

《银行综合业务系统设计报告》

目录

一、 引言	2
二、 任务分工	2
三、 系统架构设计	3
1. 模块划分	3
2. 数据结构设计	4
四、 功能模块详细设计	5
(一) 职员管理模块	5
1. 初始化菜单 (InitList1)	5
2. 查询职员信息 (SelectStaff)	5
3. 入职 (AddStaff)	7
4. 离职 (DeleteStaff)	8
5. 修改职员信息 (Revise)	10
6. 显示所有职员信息 (Display1)	12
(二) 客户管理模块	14
1. 初始化菜单 (InitList2)	14
2. 判断账户是否存在 (account_num_exist)	14
3. 查询账户信息 (SelectAccount)	14
4. 开户 (Open_an_ccount)	17
5. 销户 (Pin_households)	18
6. 修改账户密码 (Modify_account_password)	20
7. 账户排序 (Sort_accountInfo)	22
8. 存款 (deposit)	25
9. 取款 (withdrawals)	26
五、 特色创新	46
六、 用户手册	47
七、 总结体会	47

一、引言

本银行综合业务系统旨在实现银行的职员管理、客户账户管理、排队叫号管理以及网点查询等功能，为银行的日常运营提供高效、便捷的信息化支持。

二、任务分工

我们将作业要求整体分为：

1. 职员管理
2. 客户管理
3. 排队系统
4. 网点查询
5. 网页版管理系统制作
6. 设计报告撰写

分工如下

林烨洋：客户管理 网点查询 网页版管理系统制作

孙浩源：职员管理 设计报告撰写

代佳楠：排队系统 设计报告撰写

完成情况：

内容	完成情况
1. 银行职员管理：输入数据建立职员表，添加、删除、修改、查询职员信息（参考通讯录）	完成 100%
2. 客户账户管理：输入数据建立客户主账户表、银行卡分账号表，添加、删除、修改、查询客户和账户信息（参考广义表和通讯录）	完成 100%
3. 存取贷业务管理：存款、取款、贷款、利息计算、账户余额等（参考航空订票和图书管理）	完成 100%

4. 业务查询：根据不同关键字（时间段、客户类型、银行卡、金额区间、收入类型、支出类型等）查询业务明细和汇总信息。（参考航班查询）	完成 100%
5. 银行排队管理：分普通客户和 VIP 客户，排队取号，VIP 优先。银行窗口分 VIP 窗口和普通窗口，一位客户完成 100% 业务顺序叫号。客户给职员打分。	完成 100%
6. 银行网点查询：建立网点地图，查询网点信息，网点路径导航，网点增删改（参考校园导游）	完成 100%
7. 客户资料管理：建立客户资料文件（账号、不动产、信用等），存取客户信息。（参考文件）	完成 100%
8. 客户资料查询：查询客户资料中的信息，客户分类（普通和 VIP）。 （参考文学助手）	完成 100%
9. 主界面：管理、职员、客户登录审核与管理。图形或字符界面。主子菜单。（主界面设计）	完成 100%
10. 将字符界面的系统转换成网页版图形界面	完成 100%

三、系统架构设计

1. 模块划分

职员管理模块：负责银行职员信息的维护，包括入职、离职、信息查询、修

改等操作。

客户管理模块：处理客户账户相关事务，如开户、销户、查询、转账、存款、取款、利息计算等。

排队管理模块：实现排队叫号功能，区分 VIP 和普通客户，包括放号、叫号服务完成、客户打分等操作。

网点查询模块：提供银行网点信息查询，如网点介绍、最短路径查询、游览路线规划、网点信息修改等功能。

2. 数据结构设计

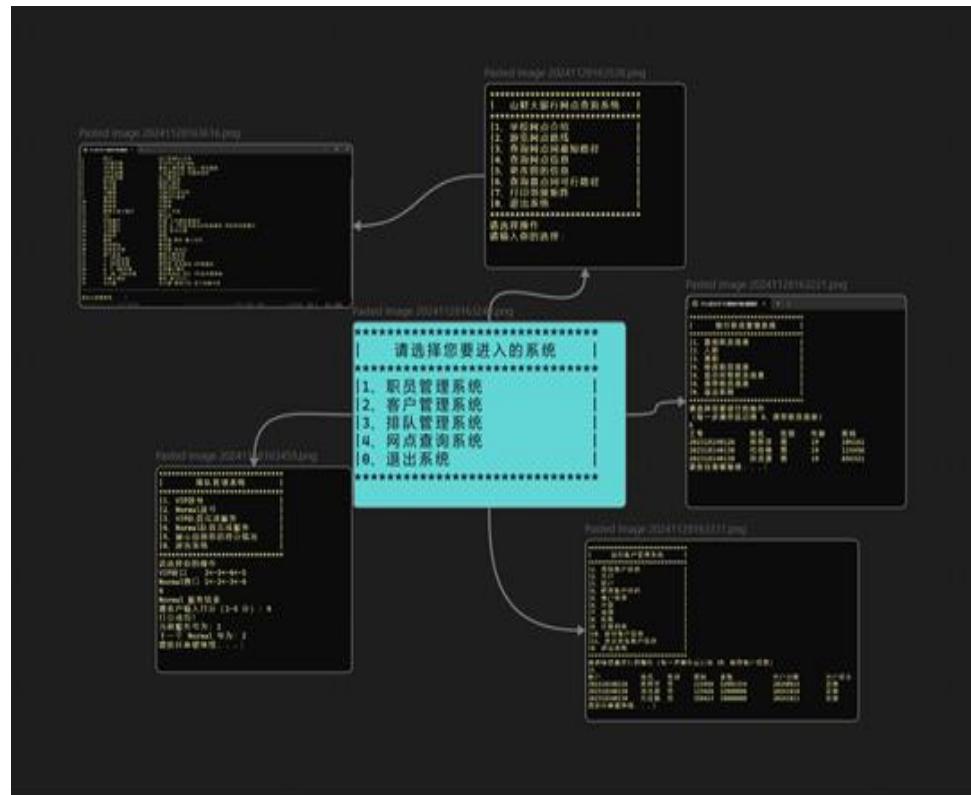
职员结构体 (Staff)：包含工号 (Number)、姓名 (name)、性别 (sex)、年龄 (age)、密码 (password)、客户所打的平均分 (grade)、总接待人数 (n) 以及指向下一职员节点的指针 (next)。

客户结构体 (Banker)：包括银行账户 (account_num)、银行卡持有人姓名 (name)、性别 (sex)、银行卡密码 (password)、银行卡金额 (money)、开户日期 (open_time)、开户状态 (states)、客户类型 (type) 以及指向下一客户节点的指针 (next)。

图结构体 (MGraph)：

用于表示银行网点信息，包含顶点个数 (vexnum)、边的个数 (arcnum)、顶点数组 (vexs) 以及邻接矩阵 (arcs)，用于存储网点间的连接关系和距离信息。

队列 (queue)：用于排队管理模块，分别存储 VIP 客户排队号 (VIPQue) 和普通客户排队号 (normalQue)。



四、功能模块详细设计

(一) 职员管理模块

1. 初始化菜单 (InitList1)

- 功能：创建职员链表的头节点，并将其 next 指针初始化为 NULL。

- 实现：

```
```cpp
void InitList1(StaffList& SL)
{
 SL = new Staff;
 SL->next = NULL;
}
```

```

2. 查询职员信息 (SelectStaff)

- 功能：根据用户选择的查询方式（工号或姓名），在职员链表中查找并显示相应职员的信息。

- 系统测试：



The screenshot shows a terminal window with the following content:

```
*****
|      银行职员管理系统      |
*****
|1、查询职员信息          |
|2、入职                  |
|3、离职                  |
|4、修改职员信息          |
|5、显示所有职员信息      |
|6、保存职员信息          |
|0、退出系统              |
*****
请选择您要进行的操作
(每一步操作后记得 6、保存职员信息)
1
1:按工号查找
2:按员工姓名查找
2
请输入要查询的姓名：小明
工号      姓名      性别      年龄      密码
202318140138  小明      男        20       123456
请按任意键继续... 
```

