تمرین اول برنامه سازی پیشرفته

کار با ماتریس ها

مسعود یوسفی – 9912358043

استاد: مهندس بطحائیان

مقدمه

این برنامه همانطور که از اسمش مشخص است برنامه ای برای کار با ماتریس و دریافتشان به چند شیوه و چک کردن خصوصیات و ویژگی های انواع ماتریس است.

شرح برنامه

همانطور که در توضیحات برنامه امده بود باید برنامه ای مینوشتیم که کاربر با توجه به دستوراتی که در خط فرمان وارد میکند بتواند با ماتریس های یک بعدی کار کند و آن ها را مدیریت کند.

این مدیریت و کار با ماتریس ها شامل دریافت ماتریس به چند شیوه مختلف , چک و بررسی کردن ویژگی های ماتریس ها مانند قطری بودن , بالا مثلثی و پایین مثلثی بودن , همانی بودن, متقارن یا پاد متقارن بودن و ... میشود.

همچنین کاربر باید امکان دریافت , حدف و ویرایش ماتریس های با نوع داده ای متفاوت را داشته باشد.

ساختار برنامه

این برنامه متشکل از چند فایل است:

- فایل اصلی (main.cpp) که در آن تابع main برنامه وجود دارد.
- هدر فایل (funcs.hpp) که در آن تابع هایی که مربوط به چک کردن ویژگی های ماتریس هستند قرار دارند.
- هدر فایل (commands.hpp) که در آن تابع های مربوط به مدیریت دستور وارد شده و انجام عملیات بر روی ماتریس ها قرار دارد.
- هدر فایل (matrix_struct.hpp) که در آن استراکت مربوط به ماتریس ها و نگهداری اطلاعات ماتریس ها قرار دارد.

فایل اصلی (main.cpp)

در این فایل تابع main برنامه قرار دارد که درون آن یک وکتور(vector) از نوع استراکت ماتریس به عنوان نگهدارنده اصلی ماتریس های برنامه تعریف شده است.

و بعد از آن تابع menu که تابع مربوط به مدیریت دستورات و فراخوانی توابع دیگر است فراخوانی شده و وکتور matrix به آن پاس شده:

```cpp
#include <iostream>
#include "funcs.hpp"         //include header that has matrix based functions
#include "matrix_struct.hpp" //include header that has matrix defenition
#include "commands.hpp"      //include header that has commands related functions
#include <vector>
using namespace std;

int main()
{
    vector<matrixS> matrixV; //main container thar keeps matrixes
    menu(matrixV);           //menu function that shows menu and calls other functions based on input

    return EXIT_SUCCESS; //end of program
}
```

استراکت ماتریس

در اینجا برای مدیریت ماتریس ها استراکتی به نام matrixS تعریف شده که شامل عضو های زیر است:

- رشته name برای نگهداری نام ماتریس
- عدد صحیح size برای نگهداری اندازه ماتریس در صورت مربعی بودن ان
- عدد صحیح row برای نگهداری تعداد سطر ماتریس
- عدد صحیح col برای نگهداری تعداد ستون ماتریس
- عدد صحیح data_type برای نگهداری عدد مربوط به نوع داده موجود در ماتریس (عدد یک مربوط به اعداد و عدد دو مربوط به کاراکتر ها و عدد سه مربوط به رشته ها است.)
- و در اخر سه اشاره گر برای خود ماتریس که هنگام در یافت عناصر ماتریس با توجه به نوع داده ای عانصر فضای حافظه برای یکی از آنها در نظر گرفته میشود.

```
struct matrixS
{
    std::string name = "";
    int size = 0;
    int row;
    int col;
    int data_type;
    float **mat_n = nullptr;
    char **mat_c = nullptr;
    std::string **mat_s = nullptr;
};
```

توابع برنامه

این برنامه متشکل از چندین توابع است که اینجا کاربرد هر کدام آمده است:

```
bool is_diagonal(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس قطری است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
bool is_upper_triangular(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس بالا مثلثی است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
bool is_lower_triangular(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس پایین مثلثی است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
int is_triangular(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس پایین مثلثی است یا بالا مثلی و یا هیچکدام و نتیجه را به صورت یک عدد صحیح برمیگرداند که عدد صفر نشان دهنده هیچکدام و عدد یک نشان دهنده بالا مثلی و عدد دو نشان دهنده پایین مثلثی و عدد سه نشان دهنده هم بالا مثلثی و هم پایین مثلثی است.

```
bool is_identity(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس همانی است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
bool is_normal_symmetric(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس متقارن است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
bool is_skew_symmetric(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس پادمتقارن است یا خیر و نتیجه را به صورت true یا fasle بر میگرداند

```
int is_symmetric(matrixS matrix)
```

این تابع یک نمونه از استراکت ماتریس را دریافت میکند و بررسی میکند که آن ماتریس متقارن است یا پادمتقارن و یا هیچکدام و نتیجه را به صورت یک عدد صحیح برمیگرداند که عدد صفر نشان دهنده هیچکدام و عدد یک نشان دهنده متقارن و عدد دو نشان دهنده پادمتقارن و عدد سه نشان دهنده هم متقارن و هم پادمتقارن است.

```
bool is_reverse(std::string str1, std::string str2)
```

