# 실습 6

학과 : 전자공학과 학번 : 2023104322 이름 : 현시온

- 과제는 pdf로 변환하여 제출(과제 문서 첫 줄에 학과/학번/이름 포함)
- 과제는 순서대로 작성하며, 문제와 설명을 모두 포함(형식이 맞지 않으면 감점)
- 프로그램을 작성하는 문제는 소스코드와 실행 결과를 모두 text로 붙여넣기(그림으로 포함하지 말 것)하고 코드 설명 및 결과에 대한 설명을 포함해야 함
- 문의 사항은 이메일(nize@khu.ac.kr) 또는 오픈 카톡방을 이용

1. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
#include <cmath>
int main() {
    const double pi = acos(-1.);
    const double e = exp(1.);
```

//0 ~ pi의 범위에서 cos이 입력한 인수가 되게 하는 값을 리턴하는 함수 acos와 자연상수 e^(입력한 인수)의 연산 값을 리턴하는 함수 exp를 호출하고, 불변 double 타입 변수 pi와 e에 각각 -1(double 타입), 1(double 타입)을 인수로 넣어 리턴된 값을 할당한다. 즉 변수들은 그들의 이름에 대응하는 값을 할당 받는다.

```
    std::cout << cos(60. * pi / 180.) << std::endl;
    std::cout << atan2(1., 1.) * 180. / pi << std::endl;
    std::cout << atan2(-1., -1.) * 180. / pi << std::endl;
```

//cos(인수)의 값을 리턴하는 함수 cos와 입력한 인수 둘을 각각 x, y 좌표로 하는 벡터와 x축 사이각의 값(단위는 라디안)을 -pi ~ pi의 범위에서 리턴하는 함수 atan2를 호출하여 입력 받은 인수에 따라 알맞게 계산한 결과값을 출력한다. (cos(pi/3), pi/4*180/pi, -3*pi/4*180/pi)

```
    std::cout << pow(2., -1.) << std::endl;
    std::cout << pow(2., .5) << std::endl;
```

//인수 둘을 각각 x, y 라 했을 때, x^y 연산 값을 리턴하는 함수 pow를 호출하여 입력 받은 인수에 따라 알맞게 계산한 결과값을 출력한다. (2^(-1), 2^0.5)

```
    std::cout << log(e) << std::endl;
```

//ln(인수)의 값을 리턴하는 함수 log를 호출하여 입력 받은 인수에 따라 알맞게 계산한 결과값을 출력한다. (ln(e))

```
}
```

실행 결과:

```
0.5
45
-135
0.5
1.41421
1
```

2. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>

void Fn1(int x) {

    std::cout << x << std::endl;

}
```

//함수를 호출했을 때 입력한 인수를 정수형 파라미터 x에 할당하고, 그 x를 출력하는 함수 Fn1을 정의한다.

```
int Fn2(int x) {

    std::cout << x++ << std::endl;

    return x;

}
```

//함수를 호출했을 때 입력한 인수를 정수형 파라미터 x에 할당하고, 후위증가 연산자이므로 입력 파라미터 x를 먼저 출력하고 x에 1을 더한 다음 int 타입의 값으로 리턴하는 함수 Fn2를 정의한다.

```
int main() {

    int x = 0;
```

//int 타입의 변수 x를 선언하고 0으로 초기화한다.

```
    Fn1(x);
```

//인수로 변수 x를 받은 Fn1 함수를 함수 내용에 따라 처리한다. 여기서는 인수 x를 할당 받은 입력 파라미터 x를 출력한다.

```
    std::cout << Fn2(x) << std::endl;
```

//인수로 변수 x를 받은 Fn2 함수를 함수 내용에 따라 처리한다. 함수 내용에 따르면 인수 x를 할당 받은 파라미터 x를 출력하고 1을 더하여 리턴하고, cout 문장에 따라 다시 한 번 리턴 받은 파라미터 x를 출력한다.

```
    Fn1(x);
```

//인수로 변수 x를 받은 Fn1 함수를 함수 내용에 따라 처리한다. 위에 작업들은 전부 파라미터 x에 대해서 연산이 이루어진 것이지, 변수 x에 대해서 이루어진 것이 아니므로 여전히 인수로는 0이 들어가기 때문에 0을 출력한다.

```
}
```