- 实现：

```
```cpp
void SelectStaff(StaffList& SL) {
```

---

```
cout << "1:按工号查找 " << endl;
cout << "2:按员工姓名查找 " << endl;
string num; // 工号
string name; // 姓名
int ch;
Staff* p = SL->next;
bool isFound = false;
cin >> ch;
switch (ch) {
case 1:
{
 cout << "请输入要查询的工号: ";
 cin >> num;
 while (p) {
 if (p->Number == num) {
 cout << "工号\t\t" << "姓名\t" << "性别\t" << "年龄\t" << "密码" << endl;
 cout << p->Number << "\t" << p->name << "\t" << p->sex << "\t" << p->age << "\t" << p->password
 << endl;
 isFound = true;
 break;
 }
 p = p->next;
 }
}
break;
case 2:
{
 cout << "请输入要查询的姓名: ";
 cin >> name;
 while (p) {
 if (p->name == name) {
 cout << "工号\t\t" << "姓名\t" << "性别\t" << "年龄\t" << "密码" <<
 endl;
 cout << p->Number << "\t" << p->name << "\t" << p->sex << "\t" << p->age
 << "\t" << p->password << endl;
 isFound = true;
 break;
 }
 p = p->next;
 }
}
break;
default:
{
 cout << "输入有误! " << endl;
 break;
}
```

```
 }
 if (!isFound) cout << "未查找到!" << endl;
 }
 ...
```

### 3. 入职 (AddStaff)

- 功能：接收新职员信息，创建新的职员节点并插入到职员链表中，同时检查工号是否存在。

- 系统测试：

```
C:\Users\lenovo\Documents\> *****
| 银行职员管理系统 |

|1、查询职员信息|
|2、入职|
|3、离职|
|4、修改职员信息|
|5、显示所有职员信息|
|6、保存职员信息|
|0、退出系统|

请选择您要进行的操作
(每一步操作后记得 6、保存职员信息)
2
请输入需要入职的工号: 3
请输入入职员工的姓名: 小明
请输入入职员工的性别(男, 女): 男
请输入入职员工的年龄: 20
请输入员工账户的密码(6位): *****
```

- 实现：

```
```cpp
void AddStaff(StaffList& SL) {
    Staff* p = new Staff;
    p->next = NULL;
    cout << "请输入需要入职的工号: ";
    cin >> p->Number;
    if (staff_num_exist(SL, p->Number))
    {
        cout << "入职失败, 该员工已存在" << endl;
        return;
    }
    cout << "请输入入职员工的姓名: ";
    cin >> p->name;
    cout << "请输入入职员工的性别(男, 女): ";
    cin >> p->sex;
    cout << "请输入入职员工的年龄: ";
    cin >> p->age;
```

```
cout << "请输入员工账户的密码(6位): ";
int password;
char charPassword[20]; // 用于存储密码字符数组
int i = 0;
char ch;
while ((ch = _getch()) != '\r') // \r 是回车键
{
    if (ch == '\b' && i > 0) // 处理退格键
    {
        cout << "\b \b";
        --i;
    }
    else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
    {
        charPassword[i++] = ch;
        cout << '*';
    }
}
charPassword[i] = '\0'; // 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
p->password = password;

Staff* pp = SL;
while (pp->next)
{
    pp = pp->next;
}
pp->next = p;
cout << "入职成功!" << endl;
}
```

```

#### 4. 离职 (DeleteStaff)

- 功能：根据输入的工号和密码，在职员链表中删除相应的职员节点。
- 系统测试：

```

| 银行职员管理系统 |

|1、查询职员信息|
|2、入职|
|3、离职|
|4、修改职员信息|
|5、显示所有职员信息|
|6、保存职员信息|
|0、退出系统|

请选择您要进行的操作
(每一步操作后记得 6、保存职员信息)
3
请输入需要离职的员工工号：3
请输入需要离职的员工的密码 :******

离职成功！
请按任意键继续. . .

```

- 实现：

```

```cpp
void DeleteStaff(StaffList& SL)
{
    string account_num; // 账户
    int password; // 密码
    Staff* p = SL;
    cout << "请输入需要离职的员工工号：" ;
    cin >> account_num;
    if (staff_num_exist(SL, account_num))
    {
        cout << "请输入需要离职的员工的密码：" ;
        char charPassword[20]; // 用于存储密码字符数组
        int i = 0;
        char ch;
        while ((ch = _getch()) != '\r') // \r 是回车键
        {
            if (ch == '\b' && i > 0) // 处理退格键
            {
                cout << "\b \b";
                --i;
            }
            else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
            {
                charPassword[i++] = ch;
                cout << '*';
            }
        }
        charPassword[i] = '\0'; // 字符串结束标志
        password = atoi(charPassword); // 将字符数组转换为整数
    }
}

```

```

cout << endl;
cout << "\n";
while (p->next!= NULL)
{
    if (p->next->Number == account_num && p->next->password == password)
    {
        cout << "离职成功！" << endl;
        Staff* pt = p->next;
        p->next = pt->next;
        delete pt;
        return;
    }
    p = p->next;
}
cout << "\n 输入的密码有误！" << endl;
}
else
{
    cout << "此员工不存在！" << endl;
}
}
```

```

## 5. 修改职员信息 (Revise)

- 功能：根据输入的工号和密码，修改相应职员的密码信息。
- 系统测试：



The screenshot shows a terminal window with the following text:

```

| 银行职员管理系统 |

|1、查询职员信息
|2、入职
|3、离职
|4、修改职员信息
|5、显示所有职员信息
|6、保存职员信息
|0、退出系统

请选择您要进行的操作
(每一步操作后记得 6、保存职员信息)
4
请输入需要修改的员工工号: 2
请输入需要修改的员工的密码 :******

请输入新的密码: *****
修改成功!
请按任意键继续. . .

```

- 实现：

```

```cpp
void Revise(StaffList& SL)

```

```
{  
    string account_num; // 账户  
    int password; // 密码  
    Staff* p = SL->next;  
    cout << "请输入需要修改的员工工号: ";  
    cin >> account_num;  
    if (staff_num_exist(SL, account_num))  
    {  
        cout << "请输入需要修改的员工的密码: ";  
        char charPassword[20]; // 用于存储密码字符数组  
        int i = 0;  
        char ch;  
        while ((ch = _getch()) != '\r') // \r 是回车键  
        {  
            if (ch == '\b' && i > 0) // 处理退格键  
            {  
                cout << "\b \b";  
                --i;  
            }  
            else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)  
            {  
                charPassword[i++] = ch;  
                cout << '*';  
            }  
        }  
        charPassword[i] = '\0'; // 字符串结束标志  
        password = atoi(charPassword); // 将字符数组转换为整数  
        cout << endl;  
        cout << "\n";  
        while (p != NULL)  
        {  
            if (p->Number == account_num && p->password == password)  
            {  
                cout << "请输入新的密码: ";  
                int password;  
                char charPassword[20]; // 用于存储密码字符数组  
                int i = 0;  
                char ch;  
                while ((ch = _getch()) != '\r') // \r 是回车键  
                {  
                    if (ch == '\b' && i > 0) // 处理退格键  
                    {  
                        cout << "\b \b";  
                        --i;  
                    }  
                }  
                charPassword[i] = '\0';  
                password = atoi(charPassword);  
            }  
            p = p->next;  
        }  
    }  
}
```

```

        else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
        {
            charPassword[i++] = ch;
            cout << '*';
        }
    }

    charPassword[i] = '\0'; // 字符串结束标志
    password = atoi(charPassword); // 将字符数组转换为整数
    cout << endl;
    p->password = password;
    cout << "修改成功!" << endl;
    return;
}

p = p->next;
}
cout << "密码错误!" << endl;
}
else
{
    cout << "此员工不存在!" << endl;
}
}
```

```

## 6. 显示所有职员信息 (Display1)

- 功能：遍历职员链表，显示所有职员的信息。
- 系统测试：



The screenshot shows a terminal window with the following content:

```

| 银行职员管理系统

|1. 查询职员信息
|2. 入职
|3. 离职
|4. 修改职员信息
|5. 显示所有职员信息
|6. 保存职员信息
|0. 退出系统

请选择您要进行的操作
(每一步操作后记得 6、保存职员信息)

5
工号 姓名 性别 年龄 密码
23 大大 是 23 213154
2 小芳 女 19 123456
1 小明 男 20 123456
请按任意键继续. . .

```

- 实现：

