

C 프로그래밍 및 실습

마리모 관리

기록장

프로젝트 제안서

제출일자: 2023-12-10

제출자명: 김산하

제출자학번: 230382

1. 프로젝트 목표

1) 배경 및 필요성

집에서 마리를 키우고 있는데 마리는 일정한 주기에 따라 수조의 물을 갈아줘야 하고 밥(영양제)도 줘야 하며 가끔씩 수조 청소도 해줘야 한다. 그런데 이렇게 마리가 요구하는 요구사항이 많고 가족들이 함께 마리를 관리하고 있어 언제 수조에 물을 갈았고 언제 밥을 줬는지 등을 기록할 수 있는 전용 기록장이 필요하다. 또 해야 할 일을 잊어버리는 경우도 있어서 마지막으로 관리를 한 건 언제 인지, 지금 마리에게 필요한게 있는지 알려주는 기능 역시 필요하다.

2) 프로젝트 목표

마리 관리 물 갈아주기, 밥 주기, 수조 청소로 나누어 날짜와 함께 기록하면 이 기록을 바탕으로 마지막으로 각각의 관리를 한지 얼마나 지났는지 또 현재 어떤 관리가 필요한지 알려주는 프로그램을 만드는 것을 목표로 한다.

3) 차별점

기존에는 스마트폰 달력 앱을 이용해 마리에게 물을 준 날짜를 기록하였으나 몇 가지 문제가 있어 이를 해결하고자 한다.

첫번째 문제는 스마트폰 달력 앱을 사용하면 본인만 볼 수 있어 다른 사람과 공유가 어렵다는 점이다. 그러나 집 거실에 있는 데스크탑은 가족 모두가 이용하고 자주 켜져 있으므로 나는 집 데스크탑에서 작동할 프로그램을 만들 것이다.

둘째는 달력 앱이 가진 여러 불편한 점들 때문이다. 달력 앱에 기록하기 위해서는 "마리 물 갈", "마리 밥 줌" 같은 식으로 일일이 타자를 쳐서 기록해야 하고 지난달의 기록을 보기 위해서는 달력을 넘겨야 하며 언제 다음에 물을 갈아야 하는지 지나 잊은 게 있는지도 알 수 없다. 그러므로 나는 일일이 문장을 입력하는 대신 1,2,3같은 숫자를 이용해 간편하게 입력할 수 있고 과거의 기록을 한 번에 볼 수 있으며 다음 물 갈기, 밥 주기, 청소가 필요한 날짜를 처음 화면에 출력하게 프로그램을 작성할 것이다.

2. 기능 계획

1) 한 일 입력 받고 저장하기

- 설명: 날짜와 한 일(물 갈기, 밥 주기, 청소)를 입력 받고 저장하는 기능

(1) 날짜와 한 일을 입력 받기

- 설명 (scanf_s()함수를 이용해 날짜와 한 일을 입력 받는다. 이때 한 일을 1='물 갈아주기', 2='밥 주기', 3='수조 청소'로 하여 문자대신 정수로 입력 받는다.)

(2) 입력 받은 내용을 저장하기

- 설명 (날짜는 2차원 배열에, 한 일은 입력 받은 정수를 그대로 배열에 저장한다.)

(3) 입력 받은 내용을 관리하기

- 설명 (배열이 다 차면 가장 오래된(앞에 있는) 항목을 삭제하고 TODO리스트에서 삭제 기능처럼 나머지를 전부 아래로 한 칸씩 내린다.)

2) 날짜 계산하고 각각의 할 일을 해야 할 날짜 알려주기

- 설명: 날짜를 계산해 어떤 일을 언제까지 해야 하는지 출력하기

(1) 날짜 계산

- 설명 (xx월 xx일에서 n일 뒤가 몇 월 몇일인지 계산한다.)

(2) 각각의 할 일을 해야 할 날짜 알려주기

- 설명 (각각 가장 최근에 한 일로부터 물 갈아주기는 1주일, 밥주기는 2주일, 수조 청소는 1달전후로 해야 한다. 따라서 각각 가장 최근 한 일의 날짜에 7일, 14일, 30일 후의 날짜를 해당 할 일의 이름과 함께 출력한다.)

