2021-1 C++프로그래밍 실습과제 03

(1) 각 문제에 대한 분석과 및 해결 방법

3.1. 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] : 3.8절의 러시안 룰렛 게임을 문제의 명세대로 확장하기 위해서 탄창 구멍 개수를 추가적으로 입력받아 격발하여 맞을 확률을 구하는데 사용하고, 모든 총알이 소모되거나 모든 사람이 죽은 경우 게임을 끝내기 위해 조건 분기를 추가하여 문제를 해결하였다.

3.2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.

(1) 여러 자리의 덧셈 문제를 출제하고 맞히는 함수를 playGuguOnce()를 참고하여 작성하라.

(2) 마찬가지로 두 자리 수 곱셈 문제를 함수로 구현하라.

(3) 프로그램이 시작되면 게임을 선택하도록 한다. 예를 들어, 1: 구구단, 2: 두 자리 수 곱셈, 3~9: 3~9자리 수 덧셈 문제가 선택된다.

[문제분석 및 해결방법] : 3.8절의 스피드 구구단 게임을 문제의 명세대로 확장하기 위해서 playGuguOnce()를 참고하여 playSpeedGugu2Once(), playSpeedSumOnce(int game)을 구현하고 의미없이 반복되는 코드들을 매크로, 함수 등을 이용하여 리팩토링하여서 함수를 최적화하였다. 또한 외부 호출로 사용하지 않는 함수들을 static으로 정의하여 불필요한 함수들의 외부 참조를 막았다.

(2) 자신이 구현한 주요 코드

3.1. 3.8절의 러시안 룰렛 게임을 다음과 같이 확장하라.

```
void playRussianRoulette(int nTurns, int nBullets, int nMax) {
    int start = rand() % nTurns;
    printf("₩n총을 돌렸습니다. %d번부터 시작합니다.₩n", start +1);
    while (true) {
        int pos = rand() % nMax;
        printf("[%d번]₩t탄창을 무작위로 돌렸습니다.₩n", start +1);
        printf("₩t엔터를 누르면 격발됩니다...");
        getchar();
        if (pos < nBullets) {
            printf("₩t빵~~~~~~~~~~%d번이 총에 맞았습니다!!!₩n", start +1);
            printf("이제 남은 사람은 %d명이고 총알은 %d발입니다.₩n“,
--nTurns, --nBullets);
            start++;
            if (!nTurns ||!nBullets) {
                break;
            }
        }
        else {
            printf("₩t휴~~~ 살았습니다!!!₩n");
        }
        start = (start +1) % nTurns;
    }
}
```

3.2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.

<SpeedGugu.cpp>

```
static void startClock() {
    t0 = clock();
}

static double getScoreWithEndClock() {
    t1 = clock();
    tElapsed = (double)(t1 - t0) / CLOCKS_PER_SEC;
    Score = (NumGames > NumWins) ? 0.0 : 100 * (5.0 * NumGames -
tElapsed) / (5.0 * NumGames);
    return Score;
}

...

static bool playSpeedGugu2Once() {
    int a = rand() % 90 +10;
    int b = rand() % 90 +10;
    int result;
    NumGames++;
    printf("[문제%2d]: %2d x %2d = ", NumGames, a, b);
    scanf("%d", &result);
    if (result == a * b) NumWins ++;
    return (result == a * b);
}

static bool playSpeedSumOnce(int game) {
    int unit = pow(10, game -1);
    int a = BIG_RAND() % (unit *9) + unit;
    int b = BIG_RAND() % (unit *9) + unit;
    int result;
    NumGames++;
    printf("[문제%2d]: %d + %d = ", NumGames, a, b);
    scanf("%d", &result);
    if (result == a + b) NumWins ++;
    return (result == a + b);
}
```

