2021-1 C++프로그래밍 실습과제 03

학번	2020136129	이름	최수연

(1) 각 문제에 대한 분석과 및 해결 방법

1. 3.8절의 러시안 룰렛 게임을 다음과 같이 확장하라.

(1) 6연발 권총이 아니라 n-연발 권총이다. n은 사용자로부터 입력받는다.
(2) 모든 총알이 발사될 때까지 게임을 진행한다. 예를 들어, 5명이 2발의 총알을 넣어 게임을 한다면 두 명이 총알을 맞는다.

[문제분석 및 해결방법]

1. 메인 함수

먼저 (1), (2)의 조건에 맞춰 메인 함수에 게임 인원, 탄창 구멍의 개수, 총알 개수를 모두 입력받는다. 단, 총알 개수는 탄창 구멍의 개수보다 작아야 한다는 조건을 지켜야 한다. 이들은 모두 매개변수로 게임 함수에 전달된다. 러시안 룰렛 메인 함수에서는 본 게임에 필요한 난수 생성을 위한 srand() 함수를 선언한다. 또한, playRussianRoulette() 함수를 다른 소스 파일에서 가져와 전방 선언한다. 메인 함수에서 실제 러시안 룰렛 게임이 진행되는 함수 playRussianRoulette()로 넘어가도록 한다.

2. 게임 함수

러시안 룰렛 게임 함수 playRussianRoulette()에서는 메인함수에서 입력 받은 게임 인원을 rand() 함수를 통해 난수를 하나 생성하여 시작하는 사람을 무작위로 정한다. 시작하는 사람이 정해지면, 반복문을 통해 총알이 0발이 될 때까지 게임을 반복한다. 격발 위치를 탄창 구멍의 개수에 맞게 난수를 생성하고 이 난수가 총알 개수보다 작으면 해당 순서에 있는 사람은 총에 맞게 된다. 이후 게임 인원과 총알 개수를 하나씩 줄여가면서 총알 개수가 0발이 될 때까지 반복하고 0발이 되면 게임을 종료한다. 이때 다음 사람의 순서는 맨 처음 무작위로 결정된 시작하는 사람의 번호에서 현재 남은 인원을 나눈 나머지 값으로 결정된다.

3. 헤더 파일

러시안 룰렛 헤더 파일에는 rand()함수와 time()함수를 사용하기 위한 라이브러리 헤더 파일을 포함하고, playRussianRoulette() 함수를 외부에도 공개하기 위해 extern으로 선언한다. 이때, playRussianRoulette() 함수에서 디폴트 매개변수를 사용하여 게임 인원, 탄창 구멍의 개수, 총알 개수에 대한 기본 값을 지정해둔다.

2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.

(1) 여러 자리의 덧셈 문제를 출제하고 맞히는 함수를 playGuguOnce()를 참고하여 작성하라.
(2) 마찬가지로 한 자리 수 곱셈 문제를 함수로 구현하라. ***(스피드 구구단과 동일)***
(3) 프로그램이 시작되면 게임을 선택하도록 한다. 예를 들어, 1: 구구단, 2: 한 자리 수 곱셈, 3~9: 3~9자리 수 덧셈 문제가 선택된다.

[문제분석 및 해결방법]

1. 메인 함수

먼저 본 게임에 필요한 난수를 생성하기 위해 srand() 함수를 선언한다. 문제 수는 난이도 조정으로 3문제로 고정하고, 게임 목록 보기를 출력하여 원하는 게임의 번호를 입력받는다. 1을 입력받으면 스피드 구구단(한 자리 수 곱셈) 게임 함수로 지정되고, 3~9 중 한 숫자를 입력받으면, 입력 받은 숫자끼리의 자리 수 덧셈 게임 함수로 지정된다. 엔터를 누르면 게임이 시작되고, 시작되는 동시에 콘솔 화면을 지우고 입력 받은 숫자에 대한 게임 함수로 이동한다. 게임 함수에서 게임이 진행 및 종료되면, 게임 함수를 통해 반환 받은 점수와 소요 시간을 메인 함수에서 화면에 출력하도록 한다.

2. 게임 함수

게임 함수에서는 static을 사용하여 해당 파일에서만 전역변수처럼 사용할 수 있도록 NumGames(진행된 게임 횟수), NumWins(이긴 횟수), Score(점수) 총 3개의 변수를 지정한다. 이외에 tElapsed 변수도 선언하는데 이는 외부 파일에도 사용되므로 static을 사용하면 안 된다. 이때 위 변수들은 모두 0으로 초기화한다.

