2021-1 C++프로그래밍 실습과제 02

학번	2020136129	이름	최수연

(1) 각 문제에 대한 분석과 및 해결 방법

1. 반복문을 이용하여 다음과 같은 패턴을 출력하는 프로그램을 작성하라.

※ 2020136129 ==> 9 % 3 + 2 = 0 + 2 ==> (2)번 ※

(2)

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

[문제분석 및 해결방법]

먼저, 위의 패턴에 나오는 숫자는 모두 정수이므로 자료형 int를 사용한다. for문을 이용하여 i가 5에서 1씩 감소하면서 1까지 도달할 때, j는 5에서 5-i보다 큰 값까지 감소하면서 j를 출력한다. 따라서 i가 5일 때, j는 5부터 5-i=5-5=0보다 큰 값까지 즉, 1까지 감소하면서 출력된다. i값이 증가할수록 출력되는 j의 값의 범위는 점점 짧아지므로 숫자 j의 출력 값이 하나씩 줄어든다. 이때, j의 출력을 %2d로 설정하여 정수 간의 간격이 생기도록 설정한다. i값 하나에 의해 결정되는 j값이 for문에 의해 모두 출력되면 Enter(₩n)를 출력해주고, i값을 하나 감소하여 다시 위의 과정을 반복한다. i가 1보다 작아지면 모든 반복문을 빠져나온다.

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

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

[문제분석 및 해결방법]

위의 라이프니츠 원주율 공식에서 괄호 안의 분수들이 (+)와 (–)가 번갈아가며 더해지는 규칙을 보고, 이를 분류하여 (+)는 (+)끼리 (–)는 (–)끼리 각각 더하여 합하는 방식을 고안하였다. 위 식의 수는 모두 실수이기 때문에 자료형 double을 사용하였다. i를 분모로 두고, i가 4씩 증가하며 더해지도록 하였다. for문의 조건식 검사부에서 i의 범위는 (+)일 때는 2*n-1, (–)일 때는 2*n+1까지 증가할 수 있도록 하였다. π의 근삿값을 출력할 때, 위 식에 따라 4를 곱해주고 이 값을 %.6f로 설정하여 소수점 6번째 자릿수까지 출력되도록 하였다.

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.

- 자릿수는 3으로 고정해도 됨. 물론, 임의의 자릿수 게임으로 만들어도 좋음.

- 문제 출제 시 키가 입력되면 출력되는 문자를 자신만의 문자로 바꿀 것.

[문제분석 및 해결방법]

2.7절 게임 확장 버전으로 세 자릿수에 대한 Up-and-Down 게임을 만들었다. 세 자릿수이므로 min과 max는 100-999까지로 범위를 설정해두었고, 문자하나 입력 시 숫자는 보이지 않고 "$"가 입력되도록 하였다. _getch()로 세 자리 문자를 입력하면 한 문자 당 아스키코드 '0'을 빼어 10진수로 바꾸어 정수형 변수 n에 세 자리 숫자로 저장하였다. 숫자를 맞출 수 있는 기회는 총 10번이고 예측한 값을 input에 저장하여 for문을 통

해 n과 비교하도록 하였다. for문이 10번 반복하는 동안 숫자를 맞추지 못하면 for문을 종료한다.(실패) 그러나 n이 input과 같아져도 for문을 빠져나올 수 있도록 하였다.(성공) 성공과 실패 여부를 출력하고, 최종 점수는 실수형 변수 grade에 저장하여 소수점 1자리 수까지 출력되도록 설정하였다.

(2) 자릿수가 많아지면 점수 계산 방법이 달라져야 할 것이다. 자릿수에 따른 점수 계산 방법을 설계해 보라.
- 점수 계산은 각자가 알아서 고민해서 적용해 볼 것.

[문제분석 및 해결방법]

교재의 2.7절에서는 두 자릿수일 때는 1회당 10점씩 감점된다. 그러나 세 자릿수일 때는 두 자릿수보다 숫자를 맞출 수 있는 확률이 낮아진다. 따라서 세 자릿수일 때는 숫자를 맞출 확률이 낮아지는 대신 1회당 점수를 적게 차감하는 방식을 고안하였다. 두 자릿수일 때 1회당 10점을 기준으로, 2 : 3 = 10 : 15라는 비례식을 이용하여 다음 식을 도출하였다.