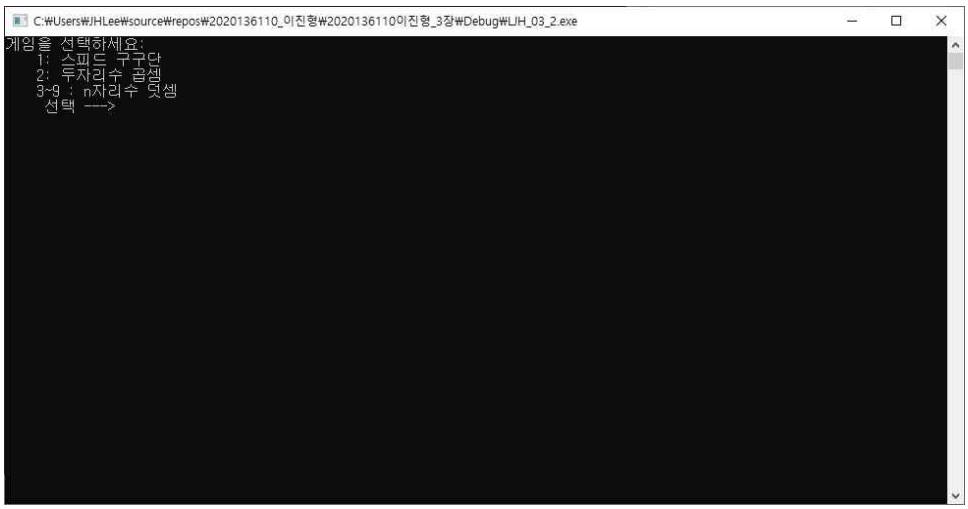
<SpeedGugu.h>

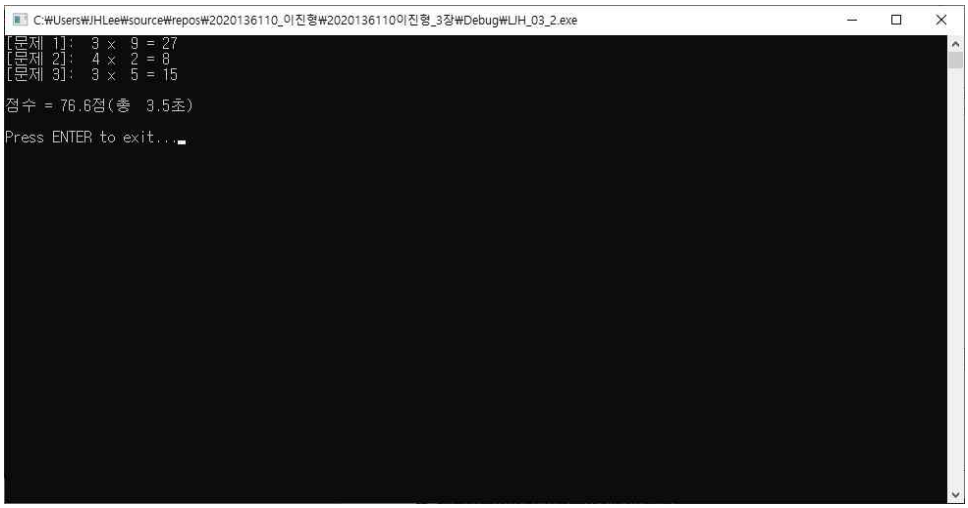
```
#define BIG_RAND() (((long) rand() <<15) | rand())
#define START_GAME(check)  {                ₩
    startClock();                           ₩
    for (int i =0; i < nPlay; i ++) {       ₩
    if (check ==false)                      ₩
        printf("₩t틀렸습니다.₩n");          ₩
    }                                       ₩
    return getScoreWithEndClock();          ₩
}
```