실행 결과:

```
0

0

1

0
```

3. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>

int main() {

    int x1 = 10; //정수형 변수 x1을 선언하고 10으로 초기화한다.

    int* p1 = &x1; //p1을 정수형 변수의 포인터로 선언하며, 포인터 p1이 가리키는 대상은 x1이다.
```

```
    int** pp1 = &p1; //pp1을 정수형 변수의 포인터의 포인터로 선언하며, 포인터의 포인터 pp1이 가리키는 대상은 p1이다. 즉 x를 가리키는 포인터 p1를 가리키는 포인터는 pp1이다.
    **pp1 = 11;
//pp1이 가리키는 대상인 p1이 가리키는 대상에 11을 할당한다. 즉 x1에 11을 할당한다.
    std::cout << x1 << std::endl;
//x1에 할당된 11을 출력한다.
    *p1 = 12;
//p1이 가리키는 대상에 12를 할당한다. 즉 x1에 12를 할당한다.
    std::cout << x1 << std::endl;
//x1에 할당된 12를 출력한다.
    int x2, * p2, ** pp2;
    x2 = 10;
    p2 = &x2;
    pp2 = &p2;
//정수형 변수 x2를 선언하고 10을 할당하며, p2는 정수형 변수 x2의 포인터, pp2는 정수형 변수 x2의 포인터 p2의 포인터로 선언 및 할당한다.
    **pp2 = 11;
//pp2가 가리키는 대상 p2가 가리키는 대상인 x2에 11을 할당한다.
    std::cout << x2 << std::endl;
//x2에 할당된 11을 출력한다.
    *p2 = 12;
//p2가 가리키는 대상인 x2에 12를 할당한다.
    std::cout << x2 << std::endl;
//x2에 할당된 12를 출력한다.
}
```

실행 결과:

```
11
12
11
12
```

4. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하라.

```
#include <iostream>
int Sum1(int x) {
    int sum = 0;
    sum += x;
    return sum;
}
//호출했을 때 입력한 인수를 정수형 파라미터 x에 할당하고, 정수형 변수 sum을 선언 및 0 할
```

당 후 변수 sum에 파라미터 x를 더하고 정수형으로 리턴하는 함수 Sum1을 정의한다.

```
int Sum2(int x) {
    static int sum = 0;
    sum += x;
    return sum;
}
```

//호출했을 때 입력한 인수를 정수형 파라미터 x에 할당하고, static 정수형 변수 sum을 선언 및 0 할당 후 변수 sum에 파라미터 x를 더하고 정수형으로 리턴하는 함수 Sum2를 정의한다.

```
int main() {
    for (int i = 0; i < 10; ++i)
        std::cout << Sum1(i) << ", ";
    std::cout << std::endl;
```

//선언 및 0으로 초기화된 정수형 변수 i에 대해서 다음 내용을 처리하는 과정을 i에 1씩 더하며 i가 10보다 작은 한 반복한다; 호출된 함수 Sum1에 변수 i를 인수로 하여 나온 리턴값을 출력한다. 일반적인 함수의 리턴값의 경우 호출되어 함수 내용을 처리한 다음 버려지기 때문에 반복 과정 한 주기마다 sum 값은 없어진다. 즉 그냥 i를 반복 출력한 것과 다름없는 실행 결과가 나온다.

```
    for (int i = 0; i < 10; ++i)
        std::cout << Sum2(i) << ", ";
    std::cout << std::endl;
```

//선언 및 0으로 초기화된 정수형 변수 i에 대해서 다음 내용을 처리하는 과정을 i에 1씩 더하며 i가 10보다 작은 한 반복한다; 호출된 함수 Sum1에 변수 i를 인수로 하여 나온 리턴값을 출력한다. 함수 내 지역 변수가 static의 특징을 갖는다면 함수 내용을 처리한 다음에도 버려지지 않기 때문에 반복 과정 주기마다 sum 값은 i의 누적된 합으로 볼 수 있다.

