**Лабораторная работа №3**

Обработка разреженных матриц

**Цель работы:** реализация алгоритмов обработки разреженных матриц, сравнение этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

**Условие задачи (вариант 1):**

Разреженная матрица хранится в форме 3-х объектов:

* вектор A содержит значения ненулевых элементов;
* вектор JA содержит номера столбцов для элементов вектора A;
* связный список IA, в элементе Nk которого находится номер компонент в A и JA, с которых начинается описание строки Nk матрицы A.

1. Смоделировать операцию сложения двух матриц, хранящихся в этой форме, с получением результата в той же форме.

2. Произвести операцию сложения.

3. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

**Наименование:**

Программа для сложения двух заданных или случайно сгенерированных матриц, сравнения алгоритмов хранения и сложения разреженных матриц.

**Область применения:**

Программа может использоваться везде, где необходимо работать с большим количеством данных, которые хранятся или представлены в матричном виде.

**Сроки исполнения:**

2 недели, начиная с 22.10.2018

**Основание для разработки:**

Заказчик – преподаватель Силантьева Александра Васильевна

Исполнитель – студент Ильясов Идрис Магомет-Салиевич, группа ИУ7-33Б

Основание – лабораторная работа №3 по предмету Типы и Структуры Данных

**Входные данные:**

Две матрицы заданных размеров

**Выходные данные:**

Матрица – результат сложения двух матриц заданных размеров.

**Назначение разработки:**

Общая концепция системы: консольное приложение.

Описание функционала системы: взаимодействие пользователя с программой осуществляется через консоль с помощью небольшого меню, где пользователь выбирает ответ на заданный вопрос [y/n]. Начальные данные вводятся пользователем или генерируются случайным образом. При завершении программы результат сложения матриц выводится на экран.

Пользователь взаимодействует с программой, отвечая на вопросы «Да/Нет» или вводя необходимые данные, требуемые программой. В случае неправильного ввода программа предложит повторить ввод.

**Возможные ошибки при работе с программой:**

1. Сложение матриц разных размеров
2. Ввод по неправильному индексу элемента
3. Ввод некорректных данных

**Требования к программе и структуры данных:**

Программа полностью написана на языке Си.

Матрицы хранятся, как и в обычном виде:

**int** \*pa, \*pb, \*pc;

Так и в сложном виде:  
   
**int** \*A1, \*A2, \*A3;  
**int** \*JA1, \*JA2, \*JA3;  
**struct** IA \*IA1 = **NULL**, \*IA2 = **NULL**, \*IA3 = **NULL**, \*tmp;

**struct** IA  
{  
 **int** i;  
 **int** Nk;  
  
 **struct** IA \*next;  
};

**Функции:**

Функции, при помощи которых выделяется память:

**int matrix\_allocation(int \*\*pa, int \*\*pb, int \*\*pc, int size);**

**void array\_allocation(int \*\*a, int \*\*b, int size);**

Функции, при помощи которых очищается память:

**void free\_list(struct IA \*head);**

**void free\_matrix\_and\_array(int \*matr, int \*A, int \*JA, struct IA \*IA);**

Функции, при помощи которых производится ввод данных:

**int read\_key(char \*text);**

**void read\_size(char \*text, int \*m, int \*n);**

Функции, при помощи которых выводятся матрицы:

**void print\_matrix(const int \*pa, int m, int n);**

**void print\_array(const int \*array, int len);**

**void print\_list(struct IA \*IA);**

**void print\_moded(const int \*A, const int \*JA, struct IA \*IA, const int n\_z\_el);**

Функции, которые сравнивают эффективность работы двух алгоритмов:

**unsigned long long tick(void);**

**void comparison(int m, int n, int percentage);**

Функции, которые необходимы для работы с матрицами при их сложении:  
**void generate(int \*pa, int m, int n, int percentage);**