먼저 스피드 구구단(한 자리 수 곱셈) 게임의 경우, playGuguOnce() 함수에서 정수형6 변수 a와 b를 무작위로 돌려 2~9사이의 수로 난수를 하나씩 생성한다. 단, 숫자 1은 스피드 구구단에서 제외하였다. 자료형 bool을 사용하여 곱셈 결과를 입력받고, 입력받은 결과 값이 a*b가 맞으면 참(true), 아닐 경우 거짓(false)으로 반환한다. playSpeedGugu() 함수에서는 문제를 푸는 동안의 걸린 소요시간을 측정하고, 매개변수를 통해 게임 문제 수(nPlay)를 메인 함수에서 가져온다. 반복문과 조건문을 이용하여 playGuguOnce()에서 반환한 값이 거짓이면 값이 틀렸다고 화면에 출력한다. 점수 계산 방법의 경우 한 문제라도 틀리면 0점을 반환하고, 문제를 모두 맞췄을 경우, [100 * (5.0초 * 게임 문제 수 – 소요 시간) / 5.0초 * 게임 문제 수] 식을 사용하여 점수를 반환하여 메인 함수로 이동한다.

3~9자리 수 덧셈 게임의 경우, playSumOnce() 함수에서는 메인함수에서 입력받은 digit을 사용하여 digit의 자리 수를 가진 정수형 변수 a와 b를 선언한다. 스피드 구구단 함수에서와 같이 자료형 bool을 사용하여 덧셈 결과를 입력받고, 입력받은 결과 값이 a+b가 맞으면 참(true), 아닐 경우 거짓(false)으로 반환한다. playSpeedSum() 함수에서는 문제를 푸는 동안의 걸린 소요시간을 측정하고, 매개변수를 통해 게임 문제 수(nPlay)를 메인 함수에서 가져온다. 반복문과 조건문을 이용하여 playSumOnce()에서 반환한 값이 거짓이면 값이 틀렸다고 화면에 출력한다. 점수 계산 방법의 경우 한 문제라도 틀리면 0점을 반환하고, 문제를 모두 맞췄을 경우, [100 * (20.0초 * 게임 문제 수 – 소요 시간) / 20.0초 * 게임 문제 수] 식을 사용하여 점수를 반환하여 메인 함수로 이동한다.

3. 헤더 파일

스피드 구구단(한 자리 수 곱셈)과 3~9자리 수 덧셈 헤더 파일에는 rand()함수와 time()함수를 사용하기 위한 라이브러리 헤더 파일을 포함하고, tElapsed 변수와 playSpeedGugu() 함수, playSpeedSum() 함수를 extern 선언하여 메인 함수에도 보일 수 있도록 설정한다.

(2) 자신이 구현한 주요 코드

if (pos < nBullets)
{
printf("\t빵~~~~~~~~~~%d번이 총에 맞았습니다!!!\n", start + 1);
printf("\t이제 남은 사람은 %d명이고 총알은 %d발입니다.\n\n", nTurns - 1, nBullets - 1);
nTurns--;
nBullets--;
}
else printf("\t휴~~~ 살았습니다!!!\n\n");
	무작위로 생성된 pos(격발 위치)가 nBullets(총알의 개수)보다 작으면, 예를 들어, nBullets가 3일 때, pos가 0~2까지의 숫자가 걸리면 해당 순서의 사람은 총에 맞는다. 따라서 nTurns - 1, nBullets - 1로 게임 인원과 총알 개수가 모두 하나씩 줄어든다. 또한, 전체 반복문의 게임 인원과 총알 개수 모두 하나씩 감소해야 한다. 만약 if (pos < nBullets)가 아니면 해당 순서의 사람은 총에 맞지 않고 바로 다음 턴으로 진행되며 게임 인원과 총알 개수 모두 그대로 남아 다음 턴이 진행된다.
	총알 개수가 하나씩 줄어들수록 pos가 nBullets보다 작을 확률도 숫자 하나씩 줄어든다.

if (nBullets == 0) break;	러시안 룰렛 게임이 진행되는 반복문에서 총알 개수가 0발이 되면 더 이상 총에 맞을 수 있는 인원이 없기 때문에 자동으로 게임을 종료해야한다.

int num = 1;
for (int i = 0; i < digit; i++)
num *= 10;
int a = rand() % (num - num / 10) + num / 10;
int b = rand() % (num - num / 10) + num / 10;
	메인 함수에서 3~9자리 수중에 입력받은 수를 게임 함수의 매개변수 digit으로 받아 for문을 통해 digit 번만큼 1로 초기화된 변수 num에 10을 곱한다.
	만약 digit가 3이라면 3자리 수 덧셈이므로 for문을 통해 num을 1000으로 만든다. 이때, rand() % (num - num / 10) + num / 10 식을 통해, rand() % (1000 - 100) + 100 = rand() % 900 + 100이 되고, 이는 0에서 899까지의 범위에 100을 더한 값이 되므로 정수형 변수 a와 b의 총 범위는 100에서 999까지의 범위로 설정된다. 이처럼 3~9사이의 digit의 값에 따라 a와 b의 범위는 digit의 자리 수로 설정된다.