```cpp

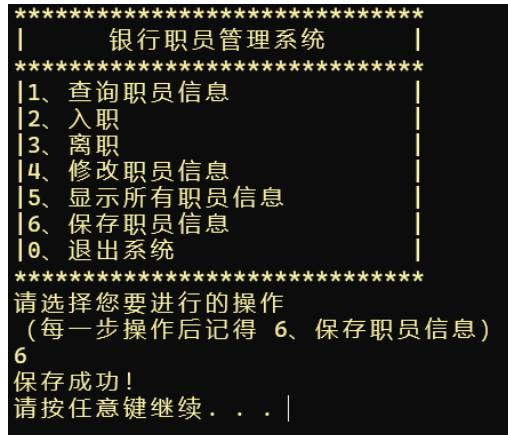
```

void Display1(StaffList& SL) {
    if (SL->next == NULL) {
        cout << "该职员表为空!" << endl;
    }
    else {
        Staff* p = SL->next;
        cout << "工号\t\t" << "姓名\t" << "性别\t" << "年龄\t" << "密码" << endl;
        while (p)
        {
            cout << p->Number << "\t" << p->name << "\t" << p->sex << "\t" << p->age << "\t" <<
p->password << endl;
            p = p->next;
        }
    }
}
```

```

## 7. 保存职员信息 (Save1)

- 功能：将职员链表中的信息保存到指定的文件中。
- 系统测试：



- 实现：

```

```cpp
void Save1(StaffList& SL, string file)
{
    ofstream ofs;
    ofs.open(file.c_str());
    Staff* p = SL->next;
    while (p)
    {
        ofs << p->Number << "\t" << p->name << "\t" << p->sex << "\t" << p->age << "\t" <<
p->password << endl;
        p = p->next;
    }
}
```

```

---

```
 }
 ofs.close();
 cout << "保存成功!" << endl;
}
```

```

(二) 客户管理模块

1. 初始化菜单 (InitList2)

- 功能：创建客户链表的头节点，并将其 `next` 指针初始化为 `NULL`。
- 实现：

```
```cpp
void InitList2(BankList& BL)
{
 BL = new Banker;
 BL->next = NULL;
}
```

```

2. 判断账户是否存在 (account_num_exist)

- 功能：在客户链表中查找指定账户是否存在。
- 实现：

```
```cpp
bool account_num_exist(BankList& BL, string account_num)
{
 Banker* p = BL->next;
 while (p)
 {
 if (p->account_num == account_num)
 return true;
 p = p->next;
 }
 return false;
}
```

```

3. 查询账户信息 (SelectAccount)

- 功能：根据用户选择的查询方式（账号、开户人姓名或开户日期），在客户链表中查找并显示相应账户的信息。
- 系统测试：

```

*****
| 银行账户管理系统 |
*****
|1、查询账户信息|
|2、开户|
|3、销户|
|4、修改账户密码|
|5、账户排序|
|6、存款|
|7、取款|
|8、转账|
|9、计算利息|
|10、保存账户信息|
|11、显示所有账户信息|
|0、退出系统|
*****
请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
1
1:按账号查找:
2:按开户人姓名查找:
3:按开户日期查找:
1
请输入需要查询的账号: 202518140138
账户      姓名    性别    密码    金额        开户日期      开户状态
202518140138    张三    男      123456  1000100000  20241216    活期
请按任意键继续...

```

- 实现:

```

```cpp
void SelectAccount(BankList& BL)
{
 cout << "1:按账号查找:" << endl;
 cout << "2:按开户人姓名查找:" << endl;
 cout << "3:按开户日期查找:" << endl;
 string account_num; // 账户
 string name; // 姓名
 int open_time; // 开户日期
 int ch;
 Banker* p = BL->next;
 cin >> ch;
 switch (ch)
 {
 case 1:
 {
 cout << "请输入需要查询的账号: ";
 cin >> account_num;
 while (p)
 {
 if (p->account_num == account_num)
 {
 cout << "账户\t\t姓名\t\t性别\t\t密码\t\t金额\t\t开户日期\t\t开户状态" << endl;
 cout << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" <<
p->password
 << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
 }
 }
 }
 }
}
```

```

```
    }
    p = p->next;
}
}

break;
case 2:
{
    cout << "请输入需要查询的开户人姓名: ";
    cin >> name;
    while (p)
    {
        if (p->name == name)
        {
            cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态" << endl;
            cout << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" <<
p->password
                << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
            return;
        }
        p = p->next;
    }
}
break;
case 3:
{
    cout << "请输入需要查询的开户日期: ";
    cin >> open_time;
    while (p)
    {
        if (p->open_time == open_time)
        {
            cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态" << endl;
            cout << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" <<
p->password
                << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
            return;
        }
        p = p->next;
    }
}
break;
default:
    cout << "输入有误! " << endl;
    break;
}
```

```
        cout << "未查找到!" << endl;
    }
```

```

#### 4. 开户 (Open\_an\_ccount)

- 功能：接收新客户账户信息，创建新的客户节点并插入到客户链表中，同时检查账户是否已存在。

- 系统测试：

```

| 银行账户管理系统 |

|1、查询账户信息|
|2、开户|
|3、销户|
|4、修改账户密码|
|5、账户排序|
|6、存款|
|7、取款|
|8、转账|
|9、计算利息|
|10、保存账户信息|
|11、显示所有账户信息|
|0、退出系统|

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
2
请输入需要开户的账户：晚安
请输入需要开户的姓名：小辉
请输入需要开户的性别(男, 女)：男
请输入需要开户的密码(6位)：*****
请输入需要开户的金额：8848
请输入需要开户的开户日期(如：20050731)：20241130
请输入需要开户的开户状态(活期, 定期)：活期
开户成功!
请按任意键继续... |
```

- 实现：

```
```cpp
void Open_an_ccount(BankList& BL)
{
    Banker* p = new Banker;
    p->next = NULL;
    cout << "请输入需要开户的账户：" ;
    cin >> p->account_num;
    if (account_num_exist(BL, p->account_num))
    {
        cout << "开户失败，账户已存在" << endl;
        return;
    }
    cout << "请输入需要开户的姓名：" ;
    cin >> p->name;
    cout << "请输入需要开户的性别(男, 女)：" ;
```

```
cin >> p->sex;
cout << "请输入需要开户的密码(6位): ";
int password;
char charPassword[20]; // 用于存储密码字符数组
int i = 0;
char ch;
while ((ch = _getch()) != '\r') // \r 是回车键
{
    if (ch == '\b' && i > 0) // 处理退格键
    {
        cout << "\b \b";
        --i;
    }
    else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
    {
        charPassword[i++] = ch;
        cout << '*';
    }
}
charPassword[i] = '\0'; // 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
p->password = password;
cout << "请输入需要开户的金额: ";
cin >> p->money;
cout << "请输入需要开户的开户日期(如: 20050731): ";
cin >> p->open_time;
cout << "请输入需要开户的开户状态(活期, 定期): ";
cin >> p->states;
Banker* pp = BL;
while (pp->next)
{
    pp = pp->next;
}
pp->next = p;
cout << "开户成功! " << endl;
}
```

```

## 5. 销户 (Pin\_households)

- 功能：根据输入的账户和密码，在客户链表中删除相应的客户节点。
- 实现：

```
```cpp
void Pin_households(BankList& BL)
```

```
{  
    string account_num; // 账户  
    int password; // 密码  
    Banker* p = BL;  
    cout << "请输入需要销户的账户: ";  
    cin >> account_num;  
    if (account_num_exist(BL, account_num))  
    {  
        cout << "请输入需要销户的密码: ";  
        char charPassword[20]; // 用于存储密码字符数组  
        int i = 0;  
        char ch;  
        while ((ch = _getch()) != '\r') // \r 是回车键  
        {  
            if (ch == '\b' && i > 0) // 处理退格键  
            {  
                cout << "\b \b";  
                --i;  
            }  
            else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)  
            {  
                charPassword[i++] = ch;  
                cout << '*';  
            }  
        }  
        charPassword[i] = '\0'; // 字符串结束标志  
        password = atoi(charPassword); // 将字符数组转换为整数  
        cout << endl;  
        cout << "\n";  
        while (p->next != NULL)  
        {  
            if (p->next->account_num == account_num && p->next->password == password)  
            {  
                cout << "销户成功! " << endl;  
                Banker* pt = p->next;  
                p->next = pt->next;  
                delete pt;  
                return;  
            }  
            p = p->next;  
        }  
        cout << "\n 输入的密码有误! " << endl;  
    }  
    else  
    {
```

```
    cout << "此账户不存在！" << endl;
}
```
``
```

## 6. 修改账户密码 (Modify\_account\_password)

- 功能：根据输入的账户和原密码，修改相应账户的密码。
- 系统测试：

The screenshot shows a terminal window for the 'Bank Account Management System'. The menu options are:

- 1. 查询账户信息
- 2. 开户
- 3. 销户
- 4. 修改账户密码
- 5. 账户排序
- 6. 存款
- 7. 取款
- 8. 转账
- 9. 计算利息
- 10. 保存账户信息
- 11. 显示所有账户信息
- 0. 退出系统

Below the menu, the system prompts the user to select an operation (step 10 for saving account information) and then performs step 4 (modify account password). It asks for the account number ('晚安'), current password ('\*\*\*\*\*'), new password ('\*\*\*\*\*'), and confirms the success of the modification.