두 자릿수일 때, 10 * (10 - (10 * (1 / 10)) * i) = (10 * (10 - i))

세 자릿수일 때, 10 * (10 - (10 * (1 / 15)) * i) = (10 * (10 - (2 / 3) * i))

(2) 자신이 구현한 주요 코드

for (int i = 5; i >= 1; i--)
{
for (int j = 5; j > 5 - i; j--)
printf("%2d", j);
printf("\n");
}
	j를 숫자로 출력하는 데, i를 이용하여 i값에 따라 j의 출력해야하는 값이 하나씩 줄어들도록 하였다. 따라서 처음에 i = 5일 때 j는 5부터 1까지 모두 출력되지만, 마지막 i = 1일 때 j의 범위는 j > 4이므로 5만 출력되는 것을 알 수 있다.

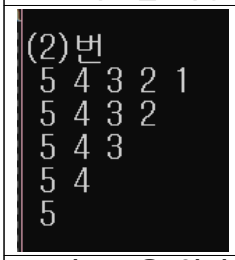
for (double i = 1; i <= 2 * (double)n - 1; i += 4)
PI += 1 / i;
for (double i = 3; i <= 2 * (double)n + 1; i += 4)
PI -= 1 / i;
	PI에 for문 두 개를 사용하여 (+)끼리 (-)끼리 더한 후 PI에 더해주는 방식으로 π의 근삿값을 구하였다. 분모는 모두 홀수이므로 i가 4씩 더해진다.

if (n == input) break;
else if (n > input) {
printf(" 더 큰 숫자입니다!\n");
if(min < input) min = input;
}
else {
printf(" 더 작은 숫자입니다!\n");
if(max > input) max = input;
}
	input이 n보다 작을 때 기존 min보다 큰 수일 때만 갱신하도록 설정하였다.
	
	input이 n보다 클 때 기존 max보다 작은 수일 때만 갱신하도록 설정하였다.

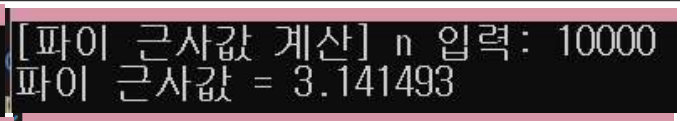
if (i == 10) grade = 0; else grade = (10 * (10 - (2.0 / 3.0) * i));	(10 * (10 - (2.0 / 3.0) * I) 세 자릿수일 때는 본 식을 이용하여 점수를 계산한다. 이때, 10회가 넘어가면 자동으로 0점을 부여한다. (10 * (10 - (2.0 / 3.0) * i) 식을 그냥 이용할 경우 10회가 넘어가면 i = 10이므로 33.333이 된다. 그러나 실패했을 경우 점수를 부여하면 안 되기 때문에 i = 10이 되어 for문을 빠져나올 경우 자동 0점으로 출력되도록 설정하였다.

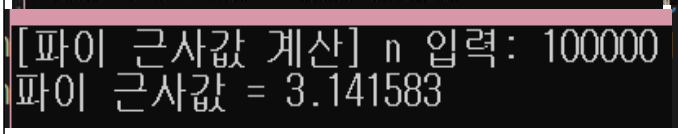
(3) 다양한 입력에 대한 테스트 결과

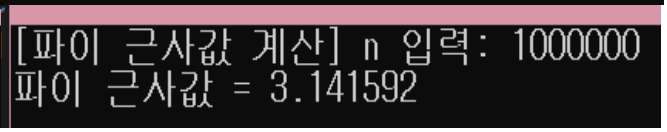
1. 반복문을 이용하여 다음과 같은 패턴을 출력하는 프로그램을 작성하라.



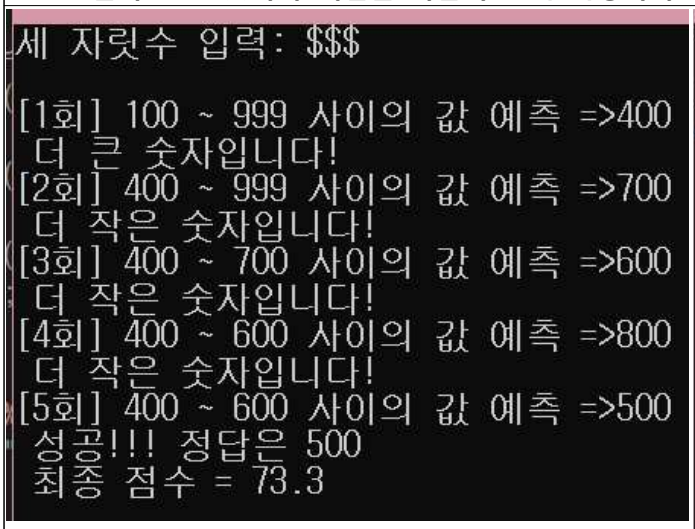
2. 정수 n을 입력받아 다음의 식을 이용하여 π의 근사값을 구하는 프로그램을 작성하라.

