2021-1 C++프로그래밍 실습과제 01

(1) 각 문제에 대한 분석과 및 해결 방법

2.1. 반복문을 이용하여 다음과 같은 패턴을 출력하는 프로그램을 작성하라. *202013611→0← => 0 % 3 + 2 = 2번 문제

```
5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

[문제분석 및 해결방법] : 각각의 줄에서 출력 개수가 줄어드는 것을 고려해서 외부 반복문(각 줄을 출력하는 루프), 내부 반복문(한 줄을 출력하는 루프)으로 구성된 중첩된 반복문을 통해 해당 패턴을 출력하는 프로그램을 작성하였다.

2.2. 정수 n을 입력받아 다음의 식을 이용하여 π의 근사값을 구하는 프로그램을 작성하라.

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + \frac{1}{2n-1} - \frac{1}{2n+1}\right)$$

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

[문제분석 및 해결방법] : 해당 식은 arctan(x)로 수렴하는 마드하바–라이프니츠 급수에 1을 대입한 후 4를 곱하여 π를 표현한 식으로 적당한 n까지 해당 급수의 값을 구하면 π의 근삿값을 얻을 수 있다. 해당 문제는 주어진 n에 대해서 1 ~ n까지 1/(2n-1) + 1/(2n+1)의 반복 덧셈을 구현하는 것으로 해결할 수 있으며 반복문을 통한 방법과 재귀함수를 이용한 방법으로 풀이할 수 있을 것으로 분석하고 재귀함수를 이용하여 반복 덧셈을 구현하였다. 반환형은 가능한 최대의 유효숫자를 사용하기 위해 long double로 결정하여 프로그램을 작성하였다.

2.3. 2.7절의 번호 맞히기 게임을 다음과 같이 확장하라.

(1) 임의의 자릿수의 숫자 맞히기 게임으로 확장하라. 이를 위해서는 여러 개의 숫자를 입력받고 엔터가 입력되면 정답을 만들어야 한다. scanf() 함수는 사용하지 않아야 하고, 숫자를 입력할 때 마다 '*'문자가 화면에 출력되도록 하라.

[문제분석 및 해결방법] : 임의의 자릿수 입력을 받고 비교를 하기 위해서는 사용할 자료형의 최대 크기를 고려해야 하는데 가 장 큰 정수형 자료형을 사용하면 최대 unsigned long long(0 ~ 18446744073709551615)을 사용할 수 있으나, 이는 20자리의 숫자까지밖에 표현할 수 없다. 따라서 입력받은 숫자를 문자열로 다루어 문제를 해결하기로 결정하였다. 동적할당(malloc, realloc)을 이용하여 getch() 함수의 반환값을 무제한으로 처리할 수 있게 처리하였고 문자열 비교를 통해 두 수의 대수관계를 판단할 수 있게 프로그램을 작성하였다.

(2) 자릿수가 많아지면 점수 계산 방법이 달라져야 할 것이다. 자릿수에 따른 점수 계산 방법을 설계해 보라.

[문제분석 및 해결방법] : 큰 자릿수의 게임을 할 수록 사용자는 한정된 시도 횟수에서 더 어려운 게임을 하게 되는 것이므로 기존 점수가 자릿수에 영향을 받아 매겨질 수 있도록 점수 계산 방법을 (남은 시도 횟수)*(자릿수)로 설계하였다.

(2) 자신이 구현한 주요 코드

2.1. 반복문을 이용하여 다음과 같은 패턴을 출력하는 프로그램을 작성하라.

