ĐẠI HỌC ĐÀ NẴNG
ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

PBL1. ĐỒ ÁN
LẬP TRÌNH TÍNH TOÁN

Đề tài 205:

Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình , n ẩn: A.X=B

GIẢNG VIÊN HƯỚNG DẪN: Đỗ Thị Tuyết Hoa

SINH VIÊN THỰC HIỆN:

Trần Nhật Nguyên LỚP: 23T_DT1 NHÓM: 23.Nh15B

Vũ Đức Minh LỚP: 23T_DT2 NHÓM: 23.Nh15B

Đà Nẵng, tháng 05/2024

LỜI MỞ ĐẦU

Hệ phương trình tuyến tính đóng vai trò vô cùng quan trọng trong nhiều lĩnh vực khoa học, kỹ thuật và đời sống. Việc giải quyết các hệ phương trình này là một bài toán cơ bản và cần thiết. Trong số các phương pháp giải hệ phương trình tuyến tính, phương pháp Gauss là một phương pháp phổ biến và hiệu quả, đặc biệt cho các hệ phương trình có nhiều ẩn.

Mục đích thực hiện đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là tạo ra một chương trình máy tính sử dụng phương pháp Gauss để giải các hệ phương trình nói trên.

Mục tiêu của đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là tạo ra một chương trình tính toán chính xác và đạt hiệu quả cao trong quá trình giải hệ phương trình tuyến tính. Cụ thể:

+) Xây dựng giao diện đơn giản.

+) Cho phép nhập các ma trận từ bàn phím người dung và từ các tệp dữ liệu đầu vào.

+) Cho phép thực hiện tính toán và tìm nghiệm trên ma trận nhập vào.

+) Cung cấp tính năng thực hiện các bước biến đổi ma trận về dạng bậc thang.

+) Cung cấp tính năng cộng dồn các phần tử phía sau cột hệ số tự do để tạo thành hệ số tự do mới.

+) Cho phép xuất kết quả tìm được ra tệp dữ liệu đầu ra.

Phạm vi áp dụng của chương trình tính nói riêng và phương pháp Gauss nói chung trải rộng trên mọi hệ phương trình tuyến tính, bất kể số lượng và loại biến số.

Báo cáo này trình bày chi tiết về chương trình tính toán nghiệm của hệ phương trình tuyến tính và các tính năng và giao diện của chương trình.

MỤC LỤC

.....1

LỜI MỞ ĐẦU 3

MỤC LỤC.....4

DANH MỤC HÌNH ẢNH 5

1. GIỚI THIỆU ĐỀ TÀI 6

2. CƠ SỞ LÝ THUYẾT.....6

2.1. Ý tưởng.....6

2.2. Cơ sở lý thuyết.....6

3. CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 6

3.1. Phát biểu bài toán 6

3.2. Cấu trúc dữ liệu 7

3.3. Thuật toán.....8

3.3.1. Thuật toán sao chép ma trận.....8

3.3.2. Thuật toán hoán vị hai số và hai dòng.....9

3.3.3. Thuật toán khử Gauss 9

3.3.4. Thuật toán tìm hạng ma trận.....9

3.3.5. Thuật toán tính nghiệm.....10

3.3.6. Một số các hàm khác và phương pháp xử lí lỗi của hàm 10

4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 13

4.1. Tổ chức chương trình 13

4.2. Ngôn ngữ cài đặt.....14

4.3. Kết quả.....14

4.3.1. Giao diện chính của chương trình 14

4.3.2. Kết quả thực thi của chương trình 15

4.3.3. Nhận xét và đánh giá 18

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 18

5.1. Kết luận.....18

5.2. Hạn chế 18

5.3. Hướng phát triển.....18

TÀI LIỆU THAM KHẢO.....19

PHỤ LỤC.....20

DANH MỤC HÌNH ẢNH

Hình 1: Tổ chức chương trình.....13
Hình 2: Giao diện giới thiệu. 14
Hình 3: Giao diện chính.....14
Hình 4: Nhập ma trận từ bàn phím thành công.....15
Hình 5: Nhập ma trận từ bàn phím thất bại (dữ liệu không hợp lệ) 15
Hình 6: Đọc ma trận từ file thành công. 15
Hình 7: Đọc ma trận từ file thất bại (thiếu dữ liệu).....15
Hình 8: Đọc ma trận từ file thất bại (thừa dữ liệu). 15
Hình 9: Đọc ma trận từ file thất bại (số dòng lớn hơn số cột). 15
Hình 10: Biến đổi ma trận.....16
Hình 11: Lời nhắc người dùng có muốn hiển thị các bước biến đổi không. 16
Hình 12: Giải hệ phương trình tuyến tính. 16
Hình 13: Hệ phương trình có nghiệm. 17
Hình 14: Hệ phương trình vô nghiệm. 17
Hình 15: Xuất kết quả. 17

1. GIỚI THIỆU ĐỀ TÀI

Đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là một đề tài trong lĩnh vực Công nghệ thông tin và Khoa học Máy Tính. Đề tài này tập trung vào việc giải và tìm nghiệm hệ phương trình tuyến tính n ẩn, n phương trình.

2. CƠ SỞ LÝ THUYẾT

2.1. Ý tưởng

2.2. Cơ sở lý thuyết

Cơ sở lý thuyết cho đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là:

+) File I/O: Sử dụng các thao tác đọc/ghi file để truy xuất dữ liệu ma trận và lưu trữ kết quả tính toán.

+) Các cấu trúc dữ liệu: Sử dụng cấu trúc dữ liệu mảng hai chiều để lưu trữ và quản lý ma trận; Sử dụng cấu trúc dữ liệu danh sách để lưu trữ và quản lý thực đơn của chương trình.

+) Thuật toán giải hệ phương trình tuyến tính bằng phương pháp Gauss để giải và tìm nghiệm của hệ phương trình tuyến tính đã cho.

+) Thiết kế giao diện: Thiết kế giao diện đơn giản.

3. CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

3.1. Phát biểu bài toán

Đầu vào (Input) của chương trình bao gồm: Dữ liệu của ma trận đầu vào bao gồm số dòng, số cột và các phần tử ma trận.

Đầu ra (Output) của chương trình bao gồm: Dữ liệu của ma trận được tính toán và nghiệm của hệ phương trình tuyến tính.

Bài toán đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" thực hiện các công việc chính sau:

+) Thực hiện đọc ma trận cấp n x m từ tệp đầu vào hoặc từ bàn phím.

+) Cộng dồn các phần tử phía sau kể từ cột n + 1 trở đi để trở thành cột hệ số tự do tương ứng.

+) Thực hiện các tính toán trên ma trận bao gồm: biến đổi ma trận về dạng bậc thang; giải hệ phương trình tuyến tính ứng với ma trận; xuất kết quả của chương trình ra màn hình và tệp đầu ra.

3.2. Cấu trúc dữ liệu

Để xây dựng chương trình cho đề tài "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" cần có các thành phần:

+) Các định nghĩa, hằng:

```
#define ENTER1          '\n'
#define ENTER2          '\r'
#define SPACE.          ' '
#define MAX             100
#define CHAR_FORMAT     "%7c"
#define DATA_FORMAT     "%7.2f"
#define INPUT_DIR       "INPUT/"
#define OUTPUT_DIR      "OUTPUT/"
#define LOG_PATH        "log/log.txt"
#define INPUT_PATH      "DATA.INP"
#define OUTPUT_PATH     "OUTPUT/DATA.OUT"
```

+) Các kiểu dữ liệu được định nghĩa của chương trình.