```

| 银行账户管理系统 |

|1、查询账户信息|
|2、开户|
|3、销户|
|4、修改账户密码|
|5、账户排序|
|6、存款|
|7、取款|
|8、转账|
|9、计算利息|
|10、保存账户信息|
|11、显示所有账户信息|
|0、退出系统|

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)

4
请输入需要修改的账户：晚安
请输入需要修改的账户密码 :*****

请输入新的密码： *****
修改成功！
请按任意键继续. . . |
```

- 实现：

```
```cpp
void Modify_account_password(BankList& BL)
{
    string account_num; // 账户
    int password; // 密码
    Banker* p = BL->next;
    cout << "请输入需要修改的账户：" ;
    cin >> account_num;
    if (account_num_exist(BL, account_num))
    {
        cout << "请输入需要修改的账户密码：" ;
        char charPassword[20]; // 用于存储密码字符数组
        int i = 0;
        char ch;
        while ((ch = _getch()) != '\r') // \r 是回车键
        {
            if (ch == '\b' && i > 0) // 处理退格键
            {
```

```
        cout << "\b \b";
        --i;
    }
    else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
    {
        charPassword[i++] = ch;
        cout << '*';
    }
}
charPassword[i] = '\0'; // 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
cout << "\n";
while (p!= NULL)
{
    if (p->account_num == account_num && p->password == password)
    {
        cout << "请输入新的密码: ";
        int password;
        char charPassword[20]; // 用于存储密码字符数组
        int i = 0;
        char ch;
        while ((ch = _getch()) != '\r') // \r 是回车键
        {
            if (ch == '\b' && i > 0) // 处理退格键
            {
                cout << "\b \b";
                --i;
            }
            else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
            {
                charPassword[i++] = ch;
                cout << '*';
            }
        }
        charPassword[i] = '\0'; // 字符串结束标志
        password = atoi(charPassword); // 将字符数组转换为整数
        cout << endl;
        p->password = password;
        cout << "修改成功! " << endl;
        return;
    }
    p = p->next;
}
cout << "密码错误! " << endl;
```

```
    }
    else
    {
        cout << "此账户不存在!" << endl;
    }
}
```
```

## 7. 账户排序 (Sort\_accountInfo)

- 功能：根据用户选择的排序方式（账号、开户人姓名、金额或开户日期），对客户链表中的账户信息进行排序并显示。
- 系统测试：

The screenshot shows a terminal window for the 'Bank Account Management System'. It displays a menu with 11 options from 1 to 11, followed by an exit option (0). The user selects option 5 (Account Sort). They then choose option 3 (Sort by Amount) and enter the number 3. The system responds with 'Sorting successful!'.

```

| 银行账户管理系统 |

|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
5
1:按账号排序
2:按开户人姓名排序
3:按金额排序
4:按开户日期排序
请输入选择的排序编号：
3
排序成功！
请按任意键继续... |
```

- 实现：

```
```cpp
void Sort_accountInfo(BankList& BL)
{
    int ch;
    cout << "1:按账号排序" << endl;
    cout << "2:按开户人姓名排序" << endl;
    cout << "3:按金额排序" << endl;
    cout << "4:按开户日期排序" << endl;
    cout << "请输入选择的排序编号：" << endl;
    cin >> ch;
    Banker* p1 = BL->next;
    switch (ch)
    {
```

```
case 1:
{
    for ( ; p1->next!= NULL; p1 = p1->next)
    {
        for (Banker* p2 = p1->next; p2!= NULL; p2 = p2->next)
        {
            if (p1->account_num > p2->account_num) // 升序排列
            {
                Banker* p11 = p1->next;
                Banker* p22 = p2->next;
                Banker t = *p1;
                *p1 = *p2;
                *p2 = t;
                p1->next = p11;
                p2->next = p22;
            }
        }
    }
    cout << "排序成功! " << endl;
}
break;

case 2:
{
    for ( ; p1->next!= NULL; p1 = p1->next)
    {
        for (Banker* p2 = p1->next; p2!= NULL; p2 = p2->next)
        {
            if (p1->name > p2->name) // 升序排列
            {
                Banker* p11 = p1->next;
                Banker* p22 = p2->next;
                Banker t = *p1;
                *p1 = *p2;
                *p2 = t;
                p1->next = p11;
                p2->next = p22;
            }
        }
    }
    cout << "排序成功! " << endl;
}
break;

case 3:
{
    for ( ; p1->next!= NULL; p1 = p1->next)
```

```
{  
    for (Banker* p2 = p1->next; p2!= NULL; p2 = p2->next)  
    {  
        if (p1->money > p2->money) // 升序排列  
        {  
            Banker* p11 = p1->next;  
            Banker* p22 = p2->next;  
            Banker t = *p1;  
            *p1 = *p2;  
            *p2 = t;  
            p1->next = p11;  
            p2->next = p22;  
        }  
    }  
}  
cout << "排序成功！" << endl;  
}  
break;  
case 4:  
{  
    for (; p1->next!= NULL; p1 = p1->next)  
    {  
        for (Banker* p2 = p1->next; p2!= NULL; p2 = p2->next)  
        {  
            if (p1->open_time > p2->open_time) // 升序排列  
            {  
                Banker* p11 = p1->next;  
                Banker* p22 = p2->next;  
                Banker t = *p1;  
                *p1 = *p2;  
                *p2 = t;  
                p1->next = p11;  
                p2->next = p22;  
            }  
        }  
    }  
    cout << "排序成功！" << endl;  
}  
break;  
default:  
    cout << "您的输入有误！" << endl;  
}  
}  
```
```

## 8. 存款 (deposit)

- 功能：根据输入的账户和密码，向相应账户中存入指定金额。
- 系统测试：

```

| 银行账户管理系统 |

|1、查询账户信息|
|2、开户|
|3、销户|
|4、修改账户密码|
|5、账户排序|
|6、存款|
|7、取款|
|8、转账|
|9、计算利息|
|10、保存账户信息|
|11、显示所有账户信息|
|0、退出系统|

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
6
请输入需要存款的账户：晚安
请输入需要存款的账户密码 :******

请输入存款金额：520
存款成功！
请按任意键继续. . . |
```

- 实现：

```
```cpp
void deposit(BankList& BL)
{
    string account_num; // 账户
    int password; // 密码
    Banker* p = BL->next;
    cout << "请输入需要存款的账户：" ;
    cin >> account_num;
    if (account_num_exist(BL, account_num))
    {
        cout << "请输入需要存款的账户密码：" ;
        char charPassword[20]; // 用于存储密码字符数组
        int i = 0;
        char ch;
        while ((ch = _getch()) != '\r') // \r 是回车键
        {
            if (ch == '\b' && i > 0) // 处理退格键
            {
                cout << "\b \b";
                --i;
            }
        }
    }
}
```

```
    else if (ch >= '0' && ch <= '9') // 只接受数字 (0 - 9)
    {
        charPassword[i++] = ch;
        cout << '*';
    }
}
charPassword[i] = '\0'; // 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
cout << "\n";
while (p!= NULL)
{
    if (p->account_num == account_num && p->password == password)
    {
        cout << "请输入存款金额: ";
        long long m;
        cin >> m;
        p->money += m;
        cout << "存款成功! " << endl;
        return;
    }
    p = p->next;
}
cout << "密码错误! " << endl;
}
else
{
    cout << "此账户不存在! " << endl;
}
}
```
```

```

9. 取款 (withdrawals)

- 功能：根据输入的账户和密码，从相应账户中取出指定金额，若余额不足则提示。
- 系统测试：

```

*****
|      银行账户管理系统      |
*****
|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统
*****
请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
7
请输入需要取款的账户: 202518140138
请输入需要取款的账户密码:*****
请输入您的取款金额: 20000
取款成功!
当前账户信息如下:
账户          姓名    性别    密码    金额        开户日期    开户状态
202518140138    张三    男      123456  1000080000  20241216    活期
请按任意键继续. . .

