C 프로그래밍 및 실습

가계부 자동화

진척 보고서 #1

제출일자: 2023-11-26
제출자명: 문준혁
제출자학번: 201595

1. 프로젝트 목표

1) 배경 및 필요성

자취를 시작하며 용돈 사용량이 늘어나고 소비의 종류도 다양해짐. 예상에 없던 지출이 자주 생겨 계획적인 소비와 절약에 어려움을 느낌. 이러한 상황이 반복되어 월 말에 경제난이 자주 발생함. 이 문제를 해결하기 위해 나만의 가계부를 체계적으로 작성해 현재 내 소비가 적당한지에 대하여 판단하게 해줄 프로그램이 필요함.

2) 프로젝트 목표

나의 과거에 대한 소비량과 현재의 소비량을 카테고리 별로 확인하여 현재 소비가 적절한지를 판단하게 해주는 프로그램 코딩을 목표로 함.

3) 차별 점

기존의 가계부들은 카테고리가 없이 소득과 지출만 보여주는 것이나 한 달을 기준으로 소득과 지출을 보여주어 해당 월 말에만 나의 소비에 대한 판단을 할 수 있음. 이 프로그램은 프로그램 종료 시 마다 전 월의 지출액 뿐만 아니라 최근 3달간의 평균 지출과 함께 현재 나의 지출을 비교하여 몇 퍼센트를 더 썼는지, 또는 덜 썼는지를 알려주어 남은 날 동안 소비를 어떻게 해야 할 지 더 확실히 알 수 있음. 그리하여 평소에 비해 많이 쓰는 항목을 인지하고 사용자의 소비에 참고할 수 있게 할 것임. 또한 계획과 달리 매달 일어나는 소비가 아닌 가끔 발생하는 소비, 예를 들어 여행, 병원비 등 이러한 것들로 인한 지출을 반영할 것임. 그리하여 주마다 예상치 못한 지출에 대한 스스로의 소비대책을 강구할 수 있게 됨.

2. 기능 계획

1) 기능 1: 메뉴 입력 받기

- 설명: '1. 지출 입력 2. 소득 입력 3. 지출 및 소득 내역 수정, 4. 특정 월 소득 및 지출 확인, 5. 종료' 목록 보여주고 입력 받기

2) 기능 2: 지출 입력 받고 저장

- 설명: 지출을 입력 받고 프로그램에 저장한다.

(1) 세부 기능 1: 여러 개의 항목으로 지출내역 입력 받기

- 설명: 술자리, 담배, 커피, 외식 및 배달, 생활용품, 옷 등 6가지 항목에 각각 지출을 입력 받고 저장한다.

3) 기능 3: 소득 입력 받고 저장

- 설명: 사용자가 얻은 소득을 입력 받고 프로그램에 저장한다.

(1) 세부 기능 1: N빵 결제 시 각 항목에 이체 받은 돈을 입력 받기.

- 설명: 평소에는 그냥 소득으로 잔고에 입력 받으면 되지만 술자리, 커피, 외식 및 배달 등 소비한 돈을 정확하게 파악하기 위해 일반 소득 + 3가지 항목을 더해 4개의 메뉴 중 선택하여 소득 입력

4) 기능 4: 소득 및 지출 수정

- 설명: 사용자가 입력했던 소득 및 지출 내역을 수정한다.

(1) 세부 기능 1: 소득과 지출을 나누어 내역을 수정한다.

- 설명: 메뉴 중 수정을 선택하면 소득과 지출을 다시 한 번 선택한 후 수정하고 싶은 내역을 수정한다.

5) 기능 5: 특정 월 소득 및 지출 확인

- 설명: 사용자가 원하는 달의 지출 내역과 소득 내역을 출력해준다

(1) 세부 기능 1: 소득과 지출의 내역의 총합을 출력해준다

- 설명: 내역 뿐만 아니라 총 금액의 합계도 같이 출력해준다.

6) 기능 6: 프로그램 종료 시 이번 달 지출 및 소득 총합 출력

- 설명: 프로그램 종료 시 이번 달에 사용한 금액의 총합과 소득의 총합을 각각 출력해준다.