**void input(int \*pa, int m, int n);**

**void count\_not\_zero(int \*pa, int m, int n, int \*not\_zero\_rows, int \*not\_zero);**

**void change\_matrix(int \*pa, int m, int n, int \*A, int \*JA, struct IA \*IA);**

**void usual\_addition(int \*pa, int \*pb, int \*pc, int m, int n);**

**void moded\_addition(int \*A1, int \*JA1, struct IA \*IA1, int not\_zero\_pa,**

**int \*A2, int \*JA2, struct IA \*IA2, int not\_zero\_pb,**

**int \*A3, int \*JA3, struct IA \*IA3, int \*not\_zero\_pc);**

**void copy\_rows(int \*A1, int \*JA1, struct IA \*IA1, int max\_iel,**

**int \*A3, int \*JA3, struct IA \*IA3, int \*not\_zero\_pc);**

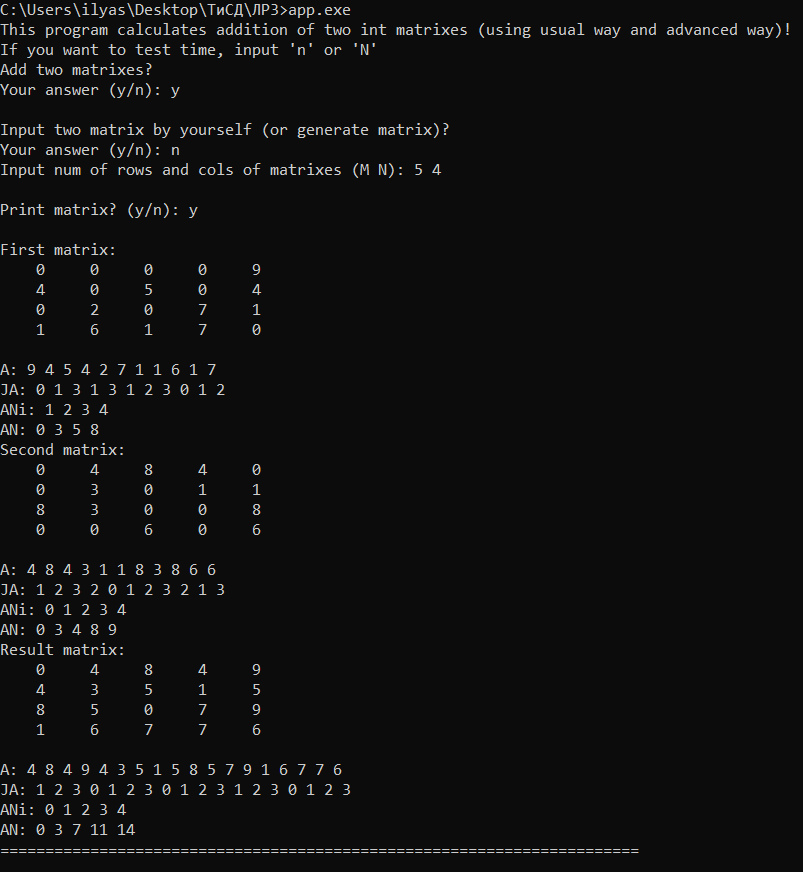
**void addition\_rows(int \*A1, int \*JA1, struct IA \*IA1, int max\_i1el,**

**int \*A2, int \*JA2, struct IA \*IA2, int max\_i2el,**

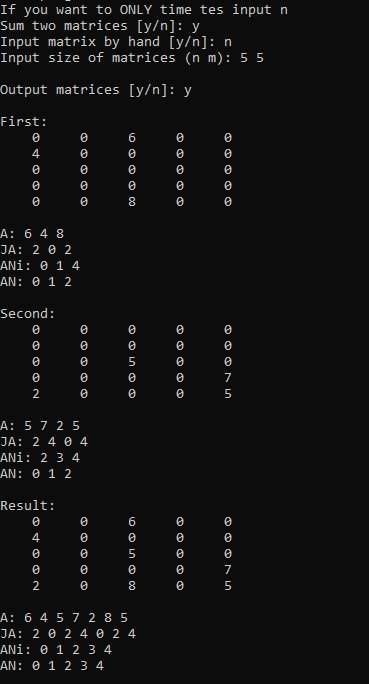
**int \*A3, int \*JA3, struct IA \*IA3, int \*not\_zero\_pc);**

