برنامه سازی پیشرفته

گزارشکار تمرین دوم

هانیه اکبری

۹۹۱۲۳۵۸۰۰۳

```
static int count = 0;     //Number of people who opened a bank account
static int bankruptcy = 0;
```

در ابتدای برنامه ۲ متغیر static تعریف شده است. اولین متغیر برای نگهداری تعداد تمام مشتریان بانک و دومین متغیر برای اعلام ورشکستگی بانک است. (اگر متغیر یک باشد بانک ورشکسته شده است)

```
vector <Customer> create (vector <Customer>cus)
```

این تابع برای افتتاح حساب می‌باشد.

```
if (bankruptcy == 1)
{
   cout << "The bank has gone bankrupt" << endl;
   exit(0);
}
```

اگر بانک ورشکست شده باشد اعلام می‌کند.

```
for (size_t i = 0; i < cus.size(); i++)
{
        if (cus[i].account_number == c.get_account_number())
        {
            c.set_account_number();
        }
}
```

اگر شماره کارت کاربر قبلا ثبت شده باشد دوباره تابع موردنظر فراخوانی می‌شود.

```
if (c.set_ip(n) == true && temp == 1)
{
    cus.push_back(c);
    count++
}
```

تنها اگر آی پی وارد شده معتبر و یوزرنیم غیر تکراری بود، حساب با موفقیت ثبت می‌شود.

```
vector <Customer> renewal (vector <Customer> & cus)
```

این تابع برای تمدید حساب است.

کاربر یوزرنیم و آی پی خود را به صورت زیر وارد می‌کند:

```
Hanie:12.38.124.3
```

با استفاده از توابع find و substr یوزرنیم و آی پی را جدا می‌کنیم.

```
for (size_t i = 0; i < cus.size(); i++)
```

تا زمانی که به انتهای وکتور نرسیده باشد می‌چرخد و چک می‌کند اگر یوزرنیم کاربر پیدا شد ابتدا باید ببینیم کاربر موجودی کافی برای تمدید حساب خود دارد یا خیر.

اگر موجودی کافی بود یک دلار از حسابش کم و یک سال حساب کاربر تمدید می‌شود.

```
int temp = 0;
```

متغیر temp که در تمام توابع استفاده شده است برای این است که ببینیم کاربری با یوزرنیم وارد شده در بانک حساب دارد یا خیر.

```
void show (vector <Customer> cus)
```

این تابع اطلاعات تمام کاربران بانک را نمایش می‌دهد.

```
vector <Customer> deposit (vector <Customer> & cus)
```

تابع deposit برای واریز است.

```
Hanie:12.38.124.3:20000
```

دوباره با استفاده از توابع find و substr یوزرنیم و آی پی و پول واریزی جدا می‌شود.

```
for (size_t i = 0; i < cus.size(); i++)
    {
        if (cus[i].username == username1)
        {
            temp = 1;
            cus[i].balance =  cus[i].balance + deposit1;
    }
```

هنگامی که یوزرنیم کاربر پیدا شد پول واریزی به موجودی حساب کاربر اضافه می‌شود.

```
vector <Customer> transfer (vector <Customer> & cus)
```

این تابع برای انتقال است.

```
Hanie:12.38.124.3:Ali:20000
```

داده‌های بالا بر اساس علامت : با استفاده از توابع find و substr جدا می‌شوند.

ابتدا باید چک شود آیا هر دو (فرستنده و دریافت‌کننده) در بانک حساب دارند یا خیر. همچنین باید موجودی حساب فرستنده چک شود که پولی که می‌خواهد انتقال دهد به همان میزان یا بیشتر در حسابش هست یا خیر.

سپس اگر خطایی نداشتیم پول از موجودی حساب فرستنده کم و به موجودی حساب دریافت‌کننده اضافه میشود.

```
vector <Customer> withdraw (vector <Customer> & cus)
```

این تابع برای برداشت است.

```
Hanie:12.38.124.3:20000
```

با استفاده از توابع find و substr یوزرنیم و آی پی و پول جدا می‌شود.

در این تابع ابتدا موجودی کل بانک محاسبه می‌شود. اگر پولی که کاربر در حسابش دارد و می‌خواهد برداشت کند در بانک موجود نباشد، بانک اعلام ورشکستگی می‌کند.

اگر این اتفاق نیفتاد باید چک شود که پولی که کاربر می‌خواهد از حسابش بردارد آیا در حساب خودش هست یا خیر.

همچنین اگر کاربر بدهی داشته باشد اجازه برداشت ندارد.

```
long double Inventory (vector <Customer> cus)
```

این تابع موجودی کل بانک را حساب می‌کند.

به این صورت که تمام موجودی های کاربرانش را باهم جمع می‌کند و وام هایی که به کاربران داده است هم محاسبه می‌کند و اختلاف این دو را به عنوان موجودی کل برمی‌گرداند.

```
vector <Customer> getLoan (vector <Customer> & cus)
```

این تابع برای درخواست وام است.

برای پذیرفتنن این درخواست دو شرط لازم است. اول اینکه باید چک شود آیا مقدار وامی که کاربر می‌خواهد در بانک موجود هست یا خیر و همچنین ۳/۴ مقدار وام باید در حساب خود کاربر موجود باشد.

```
cus[i].debt = money1 / 10 + money1;
```

وقتی کاربر وام می‌گیرد هزینه وام بعلاوه ده درصد بهره به بدهی کاربر اضافه می‌شود.

```
vector <Customer> payLoan (vector <Customer> & cus)
```

این تابع برای بازپرداخت وام است.

```
if (money > cus[i].debt)
```

اگر پولی که کاربر پرداخت کرد بیشتر از مقدار بدهیش بود، بدهی کاربر صفر میشود و اختلاف بدهی و پول پرداخت شده به موجودی حساب کاربر اضافه می‌شود.

اگر پول پرداختی کمتر از بدهی بود اون مقدار از بدهی کاربر کم میشود.

```
void showCustomer (vector <Customer> cus)
```

این تابع با دریافت یوزرنیم یا آی پی کاربر اطلاعات حسابش را نشان می‌دهد.

```
void bank (vector <Customer> cus)
```

و این تابع اطلاعات بانک را نمایش می‌دهد.