(1) 세부 기능 1: 해당 달의 지출의 총 합계와 소득의 총 합계를 출력해준다.

- 설명: 프로그램 종료마다 내 현재 내역을 보며 이번 달 소비를 평가

(2) 세부 기능 2: 최근 세 달의 지출 총 합계의 평균, 소득 총 합계의 평균 출력해준다.

- 설명: 이번 달 뿐만 아니라 최근 평균 소비와 같이 보며 내 현재 소비를 평가

(3) 세부 기능 3: 저번 달과 비교해 이번 달 지출과 소득 비율을 출력해준다

- 설명: 총액 뿐만 아니라 비율로 표시해 더 직관적으로 소비를 평가할 수 있게 해준다.

3. 진척사항

1) 기능 구현

(1) 메뉴 입력 받기

- 입출력: 메뉴 출력 및 입력 받기

- 설명: 프로그램 시작 시 메뉴를 출력해주고 switch문으로 메뉴를 입력 받는다.

- 적용된 배운 내용: 반복문, switch문

- 코드 스크린샷

```
while (!terminate) {
  printf("------------------₩n");
  printf("메뉴를 입력해주세요.₩n");
  printf(
      "1. 지출 입력₩n2. 소득 입력₩n3. 지출 및 소득 내역 수정₩n4."
      " 특정 월소득 및 지출 확인₩n5. 종료₩n");
  printf("------------------₩n");

  scanf_s("%d", &choice);

  switch (choice) {
    case 1:  // 지출 입력 및 저장
      input_Expense(expenses, expense_types);
      break;
    case 2:  // 소득 입력 및 저장
      input_Income(incomes, income_types);
      break;
    case 3:  // 지출 및 소득내역을 수정
      modify_list(expenses, incomes, expense_types, income_types);
      break;
    case 4:  //특정 월소득 및 지출 확인
      display_list(year, month);
      break;
    case 5:  //종료 및 소비평가 출력
      terminate = 1;
      break;

    default:
      printf("올바른 메뉴를 선택해주세요.₩n");
      break;
  }
```

(2) 지출 입력 받고 저장

- 입출력: 지출 입력 받기

- 설명: 지출 항목, 내역, 금액을 입력 받는 함수를 호출하여 실행하고 저장한다.

- 적용된 배운 내용: switch문, 조건문, 반복문, 문자열, 함수, 2차원 배열

- 코드 스크린샷

```c
void input_Expense(float expenses[GOODS][PRICE],   //지출 입력을 위한 함수정의
                   char expense_types[EXPENSE_CATEGORIES][20]) {
    int input_year, input_month, input_day;
    printf("저장할 날짜를 입력하세요(YY MM DD): ");
    scanf_s("%d %d %d", &input_year, &input_month, &input_day);

    if (input_year > 0 && input_year <= YEAR && input_month > 0 &&
        input_month <= MONTH && input_day > 0 && input_day <= DAY) {
        calendar[input_year - 1][input_month - 1][input_day - 1] = 1.0;

        // 전역 변수에 날짜 저장
        year = input_year;
        month = input_month;
        day = input_day;
    } else {
        printf("유효하지 않은 날짜입니다.₩n");
        return;
    }

    int count = 0;  // 입력한 개수 저장 변수

    char temp_expenses[20]; //문자열을 배열에 저장하기 위한 임시 변수

    for (int i = 0; i < GOODS; i++) {
        printf("지출 항목을 입력하세요(1 입력 시 종료): ");
        scanf_s("%19s", expense_types[i], 20);

        if (expense_types[i][0] == '1') {
            printf("'1'을 입력하셨습니다. 지출 입력을 종료합니다.₩n");
            break;
        }

        printf("지출 내역을 입력하세요: ");
        scanf_s("%19s", temp_expenses, 20);  // 문자열을 임시 배열에 저장

        strcpy_s(expense_types[i], sizeof(expense_types[i]), temp_expenses);

        printf("해당 내역의 금액을 입력하세요: ");
        scanf_s("%f", &expenses[i][0]);
        printf("입력된 지출 값: %.2f₩n", expenses[i][0]);
        count++;
    }

    printf("%d년 %d월 %d일에 %d개의 내용이 저장되었습니다.₩n", year, month, day,
           count);
}
```

(3) 소득 입력 받고 저장

- 입출력: 소득 입력 받기

- 설명: 소득 항목, 내역, 금액을 입력 받는 함수를 호출하여 실행하고 저장한다.