```

- 实现:

```

```cpp
void withdrawals(BankList& BL)
{
 string account_num; // 账户
 int password; // 密码
 Banker* p = BL->next;
 cout << "请输入需要取款的账户: ";
 cin >> account_num;
 if (account_num_exist(BL, account_num))
 {
 cout << "请输入需要取款的账户密码: ";
 char charPassword[20];// 用于存储密码字符数组
 int i = 0;
 char ch;
 while ((ch = _getch()) != '\r')// \r 是回车键
 {
 if (ch == '\b' && i > 0)// 处理退格键
 {
 cout << "\b \b";
 --i;
 }
 else if (ch >= '0' && ch <= '9')// 只接受数字 (0 - 9)
 {
 charPassword[i++] = ch;
 cout << '*';
 }
 }
 cout << endl;
 }
}

```

---

```
 }
}

charPassword[i] = '\0';// 字符串结束标志
password = atoi(charPassword);// 将字符数组转换为整数
 cout<<endl;
cout<<"\n";
while (p != NULL)
{
 if (p->account_num == account_num && p->password == password)
 {
 cout << "请输入您的取款金额：";
 long long m;
 cin >> m;
 if (p->money >= m)
 {
 p->money -= m;

 cout << "取款成功！" << endl;
 cout << "当前账户信息如下：" << endl;
 cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态" << endl;
 cout << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" << p->password
 << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
 return;
 }
 else
 {
 cout << "余额不足！" << endl;
 return;
 }
 }
 p = p->next;
}
cout << "密码错误！" << endl;
}
else
{
 cout << "此账户不存在！" << endl;
}
}

void Readfile2(BankList&, string);
void Save2(BankList& BL, string file);
```

## 10. 转账

- 功能：根据输入的账户和密码，从相应账户中取出指定金额并转入指定账

户，若余额不足或账户异常则提示。

- 系统测试：

```
| 银行账户管理系统 |

|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
8
请输入转账方用户类型: VIP1
请输入收款方用户类型: VIP1
请输入转账方账户: 202318140126
请输入密码:******

请输入收款方账户: 202318140130
请输入转账金额: 10000
转账成功!
当前账户信息如下:
账户 姓名 性别 密码 金额 开户日期 开户状态
202318140126 林烨洋 男 123456 51991314 20240815 活期
保存成功!
请按任意键继续. . . |
```

- 实现：

```
void transfer(BankList& BL)
{
 Banker* p1 = BL->next;

 BankList bk;
 InitList2(bk);

 string bank1, bank2, account_num;//银行名, 账户
 int password;//密码
 cout << "请输入转账方用户类型: ";
 cin >> bank1;
 bank1 += ".txt";
 cout << "请输入收款方用户类型: ";
 cin >> bank2;
 bank2 += ".txt";
 Readfile2(bk, bank2);
 Banker* p2 = bk->next;
 cout << "请输入转账方账户: ";
 cin >> account_num;
 if (account_num_exist(BL, account_num))
 {
 cout << "请输入密码: ";
 char charPassword[20];// 用于存储密码字符数组
```

---

```
int i = 0;
char ch;
while ((ch = _getch()) != '\r')// \r 是回车键
{
 if (ch == '\b' && i > 0)// 处理退格键
 {
 cout << "\b \b";
 --i;
 }
 else if (ch >= '0' && ch <= '9')// 只接受数字 (0 - 9)
 {
 charPassword[i++] = ch;
 cout << '*';
 }
}
charPassword[i] = '\0';// 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
cout << "\n";
while (p1 != NULL)
{
 if (p1->account_num == account_num && p1->password == password)
 {

 cout << "请输入收款方账户: ";
 cin >> account_num;
 while (p2)
 {
 if (p2->account_num == account_num)
 {
 long long m;
 cout << "请输入转账金额: ";
 cin >> m;
 if (bank1 == bank2)
 {
 if (p1->money >= m)
 {
 p1->money -= m;
 p2->money += m;
 cout << "转账成功! " << endl;
 cout << "当前账户信息如下: " << endl;
 cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态
" << endl;
 cout << p1->account_num << "\t" << p1->name << "\t" << p1->sex
<< "\t" << p1->password
```

```
 << "\t" << p1->money << "\t" << p1->open_time << "\t"
 << p1->states << endl;

 Save2(bk, bank2);
 return;
}
else
{
 cout << "余额不足!" << endl;
 return;
}
else
{
 cout << "收取手续费：" << m * 0.01 << " 元!" << endl;
 if (p1->money >= m + m * 0.01)
 {
 p1->money -= m;
 p2->money += m;
 cout << "转账成功!" << endl;
 cout << "当前账户信息如下：" << endl;
 cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态
" << endl;
 cout << p1->account_num << "\t" << p1->name << "\t" << p1->sex
 << "\t" << p1->password
 << "\t" << p1->money << "\t" << p1->open_time << "\t"
 << p1->states << endl;

 Save2(bk, bank2);
 return;
 }
}
}

p2 = p2->next;
}
cout << "账户不存在" << endl;
}
p1 = p1->next;
```

---

```
 }
 cout << "密码错误！" << endl;
}
else
{
 cout << "账户不存在！" << endl;
}
}
```

```
double cal_year(int a, int b)
{
 int year1 = a / 10000;
 int year2 = b / 10000;
 int month1 = a / 100 % 100;
 int month2 = b / 100 % 100;
 int day1 = a % 100;
 int day2 = b % 100;
 double year = 0;
 if (year1 == year2)
 year = 0.5;
 else if (month2 == month1 && day1 == day2)
 {
 year = year2 - year1;
 }
 else if (month2 < month1 || month2 == month1 && day1 > day2)
 {
 year = year2 - year1 - 0.5;
 }
 else if (month2 > month1 || month2 == month1 && day1 < day2)
 {
 year = year2 - year1 + 0.5;
 }
 return year;
}
```

## 11. 计算利息

- 功能：根据输入的账户和密码，结合利率，计算账户对应年份的利息。
- 系统测试：

```

| 银行账户管理系统 |

|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
9
请输入账户：晚安
请输入密码： *****

你输入的密码是：123456
请输入存款到期时间：20301231
您获得的利息为：203.23
请按任意键继续... |

```

- 实现：

```

/*
活期 每年 0.35% (0.0035)
不足一年按半年算，利率为活期年利率的一半，即 0.0035 * 0.5
定期 第一年 2.75% (0.0275)
第二年 3.35% (0.0335)
第三年 4% (0.04)
*/
void Calculation_interest(BankList& BL)
{
 string account_num;//账户
 int password;//密码
 Banker* p = BL->next;
 cout << "请输入账户：" ;
 cin >> account_num;
 if (account_num_exist(BL, account_num))
 {
 char charPassword[20];// 用于存储密码字符数组
 cout << "请输入密码：" ;
 int i = 0;
 char ch;
 while ((ch = _getch()) != '\r')// \r 是回车键
 {
 if (ch == '\b' && i > 0)// 处理退格键
 {
 cout << "\b \b";

```

---

```
--i;
}
else if (ch >= '0' && ch <= '9')// 只接受数字 (0 - 9)
{
 charPassword[i++] = ch;
 cout << '*';
}
}
charPassword[i] = '\0';// 字符串结束标志
password = atoi(charPassword); // 将字符数组转换为整数
cout << endl;
cout << endl;
cout << endl << "你输入的密码是: " << password << endl;
while (p != NULL)
{
 if (p->account_num == account_num && p->password == password)
{
 cout << "请输入存款到期时间: ";
 int end_time;
 double rate = 0;
 cin >> end_time;
 double year = cal_year(p->open_time, end_time);
 int year2 = year;//转化为整数年份
 if (p->states == "活期")
 {
 for (int i = 1; i <= year2; i++)
 {
 rate += (rate + p->money) * 0.0035;
 }
 if (year > year2 + 0.1)
 rate += (rate + p->money) * 0.0035 * 0.5;
 }
 else if (p->states == "定期")
 {
 for (int i = 1; i <= year; i++)
 {
 if (i == 1)
 {
 rate += (rate + p->money) * 0.0275;
 }
 else if (i == 2)
 {
 rate += (rate + p->money) * 0.0335;
 }
 else if (i >= 3)
```

```

 {
 rate += (rate + p->money) * 0.04;
 }

 int year2 = year;
 if (fabs(year - 0.5) <= 1e-6)
 rate += (rate + p->money) * 0.0275 * 0.5;
 else if (fabs(year - 1.5) <= 1e-6)
 rate += (rate + p->money) * 0.0335 * 0.5;
 else if (fabs(year - 2.5) <= 1e-6)
 rate += (rate + p->money) * 0.04 * 0.5;
}

cout << "您获得的利息为: " << rate << endl;
return;
}
p = p->next;
}
cout << "密码错误! " << endl;
}
else
{
 cout << "账户不存在! " << endl;
}
}

```

## 12. 显示所有账户信息

- 功能：显示所有账户的姓名、性别、密码、金额、开户日期、开户状态
- 系统测试：