Score = (NumGames > NumWins) ? 0.0 : 100 * (20.0 * NumGames - tElapsed) / (20.0 * NumGames);	3~9자리수를 덧셈하는 것은 스피드 구구단보다 난이도가 있기 때문에 3문제로 바꿈에도 불구하고 100 * (5.0 * NumGames - tElapsed) / (5.0 * NumGames); 위 식으로 점수를 계산하였을 때 덧셈하는 데에 시간이 많이 걸려 결과 값이 쉽게 마이너스가 된다. 따라서 마이너스가 쉽게 나오지 않도록 점수 계산법에 변화를 주었다. 100 * (20.0 * NumGames - tElapsed) / (20.0 * NumGames); 위 식과 같이 한 문제당 5.0초를 20.0초로 바꾸어 3문제를 풀 때 1분(60초)안으로 풀면 점수 값이 양수로 나올 수 있도록 하였다.

(3) 다양한 입력에 대한 테스트 결과

1. 3.8절의 러시안 룰렛 게임을 다음과 같이 확장하라.

```
게임 인원 (예:2) ==> 3
탄창 구멍 개수 ==> 6
총알 개수 (6이하) ==> 2

총을 돌렸습니다. 2번부터 시작합니다.
[2번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[3번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[1번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[2번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[3번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        빵~~~~~~~~~~3번이 총에 맞았습니다!!!
        이제 남은 사람은 2명이고 총알은 1발입니다.

[2번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[1번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[2번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        휴~~~ 살았습니다!!!

[1번]   탄창을 무작위로 돌렸습니다.
        엔터를 누르면 격발됩니다...
        빵~~~~~~~~~~1번이 총에 맞았습니다!!!
        이제 남은 사람은 1명이고 총알은 0발입니다.

        게임 종료
```

2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.

```
[문제 1]:  8 x  9 = 72
[문제 2]:  4 x  8 = 32
[문제 3]:  9 x  9 = 81

점수 = 66.2점(총  5.1초)
```

```
[문제 1]: 779 + 611 = 1390
[문제 2]: 996 + 400 = 1396
[문제 3]: 462 + 584 = 1046

점수 = 69.6점(총 18.2초)
```

```
[문제 1]: 5557 + 3921 = 9478
[문제 2]: 5822 + 1565 = 7387
[문제 3]: 4533 + 2176 = 6709

점수 = 63.9점(총 21.7초)
```

```
[문제 1]: 14749 + 24485 = 39234
[문제 2]: 29233 + 15509 = 600000
        틀렸습니다.
[문제 3]: 26501 + 22477 = 450000
        틀렸습니다.

점수 =  0.0점(총 21.7초)
```

(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

(없음.)

(5) 이번 과제에 대한 느낀점

작년에 C프로그래밍을 배우면서 한 프로젝트 파일에 소스 파일 여러 개를 두고 하나의 헤더파일을 따로 사용하는 방법을 한번 배운 적이 있는데, 이번 과제를 하기 전까지만 해도 제대로 기억이 나지 않았다. 그래서 한 프로젝트에 소스 파일을 여러 개 두면 자꾸 발생하는 링크 오류를 제대로 해결하지 못한 채, 계속 프로젝트를 여러 개 만들어 번거롭게 코딩을 하였다. C++ 첫 번째 과제에서도 그랬다. 하지만 이번 과제를 통해 링크 오류가 나는 이유와 헤더 파일을 따로 분리하는 법에 대한 정확한 개념을 잡을 수 있었다. 또, 지난여름방학 때 고등학교 친구들과 코딩스터디를 하면서 비전문가들끼리 질의응답하고 스스로 검색하며 코딩을 공부할 때는 알지 못했던 새로운 C++에 대한 개념들도 체계적으로 배우고, 이를 사용하는 법도 이번 과제를 통해 제대로 학습할 수 있어 좋았다. 그리고 평소 헷갈렸던 난수 발생 함수도 본 과제를 통해 게임도 만들고 보고서도 써보면서 개념을 제대로 정리할 수 있어서 이 또한 오래 기억에 남을 것 같다.

(6) 궁금한 점이나 건의사항

딱히 없습니다.