1. '*string*': kiểu dữ liệu chuỗi.
2. '*Matrix*': kiểu dữ liệu mảng hai chiều biểu diễn ma trận.
3. '*Vector*': kiểu dữ liệu mảng một chiều biểu diễn vector.
4. '*bool*': kiểu dữ liệu logic có hai giá trị là đúng (*true*, 1) và sai (*false*, 0).
5. '*func*': kiểu dữ liệu hàm.
6. '*MatrixRecord*': kiểu dữ liệu cấu trúc của bản ghi các thông tin của ma trận bao gồm '*n*' (số hang), '*m*' (số cột) và '*matrix*' (ma trận).
7. '*MenuOption*': kiểu dữ liệu cấu trúc của một lựa chọn trên thực đơn, bao gồm các thuộc tính:
   a. '*label*': nhãn của lựa chọn, hiển thị cho người dùng nhìn thấy.
   b. '*action*': hành động được thực hiện sau khi người dùng chọn.
8. '*Menu*': một List ADT của các lựa chọn trên thực đơn, bao gồm một số thuộc tính cơ bản của List ADT: "*list*" (mảng các lựa chọn) và "*count*" (số lượng các lựa chọn).

+) Các cờ hiệu được định nghĩa để điều khiển chương trình:

1. Cờ '*allow_color_showing_in_matrix*': nhận hai giá trị đúng hoặc sai, nếu đúng thì chương trình được phép hiển thị màu khi in ma trận. Giá trị mặc định là sai.
2. Cờ '*pivot_row*': nhận các giá trị nguyên, lưu chỉ số hiện tại của dòng biến đổi trong quá trình biến đổi ma trận về dạng bậc thang. Giá trị mặc định là -1.
3. Cờ '*dest_row*': nhận các giá trị nguyên, lưu chỉ số hiện tại của dòng bị biến đổi trong quá trình biến đổi ma trận về dạng bậc thang. Giá trị mặc định là -1.
4. Cờ '*current_row*': nhận các giá trị nguyên, lưu chỉ số lớn nhất có thể có của những dòng đã được biến đổi. Giá trị mặc định là -1.
5. Cờ '*ENABLE_COLOR*': nhận hai giá trị 0 hoặc 1, nếu nhận giá trị 1 thì chương trình cho phép hiển thị các màu sắc khác trên màn hình bao gồm: *RED* (đỏ), *GREEN* (xanh lá), *YELLOW* (vàng), *BLUE* (xanh) và *RESET* (trở về màu bình thường). Giá trị mặc định là 1.
6. Cờ '*LINUX*': nhận hai giá trị 0 hoặc 1, nhận giá trị 1 khi chương trình chạy trên các hệ điều hành dựa trên Linux.
7. Cờ '*output_file_path*': nhận giá trị là một chuỗi kí tự tương ứng với đường dẫn của tệp đầu ra, được chương trình tự động tạo ứng với tên của tệp đầu vào.

+) Các biến toàn cục được khai báo trong chương trình:

1. '*log_file*': tệp lưu các thông báo của chương trình nhưng không được in ra màn hình (*stdout*).
2. '*menu*': thực đơn chính của chương trình.
3. '*current*': bản ghi ma trận hiện tại trên chương trình được dùng để tính toán.

3.3. Thuật toán

Thuật toán của bài toán "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" được thể hiện qua:

3.3.1. Thuật toán sao chép ma trận

+) Thuật toán lặp từng gán phần tử của ma trận nguồn (src) sang ma trận đích (dest).

+) Độ phức tạp thuật toán sao chép ma trận là $O(n^2)$.

3.3.2. Thuật toán hoán vị hai số và hai dòng

+) Thuật toán hoán vị hai số bằng phương pháp đặt ẩn phụ có độ phức tạp thuật toán là $O(1)$.

+) Thuật toán hoán vị hai dòng bằng phương pháp hoán vị từng phần tử của hai dòng có độ phức tạp thuật toán là $O(n)$.

3.3.3. Thuật toán khử Gauss

+) Thuật toán biến đổi ma trận về ma trận bậc thang:

Đầu vào: Ma trận a chưa được biến đổi.

Bước 1: $\boldsymbol{i \leftarrow 0,\ j \leftarrow 0}$.

Bước 2: Nếu $i \geq n$ hoặc $j \geq m$ thì kết thúc thuật toán.

Bước 3: Nếu $a_{ij} = 0$, lặp $k \leftarrow \overline{i+1,\ n}$:

- Nếu $k = n$, thực hiện $\mathbf{j \leftarrow j + 1}$ và quay lại **bước 3.**
- Nếu $a_{kj} \neq 0$, dừng lặp.

Bước 4: Thực hiện đổi hai dòng i và k.

Bước 5: Lặp $k \leftarrow \overline{i+1,\ n-1}$:

- Nếu $a_{ij} \neq 0$: $\mathbf{w \leftarrow a_{kj} \,/\, a_{ij}}$, tiến hành lặp $l \leftarrow \overline{j,\ m-1}$:

$$\mathbf{a_{kl} \leftarrow a_{kl} - w * a_{il}}$$

Bước 6: Tiến hành $\mathbf{i \leftarrow i + 1, j \leftarrow j + 1}$, quay lại **bước 2**.

Đầu ra: Ma trận a đã được biến đổi.

+) Độ phức tạp thuật toán khử Gauss là $O(n^3)$.

3.3.4. Thuật toán tìm hạng ma trận

+) Thuật toán tìm hạng ma trận đã được biến đổi về dạng bậc thang:

Đầu vào: Ma trận a đã được biến đổi.

Bước 1: $\mathbf{j \leftarrow 0}$, $r \leftarrow 0$.

Bước 2: Lặp $i \leftarrow \overline{0,\ n-1}$:

- Lặp $j \leftarrow \overline{j,\ m-1}$:
  - Nếu $a_{ij} \neq 0$ thì kết thúc lặp.
  - Tăng j thêm một đơn vị $\mathbf{j \leftarrow j + 1}$.
- Nếu $j < m$, tăng r thêm một đơn vị $\mathbf{r \leftarrow r + 1}$.

Bước 3: Trả về giá trị r, kết thúc thuật toán.

Đầu ra: Số nguyên r là hạng của ma trận a.

+) Độ phức tạp thuật toán tìm hạng ma trận là $O(n^2)$.

3.3.5. Thuật toán tính nghiệm

+) Thuật toán tính nghiệm của một ma trận đã được biến đổi về dạng bậc thang và biết được hạng của ma trận đó.

Đầu vào: Ma trận a đã được biến đổi, hạng của ma trận hệ số ($r(A)$), hạng của ma trận mỡ rộng ($r(\bar{A})$).

Bước 1: Nếu $r(A) \neq r(\bar{A})$, thông báo hệ phương trình vô nghiệm, kết thúc thuật toán.

Bước 2: Nếu $r(A) < (số\ nghiệm = n)$, thông báo hệ phương trình có vô số nghiệm, kết thúc thuật toán.

Bước 3: $S_{n-1} \leftarrow a_{n-1,n} \,/\, a_{n-1,n-1}$.

Bước 4: Lặp $i \leftarrow \overline{n-2,\ 0}$:

- $c \leftarrow 0,$ lặp $j \leftarrow \overline{n-1,\ i}: c \leftarrow a_{ij} * S_j$
- $S_i = (a_{in} - c) \,/\, a_{ii}$

Bước 5: Thông báo hệ phương trình có nghiệm và trả về tập S, kết thúc thuật toán.

Đầu ra: Thông báo hệ phương trình vô nghiệm, vô số nghiệm hay có nghiệm (trả về nghiệm dưới dạng vector S).