```
for (int y =1; y <= n; y ++) {
    int x = n;
    for (; x > y; x --) {
        printf("%d ", x);
    }
    printf("%d₩n", x);
}
```

2.2. 정수 n을 입력받아 다음의 식을 이용하여 π의 근사값을 구하는 프로그램을 작성하라.

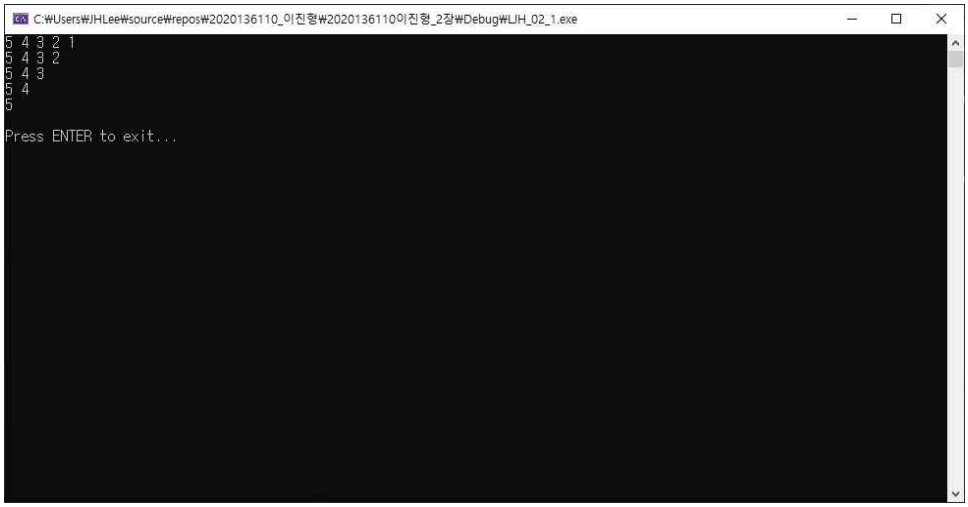
```
long double get_madhava_leibniz_series(int n, long double accumulator =0) {
    accumulator == 0 && n % 2 == 0 ? n-- : 0;
    long double value = accumulator +1.0L / (2 * n -1) -1.0L / (2 * n +1);
    return n > 1 ? get_madhava_leibniz_series(n - 2, value) : value;
}
```

2.3. 2.7절의 번호 맞히기 게임을 다음과 같이 확장하라.

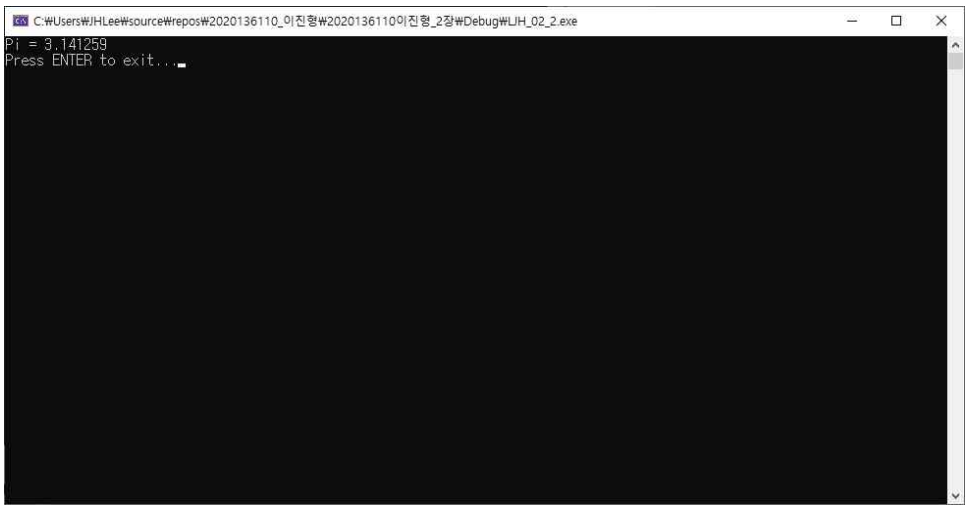
```
(1)
int compare(char * n1, char * n2) {
    int l1 = strlen(n1);
    int l2 = strlen(n2);
    return l1 < l2 ? -1 : (l1 > l2 ? 1 : strcmp(n1, n2));
}

char * next_bigint(bool use_secure) {
    char * buf = (char *)malloc(sizeof(char));
    int i =0;
    while (true) {
        char c = getch();
        if (('0'<= c && c <='9') &&!(i ==0 && c =='0')) {
            buf[i++] = c;
            buf = (char *) realloc(buf, sizeof(char) * (i + 1));
            use_secure ? printf("*") : printf("%c", c);
        }
        else if (c =='₩r' && i != 0) {
            buf[i] ='₩0';
            printf("₩n");
            return buf;
        }
    }
}

