r

C++프로그래밍및실습

간단 예산관리

진척 보고서 #2

제출일자: 2024.12.01.
제출자명: 이승진
제출자학번: 214935

1. 프로젝트 목표

1) 배경 및 필요성

자취 또는 기숙사 생활을 하는 많은 대학생들이 매달 용돈이나 생활비를 효율적으로 관리하지 못하고 있음. 이는 불필요한 소비를 지속적으로 발생시키고, 본인이 어디에 얼마나 지출하였는지조차 알지 못하게 된다. 이에 간단하게 예산을 설정하고 지출을 관리하려는 사용자에게 최소한의 입력으로 예산 상황을 쉽게 파악할 수 있는 프로그램이 필요하다고 판단함.

2) 프로젝트 목표

간단한 개인 예산 관리 프로그램을 개발하여, 사용자가 수입과 지출을 기록하고 월간 예산을 설정할 수 있도록 돕는 것을 목표로 함. 이 프로그램은 사용자가 현재 남은 예산을 즉각적으로 확인할 수 있도록 하여, 보다 체계적인 지출 관리에 기여할 것으로 예상함.

3) 차별점

기존의 가계부 앱이나 금융 관리 앱은 다양한 기능을 포함하고 있어 초보 사용자에게 부담스러울 수 있다. 이에 소비 현황에 집중도를 높여 직관적인 소비 패턴을 파악 할 수 있도록 한다. 또한 고정 지출 항목을 직접 설정할 수 있도록 하여 반복 입력의 번거로움을 줄인다.

2. 기능 계획

1) 수입 및 지출 내역 기록

- 사용자가 일일 수입과 지출을 손쉽게 입력할 수 있도록 함.

(1) 일일 지출 내역 입력

- 특정 날짜의 지출 금액과 항목을 기록한다.

(2) 일일 수입 내역 입력

- 특정 날짜의 수입 금액과 항목을 기록한다.

(3) 월간 입력 기능

- 입력하고자 하는 달을 유저에게 직접 입력받고, 1~12월의 기록을 저장할 수 있도록 함

2) 월간 예산 설정 및 잔액 확인

- 사용자가 월간 예산을 설정하고 현재 예산의 남은 잔액을 확인할 수 있도록 함.

(1) 월간 예산 설정

- 한 달의 지출 한도를 설정하여 현재 남은 예산을 계산한다.

(2) 잔액 확인

- 현재까지의 총 지출을 바탕으로 남은 예산을 실시간으로 표시한다.

3) 고정 지출 항목 자동 기록

- 매월 반복되는 고정 지출을 지출 항목에 자동으로 입력 받도록 함.

(1) 고정 지출 항목 등록

- 고정 지출되는 금액과 내용을 미리 등록하여 매달 자동으로 기록한다.

(2) 고정 지출 항목 삭제

- 추후 고정 지출이 변경 될 수 있기 때문에 지출항목을 삭제할 수 있도록 한다.

4) 설정된 월간 예산 사용 비율 표기

- 사용자가 사전에 설정한 월간 예산 금액 중 현재까지 사용한 금액을 비율로 나타내어 사용자에게 소비 금액을 조절할 수 있도록 함.

(1) 월간 예산에 대한 소비 현황 비율 표기

- 사용한 액수를 비율(%)로 나타내고, 월간 예산 대비 사용할 수 있는 금액을 표기한다.

(2) 일정 구간 별 경고메시지 출력

- 예산 25%, 50%, 75% 사용 시, 경고메시지 출력

3. 진척사항

1) 기능 구현

(1) 수입 및 지출 내역 기록

- 입출력

입력 : 월 선택, 수입 추가, 지출 추가, 거래 내역 보기

출력 : 수입기록, 지출기록, 거래내역

- 설명

Transaction 구조체를 이용해 거래 내역을 저장함(수입or지출, 설명, 금액, 날짜)

BudgetManager 클래스를 통해 수입, 지출, 거래 내역을 관리함

YearBudgetManager 클래스는 사용자가 1~12월별 거래를 관리할 수 있도록 돕고, 각 월에 대해 manageBudgetForMonth 함수를 이용해 거래 내역을 추가, 확인 할 수 있도록 한다.