3.3.6. Một số các hàm khác và phương pháp xử lí lỗi của hàm

+) Đối với việc nhập ma trận thông qua hai hàm $scan\_matrix()$ và $load\_matrix()$:

Hàm $scan\_matrix()$ thực hiện chức năng nhập ma trận từ bàn phím hoặc từ tệp đầu vào:

a. Dùng hai biến $dummy$ và $success$ để kiểm tra dữ liệu đầu vào có hợp lệ hay không, nếu dữ liệu không hợp lệ thì $dummy$ khác các kí tự xuống dòng và dấu cách, hoặc $success = 0$.
b. Dùng các hàm $fseek(), ftell()$ và $feof()$ và biến $dummy$ để kiểm tra dữ liệu trên tệp đầu vào thừa hay thiếu dữ liệu (bỏ qua kiểm tra khi nhập từ bàn phím). Cụ thể:
   - Lưu vị trí hiện tại của con trỏ $file$ vào biến $POSITION$ với hàm $ftell()$.
   - Đọc các kí tự tiếp theo vào $dummy$ cho đến khi $dummy \neq ' '$.
   - Nếu $dummy$ là kí tự xuống dòng mà chương trình chưa đọc hết dòng đó thì thông báo thiếu dữ liệu trên dòng đó và kết thúc việc đọc.
   - Nếu $dummy$ khác kí tự xuống dòng mà chương trình đã đọc hết dòng đó thì thông báo thừa dữ liệu trên dòng đó và kết thúc việc đọc.
   - Đưa con trỏ $file$ về vị trí ban đầu $POSITION$ với hàm $fseek()$.

- Sau khi đã hoàn thành việc đọc ma trận, tiếp túc kiểm tra nếu như đọc đến cuối tệp (dùng hàm $feof()$) và còn có thể lấy dữ liệu từ tệp thì thông báo thừa dòng và kết thúc việc đọc.

c. So sánh hai giá trị 'n' và 'm' và hiển thị thông báo lỗi nếu số dòng lớn hơn số cột.

d. Sau khi đọc ma trận thành công ma trận mà không có lỗi, hàm $scan\_matrix()$ sẽ thực hiện gọi hàm $cong\_don()$ để cộng dồn các phần tử nằm phía sau của cột n để tạo thành cột hệ số tự do mới.

Hàm $load\_matrix()$ thực hiện chức năng đọc dữ liệu tệp có đường dẫn được nhập từ bàn phím hoặc từ tham số truyền vào hàm. Nếu không thể tìm thấy tệp thì thông báo lỗi và kết thúc việc đọc, nếu tìm thấy tệp thì thực thi hàm $scan\_matrix()$ với đối số truyền vào là tệp đã mở. Hàm $load\_matrix()$ tự động cấp tên cho tệp đầu ra ứng với tên của tệp đầu vào gán vào cờ '$output\_file\_path$'.

+) Đối với các hàm hiển thị ma trận:

1. $show\_matrix()$: hàm hiển thị ma trận, sử dụng kí tự '|' để phân cách giữa phần ma trận hệ số và cột hệ số tự do.
2. $hien\_thi\_hpt()$: hàm hiển thị ma trận dưới dạng hệ phương trình.

+) Các hàm xây dựng cho List ADT '$Menu$':

1. $create\_menu()$: tạo một thực đơn mới.
2. $push()$: thay thế hàm $insert()$ trong List để chèn đẩy lựa chọn ('$MenuOption$') vào cuối thực đơn.
3. $show\_menu()$: hiển thị nhãn của các lựa chọn trong thực đơn theo thứ tự từ đầu đến cuối danh sách và đánh chỉ số bắt đầu từ 1, hàm đồng thời thực hiện việc đọc lựa chọn của người dùng từ bàn phím (chọn theo các chỉ số) và thực thi hành động của lựa chọn đó.

+) Các macro thực hiện việc hiển thị thông báo lỗi ra màn hình:

```
#define ERR_SO_DONG_LON_HON_COT(n, m)
\
    printf("So dong(%d) lon hon hoac bang so cot (%d).", n, m)
#define ERR_THUA_DONG      printf("Qua nhieu dong.\n")
#define ERR_THIEU_COT(i) printf(RED "Qua it du lieu tren dong %d.\n", i +
1)
#define ERR_THUA_COT(i) printf("Qua nhieu du lieu tren dong %d.\n" , i + 1)
#define ERR_DL_KHONG_HOP_LE(i, j, row, col) {                          \
if (row)        printf(RED "So dong co du lieu khong hop le.\n" RESET);  \
else if (col)   printf(RED "So cot co du lieu khong hop le.\n" RESET);   \
else            printf("Du lieu khong hop le tai dong %d, cot %d.\n", i + 1, j + 1);\
}
#define ERR_FILE_KHONG_TON_TAI(file) printf("File %s khong ton
tai.\n",\ file)
```

+) Các macro quản lí chương trình:

```
#define TAO_THU_MUC_QUAN_LI {                                          \
```

```
        printf("Tao cac thu muc can thiet...\n");                                    \
        if (system("cd INPUT && cd .."))     system("mkdir INPUT");                  \
        if (system("cd OUTPUT && cd .")) system("mkdir OUTPUT");                     \
        if (system("cd log && cd .."))                  system("mkdir log");         \
        CLEAR_SCREEN;                                                                \
    }
    #define HR(file) fprintf(file,                                                   \
    "=========================================================== \
    ========\n")
```

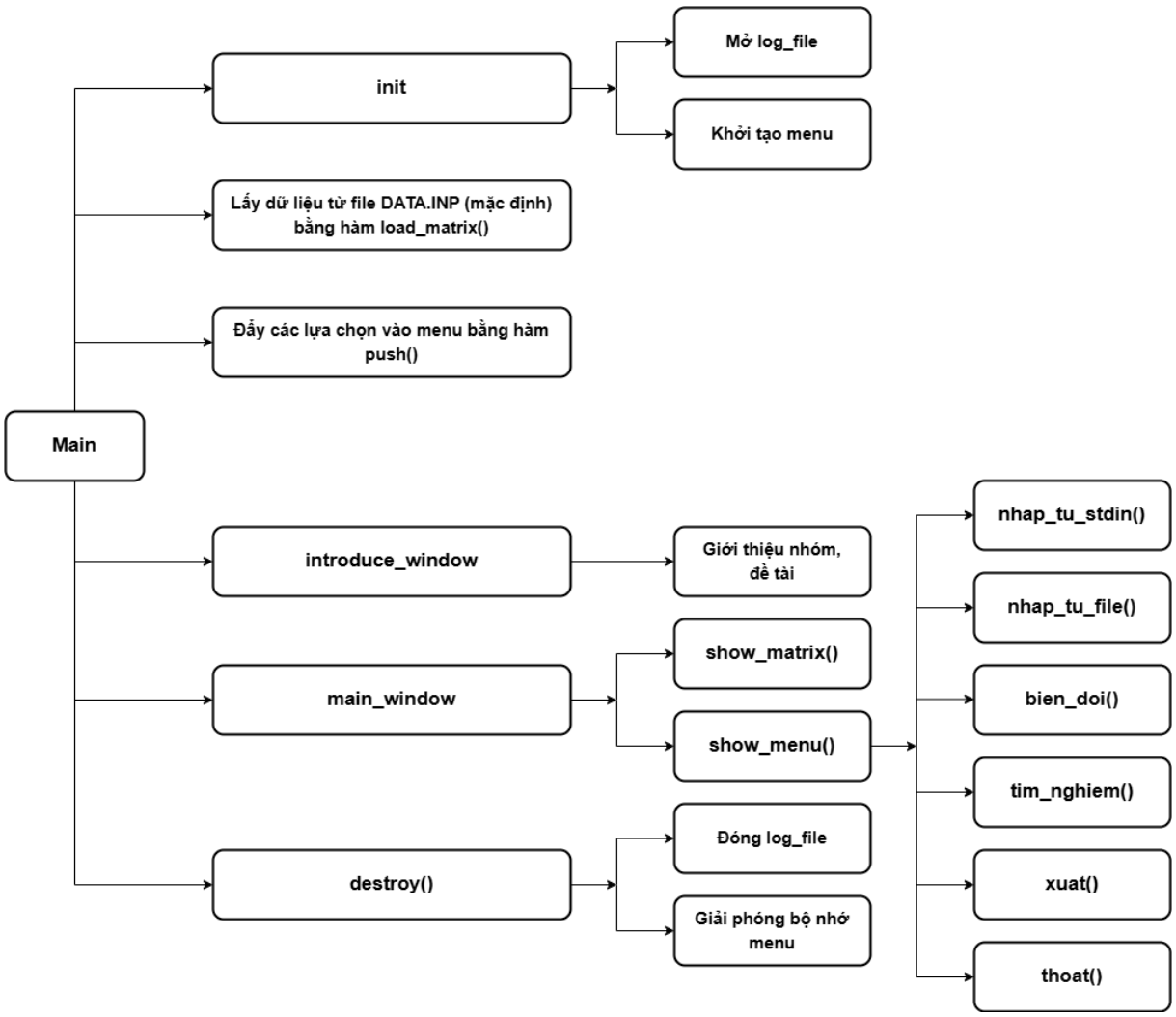
Trong đó, macro $HR(file)$ để tạo một đường ngang trong tệp hoặc màn hình để phân cách giữa các bước thực hiện của chương trình.

