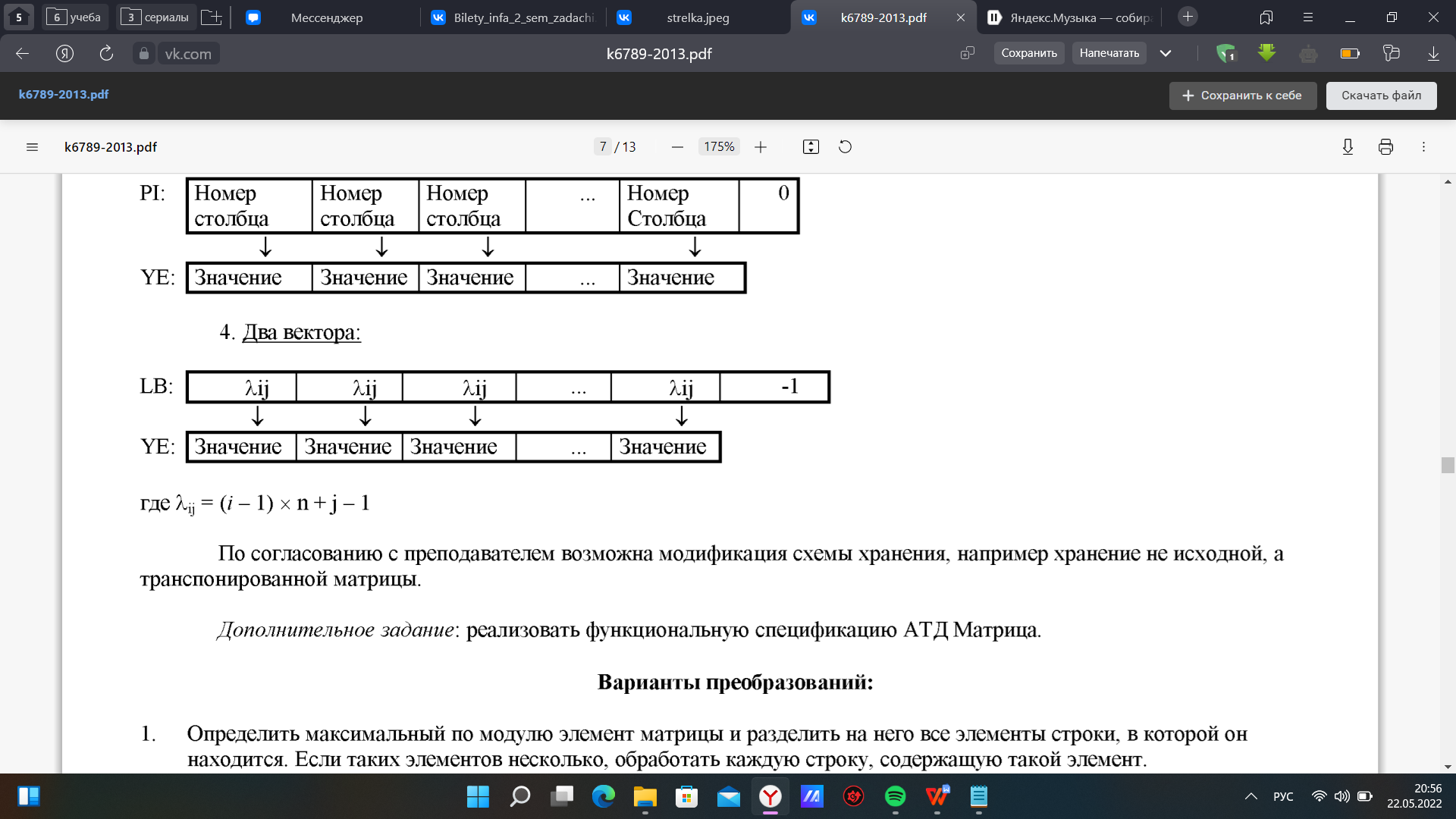
*Один вектор*

Ненулевому элементу соответствуют две ячейки: номер столбца и само значение. Если в ячейке номера строки находится, например, -1, то значит строка закончена, а в следующей ячейке находиться номер следующей строки. Если в обеих ячейках -1, то ненулевые элементы закончились.