(2)
int nl = strlen(n);
...
for (; i <10; i ++) {
...
}
...
printf(" 최종 점수 = %d₩n", nl * (10 - i));
```

(3) 다양한 입력에 대한 테스트 결과

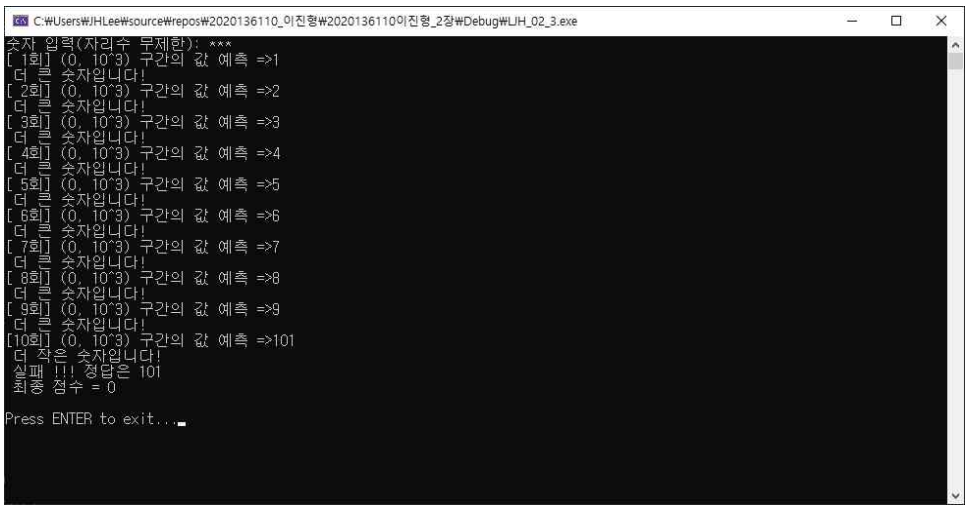
2.1. 반복문을 이용하여 다음과 같은 패턴을 출력하는 프로그램을 작성하라.

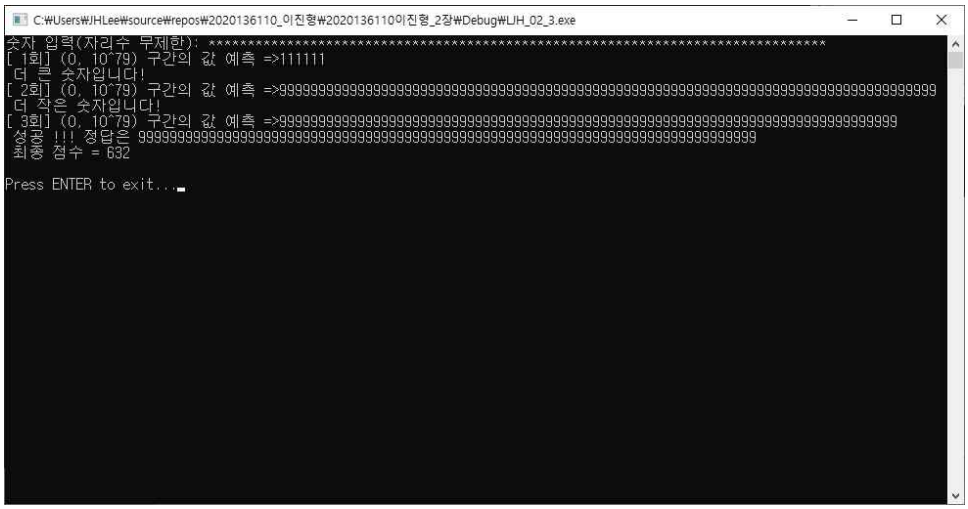


2.2. 정수 n을 입력받아 다음의 식을 이용하여 π의 근사값을 구하는 프로그램을 작성하라.



2.3. 2.7절의 번호 맞히기 게임을 다음과 같이 확장하라.





(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

*** 2.1, 2.2의 경우 위에서 설명한 것이 전부입니다!**

2.3. 2.7절의 번호 맞히기 게임을 다음과 같이 확장하라.

문자열 기반으로 확장된 번호 맞히기 게임을 만들기 위해서 숫자를 입력받는 입력 함수와, 두 숫자를 비교하는 비교 함수를 만들었다.