+) Ngoài ra, còn có các macro khác:

1. '$CLEAR\_SCREEN$': thực hiện việc xóa màn hình chính sử dụng lệnh cls (trên Windows) hoặc lệnh $clear$ (trên các hệ điều hành dựa trên UNIX).
2. '$CLEAR\_STDIN$': thực hiện việc xóa bộ nhớ đệm của $stdin$.
3. $getch()$ và $getche()$: nằm trong thư viện "$conio.h$" trên Windows. Trên các hệ điều hành dựa trên UNIX, ta định nghĩa các hàm này ứng với hàm $getchar()$.
4. '$SET\_COLOR()$': thực hiện việc thay đổi màu chữ trên màn hình nếu được cho phép (cờ '$ENABLE\_COLOR$' được bật).
5. '$RESET\_COLOR$': thực hiện việc đặt lại màu chữ mặc định.
6. '$TAO\_CAC\_THU\_MUC\_QUAN\_LI$': thực hiện việc tạo các thư mục cần thiết cho chương trình, bao gồm '$INPUT$', '$OUTPUT$' và 'log'.

4. CHƯƠNG TRÌNH VÀ KẾT QUẢ

4.1. Tổ chức chương trình



Hình 1: Tổ chức chương trình

Chú thích:

'nhap_tu_stdin()' : Đọc dữ liệu từ bàn phím bằng hàm `scan_matrix()` *với đối số là stdin.*

'nhap_tu_file()' : Đọc dữ liệu từ file bằng hàm `load_matrix()` *với đường dẫn file được nhập từ bàn phím.*

'bien_doi()': Thực hiện biến đổi ma trận bằng hàm `bien_doi_ma_tran()`.

'tim_nghiem()': Thực hiện giải hệ phương trình tuyến tính bằng hàm `bien_doi_Gauss()`.

'xuat(): Thực hiện giải hệ phương trình tuyến tính bằng hàm `bien_doi_Gauss()` *và xuất kết quả vào file có đường dẫn là* `output_file_path`.

'thoat(): Thực hiện hàm destroy() để giải phóng dữ liệu và thoát chương trình.

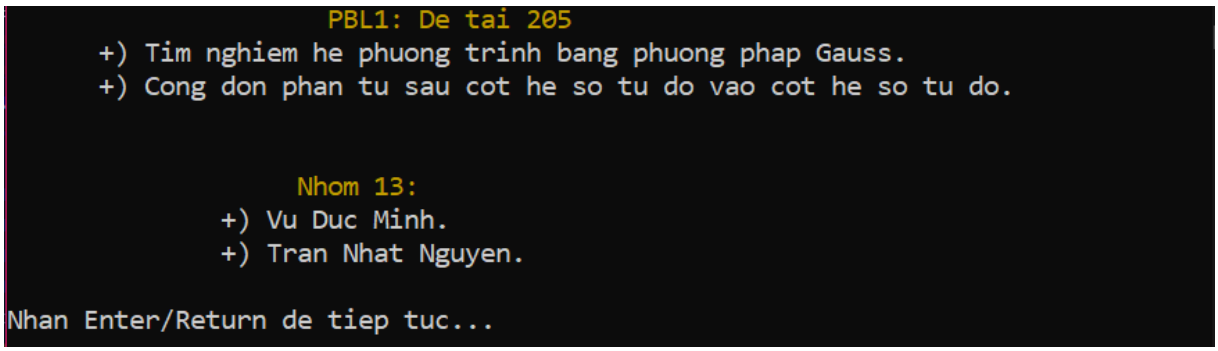
4.2. Ngôn ngữ cài đặt

Bài toán "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là một trong những bài toán phổ biến trong lập trình, được thực hiện bằng nhiều ngôn ngữ khác nhau. Trong trường hợp này, chúng ta sử dụng ngôn ngữ C để triển khai chương trình.

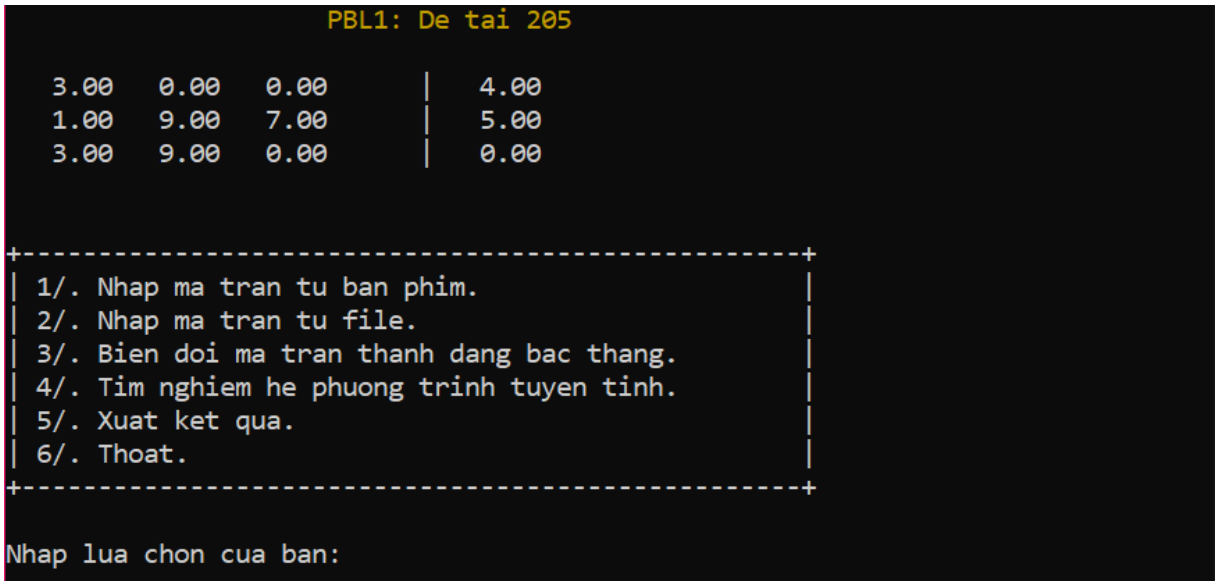
4.3. Kết quả

Chương trình có đầy đủ, hiệu quả các chức năng giải và tìm nghiệm hệ phương trình tuyến tính bằng phương pháp Gauss như biến đổi ma trận, tìm nghiệm hệ phương trình tuyến tính và xuất kết quả ra tệp.

4.3.1. Giao diện chính của chương trình



Hình 2: Giao diện giới thiệu.



Hình 3: Giao diện chính.

