**«Московский Авиационный Институт»**

(Национальный Исследовательский Университет)

**Институт: №8 «Прикладная математика и информатика»**

**Кафедра: 806 «Вычислительная математика и программирование»**

Курсовая работа

II семестр

По теме

«Разреженные матрицы»

|  |  |
| --- | --- |
| Группа | М8О-107Б-20 |
| Студент | Чекменев В.А. |
| Преподаватель | Найдёнов И.Е. |
| Оценка |  |
| Дата |  |

Москва, 2021

**Постановка задачи**

Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разреженных матриц, которая:

1) Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой

2) Печатает введённые матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном(естественном виде)

3) Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путём обращения к соответствующим процедурам и/или функциям

4) Печатает результат преобразования согласно заданной схеме размещения и в обычном виде

**Теория**

Разреженная матрица-это матрица с преимущественно нулевыми элементами. В противном случае, если бо́льшая часть элементов матрицы ненулевые, матрица считается ***плотной***.

**Схема размещения матрицы:**

2. 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 разреженной матрицы.

**Вариант преобразования**

Вычислить произведение двух разреженных матриц. Проверить, является ли полученная матрица диагональной.

**Алгоритм решения поставленной задачи**

Для обработки разреженных матриц опишем структуру вектора с его множеством операций и реализуем вектор на Си. Отдельно опишем функции для обработки разреженных матриц:

1) Считывание матриц в обычном виде из файла с преобразованием в вектор согласно заданной схеме размещения

2) Выполнение заданного преобразования

3) Печать вектора(схемы размещения ненулевых элементов разреженной матрицы)