این تابع دو رشته را به عنوان ورودی دریافت میکند و بررسی میکند که ایا این دو رشته معکوس یگدیگر هستند یا خیر و نتیجه را به صورت true یا false بر میگرداند.

```
void show_commands()
```

این تابع در صورت فراخوانی تمامی دستورات معتبر که کاربر میتواند وارد کند را چاپ کرده و در صفحه نمایش میدهد.

```
bool is_number(const string &str)
```

این تابع یک رشته را به عنوان ورودی دریافت میکند و بررسی میکند که ایا تمام کاراکتر های رشته ورودی جزو کاراکتر های اعداد هستند یا خیر و نتیجه را به صورت true یا fasle برمیگرداند

```
void new_matrix(matrixS &matrix)
```

این تابع یک نمونه استراکت ماتریس را به عنوان ورودی دریافت کرده و با توجه به متغیر data_type موجود در نمونه و تعداد سطر و ستون موجود در ان فضای حافظه مربوط به ان نوع داده ای را برای ماتریس به اشاره گر مربوطه اختصاص میدهد

```
void free_matrix(matrixS &matrix)
```

این تابع یک نمونه استراکت ماتریس را به عنوان ورودی دریافت کرده و با توجه به متغیر data_type موجود در نمونه و تعداد سطر و ستون موجود در ان فضای حافظه مربوط به ان نوع داده ای موجود را آزاد میکند.

```
void show_matrix(string commandLine,
vector<matrixS> matrixV)
```

این تابع یک رشته دستوری را به عنوان ورودی دریافت میکند و اگر ان رشته نام یک ماتریس باشد ان ماتریس را از وکتور پاس شده پیدا کرده و اعضای آن را چاپ میکند.

```
void inverse(string &commandLine,
vector<matrixS> &matrixV)
```

این تابع یک رشته دستوری و وکتور اصلی برنامه را به عنوان ورودی دریافت میکند.

اگر رشته شامل نام یک ماتریس باشه آن ماتریس را در جای خود معکوس کرده و اگر شامل نام دیگری باشد معکوس ماتریس اول را به عنوان ماتریسی جدید با نام دوم در وکتور پاس شده ذخیره میکند.

```
void delete_matrix(string &commandLine,
vector<matrixS> &matrixV)
```

این تابع یک رشته دستوری و وکتور اصلی برنامه را به عنوان ورودی دریافت میکند.

اگر رشته شامل نام صحیحی از یک ماتریس ذخیره شده باشد آن ماتریس را از وکتور حذف کرده و فضای اختصاص داده شده به آن را آزاد میکند.

```
void change_matrix(string &commandLine,
vector<matrixS> &matrixV)
```

این تابع یک رشته دستوری و وکتور اصلی برنامه را به عنوان ورودی دریافت میکند.

پس از بررسی رشته دستوری اگر رشته شامل نام صحیحی از یک ماتریس ذخیره شده و شماره سطر و ستون درستی از آن ماتریس باشد. قسمت اخر رشته را با مقدار موجود در سطر و ستون داده شده در ماتریس در صورتی که از نظر نوع داده مغایرتی نداشته باشه تعویض میکند.(قابلیت ویرایش ماتریس)

```
int is_available(const string &name, const
vector<matrixS> &matrixV)
```

این تابع یک رشته و وکتور اصلی برنامه را به عنوان ورودی دریافت میکند.

اگر رشته دریافت شده نام صحیحی از یک ماتریس ذخیره شده در وکتور باشد اندیس خانه مربوط به آن ماتریس در وکتور را برمیگرداند و در غیر اینصورت عدد منفی یک را برمیگرداند.

```
void add_matrix(string &commandsLine,
vector<matrixS> &matrixV)
```

این تابع یک رشته دستوری و وکتور اصلی برنامه را به عنوان ورودی دریافت میکند و پس از تجزیه و تحلیل رشته دستوری در صورت درست بودن اطلاعات ماتریسی را از کاربر دریافت کرده و ان را در وکتور اصلی ذخیره میکند.

```
string split_command(string &commandLine)
```

این تابع رشته ای را به عنوان ورودی دریافت کرده و با حساسیت نسبت به فضای خالی (space) رشته را به دو قسمت تقسیم کرده. قسمت اول را به عنوان خروجی بر میگرداند و قسمت دوم را به عنوان یک رشته جدید در همان رشته ورودی (commandLine) ذخیره میکند.

```
void menu(vector<matrixS> &matrixV)
```

این تابع که بعد از تابع main مهم ترین تابع این برنامه محسوب میشد وکتور اصلی برنامه را به عنوان ورودی دریافت میکند.

این تابع یک رشته این را به عنوان دستور از خط فرمان دریافت میکند و پس از تجزیه و تحلیل و ان رشته به کمک توابع موجود تابع اصلی مورد نیاز کاربر را فراخوانی میکند.

نکته: در تمامی مراحل برنامه در صورت پیدا شدن خطا حاصل از در خواست اشتباه کاربر یا اطلاعات ورودی نادرست و ... خطای و پیغام مناسب چاپ و به کاربر نمایش داده میشود.

نکته: اطلاعات بیشتر بر روی سورس کد برنامه در غالب کامنت وجود دارد

پایان.