- 적용된 배운 내용

While 반복문을 사용해 1~12월 중 선택 할 수 있는 기능 구현.

If-else 조건문을 사용해 사용자의 입력에 대한 출력을 결정.

프로그램의 구조를 명확하게 하기 위해 클래스 사용

addIncome, addExpense, showTransactions 등의 함수를 사용해 가독성과 코드 효율을 높임

항목의 개수가 정해져 있지 않은 거래 내역은 벡터를 사용해 저장하도록 함

- 코드 스크린샷

```
// 거래 내역 구조체: 수입 또는 지출 정보를 저장
struct Transaction {
    string type;        // "수입" 또는 "지출"
    string description; // 항목 설명
    int amount;         // 금액
    string date;        // 날짜 (예: "2024-11-16")
};
```

```
// 특정 월의 거래를 관리하는 클래스
class BudgetManager {
public:
    // 수입을 추가하는 함수
    void addIncome(const string& date, int amount, const string& description) {
        transactions.push_back({"수입", description, amount, date}); // 벡터에 수입 기록 추가
        cout << "수입 기록: " << date << " + " << description << " - " << amount << "원" << endl;
    }

    // 지출을 추가하는 함수
    void addExpense(const string& date, int amount, const string& description) {
        transactions.push_back({"지출", description, amount, date}); // 벡터에 지출 기록 추가
        cout << "지출 기록: " << date << " - " << description << " - " << amount << "원" << endl;
    }

    // 저장된 거래 내역을 출력하는 함수
    void showTransactions() const {
        if (transactions.empty()) { // 거래 내역이 없으면 알림
            cout << "거래 내역이 없습니다." << endl;
            return;
        }

        // 거래 내역 출력
        cout << "\n[거래 내역]" << endl;
        for (const auto& transaction : transactions) {
            cout << transaction.date << " - " << transaction.type << ": "
                 << transaction.description << " - " << transaction.amount << "원" << endl;
        }
    }
```

```
private:
    vector<Transaction> transactions; // 해당 월의 거래 내역을 저장하는 벡터
};
```

```cpp
// 연간 거래를 관리하는 클래스
class YearBudgetManager {
public:
    // 월별 거래 관리를 시작하는 함수
    void manageMonthlyBudget() {
        int month;
        while (true) {
            cout << "--- 월간 기록 프로그램 ---" << endl;
            cout << "1~12월 중 기록할 달을 선택하세요 (0 입력 시 종료): ";
            cin >> month;

            if (month == 0) { // 0 입력 시 종료
                cout << "프로그램을 종료합니다." << endl;
                break;
            } else if (month < 1 || month > 12) { // 유효하지 않은 달 입력
                cout << "잘못된 입력입니다. 1부터 12까지의 숫자를 입력하세요." << endl;
                continue;
            }

            // 선택한 월의 거래 관리 시작
            manageBudgetForMonth(month);
        }
    }
```

```cpp
private:
    BudgetManager budgetmanagers[12]; // 12개월 데이터를 관리하는 배열

    // 특정 월의 거래를 관리하는 함수
    void manageBudgetForMonth(int month) {
        int choice;
        while (true) {
            cout << "--- " << month << "월 기록 관리 ---" << endl;
            cout << "1. 수입 추가" << endl;
            cout << "2. 지출 추가" << endl;
            cout << "3. 거래 내역 보기" << endl;
            cout << "4. 돌아가기" << endl;
            cout << "선택: ";
            cin >> choice;

            if (choice == 1) { // 수입 추가
                int amount;
                string description, date;
                cout << "수입 금액을 입력하세요: ";
                cin >> amount;
                cin.ignore(); // 개행 문자 제거
                cout << "수입 항목을 입력하세요: ";
                getline(cin, description);
                cout << "수입 날짜를 입력하세요 (예: 2024-11-16): ";
                getline(cin, date);
                budgetmanagers[month - 1].addIncome(date, amount, description); // 선택한 월의 수입 추가

            } else if (choice == 2) { // 지출 추가
                int amount;
                string description, date;
                cout << "지출 금액을 입력하세요: ";
                cin >> amount;
                cin.ignore(); // 개행 문자 제거
                cout << "지출 항목을 입력하세요: ";
                getline(cin, description);
                cout << "지출 날짜를 입력하세요 (예: 2024-11-16): ";
                getline(cin, date);
                budgetmanagers[month - 1].addExpense(date, amount, description); // 선택한 월의 지출 추가

            } else if (choice == 3) { // 거래 내역 보기
                budgetmanagers[month - 1].showTransactions(); // 선택한 월의 거래 내역 출력

            } else if (choice == 4) { // 이전 메뉴로 돌아가기
                cout << "이전 메뉴로 돌아갑니다." << endl;
                break;

            } else { // 잘못된 입력
                cout << "잘못된 입력입니다. 다시 선택해 주세요." << endl;
            }
        }
    }
};

int main() {
    YearBudgetManager yearlymanager; // 연간 거래 관리 객체 생성
    yearlymanager.manageMonthlyBudget(); // 월별 거래 관리 시작
    return 0;
}
```