- 적용된 배운 내용: switch문, 조건문, 반복문, 문자열, 함수, 2차원 배열

- 코드 스크린샷

```
179
180     void input_Income(float incomes[GOODS][EARN],   //소득 입력을 위한 함수 정의
181                       char income_types[INCOME_CATEGORIES][20]) {
182         int input_year, input_month, input_day;
183         printf("저장할 날짜를 입력하세요(YY MM DD): ");
184         scanf_s("%d %d %d", &input_year, &input_month, &input_day);
185
186         if (input_year > 0 && input_year <= YEAR && input_month > 0 &&
187             input_month <= MONTH && input_day > 0 && input_day <= DAY) {
188             calendar[input_year - 1][input_month - 1][input_day - 1] = 1.0;
189
190             year = input_year;
191             month = input_month;
192             day = input_day;
193         } else {
194             printf("유효하지 않은 날짜입니다.₩n");
195             return;
196         }
197
198         char temp_incomes[20];  // 문자열을 배열에 저장하기 위한 임시 변수
199
200         int count = 0;
201         for (int i = 0; i < GOODS; i++) {
202             printf("소득 항목을 입력하세요(1 입력 시 종료): ");
203             scanf_s("%19s", income_types[i], 20);
204
205             if (income_types[i][0] == '1') {
206                 printf("'1'을 입력하셨습니다. 소득 입력을 종료합니다.₩n");
207                 break;
208             }
209
210             printf("소득 내역을 입력하세요: ");
211             scanf_s("%19s", temp_incomes, 20);  // 문자열을 임시 배열에 저장
212
213             strcpy_s(income_types[i], sizeof(income_types[i]), temp_incomes);
214
215             printf("해당 내역의 금액을 입력하세요: ");
216             scanf_s("%f", &incomes[i][0]);
217
218             count++;
219         }
220
221         printf("%d년 %d월 %d일에 %d개의 내용이 저장되었습니다.₩n", year, month, day,
222                count);
223     }
224
```

(4) 소득 및 지출 수정

- 입출력: 입력 받은 소득 및 지출을 수정을 위해 재입력 받기

- 설명: 먼저 날짜를 입력 받고, 소득과 지출을 선택하여 수정할 항목, 내역, 금액을 입력 받는 함수를 호출한다. 소득 수정 함수와 지출 수정 함수를 각각 정의하여 호출한다.