```

| 银行账户管理系统 |

|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统

请选择您要进行的操作（每一步操作后记得 10、保存账户信息）
11
账户 姓名 性别 密码 金额 开户日期 开户状态
202518140138 张三 男 123456 1000080000 20241216 活期
202514180130 李四 男 123456 1000100000 20241216 活期
请按任意键继续. . .

```

- 实现：

---

```

void Display2(BankList& BL) {
 Banker* p = BL->next;
 cout << "账户\t\t姓名\t性别\t密码\t金额\t\t开户日期\t\t开户状态" << endl;
 while (p)
 {
 cout << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" << p->password
 << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
 p = p->next;
 }
}

```

### 13. 保存账户信息

- 功能：保存用户输入的账户名、姓名、性别、密码、金额、开户日期等
- 系统测试：

```

| 银行账户管理系统 |

|1、查询账户信息
|2、开户
|3、销户
|4、修改账户密码
|5、账户排序
|6、存款
|7、取款
|8、转账
|9、计算利息
|10、保存账户信息
|11、显示所有账户信息
|0、退出系统

请选择您要进行的操作 (每一步操作后记得 10、保存账户信息)
5
1:按账号排序
2:按开户人姓名排序
3:按金额排序
4:按开户日期排序
请输入选择的排序编号：
3
排序成功！
请按任意键继续...

```

- 实现：

```

void Save2(BankList& BL, string file)
{
 ofstream ofs;
 ofs.open(file.c_str());
 Banker* p = BL->next;
 while (p)
 {
 ofs << p->account_num << "\t" << p->name << "\t" << p->sex << "\t" << p->password
 << "\t" << p->money << "\t" << p->open_time << "\t" << p->states << endl;
 p = p->next;
 }
 ofs.close();
}

```

```
 cout << "保存成功!" << endl;
}
```

### (三) 排队管理模块

-- 员工登录

#### 1. VIP 放号

- 功能：根据用户需求进行 VIP 放号
- 系统测试：

The screenshot shows a terminal window with the following text:

```

| 排队管理系统 |

| 1、VIP放号
| 2、Normal放号
| 3、VIP队首完成服务
| 4、Normal队首完成服务
| 5、展示目前您的得分情况
| 0、退出系统

请选择你的操作
VIP 队列当前为空
Normal 队列当前为空
1
请输入需要放号的个数：3
VIP 放号成功，当前 VIP 号为：3
请按任意键继续... |
```

- 实现：

```
void releaseVIP() {
int x;
cout << "请输入需要放号的个数：" << cin >> x;
for (int i = 0; i < x; i++)
 VIPQue.push(VIPNum++);
cout << "VIP 放号成功，当前 VIP 号为：" << VIPQue.back() << endl;
}
```

#### 2. Normal 放号

- 功能：根据用户需求进行 Normal 放号
- 系统测试：

```

| 排队管理系统 |

|1、VIP放号
|2、Normal放号
|3、VIP队首完成服务
|4、Normal队首完成服务
|5、展示目前您的得分情况
|0、退出系统

请选择你的操作
VIP窗口 1<-2<-3
Normal 队列当前为空
2
请输入需要放号的个数：5
Normal 放号成功，当前 Normal 号为：5
请按任意键继续... |

```

- 实现：

```

void releaseNormal() {
 int x;
 cout << "请输入需要放号的个数：" ; cin >> x;
 for (int i = 0; i < x; i++)
 normalQue.push(normalNum++);
 cout << "Normal 放号成功，当前 Normal 号为：" << normalQue.back() << endl;
}

```

### 3. 处理队首服务

- 功能：用户根据自身体验对服务进行打分
- 系统测试：

```

| 排队管理系统 |

|1、VIP放号
|2、Normal放号
|3、VIP队首完成服务
|4、Normal队首完成服务
|5、展示目前您的得分情况
|0、退出系统

请选择你的操作
VIP窗口 1<-2<-3
Normal窗口 1<-2<-3<-4<-5

3
VIP 服务结束
请客户输入打分（1-5 分）：5
打分成功！
当前服务号为：1
下一个 VIP 号为：2
请按任意键继续... |

```

- 实现：

---

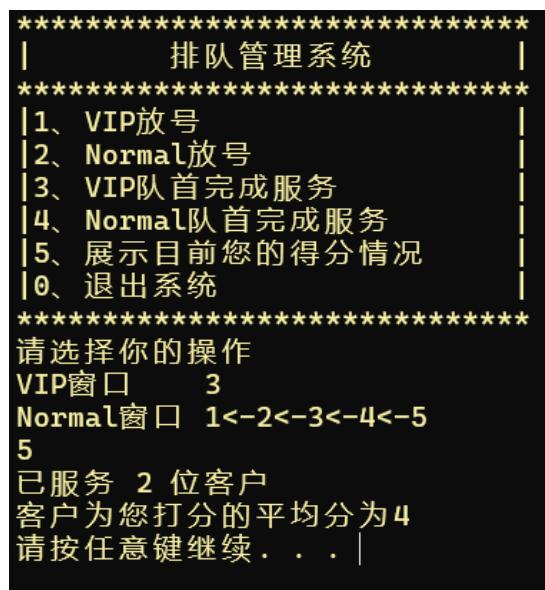
```

void Grade(Staff*& goal) {
 int score;
 while (1) {
 cout << "请客户输入打分 (1~5 分) : ";
 cin >> score;
 if (score < 1 || score>5)
 {
 cout << "请在 1~5 之间打分 " << endl;
 system("pause");
 system("cls");
 Menu3(); //菜单栏
 DisplayVIP();
 DisplayNormal();
 }
 else break;
 }
 goal->grade = (goal->grade * goal->n + score) / (goal->n + 1);
 goal->n++;
 cout << "打分成功! " << endl;
}

```

#### 4. 查看得分

- 功能：展现职员的平均得分
- 系统测试：



- 实现：

```

void DisplayScore(Staff*& goal) {
 cout << "已服务 " << cnt << " 位客户" << endl;
 cout << "客户为您打分的平均分为" << goal->grade << endl;
}

```

## (四) 排队管理模块

### 1. 介绍学校网点

- 功能：将所有网点的信息依次打印
- 系统测试：

编号	景点名称	简介
0	南门	出门有美食街
1	西门	出门有BRT公交站
2	1号教学楼	导员办公室在1461
3	2号教学楼	最常上课的楼 最后一排有插座
4	3号实验楼	上机课就在这 电梯特别挤
5	4号教学楼	办公楼旁边
6	图书馆	离南门最近
7	逸夫楼	离图书馆近
8	气膜馆	可预约打羽毛球
9	篮球场	可随时打篮球
10	排球场	打排球
11	网球场	打网球
12	纳博士地下超市	超市 + 小吃
13	711	便利店
14	西苑餐厅	吃饭 上完课吃饭最近
15	东苑餐厅	吃饭 6, 7号宿舍楼出来吃饭最近 旁边有民族餐厅
16	北苑餐厅	吃饭 近办公楼
17	校医院	看病
18	操场	体育课 跑步 晚上没灯
19	菜鸟驿站	取快递
20	海岱美术馆	美术馆 没去过
21	教工宿舍	教职工租住地
22	1, 2号宿舍楼	离菜鸟驿站近
23	3, 5号宿舍楼	离西苑 菜鸟驿站 2号楼都近
24	6, 7, 8号宿舍楼	东苑餐厅附近
25	9, 10, 11号宿舍楼	离东苑也近 在6, 7号宿舍楼前面
26	学景大酒店	酒店 南门门口
27	办公楼	办公楼 离西门近 这个电梯不挤

请按任意键继续... |

- 实现：

```
void browsocompus(MGraph c)
{
 int i;
 printf(" \n\n 编号 景点名称 简介\n");
 printf("_____");
 for (i = 0; i < c.vexnum; i++)
 printf("%-10d%-25s%-80s\n", c.vexs[i].position, c.vexs[i].name, c.vexs[i].introduction);
 printf("_____");
 \n\n");
}