2) 월간 예산 설정 및 잔액 확인

-입출력

입력 : 예산 금액

출력 : 설정된 예산 금액, 지출금액이 적용된 후의 남은 예산 금액

-설명

월간 예산을 사용자로부터 입력 받는 함수를 생성하고, 남은 잔액을 계산하는 함수를 통해 지출 내역을 등록 할 때마다 남은 사용 가능 금액을 표시하도록 한다.

-적용된 배운 내용

If-else 조건문을 사용해 사용자의 입력에 대한 출력을 결정.

함수를 사용해 가독성과 코드 효율을 높임

상수 멤버 함수(const)를 사용해 객체의 상태를 변경하지 않도록 설정

3) 고정 지출 항목 자동 기록

-입출력

입력 : 고정지출 메뉴 선택, 지출 금액, 지출 내용, 제거할 목록의 인덱스 번호

출력 : 고정지출 내역

-설명

addFixedExpense와 setFixedExpense를 통해 사용자가 고정지출을 각 월의 고정지출 목록에 추가할 수 있도록 하고, deleteFixedExpense와 removeFixedExpense를 통해 입력된 고정 지출을 삭제 할 수 있도록 한다. 이때, 인덱스 번호를 통해 원하는 내역만을 삭제할 수 있도록 한다.