- 적용된 배운 내용: switch문, 조건문, 문자열, 함수, 2차원 배열

- 코드 스크린샷

```c
// 지출, 소득 내역 수정 함수 정의
void modify_list(float expenses[GOODS][PRICE], float incomes[GOODS][EARN],
                 char expense_types[EXPENSE_CATEGORIES][20],
                 char income_types[INCOME_CATEGORIES][20]) {
  printf("수정할 내역을 선택하세요:₩n");
  printf("1. 지출 내역 수정₩n");
  printf("2. 소득 내역 수정₩n");
  printf("3. 돌아가기₩n");

  int choice;
  scanf_s("%d", &choice);

  switch (choice) {
    case 1:
      modify_Expense(expenses, expense_types); // 지출을 수정
      break;
    case 2:
      modify_Income(incomes, income_types);   //소득을 수정
      break;
    case 3:
      printf("돌아갑니다.₩n");
      break;
    default:
      printf("올바른 메뉴를 선택하세요.₩n");
      break;
  }
}
```

```c
void modify_Expense(float expenses[GOODS][PRICE],  //지출 수정 함수 정의
                    char expense_types[EXPENSE_CATEGORIES][20]) {
    int input_year, input_month, input_day;
    printf("수정할 날짜를 입력하세요(YY MM DD): ");
    scanf_s("%d %d %d", &input_year, &input_month, &input_day);

    char temp_modify[20];  // 문자열을 배열에 저장하기 위한 임시 변수

    if (calendar[input_year - 1][input_month - 1][input_day - 1] == 1.0) {
        int modify_index;
        printf("수정할 지출 항목을 선택하세요 (0부터 시작): ");
        scanf_s("%d", &modify_index);

        if (modify_index >= 0 && modify_index < GOODS) {
            printf("새로운 지출 항목을 입력하세요: ");
            scanf_s("%s", expense_types[modify_index], 20);

            printf("새로운 지출 내역을 입력하세요: ");
            scanf_s("%19s", temp_modify, 20);  // 문자열을 임시 배열에 저장
            strcpy_s(expense_types[modify_index], sizeof(expense_types[modify_index]),
                     temp_modify);



            printf("새로운 지출 내역의 금액을 입력하세요: ");
            scanf_s("%f", &expenses[modify_index][0]);

            printf("%d년 %d월 %d일의 지출 내용이 수정되었습니다.₩n", input_year,
                   input_month, input_day);
            return;
        } else {
            printf("올바른 지출 항목 인덱스를 선택하세요.₩n");
        }
    } else {
        printf("%d년 %d월 %d일에 저장된 내역이 없습니다.₩n", input_year,
               input_month, input_day);
    }
}
```

```c
void modify_Income(float incomes[GOODS][EARN],  //소득 수정 함수 정의
                   char income_types[INCOME_CATEGORIES][20]) {
    int input_year, input_month, input_day;
    printf("수정할 날짜를 입력하세요(YY MM DD): ");
    scanf_s("%d %d %d", &input_year, &input_month, &input_day);

    char temp_modify[20]; // 문자열을 배열에 저장하기 위한 임시 변수

    if (calendar[input_year - 1][input_month - 1][input_day - 1] == 1.0) {
        int modify_index;
        printf("수정할 소득 항목을 선택하세요 (0부터 시작): ");
        scanf_s("%d", &modify_index);

        if (modify_index >= 0 && modify_index < GOODS) {
            printf("새로운 소득 항목을 입력하세요: ");
            scanf_s("%s", income_types[modify_index], 20);

            printf("새로운 소득 내역을 입력하세요: ");
            scanf_s("%19s", temp_modify, 20); // 문자열을 임시 배열에 저장
            strcpy_s(income_types[modify_index], sizeof(income_types[modify_index]),
                     temp_modify);

            printf("새로운 내역의 금액을 입력하세요: ");
            scanf_s("%f", &incomes[modify_index][0]);

            printf("%d년 %d월 %d일의 소득 내용이 수정되었습니다.₩n", input_year,
                   input_month, input_day);
            return;
        } else {
            printf("올바른 소득 항목 인덱스를 선택하세요.₩n");
        }
    } else {
        printf("%d년 %d월 %d일에 저장된 내역이 없습니다.₩n", input_year,
               input_month, input_day);
    }
}
```

(5) 특정 월 소득 및 지출 확인

- 입출력: 원하는 달을 입력 받고 소득 및 지출 내역과 총합 출력

- 설명: 사용자가 확인하고 싶은 달을 입력 받고 소득 및 지출 내역을 출력해주는 함수를 호출한다. 그리고 총합을 출력해주는 함수를 따로 정의하여 같이 호출해준다.

- 적용된 배운 내용: switch문, 반복문, 문자열, 함수, 2차원 배열, 포인터

- 코드 스크린샷

```c
void calculateTotal(float (*expenses)[PRICE], float (*incomes)[EARN],
                    int year, int month) { // 특정 달 지출, 소비 총합 출력 함수 정의
    float totalExpenses = 0.0;
    float totalIncomes = 0.0;
    for (int i = 0; i < GOODS; i++) {
        totalExpenses += *(*(expenses + i) + 0);
        totalIncomes += *(*(incomes + i) + 0);
    }
    printf("%d년 %d월의 총 지출: %.2f원₩n", year, month, totalExpenses);
    printf("%d년 %d월의 총 소득: %.2f원₩n", year, month, totalIncomes);
}

void display_list(int year, int month) {  //특정 월 소득, 지출 내역 확인 함수 정의
    printf("년도와 월을 입력하세요 (YY MM): ");
    scanf_s("%d %d", &year, &month);
    printf("%d년 %d월의 내역을 출력합니다.₩n", year, month);

    printf("지출 내역:₩n");
    for (int i = 0; i < expense_count ; i++) {
        printf("%s: %.f원₩n", expense_types[i], *(*(expenses + i) + 0));
    }
    print("₩n");
    printf("소득 내역:₩n");
    for (int i = 0; i < income_count; i++) {
        printf("%s: %.f원₩n", income_types[i], *(*(incomes + i) + 0));
    }
    print("₩n");
    calculateTotal(expenses, incomes, year, month); // 지출, 소득 총합 출력 함수 호출
}
```