4) Печать матрицы в естественном виде

**Листинг программ**

**vector.h – структура вектора + прототипы функций.**

*#ifndef VECTOR\_H*

*#define VECTOR\_H*

*typedef struct \_vector vector;*

*struct \_vector*

*{*

*int size;*

*int \*data;*

*int count\_elem;*

*};*

*void create\_vector(vector \*v);*

*int size(vector \*v);*

*void resize(vector \*v);*

*void push\_back(vector \*v, int value);*

*int count\_elements(vector \*v);*

*int is\_empty(vector \*v);*

*void print\_vector(vector \*v);*

*void destroy(vector \*v);*

*#endif*

**matrix.h – прототипы функций.**

*#ifndef MATRIX\_H*

*#define MATRIX\_H*

*#include "vector.h"*

*#define SIZE 30*

*int cnt\_elem(char name[SIZE]);*

*int cnt\_lines(char name[SIZE]);*

*vector \*matrix\_input(vector \*v);*

*void task\_print(vector \*v);*

*void natural\_print(vector \*v);*

*int search(vector \*v, int i, int j);*

*void multiply(vector \*v1, vector \*v2);*

*#endif*

**vector\_func.c – функции для работы с векторами в си.**

*#include <stdio.h>*

*#include <stdlib.h>*

*#include "vector.h"*

*#include "matrix.h"*

*void create\_vector(vector \*v)*

*{*

*v->size = 0;*

*v->data = (int \*) malloc(sizeof(int) \* v->size);*

*v->count\_elem = 0;*

*}*

*int size(vector \*v)*

*{*

*return v->size;*

*}*

*int is\_empty(vector \*v)*

*{*

*return (v->count\_elem == 0);*

*}*

*int count\_elements(vector \*v)*

*{*

*return v->count\_elem;*

*}*

*void resize(vector \*v)*

*{*

*v->size++;*

*v->data = realloc(v->data, sizeof(int) \* v->size);*

*}*

*void push\_back(vector \*v, int value)*

*{*

*if (v->size == v->count\_elem) {*

*resize(v);*

*}*

*v->count\_elem++;*

*v->data[v->count\_elem - 1] = value;*

*}*

*void print\_vector(vector \*v)*

*{*

*if (is\_empty(v)) {*

*printf("вектор пуст\n");*

*} else {*

*printf("(");*

*for (int i = 0; i < v->count\_elem; i++) {*

*if (i == v->count\_elem - 1) {*

*printf("%d", v->data[i]);*

*} else {*

*printf("%d ", v->data[i]);*

*}*

*}*

*printf(")\n");*

*}*

*}*

*void destroy(vector \*v)*

*{*

*v->count\_elem = 0;*

*v->size = 0;*

*free(v->data);*

*}*

**matrix\_func.c – функции для работы с матрицами в си.**

*#include <stdio.h>*

*#include <stdlib.h>*

*#include "matrix.h"*

*#include "vector.h"*

*int cnt\_elem(char name[SIZE])*

*{*

*int el = 0, cnt = 0;*

*FILE\* f = fopen(name, "r");*

*while (fscanf(f, "%d", &el) && !feof(f)) {*

*cnt++;*

*}*

*fclose(f);*

*return cnt;*

*}*

*int cnt\_lines(char name[SIZE])*

*{*

*int cnt = 0;*

*FILE\* f = fopen(name, "r");*

*while (!feof(f)) {*

*if (fgetc(f) == '\n') {*

*cnt++;*

*}*

*}*

*fclose(f);*

*return cnt;*

*}*

*vector \*matrix\_input(vector \*v)*

*{*

*create\_vector(v);*

*char name[SIZE];*

*printf("введи название файла с матрицей пж: ");*

*scanf("%s", name);*

*FILE \*f = fopen(name, "r");*

*if (f == NULL) {*

*printf("Файл не открылся, может создашь его...\n");*

*exit(1);*

*}*

*int n, m;*

*n = cnt\_lines(name);*

*m = cnt\_elem(name) / n;*

*int c[n][m];*

*int el;*

*// считывание*

*for (int i = 0; i < n; i++) {*

*for (int j = 0; j < m; j++) {*

*fscanf(f, "%d", &el);*

*c[i][j] = el;*

*}*

*}*

*// заполнение вектора*

*for (int p = 0; p < n; p++) {*

*int flag = 0;*

*//*

*for (int g = 0; g < m; g++) {*

*if (c[p][g] != 0) { // смотрим только ненулевые элементы*

*if (flag == 0) { // первый элемент в строке*

*//push\_back(&v, p + 1);*

*push\_back(v, p + 1); // кладем в вектор номер строки (тут конец строки)*

*flag = 1;*

*}*

*push\_back(v, g + 1); // кладем в вектор номер столбца*

*push\_back(v, c[p][g]); // и само значение*

*}*

*}*

*//*

*if (flag != 0) { // вектор заполеен*

*push\_back(v, 0);*

*}*

*}*

*push\_back(v, 0);*

*push\_back(v, n);*

*push\_back(v, m);*

*fclose(f);*

*return v;*

*}*

*void task\_print(vector \*v)*

*{*

*if (v != NULL) {*

*printf("логоческий вид: ( ");*

*for (int i = 0; i < v->size - 2; i++) {*

*printf("%d ", v->data[i]);*

*}*

*printf(")\n");*

*printf("физический вид:\n");*

*for (int i = 1; i <= v->data[v->size - 2]; i++) {*

*for (int j = 1; j <= v->data[v->size - 1]; j++) {*

*printf("%d ", search(v, i, j));*

*}*

*printf("\n");*

*}*

*}*

*}*

*void natural\_print(vector\* v)*

*{*

*int n = v->data[v->size - 2];*

*int m = v->data[v->size - 1];*

*int a[n][m];*

*for (int i = 0; i < n; i++) {*

*for (int j = 0; j < m; j++) {*

*a[i][j] = 0;*

*}*

*}*

*if (v->size == 0) {*

*return;*

*}*

