**«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 | Степанов Д.М. |
| 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**

Найти столбец, содержащий наибольшее количество ненулевых элементов, и напечатать его номер и произведение элементов элементов этого столбца. Если таких столбцов несколько, обработать предпоследний.

**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

**Описание структуры вектора и множества операций над ним**

*#ifndef vector\_h*

*#define vector\_h*

*#include <stdio.h>*

*#include <stdbool.h>*

*typedef struct \_vector vector;*

*struct \_vector*

*{*

*int size;*

*int\* data;*

*int elements\_count;*

*};*

*void vector\_create(vector\* v, int size);*

*int size(vector\* v);*

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

*void resize(vector\* v, int new\_size);*

*void size\_pp(vector\* v);*

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

*void load(vector\* v);*

*void destroy(vector\* v);*

*#endif*

**Реализация вектора на Си**

*#include <stdio.h>*

*#include <stdlib.h>*

*#include "vector.h"*

*#include "matrix.h"*

*#include <stdbool.h>*

*void vector\_create(vector\* v, int size)*

*{*

*v->size = size;*

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

*v->elements\_count = 0;*

*}*

*int size(vector\* v)*

*{*

*return v->size;*

*}*

*bool empty(vector\* v)*

*{*

*return v->size == 0;*

*}*

*void size\_pp(vector\* v)*

*{*

*v->size++;*

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

*}*

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

*{*

*if (v->size == v->elements\_count) {*

*size\_pp(v);*

*}*

*v->data[v->elements\_count++] = value;*

*}*

*void destroy(vector\* v)*

*{*

*v->size = 0;*

*v->elements\_count = 0;*

*free(v->data);*

*}*

**Описание операций для обработки разреженных матриц**

*#ifndef MATRIX\_H*

*#define MATRIX\_H*

*#include "vector.h"*

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

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

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

*void function(vector\* v);*

*void function1(vector\* v);*

*int el\_count(char name[20]);*

*int lines\_count(char name[20]);*

*#endif*

**Реализация операций для обработки разреженных матриц на Си**

*#include "matrix.h"*

*#include "vector.h"*

*#include <stdio.h>*

*#include <stdlib.h>*

*int el\_count(char name[20])*

*{*

*int a = 0;*

*int count = 0;*

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

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

*count++;*

*}*

*fclose(f);*

*return count;*

*}*

*int lines\_count(char name[20])*

*{*

*int count = 0;*

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

*while (!feof(f)) {*

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

*count++;*

*}*

*fclose(f);*

*return count;*

*}*

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

*{*

*int a;*

*char name[20];*

*vector\_create(v, 1);*

*int n, m;*

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

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

*if (f == NULL) {*

*printf("The file not exists\n");*

*exit(1);*

*}*

*n = lines\_count(name);*

*m = el\_count(name) / n;*

*int c[n][m];*

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

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

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

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

*}*

*}*

*int g = 0;*

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

*int k = 0;*

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

*if (c[p][g] != 0) {*

*if (k == 0) {*

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

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

*k++;*

*}*

*push\_back(v, g + 1);*

*push\_back(v, c[p][g]);*

*}*

*}*

*if (k != 0) {*

*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) {*

*return;*

*}*

*printf("Matrix pattern placing\n");*

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

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