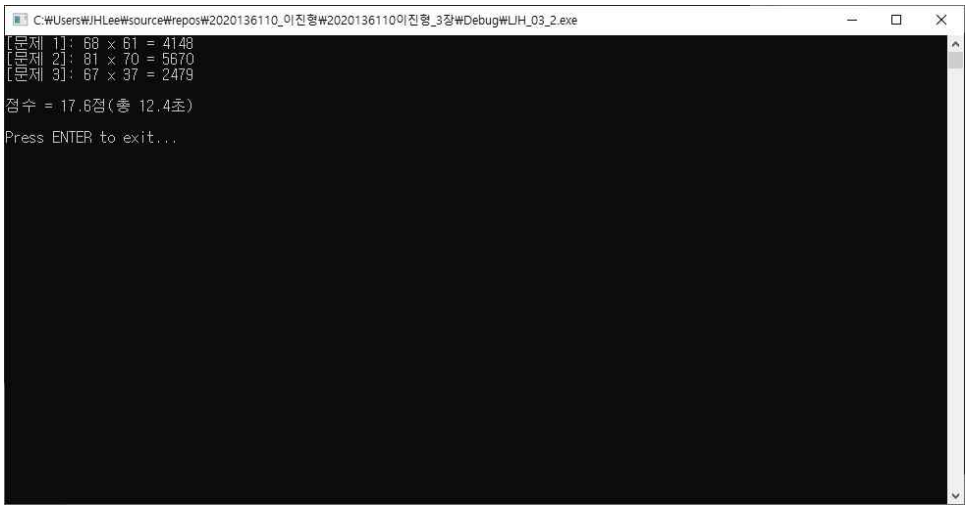
(3) 다양한 입력에 대한 테스트 결과

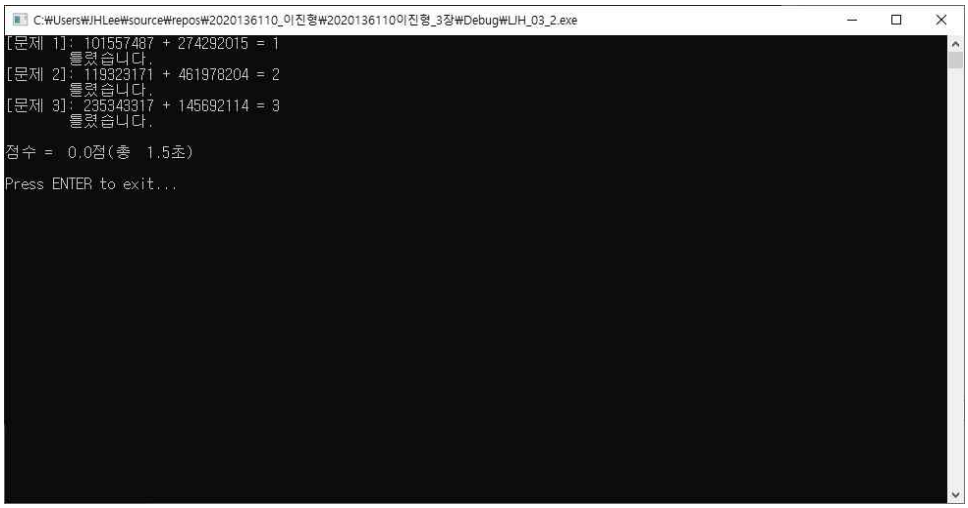
3.1. 3.8절의 러시안 룰렛 게임을 다음과 같이 확장하라.

3.2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.









(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

*** 3.1의 경우 위에서 설명한 것이 전부입니다!**

3.2. 3.8절의 스피드 구구단 게임을 다음과 같이 확장하라.

```
int a = rand() % 90 +10;
int b = rand() % 90 +10;
```

두자리수 곱셈 문제를 구현하기 위해서 10 ~ 99의 수를 표현하기 위해 기존의 playGuguOnce 함수의 코드에서 a, b 변수의 초기화를 rand() % 90 + 10으로 수정하여 playSpeedGugu2Once를 구현하였다.

<SpeedGugu.cpp>

```
static bool playSpeedSumOnce(int game) {
    int unit = pow(10, game -1);
    int a = BIG_RAND() % (unit *9) + unit;
    int b = BIG_RAND() % (unit *9) + unit;
    int result;
    NumGames++;
    printf("[문제%2d]: %d + %d = ", NumGames, a, b);
    scanf("%d", &result);
    if (result == a + b) NumWins ++;
    return (result == a + b);
}

double playSpeedGugu(int nPlay) {
    START_GAME(playGuguOnce());
}

double playSpeedGugu2(int nPlay) {
    START_GAME(playSpeedGugu2Once());
}

double playSpeedSum(int nPlay, int game) {
    START_GAME(playSpeedSumOnce(game));
}
```

<SpeedGuGu.h>

```
#define BIG_RAND() (((long) rand() << 15) | rand())
#define START_GAME(check)  {                ₩
    startClock();                           ₩
    for (int i =0; i < nPlay; i ++) {       ₩
    if (check ==false)                      ₩
        printf("₩t틀렸습니다.₩n");          ₩
    }                                         ₩
    return getScoreWithEndClock();          ₩
}
```

3~9자리수 덧셈 문제를 구현하기 위해서도 같은 코드를 활용할 수 있었지만 rand() 함수의 반환 범위가 0부터 32767(RAND_MAX)까지이므로 5자리 이상의 숫자를 만들기에는 역부족이었다. 그래서 랜덤 함수를 확장하여 더 큰 수를 지원하도록 만드는 것이 필요했는데, rand() 함수가 [0, 2^15-1] 값을 가진다는 것을 이용해서 두 번의 rand() 함수 반환 값을 이용한, 시프트 연산과 OR 비트 연산을 사용하여 최대 30 비트의 반환 난수 범위 값을 가지는 확장 난수 매크로 함수를 만들어 사용하였다. 그리고 주어진 자릿수에 대한 올바른 범위의 a, b 변수 값을 초기화하기 위해서 pow 함수를 이용하여 (10^자릿수 - 1)의 값을 구한 뒤 Modulo 연산을 활용하여 a, b 변수를 초기화했다. 또한 반복되는 부분을 최대한 줄이고 코드의 가독성을 향상시키기 위해 START_GAME(check) 매크로 함수를 정의하여 사용하였다.

(5) 이번 과제에 대한 느낀점

3.2 문제에서 상대적으로 최신 언어들에서 지원하는 0~1 사이의 소수 난수를 반환하는 함수 없이 큰 자리의 난수를 생성하는 방법에 대해서 생각해 볼 수 있어서 좋았다.