記帳小專題

組員:劉奕宏 何秉翰 溫濟澤

目錄

請至課輔系統影片檔觀看

程式解析: (記帳的基本輸入、輸出)

```
int main() {
    int choice;
    accountBook Records;

    do {
        cout << "\n============ 記帳系統選單 ============";
        cout << "\n| 1. 新增記帳                        |";
        cout << "\n| 2. 顯示所有記帳紀錄                |";
        cout << "\n| 3. 修改資料                        |";
        cout << "\n| 0. 結束程式                        |";
        cout << "\n======================================";
        cout << "\n請輸入選項: ";
        cin >> choice;
```

宣告一個整數變數 choice 來儲存使用者輸入的選項。
建立一個 accountBook 類別的物件 Records，用來處理記帳功能。這個類別的定義在 accountBook.h 檔案中。
使用 cout 印出選單，讓使用者知道有哪些功能可以選擇。
使用 cin 接收使用者輸入的選項。

主要功能:
提供一個使用者介面（透過命令列選單）來操作記帳功能
支援三個主要操作：
1.新增記帳紀錄（addRecord）
2.顯示所有記帳紀錄（displayRecords）
3.修改記帳資料（editRecords）
當使用者輸入 0，程式將會結束執行

```
    switch (choice) {
    case 1:
        Records.addRecord();
        break;
    case 2:
        Records.displayRecords();
        break;
    case 3:
        Records.editRecords();
        break;
    case 0:
        cout << "再見！" << endl;
        break;
    default:
        cout << "無效的選項，請重新輸入。" << endl;
    }

} while (choice != 0);
```

switch-case 控制流程根據使用者輸入的選項執行對應功能。
每個 case 都是呼叫 accountBook 類別中的方法。

程式解析: (輸入資料)

```
void createRecord::input() {
    ofstream outFile("records.txt", ios::app);
    if (!outFile) {
        cout << "無法打開檔案寫入記錄。" << endl;
        return;
    }
```

看是否有資料能夠使用

```
cout << "輸入金額: ";
cin >> money;

cout << "輸入類別(例如: 交通 早餐..): ";
cin >> category;

cout << "輸入備註(例如:食物名稱): ";
cin.ignore();
getline(cin, note);

int month = 0, day = 0;
cout << "輸入日期(月 日): ";
cin >> month >> day;

if (cin.fail() || month < 1 || month > 12 || day < 1 || day > 31) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "日期輸入錯誤，記錄未新增。" << endl;
    return;
}
```

依序輸入：金額、類別、備註、日期（月/日）。
使用 cin.ignore() + getline() 讓備註能輸入空白。

驗證日期輸入是否合理，避免錯誤資料寫入

```
void createRecord::display() {
    ifstream inFile("records.txt");
    if (!inFile) {
        cout << "尚無任何記錄。" << endl;
        return;
    }

    string category, note;
    double money;
    int month, day;

    while (inFile >> category >> note >> money >> month >> day) {
        cout << "項目: " << category << endl;
        cout << "備註: " << note << endl;
        cout << "金額: $" << fixed << setprecision(2) << money << endl;
        cout << "日期: " << month << " 月 " << day << " 日" << endl;
        cout << "--------------------------" << endl;
    }
```

使用迴圈逐筆讀取並顯示格式化後的內容

程式解析: (程式關掉後重啟還能保留檔案)

(records.txt)

records.txt 是一個 純文字檔案，用來儲存使用者輸入的每一筆記帳紀錄，這樣即使程式關掉、電腦關機，資料也不會消失。

只要"records.txt"這個檔案還存在，程式每次啟動時都能讀取先前記錄

```
void createRecord::display() {
    ifstream inFile("records.txt");
    if (!inFile) {
        cout << "尚無任何記錄。" << endl;
        return;
    }
```

```
void createRecord::input() {
    ofstream outFile("records.txt", ios::app);
    if (!outFile) {
        cout << "無法打開檔案寫入記錄。" << endl;
        return;
    }
```

每一筆資料在程式執行 addRecord() 時，會寫入這個檔案，像這樣：

```
apple apple 99.00 9 5
pizza banana 100.00 6 9
apple apple 99.00 9 3
food apple 90.00 8 11
```

