برنامه سازی پیشرفته

گزارشکار تمرین اول

هانیه اکبری

شماره دانشجویی : ۹۹۱۲۳۵۸۰۰۳

برنامه Main

```
int main()
{
    vector <matrix> mats;
    char ch;
    string command;
    system("clear");
    Menu();
    while (command != "exit")
    {

        getline(cin, command);

        if (command == "add matrix")
        {
            matrix temp = addMatrix<string>();
            if (temp.name != "error")
            mats.push_back(temp);
            //print all matrixes
            print_all(mats);
            cout << "Press the [m] key to return to the menu, press other key to exit : ";
            cin >> ch;
            if (ch == 'm')
            {
                system("clear");
                Menu();
                continue;
            }
            else
            {
                break;
            }
        }
```

در main برنامه ابتدا چند متغیر تعریف شده است و سپس با فراخوانی تابع Menu() منویی به کاربر نمایش داده می‌شود،

که شامل گزینه‌های زیر می‌باشد:

- add matrix
- is_diagonal
- is_upper_triangular
- is_lower_triangular
- is_triangular
- is_identity

- is_normal_symmetric
- is_skew_symmetric
- is_symmetric
- show

وقتی یکی از دستورات توسط کاربر وارد می‌شود، برنامه وارد حلقه while شده و تا زمانی که کاربر دستور exit را نزده داخل حلقه می‌ماند. (البته بعد از فراخوانی هر تابع جمله‌ای نمایش داده می‌شود به این منظور که برای بازگشت به منوی برنامه کاربر باید حرف m را وارد کرده و اگر هر کلید دیگری را فشار دهد از برنامه خارج می‌شود.)

add matrix

```
struct matrix
{
    string name;
    vector<vector<string>> row;
};
```

برای دریافت ماتریس از کاربر یک استراکت تعریف کردیم که داخلش یه متغیر برای اسم ماتریس و یک وکتور دو بعدی برای نگهداری ماتریس‌ها وجود دارد.

```
    struct matrix mat;
    int rowsCount, colsCount;
    cin >> mat.name;
    cin >> rowsCount;
```

داخل تابع add matrix یک متغیر از نوع استراکت ماتریکس و دو متغیر برای سطر و ستون ماتریس تعریف کردیم.

```
    char test = cin.peek();
    if (isspace(test))
    {
        cin.ignore(1);
        test = cin.peek();
    }
    if (isdigit(test))
    {
        cin >> colsCount;
    }
    else
        colsCount = rowsCount;
    if (rowsCount <= 0 || colsCount <= 0)
    {
        mat.name = "error";
        cout << "sizes must be positive" << endl;
        return mat;
    }
```

تابع cin.ignore() ⟵ اولین کاراکتر بافر را حذف می‌کند

تابع cin.peek() ⟵ کاراکتر بعدی بافر را می‌گیرد ولی از بافر حذفش نمی‌کند (برای تشخیص نوع ورودی استفاده کردم)

اگر آخرین خونه بافر عدد بود که نشان‌دهنده تعداد سطرهاست از کاربر تعداد ستون را می‌گیرد در غیر این صورت تعداد ستون را از کاربر نگرفته و یک ماتریس مربعی می‌سازد. در صورت منفی بودن تعداد سطر و ستون به کاربر اروری نشان داده می‌شود.

```
if (test == '[')
```

اگر به [برسیم که نشان‌دهنده مقداردهی ماتریس توسط کاربر می‌باشد، با دستور زیر تا زمانی که به] برسیم از کاربر مقادیر گرفته می‌شود. (declaration از نوع استرینگ است)

```
getline(cin, declaration, ']');
```

یک وکتور به صورت زیر تعریف کردیم که نوع آن هرچیزی می‌تواند باشد (int, float, string, ...)

```
vector <T> members;
```

```
while (index < declaration.size())
```

تا زمانی که به انتهای استرینگ نرسیدیم داخل حلقه‌ای با دستورت زیر باقی می‌مانیم.

```
string member;
for (; index < declaration.size(); index++)
{
// cout << "checking---> " << declaration[index] << endl;
if (isdigit(declaration[index]) || isalpha(declaration[index]) ||
declaration[index] == '.' || declaration[index] == '-')
{
   member += declaration[index];
}
if (!isdigit(declaration[index + 1]) && declaration[index + 1] != '.'
&& !isalpha(declaration[index + 1]))
{
   index++;
   break;
}
```

مقدار دهی کاربر به صورت روبه‌رو باید باشد(برای مثال) : [-۱۲،۵،۲،۷،۹] *

کلوشه اولی توسط تابع cin.ignore() حذف شده و تنها اعداد (که می‌تواند رشته یا هرچیز دیگری باشد) باقی مانده‌اند.