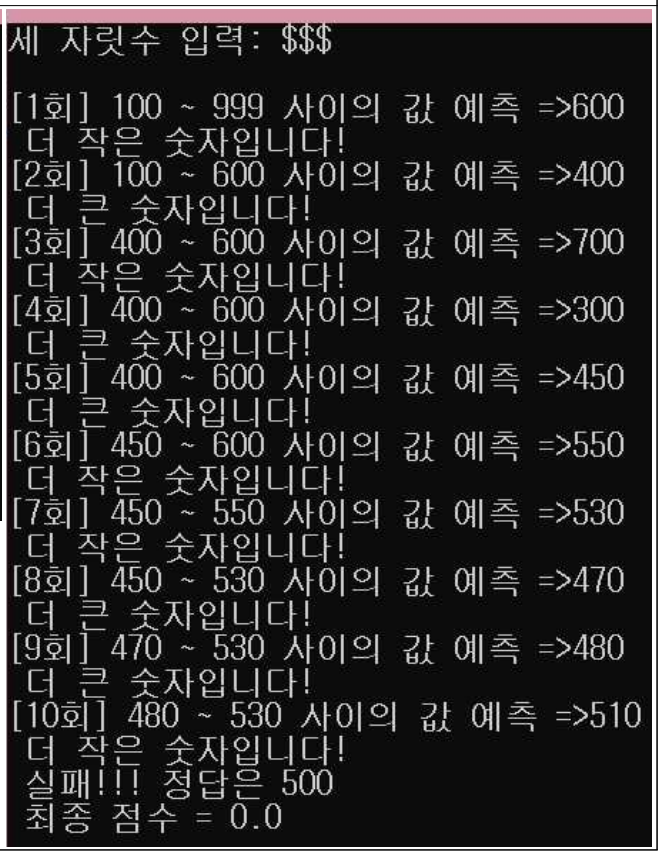






3. 2.7절의 번호 맞히기 게임을 다음과 같이 확장하라.





(4) 코드에 대한 설명 및 해당 문제에 대한 고찰

for (double i = 1; i <= 2 * (double)n - 1; i += 4)
PI += 1 / i;
for (double i = 3; i <= 2 * (double)n + 1; i += 4)
PI -= 1 / i;
	int에서 double로 바꾸니 답이 정확히 나왔다.
	for문의 증감식 처리부에서 i+4로 표현했을 때 답이 나오지 않았는데, i+=4로 바꾸었더니 답이 나왔다.

else grade = (10 * (10 - (2.0 / 3.0) * i)); printf(" 최종 점수 = %.1f\n", grade);	(2/3)을 (2.0/3.0)과 같이 소수점으로 표현하지 않았다면 제대로 된 값이 나오지 않았을 것이다.

(5) 이번 과제에 대한 느낀점

항상 코딩을 할 때, 과제 외에는 이런 코딩 보고서를 작성하지 않았다. 평소 어떤 문제에 대해 코딩을 작성하고 풀어버리면 그 다음날 어제 무슨 문제들을 풀었는지 기억이 잘 안 난다. 또, 주석도 어쩌다가 어려운 부분만 작성하는 편이라 본인이 풀었던 코딩을 보면서 다시 천천히 읽어봐야 생각나는 경우가 종종 있다. 그러나 본 코딩 보고서는 한 문제마다 코딩에 대한 설명과 출력 결과, 내가 막혔던 부분까지 작성하기 때문에 작성하는 과정에서 기억도 더 오래 남게 해주고, 이 보고서를 통해 추후에 내가 무슨 문제를 풀었었는지 쉽게 찾아볼 수 있어 매우 좋은 과제라고 생각한다. 앞으로도 C++ 보고서 과제를 열심히 해서 본 수업이 끝나더라도 매번 스스로 코딩 보고서를 작성할 수 있도록 습관을 길들여놔야겠다.

(6) 궁금한 점이나 건의사항

딱히 없습니다.