(6) 프로그램 종료 시 이번 달 지출 및 소득 총합 출력

- 입출력: 종료 메뉴 선택 시 이번 달 지출 및 소득 총합 출력

- 설명: 사용자가 프로그램을 종료 메뉴를 선택하면 자동으로 이번 달 지출 및 소득의 총합을 출력해주는 함수를 호출 후 종료한다.

- 적용된 배운 내용: switch문, 반복문, 함수, 2차원 배열, 포인터

- 코드 스크린샷

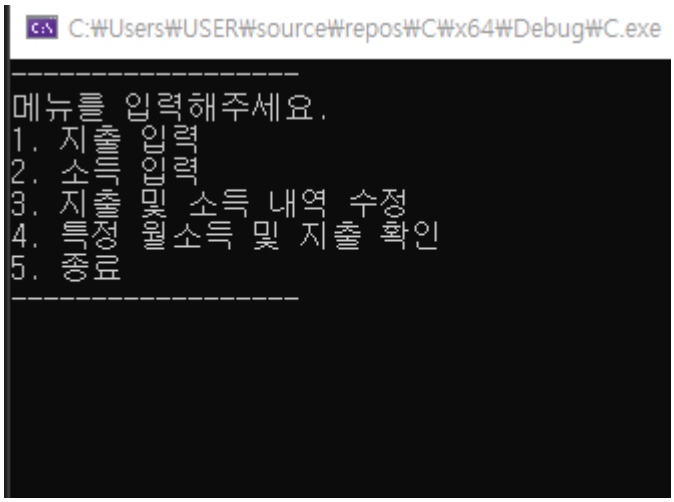
```c
void printSummary(float (*expenses)[PRICE], //종료 시 이번 달 지출 및 소득 총합 출력 함수 정의
                  float (*incomes)[EARN], int year,
                  int month) {
    float currentMonthExpense = 0.0, currentMonthIncome = 0.0;

    // 현재 달의 지출과 소득 총합 구하기
    for (int i = 0; i < GOODS; i++) {

        currentMonthExpense += *(*(expenses + i) + 0);
        currentMonthIncome += *(*(incomes + i) + 0);


    }

    printf("%d년 %d월의 지출 합: %.2f원₩n", year, month,
           currentMonthExpense);
    printf("%d년 %d월의 소득 합: %.2f원₩n", year, month,
           currentMonthIncome);
}
```

2) 테스트 결과

(1) 메뉴 입력 받기

- 설명: 프로그램 시작 시 메뉴를 출력해주고 switch문으로 메뉴를 입력 받는다.