(3) 한 일 배열에 있는 내용을 문자열로 변환해 출력

- 설명 (한 일은 한 일을 저장하는 배열에 정수형태로 저장되어 있으므로 일을 1="물 갈아주기", 2="밥 주기", 3="수조 청소"의 문자열로 변환하고 날짜와 함께 출력한다.)

3) 이전까지 한 일 출력

- 설명: 이전까지 모든 한 일과 했던 날짜를 함께 출력한다

(1) 한 일 배열에 있는 내용을 문자열로 변환해 출력

- 설명 (한 일은 한 일을 저장하는 배열에 정수형태로 저장되어 있으므로 일을 1="물 갈아주기", 2="밥 주기", 3="수조 청소"의 문자열로 변환하고 날짜와 함께 출력한다.)

(2) 한 일과 날짜 전부 출력

- 설명 (for문을 이용해 저장된 모든 한 일과 그 일을 한 날짜를 함께 출력한다.)

3. 진척사항

1) 기능 구현

(1) 메뉴 출력하고 입력 받기

- 입출력: 1. 한 일 추가, 2. 날짜 계산 및 할 일 알림, 3. 이전까지 한 일 출력, 4. 종료"가 적힌 메뉴를 출력하고 (1-4)의 정수를 입력 받는다.

- 설명: 이 프로그램의 가장 기본적인 부분으로 메뉴 목록을 출력하고 이 중 어떤 기능을 사용할 것인지 입력 받아 그 기능에 해당하는 함수를 작동시킨다.

- 적용된 배운 내용: 반복문, 조건문, 함수

- 코드 스크린샷

```

int main() {
    int choice;

    do {
        printf("\n=== TODO 관리 프로그램 ===\n");
        printf("1. 한 일 추가\n");
        printf("2. 날짜 계산 및 할 일 알림\n");
        printf("3. 이전까지 한 일 출력\n");
        printf("4. 종료\n");
        printf("선택: ");
        scanf_s("%d", &choice);

        switch (choice) {
            case 1:
                addTodo();
                break;
            case 2:
                calculateDueDate();
                break;
            case 3:
                printAllTodo();
                break;
            case 4:
                printf("프로그램을 종료합니다.\n");
                break;
            default:
                printf("올바른 선택이 아닙니다. 다시 선택해주세요.\n");
        }

    } while (choice != 4);

    return 0;
}

```

(2) 한 일 입력 받고 저장하기

- 입출력: 날짜와 한 일의 종류를 입력 받는다.
- 설명: 날짜와 한 일의 종류를 입력 받아 저장한다. 저장된 한 일의 개수는 10개가 최대이며 10개인 상태에서 새로운 한 일을 입력 받으면 가장 오래된 내용을 지우고 새로운 한 일을 저장한다.
- 적용된 배운 내용: 반복문, 조건문, 함수, 배열, 헤더 파일

- 코드 스크린샷

```

// 오래된 항목 삭제 기능
void removeOldestTodo() {
    if (itemCount <= 0) {
        printf("삭제할 항목이 없습니다.\n");
        return;
    }

    // 가장 오래된 항목 삭제
    for (int i = 0; i < itemCount - 1; i++) {
        strcpy_s(dates[i], sizeof(dates[i]), dates[i + 1]);
        tasks[i] = tasks[i + 1];
    }

    itemCount--;
}

// 한 일 추가 기능
void addTodo() {
    if (itemCount >= MAX_TODO_ITEMS) {
        printf("TODO 리스트가 가득 찼습니다. 오래된 항목을 삭제합니다.\n");
        removeOldestTodo();
    }

    printf("날짜 입력 (예: 2023-01-01): ");
    scanf_s("%s", &dates[itemCount], sizeof(dates[itemCount]));

    printf("한 일 입력 (1=물 갈아주기, 2=밥 주기, 3=수조 청소): ");
    scanf_s("%d", &tasks[itemCount]);

    itemCount++;
    printf("TODO가 추가되었습니다.\n");
}