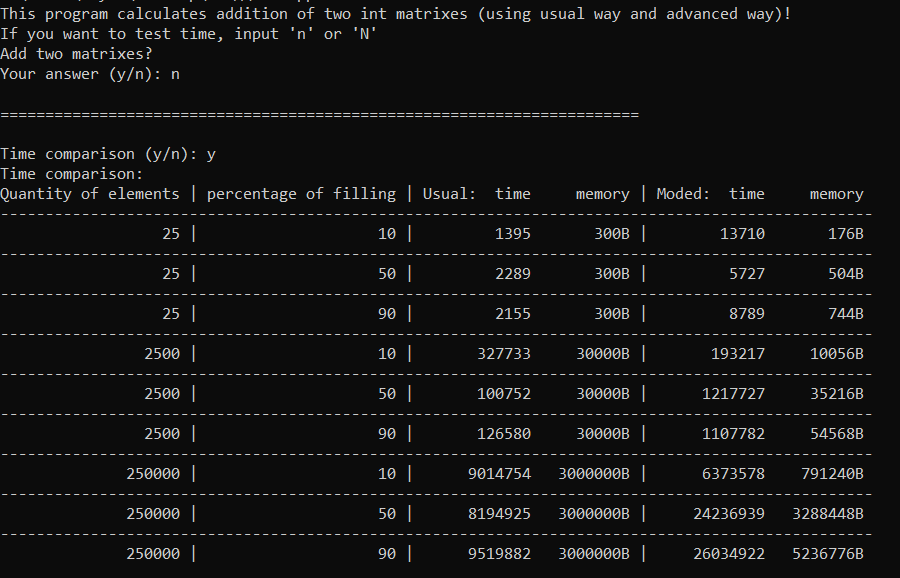
**Интерфейс программы:**

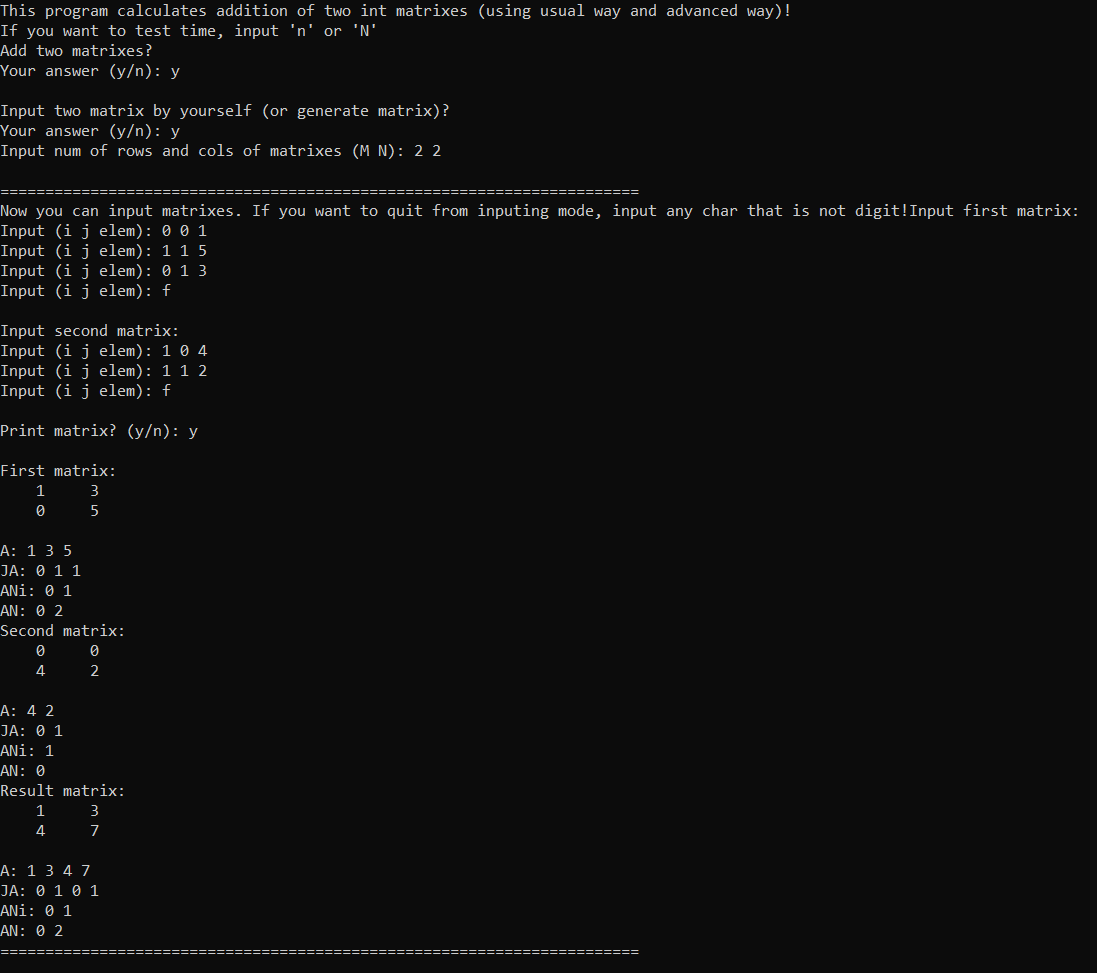
Матрицы могут быть введены пользователем вручную или сгенерированы при желании:



**Интерфейс:**Матрицу можно вводить как вручную или сгенерировать автоматически.



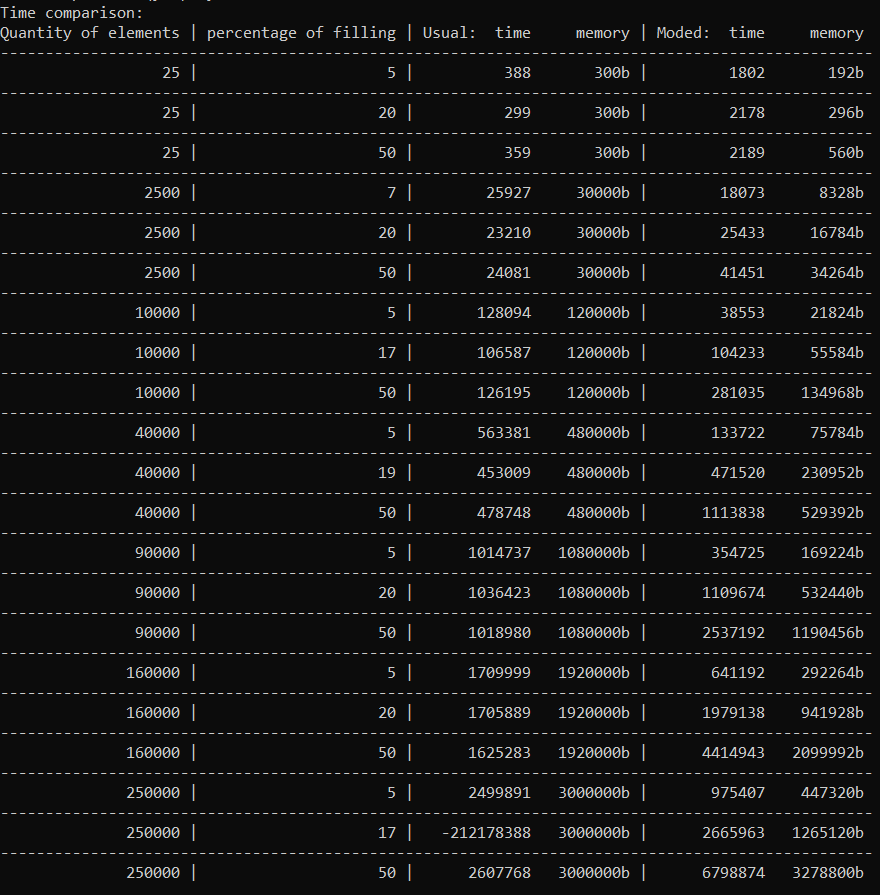




**Тесты и модули:**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Вывод** | **Класс эквивалентности** |
| Меню: y | Переход к следующему пункту меню | Корректный ввод в меню |
| Меню: f | (y/n): | Некорректный ввод в меню |
| Ввод размера матрицы: 3 3 | Успешный ввод | Корректный ввод размера матрицы |
| Ввод размера матрицы: 3 а | Wrong size input!  Try again: | Некорректный ввод размера матрицы |
| Ввод индексов и значение элемента матрицы: 2 2 2 | Успешный ввод | Корректный ввод элемента матрицы |
| Ввод индексов и значение элемента матрицы: 5 3 2 (при размере матрицы 3x3) | Wrong index input!  Must be (i: [0;2], j: [0;2]) | Некорректный ввод элемента матрицы |

**Сравнение алгоритмов по памяти и по времени:**



Операция сложения над матрицами небольших размеров (5х5) выполняется быстрее, если их хранить, как обычные (при любом проценте разреженности). По памяти эффективнее использовать хранение матрицы в специальном виде при 5% и 20% элементов.

Операция сложения над матрицами средних размеров (50х50) выполняется быстрее, если их хранить, как обычные. По памяти эффективнее использовать матрицы в специальном виде. При 7% заполненности, операция сложения обычной и специальной матриц показывают приблизительно одинаковый результат по времени.

Операцию сложения над матрицами размерам (100х100) при 5% заполненности эффективнее выполнять, используя специальные матрицы. При 19% заполненности два этих типа матриц показывают приблизительно одинаковый результат по времени. При 50% заполненности (т.е. при заполненности > 19%) эффективнее использовать обычные матрицы.

Операцию сложения над матрицами размерам (200х200) при 5% заполненности эффективнее выполнять, используя специальные матрицы. При 17% заполненности два этих алгоритма показывают приблизительно одинаковый результат по времени. При 50% заполненности (т.е. при заполненности > 17%) эффективнее использовать обычные матрицы.

Операцию сложения над матрицами размерам (300х300) при 5% заполненности эффективнее выполнять, используя специальные матрицы. При приблизительно 20% заполненности оба алгоритма показывают приблизительно одинаковый результат по времени. При 50% заполненности (т.е. при заполненности > 20%) эффективнее использовать обычные матрицы.