4.3.2. Kết quả thực thi của chương trình

1/. Chức năng đầu tiên là nhập ma trận từ bàn phím

```
Nhap so dong: 3
Nhap so cot: 4
Nhap ma tran:
3 0 0 4
1 9 7 5
3 9 0 0
Doc ma tran thanh cong!

Nhan Enter/Return de tiep tuc...
```

Hình 4: Nhập ma trận từ bàn phím thành công.

```
Nhap so dong: 3
Nhap so cot: 4
Nhap ma tran:
2 2 2 2
2 2a
Du lieu khong hop le tai dong 2, cot 2.

Nhan Enter/Return de tiep tuc...
```

Hình 5: Nhập ma trận từ bàn phím thất bại (dữ liệu không hợp lệ)

2/. Chức năng nhập ma trận từ file

```
Nhap duong dan file: dung.inp
Doc ma tran thanh cong!

Nhan Enter/Return de tiep tuc...
```

```
dung.inp
3 4
1 4 1 5
2 3 0 5
1 1 2 0
```

Hình 6: Đọc ma trận từ file thành công.

```
Nhap duong dan file: thieucot.inp
Qua it du lieu tren dong 2.

Nhan Enter/Return de tiep tuc...
```

```
thieucot.inp
3 4
1 2 3 4
2 2 3
0 1 3 2
```

Hình 7: Đọc ma trận từ file thất bại (thiếu dữ liệu)

```
Nhap duong dan file: thuadong.inp
Qua nhieu dong.

Nhan Enter/Return de tiep tuc...
```

```
thuadong.inp
3 4
1 2 3 1
4 5 6 2
7 8 9 3
1 1 2 3
```

Hình 8: Đọc ma trận từ file thất bại (thừa dữ liệu).

```
Nhap duong dan file: dongcot.inp
So dong(5) lon hon hoac bang so cot (3).
Nhan Enter/Return de tiep tuc...
```

```
dongcot.inp
5 3
1 2 5
1 1 0
3 1 1
2 2 4
5 5 0
```

Hình 9: Đọc ma trận từ file thất bại (số dòng lớn hơn số cột).

3/. Chức năng biến đổi ma trận thành dạng bậc thang

```
+) Ma tran ban dau:
   0.00   2.00   1.00      |   2.00
   2.00   1.00   3.00      |   5.00
   0.00   4.00   2.00      |   6.00



============================================================
+)BUOC 1:
Doi dong 1 va 2.
   2.00   1.00   3.00      |   5.00
   0.00   2.00   1.00      |   2.00
   0.00   4.00   2.00      |   6.00



============================================================
+) BUOC 2:
Lay dong 3 tru di 2 lan dong 2.
   2.00   1.00   3.00      |   5.00
   0.00   2.00   1.00      |   2.00
   0.00   0.00   0.00      |   2.00

Ma tran da duoc bien doi!
   2.00   1.00   3.00      |   5.00
   0.00   2.00   1.00      |   2.00
   0.00   0.00   0.00      |   2.00
```

Hình 10: Biến đổi ma trận.

4/. Chức năng tìm nghiệm hệ phương trình tuyến tính

Chương trình sẽ hiện ra một lời nhắc người dùng có hiện các bước biến đổi hay không:

```
Hien cac buoc bien doi?
+-----------------------------+
| Chon Enter/Return de dong y. |
| Chon phim bat ki de tu choi. |
+-----------------------------+

Chon:
```

Hình 11: Lời nhắc người dùng có muốn hiển thị các bước biến đổi không.

+) Nếu người dùng từ chối (nhấn phím bất kì) thì chương trình sẽ tự động giải hệ phương trình tuyến tính và ẩn đi các bước biến đổi.

```
       TIM NGHIEM HE PHUONG TRINH TUYEN TINH

============================================================
             2[x2]  +   [x3]  =  2
    2[x1]  +   [x2]  +  3[x3]  =  5
             4[x2]  +  2[x3]  =  6
============================================================
Ma tran duoc bien doi la:
   2.00   1.00   3.00      |   5.00
   0.00   2.00   1.00      |   2.00
   0.00   0.00   0.00      |   2.00

============================================================
He phuong trinh vo nghiem.

Nhan Enter/Return de tiep tuc...
```

Hình 12: Giải hệ phương trình tuyến tính.

+) Nếu người dùng đồng ý (nhấn Enter) thì chương trình sẽ hiện các bước biến đổi tương tự như tính năng "Biến đổi ma trận về dạng bậc thang" và đồng thời thực hiện việc giải và tìm nghiệm hệ phương trình tuyến tính.

Một số trường hợp giải hệ phương trình tuyến tính:

+) Có nghiệm:

```
       TIM NGHIEM HE PHUONG TRINH TUYEN TINH

============================================================
     [x1]  +  4[x2]  +   [x3]  =  5
    2[x1]  +  3[x2]           =  5
     [x1]  +   [x2]  +  2[x3]  =  0
============================================================
Ma tran duoc bien doi la:
   1.00   4.00   1.00      |   5.00
   0.00  -5.00  -2.00      |  -5.00
   0.00   0.00   2.20      |  -2.00

============================================================
He phuong trinh co nghiem la:
x1 = 0.45
x2 = 1.4
x3 = -0.91
```

Hình 13: Hệ phương trình có nghiệm.

+) Vô số nghiệm:

```
       TIM NGHIEM HE PHUONG TRINH TUYEN TINH

============================================================
     [x1]  +  2[x2]  +  4[x3]  =  5
    2[x1]  +  4[x2]  +  8[x3]  = 10
             [x2]  +  4[x3]  =  5
============================================================
Ma tran duoc bien doi la:
   1.00   2.00   4.00      |   5.00
   0.00   1.00   4.00      |   5.00
   0.00   0.00   0.00      |   0.00

============================================================
He phuong trinh co vo so nghiem.
```

Hình 14: Hệ phương trình vô nghiệm.

+) Vô nghiệm: Hình 12.

5/. Chức năng giải và xuất kết quả tìm được từ hệ phương trình

```
       TIM NGHIEM HE PHUONG TRINH TUYEN TINH

============================================================
     [x1]  +  4[x2]  +   [x3]  =  5
    2[x1]  +  3[x2]           =  5
     [x1]  +   [x2]  +  2[x3]  =  0
============================================================
Ma tran duoc bien doi la:
   1.00   4.00   1.00      |   5.00
   0.00  -5.00  -2.00      |  -5.00
   0.00   0.00   2.20      |  -2.00

============================================================
He phuong trinh co nghiem la:
x1 = 0.45
x2 = 1.4
x3 = -0.91

Xuat thanh cong vao file OUTPUT/dung.OUT
```

Hình 15: Xuất kết quả.

Kết quả tính toán sẽ được tự động xuất vào tệp đầu ra có đường dẫn tương ứng với tệp đầu vào. Nếu dữ liệu đầu vào được người dùng nhập vào từ bàn phím thì kết quả sẽ được xuất ra đường dẫn mặc định là "$OUTPUT/DATA.OUT$".

Chú thích:

+) Sau mỗi chức năng hay lựa chọn của người dùng, chương trình sẽ dừng để người dùng có thể xem kết quả thực thi. Sau đó, người dùng ấn Enter để quay về giao diện chính.

+) Người dùng chỉ được phép lựa chọn các lựa chọn từ 1-6, nếu người dùng nhập sai, chương trình sẽ thông báo lựa chọn không hợp lệ.

+) Tại giao diện chính, người dùng có thể kết thúc chương trình bằng cách chọn 6, lúc này một lời chào tạm biệt sẽ xuất hiện và kết thúc chương trình.

4.3.3. Nhận xét và đánh giá

Nhìn chung, chương trình đã đáp ứng đầy đủ các chức năng như là nhập ma trận từ bàn phím, nhập ma trận từ file, biến đổi ma trận về dạng bậc thang, tìm nghiệm hệ phương trình tuyến tính và xuất kết quả. Đồng thời chương trình có thể chạy trên nhiều nền tảng hệ điều hành khác nhau. Hạn chế lớn nhất của chương trình là số lượng và phạm vi các tính năng còn ít và chỉ gói gọn trong việc giải hệ phương trình tuyến tính.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Tổng quan lại, bài toán "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" là khá cơ bản nhưng vô cùng hữu ích và đóng vai trò hết sức quan trọng trong rất nhiều lĩnh vực khoa học, kĩ thuật và đời sống. Với các tính năng của chương trình, người dùng có thể dễ dàng giải hệ phương trình tuyến tính và xem được các bước biến đổi ma trận thành dạng bậc thang.

