공과대학
소프트웨어학과

2019/5/2

(C Programming) Programming Assignment #2

(마감일: 2019년 5월 17일 오후 5시)

목적

1. C의 다양한 제어문 (조건문 및 반복문)의 활용
2. 함수의 활용
3. 다중 소스프로그램의 활용

문제: 교육용 게임 프로그램

건우는 어린이 날을 맞이하여 유독 산수에 약한 초등학생 동생을 위한 교육용 게임 프로그램을 만들어 선물해 주기로 했다. 프로그램은 두 가지 게임으로 구성되어 있는데, 하나는 간단한 덧셈, 뺄셈 및 곱셈 문제를 해결하는 산수 게임 프로그램이고 다른 하나는 컴퓨터의 비밀 키를 맞추는 추측 게임이다. 프로그램은 사용자에게 다음 <그림 1>과 같은 메뉴를 보여주고 사용자 선택에 따라서 아래에 기술된 기능대로 동작한다. 구현 방법은 문제에 주어진 "**구현 요구사항**"을 만족하도록 구현한다.

```
1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요:
```

그림 1. 사용자 선택 메뉴 예

(1) 사용자가 메뉴 1)을 선택하면, 프로그램은 다음 <그림 2>와 같이 수행할 연산의 종류를 사용자에게 제시한다.

```
    1) 더하기 할래?
    2) 빼기 할래?
    3) 곱하기 할래?
    메뉴를 선택하세요:
```

그림 2. 산수 문제 세부 메뉴 예

사용자가 메뉴를 선택하면, 사용자로부터 연산에 사용할 최대 정수 값을 입력 받는다. 사용자가 입력한 정수 값을 max라고 했을 때, 프로그램은 1에서 max사이 (1과 max포함)의 임의의 난수를 선택하여 총 다섯 번의 서로 다른 문제를 제시한다. 처음 문제 출제시간부터, 사용자가 다섯 개의 연산 문제를 모두 해결하는 데 소요된 시간을 체크하여 총 소요 시간에 따라서 점수를 배점한다. 오답인 경우에는 5초의 벌점 시간이 추가된다. 배점 기준은 문제에 주어진 "**배점 기준**"을 따른다.

(2) 사용자가 메뉴 2)를 선택하면, 프로그램은 사용자로부터 추측 게임에 사용할 최대 정수 값 (max)를 입력 받는다. (1)에서와 마찬가지로, 프로그램은 1에서 max사이의 임의의

난수를 생성하여 컴퓨터 비밀 값으로 선택한 후, 사용자에게 컴퓨터 비밀 값이 무엇인지 추측하도록 한다. 사용자가 값을 입력할 때마다, 프로그램은 컴퓨터 비밀 값에 근접할 수 있도록 "입력 값이 너무 큽니다" 또는 "입력 값이 너무 작습니다"와 같은 힌트를 제공한다. 이 과정은 사용자가 정답을 맞출 때까지 계속 반복된다. 사용자가 정답을 입력할 때까지 소요된 시간에 따라서 점수를 부여한다. 배점 기준은 문제에 주어진 "**배점 기준**"을 따른다.

(3) 사용자가 3을 선택하면 지금까지 누적된 총 점수를 출력한다.

(4) 사용자가 4을 선택하면 프로그램은 종료한다.

(5) 사용자가 메뉴 선택 시, 사용자 입력 값에 대한 예외 처리를 수행한다. 즉, 메뉴 범위 밖의 값을 입력하면, 메뉴를 다시 보여주고 재 입력 받는다.

배점 기준

1) 산수가 좋아: 점수 산정에 필요한 시간 T

T = (다섯 개의 연산을 모두 해결하는데 걸린 총 소요 시간 + 총 벌점 시간) / 5

T에 따른 점수 (score)는 다음 <표 1>에 따라서 산정된다.

2) 비밀을 맞춰봐: 점수 산정에 필요한 시간 T

T = 정답을 입력할 때까지 걸린 총 소요 시간 / (2 * 최대 정수 값(max)의 자리 수)

예를 들어서 사용자가 입력한 최대 정수 값이 100이고, 총 15초가 걸렸다면, T = 15/(2*3) = 2.5초이고 점수는 8점이다. T에 따른 점수 (score)는 다음 <표 1>에 따라서 산정된다.

표 1. 시간에 따른 점수 환산 표

시간 T	점수
T < 1	10
1 ≤ T < 2	9
2 ≤ T < 3	8
3 ≤ T < 4	7
4 ≤ T < 5	6
5 ≤ T < 6	5
6 ≤ T < 7	4
7 ≤ T < 8	3
8 ≤ T < 9	2
9 ≤ T < 10	1
T ≥ 10	0

구현 요구사항

이번 과제는 **다중 소스 파일을 생성하여 작성**하며, **문제에서 주어진 함수 및 요구사항은 반드시 문제에 기술된 대로 구현**한다. 가급적이면 기능 별로 독립된 함수로 구현한다. 따라서, 문제에서 명시한 함수 이외에, 필요에 따라서 다양한 사용자 정의 함수를 자유롭게 추가하여 구현한다.