Операцию сложения над матрицами размерам (400х400) при 5% заполненности эффективнее выполнять, используя специальные матрицы. При приблизительно 20% заполненности алгоритмы показывают приблизительно одинаковый результат по времени. При 50% заполненности (т.е. при заполненности > 20%) эффективнее использовать обычные матрицы.

Операция сложения над матрицами больших размеров (500х500) выполняется быстрее при использовании матрицы в специальном виде (при 5% и 20% элементов). По памяти эффективнее использовать матрицы в специальном виде (при 5% и 20% элементов). При 17% заполненности большой матрицы, операция сложения обычной и специальной матриц показывают приблизительно одинаковый результат по времени.

**Вывод:**

Специальные алгоритмы работы с разреженными матрицами эффективнее использовать по памяти и по времени при проведении операций над большими матрицами с большим процентом разреженности.

**Контрольные вопросы:**

1.Что такое разреженная матрица, какие схемы хранения таких матриц вы знаете?

Разреженная матрица – матрица, большая часть которой заполнена нулями.

Существуют различные схемы хранения таких матриц. Так, например, можно хранить только информацию о ненулевых элементах (индексы i, j, значение ненулевого элемента). Также можно хранить значения ненулевых элементов, индексы i, j, номера элементов, с которых начинается очередная строка.

2.Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы.

Для хранения обычной матрицы необходимо выделить m\*n\*sizeof(тип элемента) байт памяти.

Память, которая необходима для хранения разреженной матрицы, зависит от выбранного типа хранения, количества ненулевых элементов.

3.Каков принцип обработки разреженной матрицы?

При работе с разреженной матрицы обрабатываются только ненулевые элементы, что позволяет сократить время обработки.

4.В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?

Эффективнее применять стандартные алгоритмы обработки матриц при достижении определенного уровня заполненности матрицы. Это зависит от размеров матрицы, от процента их заполненности.