5.2. Hạn chế

Chương trình giải quyết bài toán "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" được viết bằng ngôn ngữ C có những hạn chế về số lượng các tính năng và phạm vi sử dụng của các tính năng chỉ nằm trong việc giải hệ phương trình tuyến tính, dẫn đến chương trình chưa có được sự đa dụng và khả năng phát triển tốt như các ứng dụng và phần mềm tính toán trên ma trận khác.

5.3. Hướng phát triển

Bài toán "Dùng phương pháp Gauss để giải hệ phương trình gồm n phương trình, n ẩn: A.X=B" có thể phát triển theo các hướng sau:

1. Thêm các tính năng tính toán đặc biệt quan trọng trên ma trận như tính định thức, tìm giá trị riêng, vector riêng.
2. Thêm các thông báo về các bước tìm hạng của ma trận.
3. Tích hợp phương pháp giải hệ phương trình tuyến tính bằng Gauss-Jordan và triển khai thuật toán phân tích LU.

TÀI LIỆU THAM KHẢO

1. Giáo trình tin học đại cương, trường Đại học Bách Khoa – Đại học Đà Nẵng.

2. Bài giảng môn Phương pháp tính, trường Đại học Bách Khoa – Đại học Đà Nẵng.

PHỤ LỤC

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Kiem tra dang chay he dieu hanh gi.
#ifdef _WIN32
#define CLEAR_SCREEN system("cls")
#include <conio.h>
#define CLEAR_STDIN fflush(stdin)
#define LINUX 0
#else
#define CLEAR_SCREEN system("clear")
#ifdef __APPLE__
#define CLEAR_STDIN fflush(stdin)
#define LINUX 0
#else
#define CLEAR_STDIN scanf("%*c")
#define LINUX 1
#endif
#define getch() getchar()
#define getche() getchar()
#endif

#define ENABLE_COLOR 1
#if ENABLE_COLOR
// Dinh nghia cac mau sac.
#define RED   "\x1B[31m"
#define GREEN   "\x1B[32m"
#define YELLOW      "\x1B[33m"
#define BLUE "\x1B[34m"
#define RESET         "\x1B[0m"

#define SET_COLOR(color) printf(color)
#define RESET_COLOR                         printf(RESET)
#else
#define RED
#define GREEN
#define YELLOW
```

```
#define BLUE
#define RESET

#define SET_COLOR(color)
#define RESET_COLOR
#endif

#define TAO_THU_MUC_QUAN_LI {\
        printf("Tao cac thu muc can thiet...\n");\
        if (system("cd INPUT && cd .."))     system("mkdir INPUT");       \
        if (system("cd OUTPUT && cd .")) system("mkdir OUTPUT");   \
        if (system("cd log && cd .."))                  system("mkdir log");  \
        CLEAR_SCREEN;\
}

#define HR(file) fprintf(file,\
"============================================================\n
")
#define ENTER1 '\n'
#define ENTER2 '\r'
#define SPACE  ' '

#define MAX 100
#define CHAR_FORMAT "%7c"
#define DATA_FORMAT "%7.2f"
#define INPUT_DIR           "INPUT/"
#define LOG_PATH            "log/log.txt"
#define INPUT_PATH                  "DATA.INP"
#define OUTPUT_DIR                  "OUTPUT/"
#define OUTPUT_PATH                 "OUTPUT/DATA.OUT"

#define XUAT_NGHIEM(file, nghiem, i)     fprintf(file, "x%d = %.2g\n", i + 1, nghiem)
#define ERR_SO_DONG_LON_HON_COT(n, m)\
        printf(RED "So dong(%d) lon hon hoac bang so cot (%d)." RESET, n, m)
#define ERR_THUA_DONG  printf(RED "Qua nhieu dong.\n" RESET)
#define ERR_THIEU_COT(i) printf(RED "Qua it du lieu tren dong %d.\n" RESET, i + 1)
#define ERR_THUA_COT(i)  printf(RED "Qua nhieu du lieu tren dong %d.\n" RESET, i +
1)
#define ERR_DL_KHONG_HOP_LE(i, j, row, col) {
        if (row)          printf(RED "So dong co du lieu khong hop le.\n" RESET);\
        else if (col)     printf(RED "So cot co du lieu khong hop le.\n" RESET);\
        else\
        printf(RED "Du lieu khong hop le tai dong %d, cot %d.\n" RESET, i + 1, j + 1);\
}
```

```
#define ERR_FILE_KHONG_TON_TAI(file)\
printf(RED "File %s khong ton tai.\n" RESET, file)

#define TB_BD_THANH_CONG\
printf(GREEN "Ma tran da duoc bien doi!\n" RESET);
#define TB_DOI_DONG(i1, i2)\
printf(YELLOW "Doi dong %d va %d.\n" RESET, i1 + 1, i2 + 1)
#define TB_BIEN_DOI(i1, k, i2)\
printf(YELLOW "Lay dong %d tru di %g lan dong %d.\n" RESET, i2 + 1, k, i1 + 1)
#define MSG_VO_SO_NGHIEM     "He phuong trinh co vo so nghiem."
#define MSG_VO_NGHIEM                "He phuong trinh vo nghiem."
#define MSG_CO_NGHIEM                "He phuong trinh co nghiem la: "
#define OUTPUT_LABEL\
"\tTIM NGHIEM HE PHUONG TRINH TUYEN TINH"

typedef char *string;
typedef float Matrix[MAX][MAX];
typedef float Vector[MAX];
typedef unsigned char bool;
#define true 1
#define false 0
typedef void (*func)();

typedef struct {
        Matrix matrix;
        int n;
        int m;
} MatrixRecord;

typedef struct {
        string label;
        func action;
} MenuOption;

typedef struct Menu {
        MenuOption list[MAX];
        int count;
} * Menu;

Menu menu;
FILE* log_file;
MatrixRecord current;

void thoat();  // Ham thoat khoi chuong trinh.
```

```
Menu create_menu();  // Ham tao menu;
void show_matrix(Matrix, int, int, FILE*);
void hien_thi_hpt(Matrix, int, FILE*);

bool allow_color_showing_in_matrix = false;  // Cho phep hien mau trong qua trinh bien doi ma tran.
int pivot_row = -1;      // Chi so cua dong tru/doi dong (dong o tren).
int dest_row = -1;       // Chi so cua dong bi tru/doi dong (dong o duoi).
int current_row = -1;  // Chi so cua dong hien tai dang bien doi.
char output_file_path[MAX]; // Ten cua file dau ra.

void set_row_for_showing_color(int pivot, int dest) {
	pivot_row = pivot;
	dest_row = dest;
	if (pivot == -1 && dest == -1)
		current_row = 10e3;
	else
		current_row =((current_row < dest_row) ? dest_row : current_row);
}

void init() {  // Khoi tao cac bien toan cuc
	log_file = fopen(LOG_PATH, "w");
	menu = create_menu();
}

void destroy() {  // Giai phong cac bien toan cuc duoc cap phat.
	fclose(log_file);
	free(menu);
}

void enter_to_continue() {
	printf("\nNhan Enter/Return de tiep tuc...");
	getch();
}

// Cong don phan tu sau cot he so tu do vao cot he so tu do.
void cong_don(Matrix matrix, int *n, int *m) {
	for(int i = 0; i < *n; ++i)
		for (int j = *n + 1; j < *m; ++j)
		matrix[i][*n] += matrix[i][j];
	*m = *n + 1;
}

void cpy_mat(Matrix dest, int *n_dest, int *m_dest, Matrix src, int n, int m) {
```