*int l = 0; // индекс строки*

*int k = 0; // индекс столбца*

*int count = 0;*

*int value;*

*for (int i = 0; i < v->size - 5; i++) {*

*if (count == 0) {*

*l = v->data[i] - 1;*

*k = v->data[i + 1] - 1;*

*value = v->data[i + 2];*

*a[l][k] = value;*

*count++;*

*if (v->data[i + 3] == 0) {*

*count--;*

*i = i + 4;*

*} else {*

*i = i + 3;*

*}*

*} else {*

*k = v->data[i] - 1;*

*value = v->data[i + 1];*

*a[l][k] = value;*

*if (v->data[i + 2] == 0) {*

*count--;*

*i = i + 3;*

*} else {*

*i = i + 2;*

*}*

*}*

*}*

*printf("Естественная форма матрицы:\n");*

*for (int i = 0; i < n; i++) {*

*for (int j = 0; j < m; j++) {*

*printf("%d ", a[i][j]);*

*}*

*printf("\n");*

*}*

*}*

*int search(vector \*v, int i, int j)*

*{*

*int a = 0;*

*while (v->data[a] <= i) {*

*if (v->data[a] == i)*

*{*

*a++;*

*while (v->data[a] != j) {*

*if (v->data[a] != 0) {*

*a += 2;*

*} else {*

*return 0;*

*}*

*}*

*if (v->data[a] == j) {*

*return v->data[a + 1];*

*}*

*}*

*else*

*{*

*while (v->data[a] != 0) {*

*a++;*

*}*

*a++;*

*}*

*}*

*}*

*void multiply(vector \*v1, vector \*v2)*

*{*

*printf("файл с мат1: ");*

*v1 = matrix\_input(v1);*

*task\_print(v1);*

*printf("файл с мат2: ");*

*v2 = matrix\_input(v2);*

*task\_print(v2);*

*int g = v1->data[v1->count\_elem - 2];*

*int l = v2->data[v2->count\_elem - 1];*

*int c[g][l];*

*if (v1->data[v1->count\_elem - 1] != v2->data[v2->count\_elem - 2]) {*

*printf("умножение невозможно!\n");*

*}*

*else*

*{*

*for (int i = 1; i <= g; i++) { // строки первой*

*for (int j = 1; j <= l; j++) { // столбцы второй*

*c[i][j] = 0;*

*for (int k = 1; k <= v1->data[v1->count\_elem - 1]; k++) {*

*c[i][j] = c[i][j] + search(v1, i, k) \* search(v2, k, j);*

*}*

*}*

*}*

*printf("результат\n");*

*for (int i = 1; i <= g; i++) {*

*for (int j = 1; j <= l; j++) {*

*printf("%d ", c[i][j]);*

*}*

*printf("\n");*

*}*

*printf("диагональная?\n");*

*int flag = 1;*

*for (int i = 1; i <= g; i++) {*

*for (int j = 1; j <= l; j++) {*

*if (i != j && c[i][j] != 0) {*

*flag = 0;*

*break;*

*}*

*}*

*if (flag == 0) {*

*break;*

*}*

*}*

*if (flag == 1) printf("Да\n");*

*if (flag == 0) printf("Нет\n");*

*}*

*}*

**client.c – интерфейс программы.**

*#include <stdio.h>*

*#include "matrix.h"*

*#include "vector.h"*

*int main(void)*

*{*

*vector v, v1, v2;*

*printf("Напишите '?' для получения помощи в использовании программы:\n");*

*char c;*

*while ((c = getchar()) != EOF) {*

*if (c == '?') {*

*printf("Набор команд:\n");*

*printf("i - считать матрицу из файла и вывести в логическом виде, в виде вектора (следуйте указаниям в терминале).\n");*

*printf("t - выполнить задание (следуйте указаниям в терминале).\n");*

*printf("e - закончить сеанс.\n");*

*} else if (c == 'i') {*

*task\_print(matrix\_input(&v));*

*printf("готово\n");*

*} else if (c == 't') {*

*multiply(&v1, &v2);*

*printf("все, результат получен\n");*

*} else if (c == 'e') {*

*printf("все на сегодня...\n");*

*return 0;*

*} else if (c != '?' && c != 'c' && c != 'p' && c != 'f' && c != 'd' && c != '\n' && c != '\t' && c != ' ') {*

*printf("Не та буковка, попробуйте еще раз...\n");*

*}*

*}*

*return 0;*

*}*

**Тестирование программы**

**Листинг терминала, пользовательский ввод жирным шрифтом**

**Ответ на задачу подчеркнут**

[suraba04@asusx512fl cp7]$ .**/test1**

Напишите '?' для получения помощи в использовании программы:

**?**

Набор команд:

i - считать матрицу из файла и вывести в логическом виде, в виде вектора (следуйте указаниям в терминале).

t - выполнить задание (следуйте указаниям в терминале).

e - закончить сеанс.

**i**

введи название файла с матрицей пж: **data**

логоческий вид: ( 1 1 1 0 2 2 5 0 3 3 4 0 0 )

физический вид:

1 0 0

0 5 0

0 0 4

готово

**t**

файл с мат1: введи название файла с матрицей пж: **mat1**

логоческий вид: ( 1 1 1 2 3 0 2 1 3 2 2 3 2 0 3 2 2 0 0 )

физический вид:

1 3 0

3 2 2

0 2 0

файл с мат2: введи название файла с матрицей пж: **mat2**

логоческий вид: ( 1 1 1 0 2 2 1 0 3 3 1 0 0 )

физический вид:

1 0 0

0 1 0

0 0 1

**результат**

**1 3 0**

**3 2 2**

**0 2 0**

**диагональная?**

**Нет**

все, результат получен

**e**

все на сегодня...

[suraba04@asusx512fl cp7]$

**Вывод:**

Данная курсовая работа оказалась полезной и сложной, в ней я научился работать с разреженными матрицами со способом хранения – в одном векторе. Интереснее и сложнее всего было писать функцию search().