1) **사용자 정의 헤더파일 사용: my.h**
   프로그램 실행에 필요한 모든 전처리 과정 (헤더파일 포함) 및 사용자 정의 함수에 대한 함수 원형 선언을 포함한다.
2) **main() 함수**
   - **pa2.c 파일**에 작성한다.
   - <그림 1>에 있는 메뉴를 화면에 출력하고, 사용자 입력을 받은 후, 해당하는 게임을 수행할 수 있는 함수를 호출한다.
   - 사용자는 숫자만 입력한다고 가정하고, 숫자 입력에 대한 예외 처리를 포함한다. 즉, 1 ~ 4 이외의 입력 값이 주어지면, 메뉴를 다시 보여주고 재입력 받는다.
   - 사용자가 **1을 선택하면 arithGame() 함수가 호출**되고, **2를 선택하면 findSecret() 함수가 호출**된다.
   - 사용자가 지금까지 획득한 **총 누적 점수**는 **전역변수**로 선언한다.
   - 프로그램은 사용자가 4을 입력하기 전까지 계속해서 **반복적으로 수행**된다.
3) **`void arithGame()`**
   - **arithGame.c 파일**에 작성한다.
   - <그림 2>에 있는 메뉴를 화면에 출력하고, 사용자 입력을 받은 후, 해당 연산을 수행할 수 있는 함수를 호출한다. 각 연산에 대한 함수는 자유롭게 정의하여 사용한다.
   - 사용자는 숫자만 입력한다고 가정하고, 숫자 입력에 대한 예외 처리를 포함한다. 즉, 1 ~ 3 이외의 입력 값이 주어지면, 메뉴를 다시 보여주고 재입력 받는다.
   - 사용자 선택에 따른 연산 함수를 호출하기 전에, 해당 연산에 사용될 숫자의 범위를 결정하기 위한 최대 정수 값(max)를 입력 받는다. **최대 정수 값(max)는 반드시 1이상의 값이어야 하며, 1보다 작은 값이 주어지면 재입력** 받는다.
   - 각 연산 함수는 **총 다섯 세트의 연산 문제를 제시**한다. 이 때, 문제에 사용되는 숫자는 **1에서 max사이의 숫자 (1과 max 포함)으로 랜덤하게 구성**된다.
   - 사용자가 오답을 입력했을 경우에는 "오답입니다. 정답은 ~입니다. 벌점 5초가 추가됩니다"와 같은 안내 문구를 출력하고 오답 횟수를 카운트한다.
   - 다섯 문제를 모두 풀면 게임이 종료되고, 다섯 개의 연산을 수행하는 데 소요된 총 시간과 오답에 따른 벌점 시간을 합한 후, **"배점기준"**에 명시된 대로 **점수 배점에 필요한 시간(T)**을 계산하고, 현재 수행한 게임에 대한 score를 계산한다.
   - "문제당 평균 ~초의 시간이 소요되었으며, 이번 게임의 점수는 ~입니다"와 같이 T의 값과 score 값을 출력한다.
   - 최종적으로 누적 점수에 현재 score를 추가하여 누적 점수를 갱신한 후, 게임을 종료한다.
4) **`void findSecret()`**
   - **findSecret.c 파일**에 작성한다.
   - 컴퓨터 비밀 값은 랜덤하게 할당되는데, 먼저 사용자로부터 해당 게임에 사용될 숫자의 범위를 결정하기 위한 최대 정수 값(max)를 입력 받는다. **최대 정수**

값(max)는 반드시 1이상의 값이어야 하며, 1보다 작은 값이 주어지면 재입력 받는다.

- 사용자가 입력한 max값을 기준으로, 컴퓨터 비밀 값은 **1에서 max사이의 숫자 (1과 max 포함)으로 랜덤하게 구성**된다.
- 사용자가 컴퓨터 비밀 값을 유추할 때까지, 사용자 입력을 반복해서 입력받는다.
- 총 시도 회수를 카운트하고, 사용자 입력 값에 대해서 "입력 값이 너무 큽니다" 또는 "입력 값이 너무 작습니다"와 같은 힌트를 제공한다.
- 사용자가 정답을 입력하면 게임이 종료되고, "정답입니다. 축하합니다. 총 ~초 동안, ~번의 시도로 성공하였습니다"와 같이 총 소요 시간과 시도 횟수를 출력한다.
- 정답을 유추할 때까지 소요된 총 시간과 컴퓨터 비밀 값 생성에 사용된 최대 정수 값의 자릿수를 고려하여, **"배점기준"**에 명시된 대로 **점수 배점에 필요한 시간(T)**을 계산하고, 현재 수행한 게임에 대한 score를 계산한다.
- 최종적으로 누적 점수에 현재 score를 추가하여 누적 점수를 갱신한 후, 게임을 종료한다.