*Три вектора*

При таком хранение в одном векторе хранятся ненулевые значения, во втором векторе соответствующие этим значениям столбцы, а в третьем индексы предыдущих векторов, с которых начинается строка.

*Два вектора*



В первом векторе хранится номер элемента при сквозной нумерации матрицы, он считается по формуле . Во втором векторе хранятся соответствующие ненулевые значения.

# Практическая часть

## О программе

Для представления матрицы мне достался вариант три вектора. В качестве преобразования надо было найти максимальный по модулю элемент и разделить на него каждую строку, в которой он встречается. На ввод подается сначала размер матрицы, а затем сама матрица. Программа выводит преобразованную матрицу и три вектора, соответствующие ей. Сложность ввода и вывода матрицы O(n\*m), где n и m - размеры матрицы. Сложность поиска максимального элемента матрицы O(k), где k - количество ненулевых элементов. Сложность деления строк, в которых содержится максимальный элемент, на него O(k+n), где l - количество ненулевых элементов строки.

## Протокол

*Реализация векторов*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "vector\_dbl.h"

#include "vector\_int.h"

double get\_dbl(v\_dbl \*v, int i){

return v->begin[i];

}

int get\_int(v\_int \*v, int i){

return v->begin[i];

}

void print\_dbl(v\_dbl \*v) {

for (int i = 0; i < v->size; ++i) {

printf("%lf ", get\_dbl(v, i));

}

printf("\n");

}

void print\_int(v\_int \*v) {

for (int i = 0; i < v->size; ++i) {

printf("%d ", get\_int(v, i));

}

printf("\n");

}

bool is\_empty\_dbl(v\_dbl \*v) {

return v->size == 0;

}

bool is\_empty\_int(v\_int \*v) {

return v->size == 0;

}

bool is\_full\_dbl(v\_dbl \*v) {

return v->size == v->allocated;

}

bool is\_full\_int(v\_int \*v) {

return v->size == v->allocated;

}

size\_t size\_dbl(v\_dbl \*v) {

return v->size;

}

size\_t size\_int(v\_int \*v) {

return v->size;

}

void create\_dbl(v\_dbl \*v) {

v->size = 0;

v->allocated = 0;

v->begin = NULL;

}

void create\_int(v\_int \*v) {

v->size = 0;

v->allocated = 0;

v->begin = NULL;

}

void destroy\_dbl(v\_dbl \*v) {

free(v->begin);

v->size = 0;

v->allocated = 0;

v->begin = NULL;

}

void destroy\_int(v\_int \*v) {

free(v->begin);

v->size = 0;

v->allocated = 0;

v->begin = NULL;

}

void push\_dbl(v\_dbl \*v, double value) {

if (is\_full\_dbl(v)) {

v->allocated += 10;

v->begin = realloc(v->begin, v->allocated \* sizeof(double));

}

v->begin[v->size] = value;

v->size++;

}

void push\_int(v\_int \*v, int value) {

if (is\_full\_int(v)) {

v->allocated += 10;

v->begin = realloc(v->begin, v->allocated \* sizeof(int));

}

v->begin[v->size] = value;

v->size++;

}

double pop\_dbl(v\_dbl \*v) {

double res = get\_dbl(v, v->size - 1);

v->size--;

return res;

}

int pop\_int(v\_int \*v) {

int res = get\_int(v, v->size - 1);

v->size--;

return res;

}

double fabs(double a) {

if (a < 0) {

return -a;

}

return a;

}

double max(v\_dbl \*YE) {

double max\_el = get\_dbl(YE, 0);

for (int i = 0; i < size\_dbl(YE); i++) {

if (fabs(get\_dbl(YE, i)) > max\_el) {

max\_el = get\_dbl(YE, i);