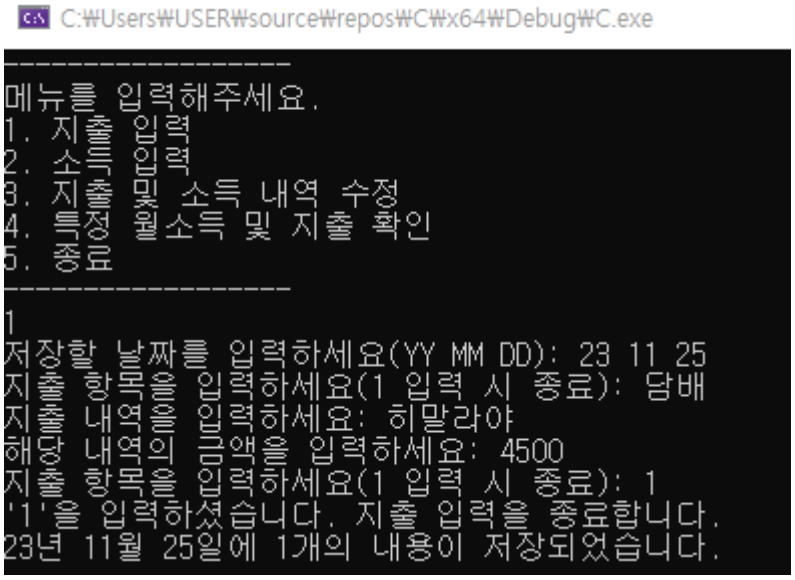
- 테스트 결과 스크린샷



(2) 지출 입력 받고 저장

- 설명: 지출 항목, 내역, 금액을 입력 받는 함수를 호출하여 실행하고 저장한다.

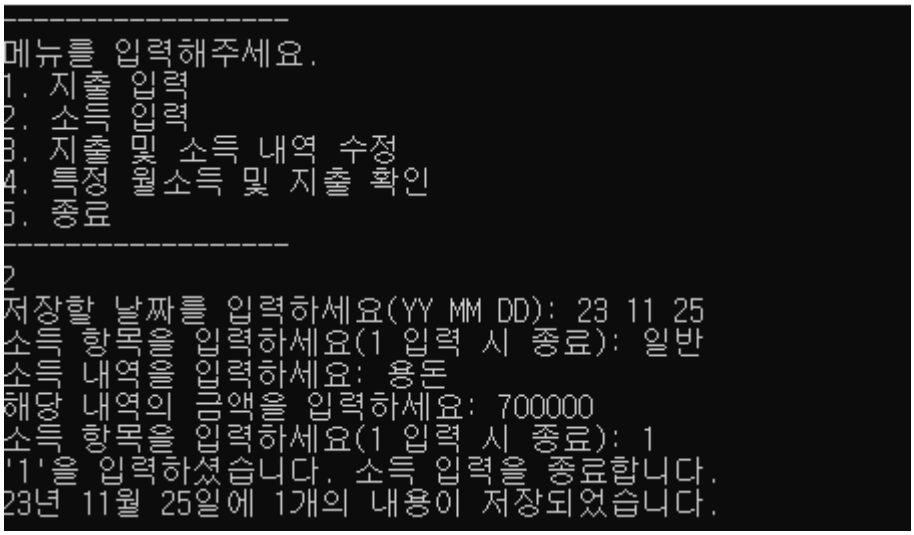
- 테스트 결과 스크린샷



(3) 소득 입력 받고 저장

- 설명: 소득 항목, 내역, 금액을 입력 받는 함수를 호출하여 실행하고 저장한다.