```
char * next_bigint(bool use_secure) {
    char * buf = (char *)malloc(sizeof(char));
    int i =0;
    while (true) {
        char c = getch();
        if (('0'<= c && c <='9') &&!(i ==0 && c =='0')) {
            buf[i++] = c;
            buf = (char *) realloc(buf, sizeof(char) * (i + 1));
            use_secure ? printf("*") : printf("%c", c);
        }
        else if (c =='₩r' && i != 0) {
            buf[i] ='₩0';
            printf("₩n");
            return buf;
        }
    }
}
```

<입력 함수>

메모리가 가능한 한 입력을 받기 위해 동적 할당 함수를 이용하여 추가 입력이 들어오는 경우를 처리했고 Enter 키를 친 경우에 해당 문자열 포인터를 반환하도록 했다. 최초 숫자 입력시의 * 처리와, 이후 숫자 입력시의 숫자 표시 처리를 위해 부울 매개변수로 use_secure를 받아서 해당 부분을 처리하였다.

```
int compare(char * n1, char * n2) {
    int l1 = strlen(n1);
    int l2 = strlen(n2);
    return l1 < l2 ? -1 : (l1 > l2 ? 1 : strcmp(n1, n2));
}
```

Dec	Hx	Oct	Html	Chr
32	20	040		Space
33	21	041	!	!
34	22	042	"	"
35	23	043	#	#
36	24	044	$	$
37	25	045	%	%
38	26	046	&	&
39	27	047	'	'
40	28	050	(	(
41	29	051	)	)
42	2A	052	*	*
43	2B	053	+	+
44	2C	054	,	,
45	2D	055	-	-
46	2E	056	.	.
47	2F	057	/	/
48	30	060	0	0
49	31	061	1	1
50	32	062	2	2
51	33	063	3	3
52	34	064	4	4
53	35	065	5	5
54	36	066	6	6
55	37	067	7	7
56	38	070	8	8
57	39	071	9	9

<비교 함수>

두 문자열 기반 자연수를 비교할 때 문자열의 길이가 큰 쪽이 항상 큰 수이고, 같은 경우에는 '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' 순으로 배치된 아스키 코드 배열을 고려할 때 strcmp 함수를 이용하면 숫자를 비교하는데 사용할 수 있는 점을 이용하여 n1 < n2일 때는 –1, n1 > n2일 때는 1, n1과 n2가 같은 경우에는 0을 반환하도록 처리하였다.

(5) 이번 과제에 대한 느낀점

2-3번 문제를 해결하면서 길이 제한 없는 두 수를 비교하기 위한 방법에 많은 생각을 할 수 있어서 좋았고 BigInt/BigInteger 클래스를 구현하거나 사용했다면 |가장 근접한 수 - 정답인 수|를 점수 계산식에 사용하는 등의 구현도 가능했을 것 같은데 점수 계산식을 자릿수를 남은 시도 횟수에 곱해주는 식으로만 구현하게 된 부분이 아쉬웠다.