기타 프로그램 구현에 유용한 함수들

- 난수 생성 시, 난수 초기 값 랜덤 설정: `srand(time(0));`
- 소요 시간 계산

```
int start, end, timespent;
start=time(0);
// 프로그램 코드
end = time(0);
timespent = end - start;
```

실행 예

다음은 프로그램 실행 예제이다. 다음의 예는 한 경우에 대한 실행 예로, 아래에 주어진 예뿐만이 아니라, 여러 다양한 데이터들에 대한 테스트가 필요하다.

```
1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요: 7

1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요: 1

1) 더하기 할래?
2) 빼기 할래?
3) 곱하게 할래?
```

```
메뉴를 선택하세요: 8

1) 더하기 할래?
2) 빼기 할래?
3) 곱하게 할래?
메뉴를 선택하세요: 1

게임에 사용할 최대 양의 정수 값 (1 이상)을 입력하시오: -3

게임에 사용할 최대 양의 정수 값 (1 이상)을 입력하시오: 0

게임에 사용할 최대 양의 정수 값 (1 이상)을 입력하시오: 10
2 + 8 = 10
5 + 1 = 6
10 + 5 = 3
오답입니다. 정답은 15 입니다. 벌점 5초 추가됩니다.
9 + 9 = 18
3 + 5 = 8

문제당 평균 3.00 초의 시간이 소요되었으며, 이번 게임의 점수는 7 입니다.

1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요: 2

게임에 사용할 최대 양의 정수 값 (1 이상)을 입력하시오: 0

게임에 사용할 최대 양의 정수 값 (1 이상)을 입력하시오: 10
비밀 값을 추측해 보세요: 8
입력 값이 너무 큽니다.
비밀 값을 추측해 보세요: 5
입력 값이 너무 작습니다.
비밀 값을 추측해 보세요: 7
입력 값이 너무 큽니다.
비밀 값을 추측해 보세요: 6
정답입니다.
축하합니다. 총 7 초 동안 4 번의 시도로 성공하였습니다.
이번 게임의 점수는 9 입니다.

1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요: 3
지금까지의 점수는 16 입니다.
```

```
1) 산수가 좋아
2) 비밀을 맞춰봐
3) 점수 보기
4) 끝내기
메뉴를 선택하세요: 4
```

제출 사항

1. 소스 파일 및 헤더파일: my.h, pa2.c, arithGame.c, findSecret.c
   - 모든 소스 파일 명은 반드시 문제 정의에 주어진 대로 한다.
   - 모든 소스 파일은 프로그램의 이해를 돕기 위해서 주요 변수 및 함수에 대한 적절한 주석을 포함하여 하며, 소스 파일의 최 상단에는 학번, 이름, 교과목명, 과제명, 작성일 등의 정보를 프로그램 서두에 주석으로 포함한다.
2. 보고서: PA2.docx (또는 PA2.hwp)

보고서는 다음의 내용을 포함하여야 한다.

- 표지: 과제명, 이름, 학번, 제출일 포함
- 다음의 내용을 순차적으로 작성한다.
  - **문제 정의**: 문제 정의를 기술
  - **주요 변수 설명**: 본 과제를 해결하기 위해 선언한 주요 변수에 대해서 설명
  - **Idea 또는 알고리즘**: 문제 해결을 위해 제안한 아이디어 (또는 알고리즘)을 기술
  - **수행 결과**
    - ▸ 다양한 사용자 입력에 대해서 프로그램의 정확성 테스트
    - ▸ 수행 결과를 보고서에 첨부
  - **토의 사항**: 본 과제를 수행하면서 발견한 문제점, 새롭게 발견한 사실 및 토의 사항 등을 기술

3. 최종 제출파일: (9자리학번-이름)PA2.zip

모든 소스 파일과 헤더 파일 및 보고서를 (9자리학번-이름)PA2.zip 파일로 압축하여, (9자리학번-이름)PA2.zip파일을 eCampus 과제함에 제출한다.

주의 사항

- 주어진 문제는 "제어문 및 함수"까지 다룬 기법으로 해결하며, 배열과 포인터 등 아직 배우지 않은 내용을 포함하여 프로그램을 작성하여서는 안 된다.
- 어떤 C 컴파일러를 사용해도 무방하나, ANSI C 표준에 맞게 구현되어야 한다. 본인이 사용하는 컴파일러의 확장성으로 인해 문법적 오류가 발생하지 않는다 하더라도, ANSI C 문법에 어긋나는 경우는 에러가 발생하는 것으로 간주한다

주요 평가 항목

- 구현 부분 (80%)
  - 적절한 제어문 및 함수의 활용
  - 사용자 입력에 대한 적절한 예외 처리
  - 문제에서 주어진 요구사항 수행
  - 실행 결과의 정확성
  - 프로그램의 반복 수행
  - 적절한 주석 처리 및 가독성을 제공하는 코딩 스타일 (띄어쓰기, 들여쓰기 등)
- 성실한 보고서 작성 (20%)