-적용된 배운 내용

Cin, cout 을 사용해 사용자와 상호작용

If else문을 사용해 유효한 내역이 있는지 확인함

If 문을 통해 잘못된 입력에 대한 에러처리 및 피드백 제공

클래스와 함수를 통해 특정 기능을 수행하도록 설정

- 코드 스크린샷

```
// 예산 설정 함수
void setBudget(int amount) {
    budget = amount; // 예산 설정
    cout << "예산이 " << budget << "원으로 설정되었습니다." << endl;
}
```

```
// 남은 잔액을 계산하는 함수
int getRemainingBalance() const {
    return budget - expenses; // 남은 잔액 계산
}
```

```
if (choice == 1) { // 예산 설정
    int amount;
    cout << "예산 금액을 입력하세요: ";
    cin >> amount;
    budgetmanagers[month - 1].setBudget(amount); // 선택한 월의 예산 설정
```

```
cout << "남은 예산: " << budgetmanagers[month - 1].getRemainingBalance() << "원" << endl; // 남은 예산 출력
```

```
// 고정 지출을 설정하는 함수
void setFixedExpense(int amount, const string& description) {
    // 고정 지출을 Transaction으로 추가
    fixedExpenses.push_back({"고정지출", description, amount, "고정지출"}); // 고정 지출 추가
    cout << "고정 지출이 추가되었습니다: " << description << " - " << amount << "원" << endl;
}

// 매월 고정 지출을 적용하는 함수
void applyFixedExpenses() {
    for (const auto& expense : fixedExpenses) {
        addExpense("고정지출", expense.amount, expense.description); // 고정 지출 추가
    }
}
```

```cpp
// 고정 지출 목록을 출력하는 함수
void showFixedExpenses() const {
    if (fixedExpenses.empty()) {
        cout << "고정 지출이 없습니다." << endl;
        return;
    }
    cout << "\n[고정 지출 목록]" << endl;
    for (size_t i = 0; i < fixedExpenses.size(); ++i) {
        cout << i << ": " << fixedExpenses[i].description << " - " << fixedExpenses[i].amount << "원" << endl;
    }
}

// 고정 지출을 삭제하는 함수
void removeFixedExpense(int index) {
    if (index < 0 || index >= fixedExpenses.size()) {
        cout << "잘못된 번호입니다." << endl;
        return;
    }
    fixedExpenses.erase(fixedExpenses.begin() + index); // 고정 지출 삭제
    cout << "고정 지출이 삭제되었습니다." << endl;
}
```

```cpp
if (choice == 0) {
    cout << "프로그램을 종료합니다." << endl;
    break;
} else if (choice == 1) {
    addFixedExpense();
} else if (choice == 2) {
    deleteFixedExpense(); // 삭제 함수 호출
```

```cpp
// 고정 지출 추가하는 함수
void addFixedExpense() {
    int amount;
    string description;
    cout << "고정 지출 금액을 입력하세요: ";
    cin >> amount;
    cin.ignore(); // 개행 문자 제거
    cout << "고정 지출 항목을 입력하세요: ";
    getline(cin, description);
    for (int month = 0; month < 12; ++month) {
        budgetmanagers[month].setFixedExpense(amount, description); // 모든 월에 고정 지출 추가
    }
}

// 고정 지출 삭제하는 함수
void deleteFixedExpense() {
    showFixedExpenses(); // 현재 고정 지출 목록 출력
    int index;
    cout << "삭제할 고정 지출의 번호를 입력하세요: ";
    cin >> index;
    for (int month = 0; month < 12; ++month) {
        budgetmanagers[month].removeFixedExpense(index); // 모든 월에서 고정 지출 삭제
    }
}
```

```
// 고정 지출 적용
budgetmanagers[month - 1].applyFixedExpenses(); // 선택한 월의 고정 지출 적용
```

```
// 고정 지출 목록을 출력하는 함수
void showFixedExpenses() const {
    for (int month = 0; month < 12; ++month) {
        cout << "\n[" << (month + 1) << "월 고정 지출 목록]" << endl;
        budgetmanagers[month].showFixedExpenses(); // 각 월의 고정 지출 목록 출력
    }
}
```

2) 테스트 결과

(1) 일일 지출 내역 입력

- 특정 날짜의 지출 금액과 항목을 기록한다.

```
선택 : 2
지출 금액을 입력하세요 : 11000
지출 항목을 입력하세요 : 영화
지출 날짜를 입력하세요 (예 : 2024-11-16): 2024-01-04
지출 기록 : 2024-01-04 - 영화 - 11000원
```

(2) 일일 수입 내역 입력

- 특정 날짜의 수입 금액과 항목을 기록한다.

```
선택 : 1
수입 금액을 입력하세요 : 120000
수입 항목을 입력하세요 : 쿠팡
수입 날짜를 입력하세요 (예 : 2024-11-16): 2024-01-01
수입 기록 : 2024-01-01 - 쿠팡 + 120000원
```

(3) 월간 입력 기능

- 입력하고자 하는 달을 유저에게 직접 입력받고, 1~12월의 기록을 저장할 수 있도록 함

```
--- 월간 기록 프로그램 ---
1~12월 중 기록할 달을 선택하세요 (0 입력 시 종료): 1
--- 1월 기록 관리 ---
1. 수입 추가
2. 지출 추가
3. 거래 내역 보기
4. 돌아가기
```

(3) 거래 내역 확인

- 입력된 수입, 지출 금액 내역 확인

```
선택 : 3

[거래 내역]
2024-01-01 - 수입 : 쿠팡 - 120000원
2024-01-04 - 지출 : 영화 - 11000원
```

2) 월간 예산 설정 및 잔액 확인

(1) 월간 예산 설정

- 한 달의 지출 한도를 설정하여 현재 남은 예산을 계산한다.

```
1~12월 중 기록할 달을 선택하세요 (0 입력 시 종료): 1
--- 1월 기록 관리 ---
남은 예산 : 0원
1. 예산 설정
2. 수입 추가
3. 지출 추가
4. 거래 내역 보기
5. 돌아가기
선택 : 1
예산 금액을 입력하세요 : 300000
예산이 300000원으로 설정되었습니다.
```