} //browsocompus
```

### 2. 规划旅游路线

- 功能：根据用户的出发点，规划路线
- 系统测试：

```

请输入一个起始景点的编号: 18

操场--->东苑餐厅--->6, 7, 8宿舍楼--->9, 10, 11宿舍楼--->学景大酒店---->南门
总路线长为90米

操场--->4号教学楼--->篮球场--->711--->校医院--->3, 5号宿舍楼--->办公楼---->西门
总路线长为350米

操场--->4号教学楼--->篮球场--->711--->校医院--->3, 5号宿舍楼--->办公楼---->1号教学楼
总路线长为330米

操场--->南门--->3号实验楼--->图书馆--->逸夫楼--->西苑餐厅--->东苑餐厅--->菜鸟驿站--->1, 2号宿舍楼--->6, 7, 8宿舍楼--->9
10, 11宿舍楼--->学景大酒店--->2号教学楼
总路线长为340米

```

- 实现:

```

void shortestpath_floyd(MGraph c)
{
 //用 floyd 算法求各对顶点 v 和 w 间的最短路径及其带权长度的 d[v][w]。
 //若 p[v][w][u]==1；则 u 是 v 到 w 的当前求得的最短路径上的顶点

 int i, j, k, d[35][35], p[35][35][35];
 int v, u, w;
 for (v = 0; v < c.vexnum; v++) //初始化各对顶点 v, w 之间的起始距离 d[v][w] 及 路
 径 p[v][w][] 数组
 {
 for (w = 0; w < c.vexnum; w++)
 {
 d[v][w] = c.arcs[v][w].adj; //d[v][w] 中存放 v 至 w 间初始权值
 for (u = 0; u < c.vexnum; u++) //初始化最短路径 p[v][w][] 数组, 第 3 个分量全部清
0
 p[v][w][u] = 0;
 if (d[v][w] < Infinity) //如果 v 至 w 间有边相连
 {
 p[v][w][v] = 1; // v 是 v 至 w 最短路径上的顶点
 p[v][w][w] = 1; // w 是 v 至 w 最短路径上的顶点
 } //if
 } //for
 } //endfor

 for (u = 0; u < c.vexnum; u++) // 求 v 至 w 的最短路径及距离。对任意顶点 u，试探其是否为 v 至 w
 最短路径上的顶点
 {
 for (v = 0; v < c.vexnum; v++)
 for (w = 0; w < c.vexnum; w++)
 if (d[v][u] + d[u][w] < d[v][w]) //从 v 经 u 到 w 的一条路径更短
 {
 d[v][w] = d[v][u] + d[u][w]; //修改 v 至 w 的最短路径长度
 for (i = 0; i < c.vexnum; i++) //修改 v 至 w 的最短路径数组。 若 i 是 v 至 u
 的最短路径上的顶点,
 p[v][w][i] = p[v][u][i] || p[u][w][i]; //或 i 是 u 至 w 的最短路径上的顶点,
 则 i 是 v 至 w 的最短路径上的顶点
 }
 }
}

```

---

```

} //endfor

printf("\n 请输入出发点和目的地编号: ");
scanf("%d%d", &k, &j);
printf("\n\n");
while (k < 0 || k > c.vexnum || j < 0 || j > c.vexnum)
{
 printf("\n 你所输入的景点编号不存在! ");
 printf("\n 请重新输入出发点和目的地编号: \n\n");
 scanf("%d%d", &k, &j);
 printf("\n\n");
}
printf("%s", c.vexs[k].name); //输出出发景点名称
for (u = 0; u < c.vexnum; u++)
 if (p[k][j][u] && k!= u && j!= u) //输出最短路径上中间景点名称
 printf("—>%s", c.vexs[u].name);
printf("—>%s", c.vexs[j].name); //输出目的地景点名称

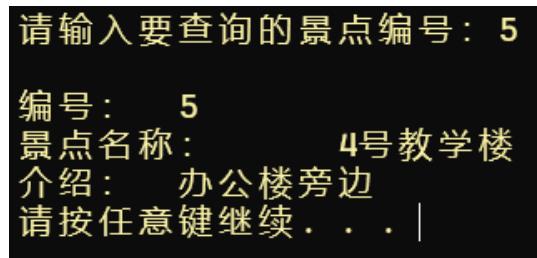
printf("\n\n\n 总长为%d 米\n\n", d[k][j]);

} //shortestpath_floyd

```

### 3. 查询网点详情

- 功能：根据用户需求打印对应景点信息
- 系统测试：



```

请输入要查询的景点编号: 5
编号: 5
景点名称: 4号教学楼
介绍: 办公楼旁边
请按任意键继续. . .

```

- 实现：

```

void seeabout(MGraph c)
{
 int k;
 printf("\n 请输入要查询的景点编号: ");
 scanf("%d", &k);
 while (k < 0 || k > c.vexnum)
 {
 printf("\n 你所输入的景点编号不存在! ");
 printf("\n 请重新输入: ");
 scanf("%d", &k);
 }
}

```

```
}

printf("\n\n 编号: %4d\n", c->vexs[k].position);
printf("\n\n 景点名称: %10s\n", c->vexs[k].name);
printf("\n\n 介绍: %80s\n\n", c->vexs[k].introduction);

} //seeabout
```

#### 4. 更改图信息

- 功能：更改对应景点的各种信息
- 实现：

```
int newgraph(MGraph *c)
{
 int changenum; //计数。用于记录要修改的对象的个数
 int i, m, n, t, distance, v0, v1;
 printf("\n 下面请输入你要修改的景点的个数: \n");
 scanf("%d", &changenum);
 while (changenum < 0 || changenum > c->vexnum)
 {
 printf("\n 输入错误！请重新输入");
 scanf("%d", &changenum);
 }

 for (i = 0; i < changenum; i++)
 {
 printf("\n 请输入景点的编号: ");
 scanf("%d", &m);
 t = locatevex(*c, m);
 printf("\n 请输入景点的名称: ");
 scanf("%s", &c->vexs[t].name);
 printf("\n 请输入景点的简介: ");
 scanf("%s", &c->vexs[t].introduction);
 }

 printf("\n 下面请输入你要更新的边数");
 scanf("%d", &changenum);
 while (changenum < 0 || changenum > c->arcnum)
 {
 printf("\n 输入错误！请重新输入");
 scanf("%d", &changenum);
 }

 printf("\n 下面请输入更新边的信息: \n");
 for (i = 1; i <= changenum; i++)
```

```

{
 printf("\n 修改的第%d 条边的起点 终点 长 度为: ", i);
 scanf("%d", &v0, &v1, &distance);
 m = locatevex(*c, v0);
 n = locatevex(*c, v1);
 if (m >= 0 && n >= 0)
 {
 c->arcs[m][n].adj = distance;
 c->arcs[n][m].adj = c->arcs[m][n].adj;
 }
}
return 1;
} //newgraph

```

## 5. 查询景点间路径

- 功能：查询两景点间的所有路径
- 系统测试：

```

请输入你要查询的两个景点编号：
3 6

2号教学楼--->1号教学楼--->3号实验楼--->西苑餐厅--->菜鸟驿站--->逸夫楼--->图书馆
2号教学楼--->1号教学楼--->3号实验楼--->西苑餐厅--->1, 2号宿舍楼--->菜鸟驿站--->逸夫楼--->图书馆
2号教学楼--->3号实验楼--->西苑餐厅--->菜鸟驿站--->逸夫楼--->图书馆
2号教学楼--->3号实验楼--->西苑餐厅--->1, 2号宿舍楼--->菜鸟驿站--->逸夫楼--->图书馆
请按任意键继续... |

```

- 实现：

```

void path(MGraph c, int m, int n, int k)
{
 int s, x = 0;
 int t = k + 1; //t 记载路径上一个中间顶点在 d[] 数组中的下
 //标
 if (d[k] == n && k < 12) //d[k] 存储路径顶点。若 d[k] 是终点 n 且景
 个数<=8，则输出该路径
 {
 //递归出口，找到一条路径
 for (s = 0; s < k; s++)
 printf("%s--->", c.vexs[d[s]].name); //输出该路径。s=0 时为起点 m
 printf("%s", c.vexs[d[s]].name); //输出最后一个景点名(即顶点 n 的名字, 此时 s ==
k)
 printf("\n\n");
 }
 else
 {

```

---