```

(3) 이전까지 한 일 출력

- 입출력: 함수를 작동시키면 이전까지 한 일을 출력한다.
- 설명: 기능 구현2에서 숫자형태로 입력 받아 저장한 할 일을 문자로 바꾸어 순서대로 출력한다.
- 적용된 배운 내용: 반복문, 조건문, 함수, 배열, 헤더 파일
- 코드 스크린샷

```

// 이전까지 한 일 출력 기능
void printAllTodo() {
    if (itemCount <= 0) {
        printf("이전에 한 일이 없습니다.\n");
        return;
    }

    printf("이전까지 한 일과 날짜를 출력합니다.\n");
    for (int i = 0; i < itemCount; i++) {
        printf("날짜: %s, ", dates[i]);

        switch (tasks[i]) {
            case 1:
                printf("물 갈아주기\n");
                break;
            case 2:
                printf("밥 주기\n");
                break;
            case 3:
                printf("수조 청소\n");
                break;
            default:
                printf("알 수 없는 작업\n");
        }
    }
}

```

(4) 날짜 계산 및 할 일 알림

- 입출력: 함수를 작동시키면 한 일의 종류별로 가장 최근에 한 날짜와 다시 해야 하는 날짜를 출력한다.
- 설명: 한 일의 종류별로 가장 최근에 한 날짜와 그 날짜에서 일정 기간을 더해 다시 해야 하는 날짜를 출력한다. 달 별로 날짜 수가 다른 것은 물론 윤년도 고려하여 날짜를 계산한다.
- 적용된 배운 내용: 반복문, 조건문, 함수, 배열, 구조체, 헤더 파일
- 코드 스크린샷

```

// 날짜 계산 및 할 일 알림 기능: 수정 완료
void calculateDueDate() {
    if (itemCount <= 0) {
        printf("할 일이 없습니다.\n");
        return;
    }

    printf("최근에 한 일로부터 날짜를 계산합니다.\n");
    for (int i = 0; i < itemCount; i++) {
        printf("최근에 한 일: %s, ", dates[i]);

        switch (tasks[i]) {
            case 1:
                printf("물 갈아주기: ");
                break;
            case 2:
                printf("밥 주기: ");
                break;
            case 3:
                printf("수조 청소: ");
                break;
            default:
                printf("알 수 없는 작업: ");
        }

        // 이 부분을 DateCalculator.h의 calculateDate 함수로 대체
        struct Date currentDate;
        sscanf_s(dates[i], "%d-%d-%d", &currentDate.year, &currentDate.month, &currentDate.day);
        struct Date dueDate = calculateDate(currentDate, tasks[i] == 1 ? 7 : (tasks[i] == 2 ? 14 : 30));

        printf("%d년 %d월 %d일에 다시 해야 합니다.\n", dueDate.year, dueDate.month, dueDate.day);
    }
}

```

```

#include <stdio.h>

// 날짜를 나타내는 구조체
struct Date {
    int year;
    int month;
    int day;
};

// 각 월의 일수 배열
int daysInMonth[2][12] = {
    {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}, // 평년
    {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31} // 윤년
};

// 윤년 여부를 판단하는 함수
int isLeapYear(int year) {
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}

// n일 후의 날짜를 계산하는 함수
struct Date calculateDate(struct Date currentDate, int n) {
    // 윤년 여부 확인
    int isLeap = isLeapYear(currentDate.year);

    // 현재 날짜에 n일을 더함
    currentDate.day += n;

    // 현재 달의 일수를 초과한 만큼 날짜를 감소시키고, 이제 다음 달로 이동
    while (currentDate.day > daysInMonth[isLeap][currentDate.month - 1]) {
        currentDate.day -= daysInMonth[isLeap][currentDate.month - 1];
        currentDate.month++;

        if (currentDate.month > 12) {
            currentDate.month = 1;
            currentDate.year++;
            isLeap = isLeapYear(currentDate.year);
        }
    }

    return currentDate;
}

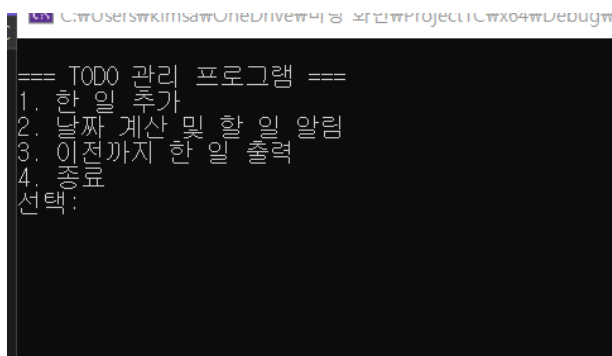
```