}

}

return max\_el;

}

*Работа с матрицей*

#include <stdio.h>

#include <stdlib.h>

#include "vector\_dbl.h"

#include "vector\_int.h"

#include "matrix.h"

void print\_line(v\_int \*CIP, v\_int \*PI, v\_dbl \*YE, int n, int i){

int begin = 0;

int end = 0;

begin = get\_int(CIP, i);

if (i == size\_int(CIP) - 1) {

end = size\_dbl(YE);

} else {

end = get\_int(CIP, i+1);

}

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

if ((begin<end) && get\_int(PI, begin)==j+1){

printf("%lf ", get\_dbl(YE, begin));

begin++;

}

else {

printf("%d ", 0);

}

}

}

void print\_matrix(v\_int \*CIP, v\_int \*PI, v\_dbl \*YE, int m, int n){

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

print\_line(CIP, PI, YE, n, i);

printf("\n");

}

}

void task(v\_int \*CIP, v\_dbl \*YE) {

int begin = 0;

int end = 0;

double max\_el = max(YE);

int i=0;

while (i < size\_dbl(YE)){

if (get\_dbl(YE, i) == max\_el) {

for (int j = 0; j < size\_int(CIP); j++) {

begin = get\_int(CIP, j);

if (j == size\_int(CIP) - 1) {

end = size\_dbl(YE);

} else {

end = get\_int(CIP, j+1);

}

if ((begin <= i) && (end > i)) {

for (int k = begin; k < end; k++) {

YE->begin[k] = get\_dbl(YE, k)/max\_el;

}

i=end;

break;

}

}

}

else {

i++;

}

}

}

*Основная программа*

#include <stdio.h>

#include <stdlib.h>

#include "vector\_dbl.h"

#include "vector\_int.h"

#include "matrix.h"

int main(void)

{

int m, n;

double num;

v\_int CIP, PI;

v\_dbl YE;

printf("Input size: ");

scanf("%d%d", &m, &n);

int count = 0;

create\_int(&CIP);

create\_int(&PI);

create\_dbl(&YE);

for (int i = 1; i <= m; i++) {

push\_int(&CIP, count);

for (int j = 1; j <= n; j++) {

scanf("%lf", &num);

if (num != 0) {

push\_int(&PI, j);

push\_dbl(&YE, num);

count++;

}

}

}

printf("\n");

if (is\_empty\_dbl(&YE)) {

printf("Can't divide by zero\n");

destroy\_int(&CIP);

return 1;

}

task(&CIP, &YE);

print\_matrix(&CIP, &PI, &YE, m, n);

printf("\n");

printf("CIP: ");

print\_int(&CIP);

printf("PI: ");

print\_int(&PI);

printf("YE: ");

print\_dbl(&YE);

destroy\_int(&CIP);

destroy\_int(&PI);

destroy\_dbl(&YE);

return 0;

}

## Тесты

Input size: 4 5

0 0 0 0 9

0 0 0 0 0

0 9 1 18 0

0 0 0 0 1

0 0 0 0 9.000000

0 0 0 0 0

0 0.500000 0.055556 1.000000 0

0 0 0 0 1.000000

CIP: 0 1 1 4

PI: 5 2 3 4 5

YE: 9.000000 0.500000 0.055556 1.000000 1.000000

--------------------------------------------------------------------

Input size: 2 3

0 0 0

0 0 0

Can't divide by zero

# Заключение

Разреженные матрицы требуют особого представления для работы с ними. Существует несколько вариантов этого представления, которые были рассмотрены в данной работе. Каждый обладает своими преимуществами и недостатками и нюансами при работе с ними.

# Список источников

1. О разреженных матрицах

<https://www.machinelearningmastery.ru/sparse-matrices-for-machine-learning/>

1. Зайцев В.Е. «Практикум по циклу дисциплин «Информатика»»