對應的意義是：

欄位	說明
類別（category）	早餐、交通、飲料等
備註（note）	食物名稱、公車、手搖等說明
金額（money）	花費金額
月（month）	幾月花的
日（day）	幾號花的

程式解析:(在紀錄時打錯可以修改並重新寫入整份檔案)

類別：accountBook

accountBook 是一個管理記帳資料的類別，裡面主要有三個功能：

新增記錄（addRecord()）

顯示記錄（displayRecords()）

編輯記錄（editRecords()）

(新增紀錄addRecord)

```
void accountBook::addRecord() {
    createRecord newRecord;
    newRecord.input();          // 由使用者輸入一筆記錄並寫入檔案
    records.push_back(newRecord);
}
```

建立一個 createRecord 物件 newRecord。
呼叫 input() 讓使用者輸入記帳資訊（寫進 records.txt）。
用 vector 把這筆資料加到記憶體中

(顯示記錄（displayRecords()）

```
void accountBook::displayRecords() {
    cout << "\n=== 你輸入的所有記帳內容 ===" << endl;
    createRecord record;
    record.display();  // 讀取 records.txt 並顯示所有紀錄
}
```

呼叫 createRecord 裡的 display() 函式。
從 records.txt 讀取所有記錄並逐一顯示。

程式解析:(在紀錄時打錯可以修改並重新寫入整份檔案)

編輯記錄（editRecords()）

```
void accountBook::editRecords() {
    ifstream inFile("records.txt");
    if (!inFile) {
        cout << "尚無任何記錄。" << endl;
        return;
    }

    struct Record {
        string category = "Unknown";
        string note = "None";
        double money = 0.0;
        int month = 1;
        int day = 1;
    };
```

先讀取 records.txt 的所有記錄再把檔案內容存進一個 vector<Record> 中，準備修改。

顯示所有記錄，並讓使用者選要改哪一筆

```
cout << "\n=== 所有記帳紀錄 ===" << endl;
for (size_t i = 0; i < records.size(); ++i) {
    cout << i + 1 << ". "
        << "項目: " << records[i].category << ", "
        << "備註: " << records[i].note << ", "
        << "金額: $" << fixed << setprecision(2) << records[i].money << ", "
        << "日期: " << records[i].month << "/" << records[i].day << endl;
}
```

程式解析:(在紀錄時打錯可以修改並重新寫入整份檔案)

```
switch (choice) {
case 1:
    cout << "請輸入新的項目(category): ";
    cin >> rec.category;
    break;
case 2:
    cout << "請輸入新的備註(note): ";
    cin >> rec.note;
    break;
case 3:
    cout << "請輸入新的金額: ";
    cin >> rec.money;
    break;
case 4:
    cout << "請輸入新的月份: ";
    cin >> rec.month;
    cout << "請輸入新的日期: ";
    cin >> rec.day;
    break;
default:
    cout << "無效選項。" << endl;
    return;
}
```