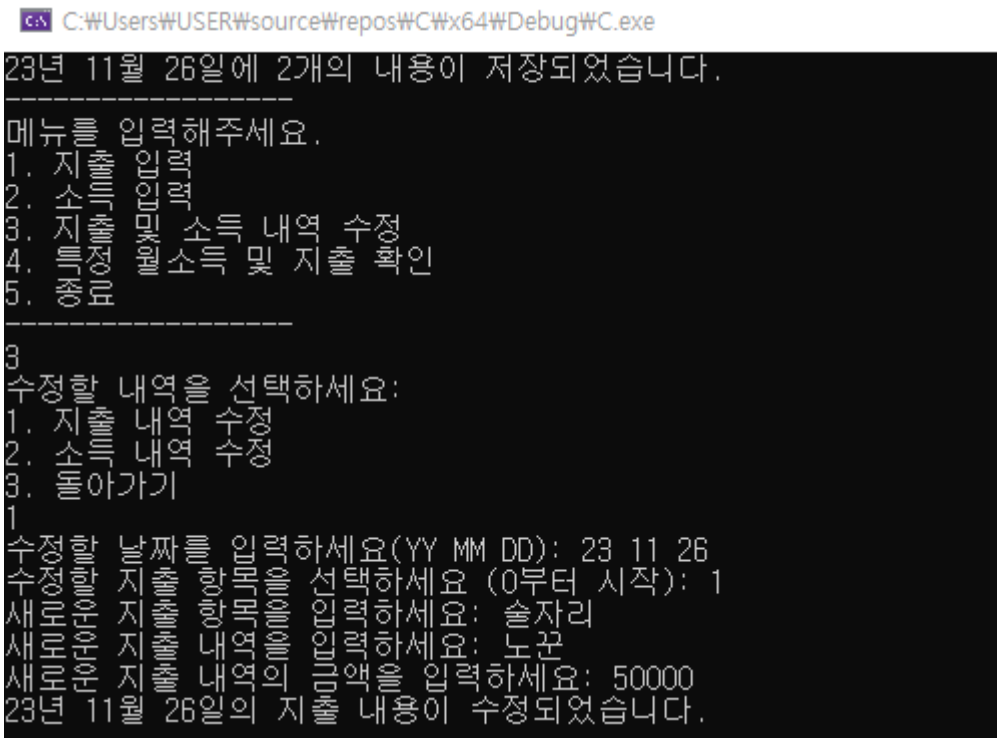
- 테스트 결과 스크린샷



(4) 소득 및 지출 수정

- 설명: 먼저 날짜를 입력 받고, 소득과 지출을 선택하여 수정할 항목, 내역, 금액을 입력 받는 함수를 호출한다. 소득 수정 함수와 지출 수정 함수를 각각 정의하여 호출한다.

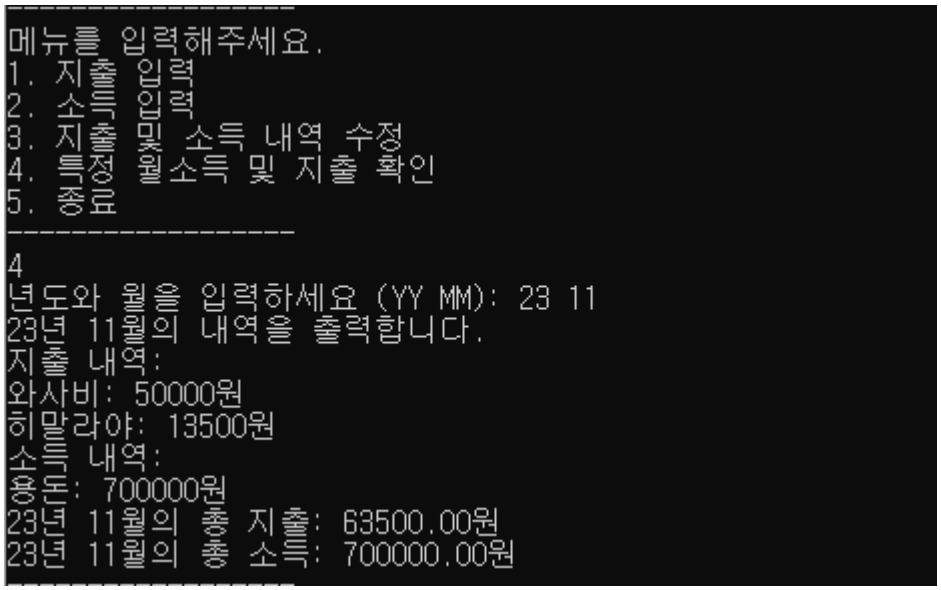
- 테스트 결과 스크린샷



(5) 특정 월 소득 및 지출 확인

- 설명: 사용자가 확인하고 싶은 달을 입력 받고 소득 및 지출 내역을 출력해주는 함수를 호출한다. 그리고 총합을 출력해주는 함수를 따로 정의하여 같이 호출해준다.