```
}
```

실행 결과:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
0, 1, 3, 6, 10, 15, 21, 28, 36, 45,
```

5. 아래 C++ 코드의 출력을 확인하고, gcd 함수의 호출과 반환 과정을 순서대로 진행하면서 결과를 설명하라.

```
#include <iostream>
int gcd(int m, int n) {
    if (n == 0)
        return m;
    else
        return gcd(n, m % n);
}
```

//호출했을 때 입력한 인수를 정수형 파라미터 m과 n으로 할당하고 정수형 타입의 값을 반환하는 함수 gcd는 다음과 같은 내용을 가진다; 파라미터 n이 0이면 m을 출력하고, 0이 아니면 gcd(n, m % n)의 반환 값을 출력한다. 즉 파라미터 n에 할당되는 값이 0이 될 때의 m에 할당되는 값이 최종적으로 반환될 것임을 예측할 수 있다.

```
int main() {
```

```
    std::cout << gcd(48, 30) << std::endl;
```

//함수 gcd를 호출하고 48, 30을 각각 인수로 하여 반환한 값을 출력한다. 파라미터 m과 n에 할당되는 값이 변화하는 과정은 다음과 같다. (48, 30 -> 30 18 -> 18, 12 -> 12, 6 -> 6, 0)

```
}
```

실행 결과:

```
6
```

6. 아래 C++ 코드의 출력을 확인하고, 동작을 설명하시오.

```
#include <iostream>
int* Fn(int* p1, int *p2) {
    return *p1>*p2?p1:p2;
}
```

//호출하면 입력한 인수를 정수형 변수의 포인터 p1, p2에 할당하고 정수형 변수의 포인터를 반환하는 함수 Fn은 다음과 같은 내용을 처리한다; p1의 실체인 정수형 변수가 p2의 실체인 정수형 변수보다 크다면 p1을, 아니라면 p2를 반환한다. (파라미터 p1과 p2의 선언에서 별의 위치의 차이는 아무런 의미가 없다.)

```
int main() {
    int x = 10, y = 20;
    int* p = Fn(&x, &y);
```

//10이 할당된 정수형 변수 x의 주소와 20이 할당된 정수형 변수 y의 주소를 인수로 하여 함수 Fn은 함수의 내용에 따라 호출 및 반환되어 정수형 변수의 포인터 p에 할당된다. 여기서는 x가 y보다 크지 않기 때문에 &y가 반환되어 p에 할당된다.

```
    *p += 5;
```

//포인터 p의 실체에 5를 더한다. 즉 &y의 실체인 y에 5를 더하므로 y는 25가 할당된다.

```
    std::cout << x << std::endl;
    std::cout << y << std::endl;
```

//x에는 아무런 변화가 일어나지 않았기에 그대로 10이, y는 5가 더해졌으므로 25가 출력된다.

```
}
```

실행 결과:

```
10
25
```

7. 실수를 반복문으로 입력 받아서 (입력 종료는 EOF), 산술 평균(average), 기하 평균(geometric mean)과 조화평균(harmonic mean)을 계산해서 출력하는 프로그램을 작성하라.

```
#include <iostream>
int main() {
        int input = 0;
        //입력할 정수를 할당받을 정수형 변수 input을 선언 및 0으로 초기화
        double average = 0, geometric_mean = 0, harmonic_mean = 0, count = 0, sum = 0,
reverse_sum = 0;
        //평균, 기하 평균, 조화 평균을 할당받고 이들을 계산하기 위해 사용할 변수들 선언 및
0으로 초기화
```

```
        while (static_cast<bool>(std::cin)) { //eof를 입력받기 전까지 반복
                std::cin >> input; //정수 입력 받기
                if (static_cast<bool>(std::cin)) { //조건문을 통해 eof가 입력되고 계산되는
것을 방지
                        count++; //입력받은 정수의 개수
                        sum += static_cast<double>(input); //입력받은 정수의 합
                        reverse_sum += pow(static_cast<double>(input), -1.); //입력받은
정수의 역수의 합
                        average = sum / count; //평균은 입력받은 정수의 합을 정수의 개수로
나눈 결과
                        geometric_mean = pow(static_cast<double>(sum), pow(count, -1.));
//기하 평균은 (입력받은 정수의 합)^(정수의 개수의 역수)
                        harmonic_mean = count / reverse_sum; //조화 평균은 (정수의 개수) /
(입력받은 정수의 역수의 합)
                }
        }
        std::cout << average << ", " << geometric_mean << ", " << harmonic_mean << std::endl;
//계산한 결과 출력


    }
```