選擇要更改的項目

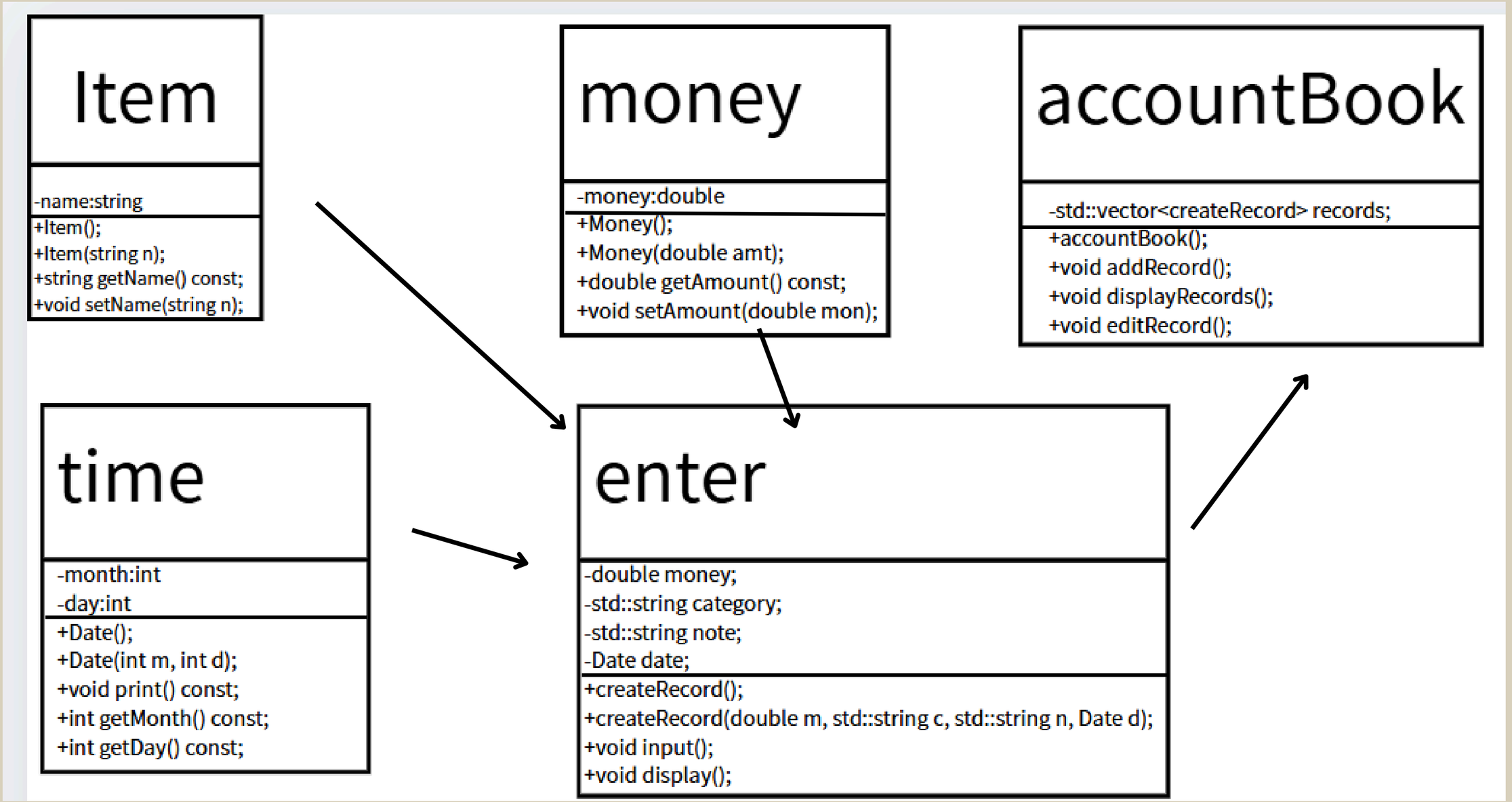
switch-case 控制流程根據使用者輸入的選項執行對應功能。
每個 case 都是呼叫 accountBook 類別中的方法

寫回檔案

```
ofstream outFile("records.txt");
for (const auto& r : records) {
    outFile << r.category << " "
        << r.note << " "
        << fixed << setprecision(2) << r.money << " "
        << r.month << " "
        << r.day << endl;
}
outFile.close();

cout << "紀錄修改完成！" << endl;
```

UML 流程圖


Item
-name:string
+Item();
+Item(string n);
+string getName() const;
+void setName(string n);
money
-money:double
+Money();
+Money(double amt);
+double getAmount() const;
+void setAmount(double mon);
accountBook
-std::vector<createRecord> records;
+accountBook();
+void addRecord();
+void displayRecords();
+void editRecord();
time
-month:int
-day:int
+Date();
+Date(int m, int d);
+void print() const;
+int getMonth() const;
+int getDay() const;
enter
-double money;
-std::string category;
-std::string note;
-Date date;
+createRecord();
+createRecord(double m, std::string c, std::string n, Date d);
+void input();
+void display();

分工細節

溫濟澤

1.完成main函式
2.完成accountBook.cpp
3.完成enter.cpp
4.參與報告製作
5.參與主題討論
6.程式碼除錯

何秉翰

1.UML設計與完成
2.參與主題討論
3.程式基礎建置
4.參與報告製作
5.程式碼除錯
6.新增檔案紀錄功能

劉奕宏

1.完成報告大部分內容
2.參與主題討論
3.討論程式碼設計
4.程式碼除錯
5.程式基礎建置