*}*

*printf("\n");*

*}*

*void function1(vector\* v)*

*{*

*int count = 0;*

*int column\_count = 1;*

*vector a;*

*vector\_create(&a, 0);*

*for (int i = 0; i < v->size - 3; i++) {*

*if (v->data[i] != 0 && (i + 1) % 2 == 0) {*

*push\_back(&a, v->data[i]);*

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

*}*

*}*

*printf("\n");*

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

*return;*

*}*

*int ind = 0;*

*int e = 0;*

*int max = 1;*

*int old\_max = max;*

*for (int i = 0; i < a.size - 1; i++) {*

*if (a.data[i] == a.data[i + 1]) {*

*count++;*

*e++;*

*}*

*if (count > max - 1) {*

*max = count;*

*}*

*if (a.data[i] != a.data[i + 1]) {*

*count = 0;*

*}*

*}*

*if (e != 0) {*

*max++;*

*}*

*printf("%d\n", max);*

*}*

*void function(vector\* v)*

*{*

*vector a;*

*vector\_create(&a, 0);*

*for (int i = 0; i < v->size - 3; i++) {*

*if (v->data[i] != 0 && (i + 1) % 2 == 0) {*

*push\_back(&a, v->data[i]);*

*}*

*}*

*vector b;*

*vector\_create(&b, v->data[v->size - 1]);*

*for (int i = 0; i < b.size; i++) {*

*b.data[i] = 0;*

*}*

*for (int i = 0; i < a.size; i++) {*

*b.data[a.data[i] - 1]++;*

*}*

*int max = b.data[0];*

*for (int i = 0; i < b.size - 1; i++) {*

*if (b.data[i + 1] > max) {*

*max = b.data[i + 1];*

*}*

*}*

*int alone\_st = -1; // индекс единственного столбца*

*int back\_second\_st = -1; // индекс предпоследнего столбца*

*int k = 0;*

*int ind; // индекс столбца,который необходимо обработать*

*for (int i = b.size - 1; i >= 0; i--) {*

*if (b.data[i] == max && k == 0) {*

*alone\_st = i + 1;*

*k++;*

*} else if (b.data[i] == max && k == 1) {*

*back\_second\_st = i + 1;*

*k++;*

*}*

*}*

*if (back\_second\_st == -1) {*

*ind = alone\_st;*

*} else {*

*ind = back\_second\_st;*

*}*

*int p = 1;*

*for (int i = 0; i < v->size - 3; i++) {*

*if (v->data[i] != 0 && (i + 1) % 2 == 0 && v->data[i] == ind) {*

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

*}*

*}*

*//printf("The p of column ! %d\n", p);*

*printf("The elements composition of processed column: %d\n", p);*

*printf("The index of processed column: %d\n", ind);*

*}*

*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;) {*

*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("Natural form of matrix\n");*

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

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

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

*}*

*printf("\n");*

*}*

*}*

**Тестирование программы**

|  |  |
| --- | --- |
| Матрица из входного файла | Вывод программы |
| 1 0 0  0 5 0  0 0 4 | Matrix pattern placing  1 1 1 0 2 2 5 0 3 3 4 0 0  Natural form of matrix  1 0 0  0 5 0  0 0 4  The elements composition of processed column: 5  The index of processed column: 2 |
| 1 0 3 1  0 0 4 0  1 0 0 1 | Matrix pattern placing  1 1 1 3 3 4 1 0 2 3 4 0 3 1 1 4 1 0 0  Natural form of matrix  1 0 3 1  0 0 4 0  1 0 0 1  The elements composition of processed column: 12  The index of processed column: 3 |
| 0 0 4 0 0  0 0 5 0 0  0 0 1 0 0 | Matrix pattern placing  1 3 4 0 2 3 5 0 3 3 1 0 0  Natural form of matrix  0 0 4 0 0  0 0 5 0 0  0 0 1 0 0  The elements composition of processed column: 20  The index of processed column: 3 |
| 3 0 0 10 8  0 2 0 0 9  0 5 1 0 0  0 0 0 7 0 | Matrix pattern placing  1 1 3 4 10 5 8 0 2 2 2 5 9 0 3 2 5 3 1 0 4 4 7 0 0  Natural form of matrix  3 0 0 10 8  0 2 0 0 9  0 5 1 0 0  0 0 0 7 0  The elements composition of processed column: 70  The index of processed column: 4 |
| 5 0 0  3 0 0  0 0 0  0 1 0 | Matrix pattern placing  1 1 5 0 2 1 3 0 4 2 1 0 0  Natural form of matrix  5 0 0  3 0 0  0 0 0  0 1 0  The elements composition of processed column: 15  The index of processed column: 1 |

**Вывод:**

В курсовой работе №7 я научился обрабатывать разреженные матрицы. Работа была сложной при реализации операций для обработки матриц с помощью вектора, содержащего индексы строк и столбцов с ненулевыми элементами. Но вместе с этим работа была интересной.