```
        for (int i = 0; i < n; ++i)                 for (int j = 0; j < m; ++j)
                dest[i][j] = src[i][j];
        *n_dest = n;    *m_dest = m;
}

void xac_dinh_nghiem(Matrix matrix, int n, int m, Vector nghiem) {
   nghiem[n - 1] = matrix[n - 1][n]/matrix[n - 1][n - 1];
   float c = 0;
   for (int i = n - 2; i >= 0; --i) {
      c = 0;
      for (int j = n - 1; j > i; --j) {
         c += matrix[i][j] * nghiem[j];
      }
      nghiem[i] = (matrix[i][n] - c)/matrix[i][i];
      if (nghiem[i] == 0)     nghiem[i] = 0;
   }
}

void xuat_nghiem(FILE* file, MatrixRecord backup,
                Matrix matrix, int r, int r_mr, Vector nghiem) {
        const int SO_NGHIEM = backup.m - 1;
        fprintf(file, "\n\n\n%s\n\n", OUTPUT_LABEL);
        HR(file);
        hien_thi_hpt(backup.matrix, SO_NGHIEM, file);

        HR(file);
        fprintf(file, "%s\n", "Ma tran duoc bien doi la:");
        show_matrix(matrix, backup.n, backup.m, file);
        HR(file);

        if (r != r_mr)
                fprintf(file, "%s\n", MSG_VO_NGHIEM);
        else if (r < SO_NGHIEM)
                fprintf(file, "%s\n", MSG_VO_SO_NGHIEM);
        else {
                xac_dinh_nghiem(matrix, backup.n, backup.m, nghiem);
                fprintf(file, "%s\n", MSG_CO_NGHIEM);
                for (int i = 0; i < SO_NGHIEM; ++i)
                        XUAT_NGHIEM(file, nghiem[i], i);
        }
}

void swap(float *a, float *b) {
        float tmp = *a;
```

```
        *a = *b;
        *b = tmp;
}

void swap_row(Matrix matrix, int m, int i1, int i2) {
        for (int j = 0; j < m; ++j) swap(&matrix[i1][j], &matrix[i2][j]);
}

void bien_doi_ma_tran(Matrix matrix, int i0, int j0, int n, int m, bool show_step, int step) {
   if (i0 >= n || j0 >= m) {
                if(show_step)  TB_BD_THANH_CONG;
                set_row_for_showing_color(-1, -1);
                if (show_step)
                        show_matrix(matrix, n, m, stdout);
                allow_color_showing_in_matrix = false;
                current_row = -1;
                return;
        }
        allow_color_showing_in_matrix = show_step;
   int k;
   if (matrix[i0][j0] == 0) {
      // Tim vi tri dau tien khac 0.
      k = i0 + 1;
      while (k < n && matrix[k][j0] == 0)     ++k;
      if (k >= n)
                        return bien_doi_ma_tran(matrix, i0, j0 + 1, n, m, show_step, step);
      swap_row(matrix, m, i0, k);

      if (show_step) {
                        printf("\n\n");             HR(stdout);
                        printf("+)BUOC %d:\n", step++);
                        TB_DOI_DONG(i0, k);
                        set_row_for_showing_color(i0, k);
         show_matrix(matrix, n, m, stdout);
      }
   }

   float tmp;
   for (int i = i0 + 1; i < n; ++i) {
      if (matrix[i][j0] == 0)    continue;
      tmp = matrix[i][j0] / matrix[i0][j0];
      for (int j = j0; j < m; ++j)    matrix[i][j] -= tmp * matrix[i0][j];

      if (show_step) {
```

```
                        printf("\n\n");  HR(stdout);
                        printf("+) BUOC %d:\n", step++);
                        TB_BIEN_DOI(i0, tmp, i);
                        set_row_for_showing_color(i0, i);
            show_matrix(matrix, n, m, stdout);
        }
            }
            return bien_doi_ma_tran(matrix, i0 + 1, j0 + 1, n, m, show_step, step);
}

int rank(Matrix matrix, int n, int m) {
    int j = 0, r = 0;
    for (int i = 0; i < n; ++i) {
                while (j < m && matrix[i][j] == 0)  ++j;
                if (j < m)          ++r;
            }
    return r;
}

void bien_doi_Gauss(Matrix matrix, int n, int m, string output_path, bool check) {
    Vector nghiem;
    FILE* file = fopen(output_path, "a");

    MatrixRecord backup;
    cpy_mat(backup.matrix, &backup.n, &backup.m, matrix, n, m);

            printf("Hien cac buoc bien doi?\n");
            printf("+------------------------------+\n");
            printf("| Chon Enter/Return de dong y. |\n");
            printf("| Chon phim bat ki de tu choi. |\n");
            printf("+------------------------------+\n\n");
            printf("Chon: ");

            CLEAR_STDIN;
            char c = getche();
            bool show_step = (c == ENTER1 || c == ENTER2);

            bien_doi_ma_tran(matrix, 0, 0, n, m, show_step, 1);
            int r_mr = rank(matrix, n, m);
            int r = rank(matrix, n, m - 1);
            int so_nghiem = m - 1;

            xuat_nghiem(stdout, backup, matrix, r, r_mr, nghiem);
            if (check)
```

```
                xuat_nghiem(file, backup, matrix, r, r_mr, nghiem);
        fclose(file);
}

Menu create_menu() {
        Menu menu = malloc(sizeof(struct Menu));
        menu->count = 0;
        return menu;
}

void push(string label, func action, Menu menu) {
        MenuOption menu_option;
        menu_option.label = label;
        menu_option.action = action;
        menu->list[menu->count++] = menu_option;
}

void show_menu(Menu menu) {
        printf("\n");

        int size = strlen(menu->list[0].label) + 6;
        printf("+");
        for (int i = 1; i < size - 1; ++i)
                printf("-");
        printf("+\n");

        for (int i = 0; i < menu->count; ++i)
                printf("| %d/. %s\n", i + 1, menu->list[i].label);

        printf("+");
        for (int i = 1; i < size - 1; ++i)
                printf("-");
        printf("+\n\n");
        printf("%s", "Nhap lua chon cua ban: ");
        int op = (int)getchar();
        op -= '0' + 1;
        CLEAR_SCREEN;
        printf("\n");
        if (op >= menu->count || op < 0) {
                printf(RED "%s\n" RESET, "Canh bao: Lua chon khong hop le.");
        }
        else
                menu->list[op].action();
        CLEAR_STDIN;
```

```
        enter_to_continue();
   if (!LINUX)
            CLEAR_STDIN;
        if (menu->list[op].action == thoat) {
                // Neu user chon thoat thi giai phong bo nho va thoat chuong trinh.
                destroy();
                exit(0);
        }
}

// Nhap ma tran tu file (hoac stdin).
bool scan_matrix(Matrix matrix, int *n, int *m, FILE* file, FILE* log_file) {
        char dummy;
        int success;
        fprintf(log_file, "%s", "Nhap so dong: ");
        success = fscanf(file, "%d%c", n, &dummy);
        if (dummy != ENTER1 && dummy != SPACE && dummy != ENTER2 || success
== 0) {
                ERR_DL_KHONG_HOP_LE(0, 0, true, false);
                return false;
        }

        fprintf(log_file, "%s", "Nhap so cot: ");
        success = fscanf(file, "%d%c", m, &dummy);
        if ((dummy != ENTER1 && dummy != SPACE && dummy != ENTER2) || success
== 0) {
                ERR_DL_KHONG_HOP_LE(0, 0, false, true);
                return false;
        }

        if (*n  >= *m) {
                ERR_SO_DONG_LON_HON_COT(*n, *m);
                return false;
        }
        fprintf(log_file, "%s\n", "Nhap ma tran: ");
        for (int i = 0; i < *n; ++i)
                for (int j = 0;j < *m; ++j) {
                success = fscanf(file, "%f%c", &matrix[i][j], &dummy);
                if (feof(file))
                                dummy = '\n';
                        if (dummy != ENTER1 && dummy != SPACE && dummy !=
ENTER2 || success == 0) {
                                ERR_DL_KHONG_HOP_LE(i, j, false, false);
                                return false;
```