```

s = 0;
while (s < c.vexnum) //从第 m 个顶点, 试探至所有顶点是否有路径
{
 if ((c.arcs[d[k]][s].adj < Infinity) && (visited[s] == 0)) //初态: 顶点 m 到顶点 s 有
 边, 且未被访问
 {
 visited[s] = 1;
 d[k + 1] = s; //存储顶点编号 s 至 d[k + 1]中
 path(c, m, n, t); //求从下标为 t = k + 1 的第 d[t]个顶点开始的路径(递归
 调用), 同时打印出一条 m 至 n 的路径
 visited[s] = 0; //将找到的路径上顶点的访问标志重新设置为 0, 以用于试
 探新的路径
 }
 s++; //试探从下一个顶点 s 开始是否有到终点的路径
} //endwhile
} //endelse

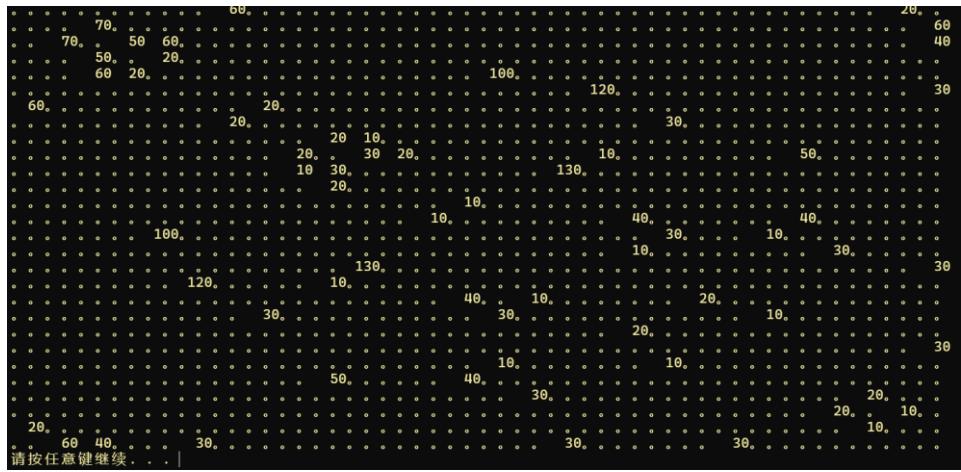
} //endpath

int allpath(MGraph c)
{
 int k, i, j, m, n;
 printf("\n\n请输入你要查询的两个景点编号:\n\n");
 scanf("%d%d", &i, &j);
 printf("\n\n");
 m = locatevex(c, i); //调用(2), 确定该顶点是否存在。若存在, 返回该顶
 点编号
 n = locatevex(c, j);
 d[0] = m; //存储路径起点 m (int d[]数组是全局变量)
 for (k = 0; k < c.vexnum; k++) //全部顶点访问标志初值设为 0
 visited[k] = 0;
 visited[m] = 1; //第 m 个顶点访问标志设置为 1
 path(c, m, n, 0); //调用(3)。k = 0, 对应起点 d[0] == m。k 为 d[]数
 组下标
 return 1;
}

```

## 6. 打印邻接矩阵

- 功能: 打印邻接矩阵
- 系统测试:



- 实现：

```
void printmatrix(MGraph c)
{
 int i, j, k = 0; //k 用于计数，控制换行

 for (i = 0; i < c.vexnum; i++)
 for (j = 0; j < c.vexnum; j++)
 {
 if (c.arcs[i][j].adj == Infinity)
 printf("----");
 else
 printf("%4d", c.arcs[i][j].adj);
 k++;
 if (k % c.vexnum == 0)
 printf("\n");
 }
} //printpath
```

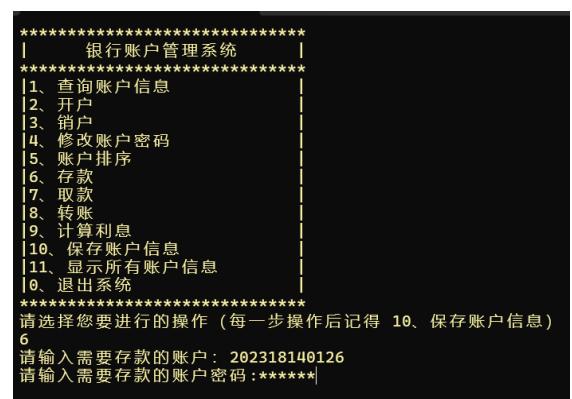
(五) 退出

- 功能：退出系统

## 五、特色创新

### 1、密码安全性保护

如图所示，当用户输入密码时，会将其输入的密码以‘\*’的形式展示用来保护用户的隐私的安全性。所应用的原理是：cout << "\b \b";：在控制台输出两个退格符，一个用于移动光标，另一个用



于覆盖前一个字符。这样可以使得用户输入的密码不显示为原本内容，而是以‘\*’的形式代替出现。

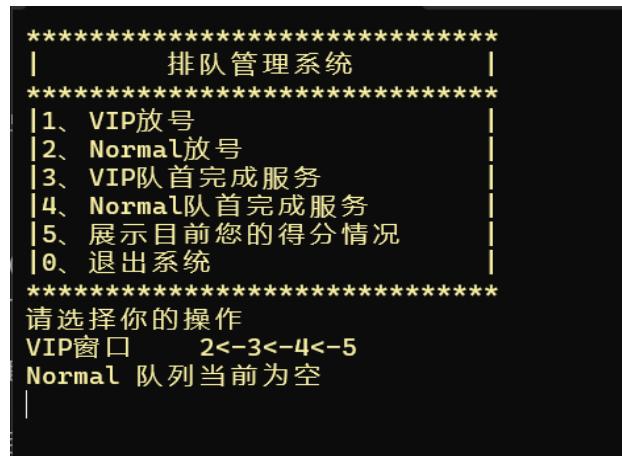
## 2. 排队管理系统需要登录

如图所示，在进入排队管理系统之前，需要提前登录验证职员身份。当在此页面输入账号密码时，系统会依次进入职员表(GroupA, GroupB, GroupC)中寻找是否存在该职员，工号和密码是否正确。工号密码正确的话会进入排队管理系统，以保证安全。



## 3. 排队管理系统状态实时显示

如图所示，在排队管理系统中，登入后，可以选择 VIP 和 normal 两个队列放号，两个队列相互独立，可以看到每个队列的所放出号的服务完成情况，剩余多少未完成。



## 六、用户手册

见附件 1。

## 七、总结体会

我们小组在数据结构程序设计课程中完成了银行管理系统的作业，这让我们深刻体会到了理论与实践相结合的重要性。在这个项目中，我们运用了多种数据结构，如链表、队列和图，以及 C++ 语言的特性，来设计和实现一个功能完备的银行管理系统。

首先，我们通过定义 `Banker` 和 `Staff` 结构体，将理论知识应用到了实际的数据存储和管理中。这些结构体不仅包含了基本的数据字段，还通过指针实现了动态内存

---

管理，让我们深刻理解了链表在实际应用中的优势，尤其是在数据项频繁插入和删除的场景下。

在实现账户管理功能时，我们使用了链表来存储客户和员工的信息，这让我体会到了链表在顺序存储结构中的灵活性。通过链表，我们可以轻松地添加和删除账户，而不必担心数组等静态数据结构的局限性。

在我们的银行管理系统作业中，我们使用了 C++ 的输入输出流（`ifstream` 和 `ofstream`）来处理数据持久化。具体方法是，在程序启动时，通过 `ifstream` 读取预定定义格式的数据文件，将员工和客户的信息解析并构建成链表；在数据更新后，使用 `ofstream` 将链表中的数据写回文件，以此保持信息的连续性。这种方法实现了数据的读取和存储，确保了系统在多次运行间数据的完整性和一致性。

在处理排队系统时，我们使用了队列这一数据结构，它完美地模拟了银行中的叫号流程。通过`queue`库的使用，我们学会了如何在程序中实现先进先出（FIFO）的逻辑，这对于理解现实世界中的排队机制非常有帮助。

此外，我们还实现了图的数据结构来处理网点查询系统，这让我对图的遍历和最短路径算法有了更深的认识。通过迪杰斯特拉算法和弗洛伊德算法的应用，我们不仅复习了这些算法的原理，还学会了如何在实际问题中应用它们。

在程序设计方面，我们学会了如何将复杂的系统分解成多个子系统，并通过函数调用来实现模块间的通信。这种模块化的设计方法让我们的代码更加清晰和易于管理，也让我们意识到了在大型软件项目中，合理划分功能模块的重要性。

通过这次作业，我们不仅加深了对数据结构的理解，还提高了我们的程序设计能力。我们学会了如何选择合适的数据结构来解决特定的问题，如何设计函数接口，以及如何组织代码以提高可读性和可维护性。这些经验对我们未来的学习和职业发展都具有重要意义。