- 테스트 결과 스크린샷

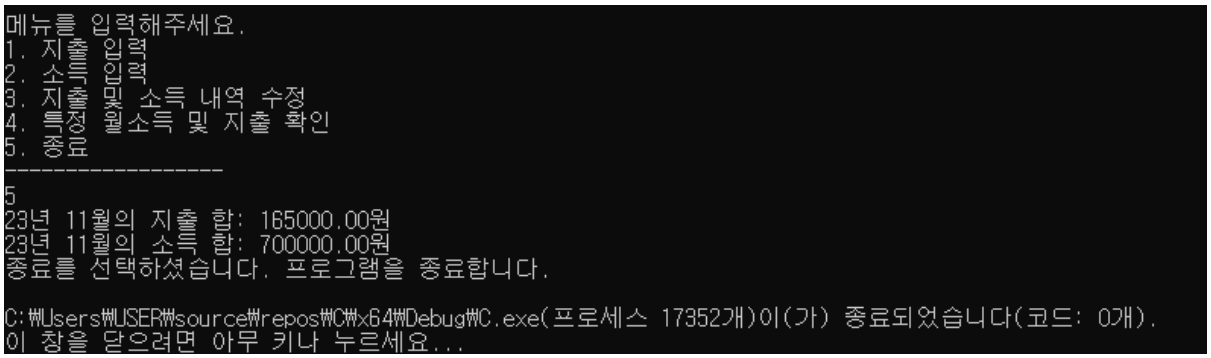


(6) 프로그램 종료 시 이번 달 지출 및 소득 총합 출력

- 설명: 사용자가 프로그램을 종료 메뉴를 선택하면 자동으로 이번 달 지출 및 소득의

총합을 출력해주는 함수를 호출 후 종료한다.

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

1) 특정 월 소득 및 지출 확인

- 이전-

[기능 5]: 지출비용과 현재잔액을 보여준다.

- 설명: 이번 달에 사용한 금액을 모두 합하여 보여준다.

세부 기능 1: 항목 별 지출비용과 총 지출비용 그리고 현재 잔액을 보여준다.

- 설명: 6개의 항목에 각각 얼마를 지출했는지, 그것을 다 더한 총 지출비용 그리고 현재 잔액을 계산해서 보여준다

- 이후-

[기능 5]: 특정 월의 소득 및 지출 확인

: 이번 달이 아닌 사용자가 원하는 달의 지출과 소득 내역, 그리고 합계를 출력해준다.

- 사유-

: 내 지출 및 소득 내역을 저장하고 관리하는 가계부의 본래 목적에 맞게 원하는 년도, 월의 내역을 보게 하기 위함.

2) 프로그램 종료 시 이번 달 지출 및 소득 총합 출력

- 이전-

[기능 6]: 이번 달 소비평가

- 설명: 이번 달에 사용한 금액을 모두 합하여 보여준다.

세부 기능 1: 항목 별 지출비용과 총 지출비용 그리고 현재 잔액을 보여준다.

- 설명: 6개의 항목에 각각 얼마를 지출했는지, 그것을 다 더한 총 지출비용 그리고 현재 잔액을 계산해서 보여준다.

- 이후-

[기능 6]: 이번 달 소비평가

- 설명: 프로그램 종료 시 이번 달에 사용한 금액의 총합과 소득의 총합을 각각 출력

세부 기능 1: 해당 달의 지출의 총 합계와 소득의 총 합계를 출력

- 설명: 프로그램 종료마다 내 현재 내역을 보며 이번 달 소비를 평가

세부 기능 2: 최근 세 달의 지출 총 합계의 평균, 소득 총 합계의 평균 출력

- 설명: 이번 달 뿐만 아니라 아닌 내 평균 소비와 같이 보며 내 현재 소비를 평가

세부 기능 3: 저번 달과 비교해 이번 달 지출과 소득 비율을 출력해준다

- 설명: 총액 뿐만 아니라 비율로 표시해 더욱 직관적으로 소비를 평가할 수 있게 해준다.

- 사유-

가계부 자동화의 취지에 맞게 지출 및 소득의 입력을 끝내고 나면 항상 이번 달 나의 소비가 적절한지에 대해 확인할 수 있도록 하기 위함

5. 프로젝트 일정

업무		11/3	11/10	11/17	11/25
제안서 작성		완료			
기능1	세부기능1	---------->			
기능2	세부기능1	---------->			
기능3	세부기능1	---------->			
기능4	세부기능1		---------->		
기능5	세부기능1		---------->		

업무		11/17	11/25	12/01	12/08
기능6	세부기능1	---------->			
프로그램 코드 오류 수정		---------->			
진척 보고서1 작성		---------->			
기능6	세부기능2		---------->		
	세부기능3			---------->	

업무	12/10	12/15	12/22	
진척 보고서2 작성	완료			
기능 추가 및 삭제	---------->			
프로그램 검토		---------->		