```
                        }
                        // Neu nhap tu ban phim thi khong can kiem tra dong, cot.
                        if (file == stdin)
                                continue;

                        const long int POSITION = ftell(file);
                        while (dummy == SPACE)    dummy = getc(file);
                        if ((dummy == ENTER1 || dummy == ENTER2) && j < *m - 1) {
                                ERR_THIEU_COT(i);
                                return false;
                        }
                        else if (dummy != ENTER1 && dummy != ENTER2 && j >= *m -
1) {
                                ERR_THUA_COT(i);
                                return false;
                        }
                        else
                                fseek(file, POSITION, SEEK_SET);
                }
        float tmp;
        if (file != stdin && !feof(file) && fscanf(file, "%f", &tmp) == 1) {
                ERR_THUA_DONG;
                return false;
        }

        printf(GREEN "%s\n" RESET, "Doc ma tran thanh cong!");

        cong_don(matrix, n, m);

        if (file == stdin)
                strcpy(output_file_path, OUTPUT_PATH);
   return true;
}

// Nhap ma tran tu file.
bool load_matrix(string path, bool read_path_stdin, Matrix matrix, int *n, int *m) {
        if (read_path_stdin) {
           CLEAR_STDIN;
                printf("%s", "Nhap duong dan file: ");
                fgets(path, MAX, stdin);
                path[strlen(path) - 1] = '\0';
        }

        FILE *file = fopen(path, "r");
```

```
        if (file == NULL) {
                char temp[MAX];
                strcpy(temp, INPUT_DIR);
                strcat(temp, path);
                file = fopen(temp, "r");
                if (file == NULL) {
                        ERR_FILE_KHONG_TON_TAI(path);
                        ERR_FILE_KHONG_TON_TAI(temp);
                        return false;
                }
        }
        bool foo = scan_matrix(matrix, n, m, file, log_file);

        // Doi phan mo rong tu .inp sang .out
        int size = strlen(path);
        int pivot = size;
        for (int i = size - 1; i >= 0; --i) {
                if (path[i] == '.')
                        pivot = i;
        }
        strcpy(output_file_path, "");
        strcat(output_file_path, OUTPUT_DIR);
        strncat(output_file_path, path, pivot + 1);
        strcat(output_file_path, "OUT");

        fclose(file);
        return foo;
}

// Ghi ma tran ra file (hoac stdout).
void show_matrix(Matrix matrix, int n, int m, FILE *file) {
        for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j){
        if (j == m - 1)
                fprintf(file, CHAR_FORMAT, '|');
        if (matrix[i][j] == 0 && i < current_row && allow_color_showing_in_matrix)
                SET_COLOR(GREEN);
        if (allow_color_showing_in_matrix && i == pivot_row)
                SET_COLOR(BLUE);
        if (allow_color_showing_in_matrix && i == dest_row)
                SET_COLOR(RED);
                        fprintf(file, DATA_FORMAT, matrix[i][j]);
                        RESET_COLOR;
```

```
                }
        fprintf(file, "\n");
        }
        fprintf(file, "\n");
}

void hien_thi_hpt(Matrix matrix, int so_nghiem, FILE* file) {
        int pivot;
        for (int i = 0; i < so_nghiem; ++i) {
                pivot = -1;
                for (int j = 0; j < so_nghiem; ++j) {
                        if (matrix[i][j] == 0) {
                                fprintf(file, "%10c", ' ');
                        } else {
                                if (pivot == -1)
                                        pivot = j;
                                float fabs_value = fabs(matrix[i][j]);
                                if (j == pivot)
                                        if (fabs_value != 1)
                                                fprintf(file, "%6.2g", matrix[i][j]);
                                        else
                                                fprintf(file, "%6c", ' ');
                                else {
                                        fprintf(file, "  %c", ((matrix[i][j] > 0) ? '+' : '-'));
                                        if (fabs_value != 1)
                                                fprintf(file, "%3.2g", fabs_value);
                                        else
                                                fprintf(file, "%3c", ' ');
                                }
                                fprintf(file, "[x%d]", j + 1);
                        }
                }
                fprintf(file, "  =%3.2g\n", matrix[i][so_nghiem]);
        }
}

void main_window() {
        CLEAR_SCREEN;
        SET_COLOR(YELLOW);
        printf("                PBL1: De tai 205\n\n");
        RESET_COLOR;
        show_matrix(current.matrix, current.n, current.m, stdout);
        show_menu(menu);
        main_window();
}
```

```
void introduce_window() {
	SET_COLOR(YELLOW);
	printf("		PBL1: De tai 205\n");
	RESET_COLOR;
	printf("	+) Tim nghiem he phuong trinh bang phuong phap Gauss.\n");
	printf("	+) Cong don phan tu sau cot he so tu do vao cot he so tu do.\n\n");
	SET_COLOR(YELLOW);
	printf("\n		Nhom 13:\n");
	RESET_COLOR;
	printf("		+) Vu Duc Minh.\n");
	printf("		+) Tran Nhat Nguyen.\n");
	enter_to_continue();
}

void nhap_tu_stdin() {
	MatrixRecord input;
	if (scan_matrix(input.matrix, &input.n, &input.m, stdin, stdout))
		cpy_mat(current.matrix, &current.n, &current.m, input.matrix, input.n,
input.m);
}

void nhap_tu_file() {
	char path[MAX];
	MatrixRecord input;
	if (load_matrix(path, true, input.matrix, &input.n, &input.m))
		cpy_mat(current.matrix, &current.n, &current.m, input.matrix, input.n,
input.m);
}

void xuat() {
	MatrixRecord cal;
	cpy_mat(cal.matrix, &cal.n, &cal.m, current.matrix, current.n, current.m);

	bien_doi_Gauss(cal.matrix, cal.n, cal.m, output_file_path, true);
	printf("\nXuat thanh cong vao file %s\n", output_file_path);
}

void bien_doi() {
	printf("+) Ma tran ban dau:\n");
	show_matrix(current.matrix, current.n, current.m, stdout);
	MatrixRecord cal;
	cpy_mat(cal.matrix, &cal.n, &cal.m, current.matrix, current.n, current.m);
	bien_doi_ma_tran(cal.matrix, 0, 0, cal.n, cal.m, true, 1);
}
```

```
void tim_nghiem() {
	MatrixRecord cal;
	cpy_mat(cal.matrix, &cal.n, &cal.m, current.matrix, current.n, current.m);

	bien_doi_Gauss(cal.matrix, cal.n, cal.m, OUTPUT_PATH, false);
}

void thoat() {
	printf("%s\n", "\tHen gap lai...");
}

int main() {
	TAO_THU_MUC_QUAN_LI;
	init();
	// Lay du lieu tu file DATA.INP
	load_matrix(INPUT_PATH, false, current.matrix, &current.n, &current.m);
	CLEAR_SCREEN;

	push("Nhap ma tran tu ban phim.                    |", nhap_tu_stdin, menu);
	push("Nhap ma tran tu file.                        |", nhap_tu_file, menu);
	push("Bien doi ma tran thanh dang bac thang.       |", bien_doi, menu);
	push("Tim nghiem he phuong trinh tuyen tinh.       |", tim_nghiem, menu);
	push("Xuat ket qua.                                |", xuat, menu);
	push("Thoat.                                       |", thoat, menu);

	introduce_window();
	main_window();

	destroy();
	return 0;
}
```

Chữ kí giảng viên