تا زمانی که به انتهای رشته نرسیدیم کاراکتر کاراکتر ازش می‌خوانیم و شرایط را چک می‌کنیم.

برای مثال: با توجه به * رشته ما ۱۲،۵،۲،۷،۹- می‌باشد. با توجه به شروط اولین if وارد آن شده و – را به member اضافه می‌کند و به ترتیب ۱ و ۲ را نیز کنار آن قرار می‌دهد وقتی به ، می‌رسد index را اضافه کرده و سراغ کاراکتر بعدی می‌رود.

در نهایت آن‌ها را به وکتور اضافه می‌کند.

square

```
bool Square(vector <matrix> mats, string name)
{
    int index = 0;
    for (matrix mat : mats)
    {
        if (mat.name == name)
        {
            for (auto col : mat.row)
            {
                for (auto num : col)
                {
                    index++;
                }
            }
        }
    }
    int temp = sqrt(index);
    if (temp * temp == index)
    {
        return true;
    }
    else
        return false;
}
```

این تابع مربعی بودن ماتریس را چک می‌کند. وقتی ماتریس مورد نظر را پیدا کرد تعداد خانه‌های آن را می‌شمارد و اگر تعداد آن مربع کامل بود true برمی‌گرداند.

is_diagonal

```
if (Square(mats, name))
    {
        for (matrix mat : mats)
        {
            if (mat.name == name)
            {
                for (auto col : mat.row)
                {
                    for (auto num : col)
                    {
                        number = stoi(num);
                        if (i != j && number != 0)
                        {
                            return false;
                        }
                        j++;
                    }
                    j = 0;
                    i++;
                }
            }
        }
        return true;
    }
```

در این تابع ابتدا مربعی بودن ماتریس چک شده چرا که برای قطری بودن ماتریس حتما باید مربعی باشد در صورت مربعی نبودن به کاربر اروری نشان داده می‌شود.

در این تابع همه‌ی خانه‌ها به جز قطراصلی ماتریس چک می‌شود و اگر عددی غیر از صفر بود false برمی‌گرداند.

تابع stoi() برای تبدیل رشته به اعداد می‌باشد.

is_triangular

برای بالامثلثی یا پایین مثلثی بودن ماتریس نیز ابتدا باید مربعی بودن آن چک شود.

```
if (i > j && number != 0)
{
    return false;
}
```

اگر خانه‌هایی از ماتریس که i > j است دارای مقادیری غیر از صفر باشد در این صورت ماتریس بالا مثلثی بوده.

```
if (j > i && number != 0)
{
    return false;
}
```

اگر خانه‌هایی از ماتریس که j > i است دارای مقادیری غیر از صفر باشد در این صورت ماتریس پایین مثلثی بوده.

is_identity

```
if (IsDiagonal(mats, name))
```

برای اینکه ماتریس همانی باشد باید قطری نیز باشد.

```
if (i == j && number != 1)
{
    return false;
}
```

اگر خانه‌های روی قطر اصلی هر عددی جز ۱ بود یعنی ماتریس همانی نبوده و false برمی‌گرداند.

is_symmetric

در این تابع هم در ابتدا باید مربعی بودن ماتریس چک شود.

```
if (mat.name == name)
{
   for (size_t i = 0; i < (mat.row).size(); i++)
  {
      for (size_t j = 0; j < mat.row[i].size(); j++)
      {
          if (mat.row[i][j] != mat.row[j][i])
          {
              return false;
          }
       }
   }
}
```

وقتی ماتریس با نام موردنظر پیدا شد خانه‌های آن چک می‌شود. اگر خانه‌ای که در سطر و ستون i و j قرار دارد با خانه‌ای که در سطر و ستون j و i قرار دارد برابر نبود، نشان دهنده نامتقارن بودن ماتریس می‌باشد.

برای پادمتقارن بودن ماتریس هم شرط زیر باید چک شود:

```
if (number1 != -1 * number2 && i != j)
```

number1 و number2 تبدیل یافته‌های خانه ماتریس از رشته به اعداد است.

Show

```
for (size_t i = 0; i < mat.row.size(); i++)
{
    for (size_t j = 0; j < mat.row[i].size(); j++)
    {
        cout << mat.row[i][j] << "\t";
```

```
    }
    puts("");
}
```

وقتی ماتریس با نام موردنظر یافت شد در حلقه بالا می‌چرخد و خانه‌های ماتریس را چاپ می‌کند.