تمرین شماره ۲ برنامه نویسی پیشرفته

طراحی سیستم بانکی

استاد: مهندس نرگس السادات بطحائیان

محمدرضا حیدرنیا – ۹۹۱۲۳۵۸۰۱۸

پاییز ۱۴۰۰

شرح پروژه

باید برنامه ایی را طراحی میکردیم که از طریق command line دستورات را از کاربر گرفته و آن ها را اجرا کند. در این تمرین چون یک سیستم بانکی طراحی میکردیم امنیت اهمیت بالایی داشت؛پس باید IP ، شماره کارت و username منحصر به فرد داشته باشیم.

در این برنامه قبل از انجام هر دستور باید username یا IP را پردازش میکردیم و مشتری را شناسایی میکردیم.

بیشتر برنامه به این صورت است که:

۱- username ورودی آیا حسابی دارد یا خیر؟

۲- سپس بعد از فهمیدن اینکه حساب دارد آیا آی پی ورودی صحیح است یا خیر؟

ساختار برنامه

در ایـن برنامـه یـک کـلاس مشـتری کـه مربـوط بـه امـور مشـتری است مانند واریز،برداشت،وام.....

Class customer{...};

****

یـک کـلاس هـم مربـوط تـراکنش هاسـت کـه در طـول برنامـه هـر تراکنشی اتفاق بـی افتـد جزئیـات آن ماننـد: تـاریخ ، مبـدا و مقصـد ، مقدار و .... در آن ثبت می شود.

Class banktransaction{...};

****

یک فایل برای توابع داریم که در آن ابتدا دستور ورودی را تشخیص می دهیم سپس به تابع مورد نظر آن را پاس میدهیم که دستور را اجرا کند.

function.hpp

function.cpp

روند برنامه به این شکل است که ابتدا در تابع main پیغام برای کاربر نمایش داده میشود سپس کاربر دستور مورد نظر خود را تایپ می کند

و بعد از آن وارد حلقه میشویم که شرط خروج از آن نوشتن کلمه exit است. دستور تایپ شده وارد تابعی میشود که دستور را تشخیص دهد و بعد تابع مربوط به دستور را فراخوانی میشود.

نکته: دو vector داریم که یکی از نوع کلاس مشتری و دیگری از کلاس تراکنش است که اشیا را در آنها ذخیر میکنیم و در طول برنامه از این دو وکتور استفاده میکنیم.

```
vector<banktransaction> alltransaction; //list for save transactions for each customer
vector<customer> cus;                   //list of customer class.
```

توابع برنامه

توابع فایل function.cpp :

```
bool command(string, vector<customer> &, vector<banktransaction> &)
```

دستور را میگیرد و با استفاده از توابع کلاس string دستور را تشخیص می دهد و به تابع مورد نظر پاس میدهد.

نکته:در کل برنامه وکتور های مر بوط به کلاس مشتری و کلاس تراکنش به صورت رفرنس به توابع پاس داده شده اند.

****

```
bool check_username(string, vector<customer> &)
```

هنگام ساخت اکانت، username را برسی میکند که شرایط لازم را داشته باشد.

****

```
bool check_ip(string)
```

هنگام ساخت اکانت، IP ورودی را برسی میکند.

****

```
bool add_ip(string, vector<customer> &)
```

اگر کاربر قصد اضافه کردن دوباره آی پی را داشته باشد از این تابع استفاده میشود. در ضمن این تابع توانایی اضافه کردن چند آی پی را به طور همزمان دارد.

****

```
int card_number(vector<customer> &)
```

بعد از تایید شدن نام کاربری و آی پی برنامه یک شماره کارت تصادفی ایجاد میکند؛ البته نباید این شماره کارت قبلا ایجاد شده باشد. برای ساختن شماره کارت از این تابع استفاده میشود.

****

```
void customer_transaction(string, vector<customer> &, vector<banktransaction> &)
```

برای واریز و برداشت مورد استفاده قرار میگیرد.

****

```
void transfer(string, vector<customer> &, vector<banktransaction> &)
```

برای انتقال از حسابی؛به حساب دیگر مورد استفاده قرار میگیرد.

****

void renewal(string, vector<customer> &, vector<banktransaction> &)

اگر کاربر قصد تمدید حساب را داشته باشد این تابع مورد استفاده قرار میگیرد.

****

void profits(string, vector<customer> &, vector<banktransaction> &)

برای دادن سود بانکی استفاده میشود.

****

void get_loan(string, vector<customer> &, vector<banktransaction> &)

اگر کاربر قصد گرفتن وام را داشته باشد. در صورتی که مشتری به هر دلیلی نتواند وام بگیرد آن دلیل به مشتری نمایش داده میشود.

****

void pay_loan(string, vector<customer> &, vector<banktransaction> &)

برای بازپرداخت وام از این تابع استفاده می کنیم.

****

bool show(string, vector<customer> &, vector<banktransaction> &)

در صورت تایپ کلمه بانک؛ اطلاعات بانک را نمایش می دهد و در صورت تایپ نام کاربری،آی پی و یا شماره کارت اطلاعت مشتری نمایش داده میشود.

****

توابع کلاس customer :

bool transaction(int, int, vector<customer> &)

در اینجا موجودی را تغییر میدهیم. برای آرگومان دوم از داده شمارشی استفاده میکنیم که خوانایی برنامه بالاتر باشد.

```
// enum for diffrent type of transaction
enum transaction
{
    //transactions
    DEPOSIT,
    WITHDRAW
};
```

****

bool checkIP(string)

درون وکتوری که آی پی هارا ذخیره میکند جستوجو میکند و در صورتی که آی پی قبلا ثبت شده باشد false را بر میگرداند.

****

void save_transactions(vector<banktransaction> &)

شماره تراکنش های مربوط به این حساب را در وکتوری ذخیره میکنیم.