(2) 잔액 확인

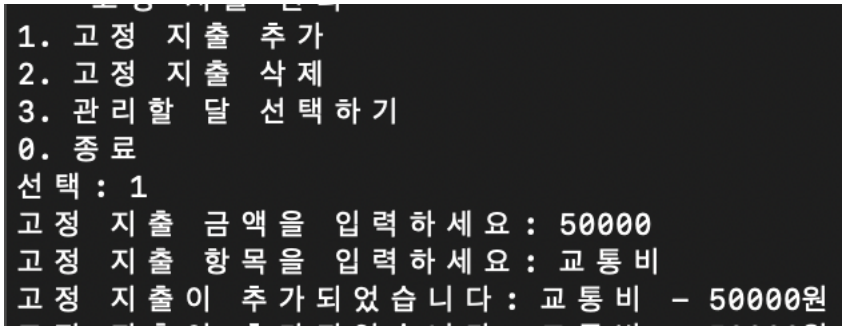
- 현재까지의 총 지출을 바탕으로 남은 예산을 실시간으로 표시한다.

```
지출 금액을 입력하세요 : 100000
지출 항목을 입력하세요 : 교재
지출 날짜를 입력하세요 (예 : 2024-11-16): 2024-01-23
지출 기록 : 2024-01-23 - 교재 - 100000원
--- 1월 기록 관리 ---
남은 예산 : 200000원
```

3) 고정 지출 항목 자동 기록

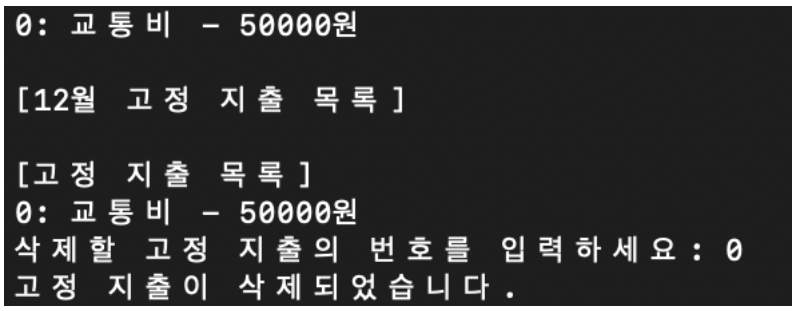
(1) 고정 지출 항목 등록

- 고정 지출되는 금액과 내용을 미리 등록하여 매달 자동으로 기록한다.



(2) 고정 지출 항목 삭제

- 추후 고정 지출이 변경 될 수 있기 때문에 지출항목을 삭제할 수 있도록 한다.



4. 계획 대비 변경 사항

1) 거래내역 확인 기능 추가

- 이전 : 기록한 거래의 내역을 확인 할 수 있는 기능이 없었음

- 이후 : 사용자 입력을 받아 기록된 내역을 확인할 수 있는 기능을 추가함

- 사유 : 거래내역을 확인할 수 없다면, 해당 프로그램의 주요 목적(자산관리 및 소비습관 확인 및 소비계획 변경)을 이루는데 어려움을 느낄 수 있다고 판단함.

2) 고정비용 등록, 삭제 시 확인문구 출력빈도 조정(예정)

- 이전 : 고정비용 등록, 삭제 시 확인문구가 12번 출력됨

- 이후 : 고정비용 등록, 삭제 시 확인문구는 1번만 출력되도록 변경

- 사유 : 확인문구가 12회 모두 출력되어 프로그램 출력 화면이 가독성이 매우 떨어진다고 판단함.

5. 프로젝트 일정

업무		11/03	11/04	11/10	11/14
제안서 작성		완료			
기능 1	세부기능1		완료		
	세부기능2		완료		
	세부기능3			완료	
업무		11/15	11/18	11/21	11/25
기능 2	세부기능1	진행중			
	세부기능2		진행중		
중간점검 및 최적화					---------->
업무		11/26	11/29	12/01	12/03
기능 3	세부기능 1	출력결과 보완중			
	세부기능 2			출력결과 보완중	
업무		12/04		12/20	12/21
기능 4	세부기능 1	진행예정			
	세부기능 2	진행예정			
최종점검 및 최적화					---------->
업무		11/17	12/1	12/15	12/22
보고서	중간보고서	---------->			
	최종보고서				---------->