4. 테스트 결과

(2) 테스트 결과

(1) 메뉴 출력하고 입력 받기

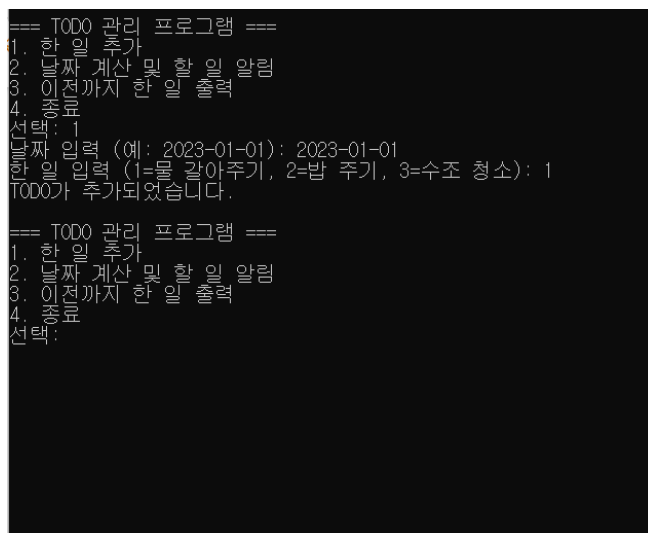
- 설명: 메뉴를 출력하고 어떤 기능을 실행할지 입력 받는다.
- 테스트 결과 스크린샷



```
C:\Users\kim\OneDrive\바탕 화면\Project\C\wx04\Debug>
=== TODO 관리 프로그램 ===
1. 한 일 추가
2. 날짜 계산 및 할 일 알림
3. 이전까지 한 일 출력
4. 종료
선택:
```

(2) 한 일 입력 받고 저장하기

- 설명: 한 일을 입력 받아 저장한다.
- 테스트 결과 스크린샷



```
=== TODO 관리 프로그램 ===
1. 한 일 추가
2. 날짜 계산 및 할 일 알림
3. 이전까지 한 일 출력
4. 종료
선택: 1
날짜 입력 (예: 2023-01-01): 2023-01-01
한 일 입력 (1=물 갈아주기, 2=밥 주기, 3=수조 청소): 1
TODO가 추가되었습니다.

=== TODO 관리 프로그램 ===
1. 한 일 추가
2. 날짜 계산 및 할 일 알림
3. 이전까지 한 일 출력
4. 종료
선택:
```

(3) 이전까지 한 일 출력

- 설명: (2)에서 저장한 내용을 출력한다.

- 테스트 결과 스크린샷

```
=== TODO 관리 프로그램 ===
1. 한 일 추가
2. 날짜 계산 및 할 일 알림
3. 이전까지 한 일 출력
4. 종료
선택: 3
이전까지 한 일과 날짜를 출력합니다.
날짜: 2023-01-01, 물 갈아주기
```

(4) 날짜 계산 및 할 일 알림

- 설명: 각각 한 일별로 가장 최근에 언제 했는지 언제 다시 해야 하는지 출력.

- 테스트 결과 스크린샷

```
=== TODO 관리 프로그램 ===
1. 한 일 추가
2. 날짜 계산 및 할 일 알림
3. 이전까지 한 일 출력
4. 종료
선택: 2
최근에 한 일로부터 날짜를 계산합니다.
최근에 한 일: 2023-01-01, 물 갈아주기: 2023년 1월 8일에 다시 해야 합니다.
최근에 한 일: 2023-01-01, 밥 주기: 2023년 1월 15일에 다시 해야 합니다.
```

5. 계획 대비 변경 사항

1) 없음

6. 느낀점

- 코딩할 때 주석을 잘 다는게 얼마나 중요한지 느꼈다. 만약 주석을 제대로 달지 않았다면 코드를 이해하는데 시간이 걸리고 계속 잊어버려서 힘들었을 것이다. 계획 세우고 지키는게 힘들었다.