8. 아래와 같은 순서를 가지는 피보나치 수열(Fibonacci series)을 출력하는 프로그램을 사용자 정의 함수(int Fibonacci(int n))를 이용하여 작성하라. (Recursive function을 사용하지 않음)

```
   0, 1, 1, 2, 3, 5, ..., 34
```

```
#include <iostream>
int Fibonacci(int n); //피보나치 함수의 프로토타입 선언
int main() {
        int input = 0; //피보나치 수열을 몇번째 항까지 출력할 것인지 입력 받을 정수형 변수
input 선언 및 0으로 초기화
        std::cin >> input; //input 입력받기
        Fibonacci(input); //input을 인수로 하여 피보나치 함수 호출
}
int Fibonacci(int n) { //파라미터는 정수형 변수 n 1개
        int front_num = 0, back_num = 1, next_num = 0; //피보나치 수열 계산을 위한 세가지
정수형 변수 선언
        std::cout << front_num << ", "; //먼저 제 1항 출력
        for (int count = 1; count < n; count++) { //제 2항부터 계산을 통해 출력
                next_num = front_num + back_num;
                front_num = back_num;
                back_num = next_num;
                std::cout << front_num << ", ";
        }
        //front_num에 back_num을 할당하고, back_num에 원래 front_num과 back_num의 합을
할당하는 방식으로 계산
        return 0; //반환값 설정
}
```

9. 두 개의 정수를 매개변수(parameter, n, r)로 받아서 순열(permutation, nPr)을 계산하여 반환하는 함수를 작성하라.

```
#include <iostream>
int factorial(int k); //팩토리얼 함수의 프로토타입 선언
double permutation(int n, int r); //순열 함수의 프로토타입 선언
int main() {
	int num1 = 0, num2 = 0;
	std::cin >> num1;
	std::cin >> num2;
	//순열의 결과를 출력하기 위한 정수형 변수 2개 선언 및 입력 받기.
	if (num1 >= num2) {
		std::cout << permutation(num1, num2) << std::endl;
	}
	//num1이 num2보다 클 경우, num1과 num2를 인수로 하여 permutation 함수를 호출하고 그
반환값을 출력.
	else {
		std::cout << "Wrong inputs" << std::endl;
	}
	//num2가 num1보다 더 크다면 계산이 불가하므로 Wrong inputs을 출력
}
int factorial(int k) { //파라미터는 정수형 변수 k
	int facto = 1;
	for (int i = 1; i <= k; i++) {
		facto *= i;
	}
	return facto;
}
//팩토리얼을 for 구문의 반복을 통해서 구현하여 반환.
double permutation(int n, int r) { //파라미터는 정수형 변수 n와 r
	if (n >= r) {
		return static_cast<double>(factorial(n)) / static_cast<double>(factorial(n -
r));
	}
}
//n이 r보다 클 경우에만 n!과 (n-r)!의 나눗셈을 factorial 함수를 통해 구현하고 반환.

    }
```

10. 이차 방정식(quadratic equation $ax^2+bx+c=0$)의 실근을 계산하여 출력하는 프로그램을 아래와 같이 처리되도록 작성하라

    A. a,b,c (type double)를 입력 받고,

    B. 다음과 같은 조건으로 계산

    - a=0이고 b=0이면, 입력 오류(not a valid equation)

    - a=0이고d b≠0이면, 1차로 근은 x=-c/b

    - $b^2$ -4ac < 0이면, 실근 없음(not a real solution)

    - $b^2$ -4ac >= 0이면, 실근 계산(근의 공식이용)

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
#include <iostream>
double quadratic_equation(double a, double b, double c); //함수 프로토타입 선언
int main() {
	double num1 = 0, num2 = 0, num3 = 0;
	std::cin >> num1;
	std::cin >> num2;
	std::cin >> num3;
	quadratic_equation(num1, num2, num3);
	//인수로 사용할 실수형 변수 3개를 선언 및 입력 받고 함수 호출.
}
double quadratic_equation (double a, double b, double c) { //파라미터는 실수형 변수 a, b, c
3개.
	if ((a == 0) && (b == 0)) {
		std::cout << "not a valid equation" << std::endl;
	}
	//a와 b가 모두 0일 경우 잘못된 방정식이므로 not a valid equation 문자열 출력.
	if ((a == 0) && (b != 0)) {
		std::cout << -c / b << std::endl;
	}
	//a가 0이지만 b는 0이 아닌 경우 1차 방정식이므로 -c/b를 출력.
	if (pow(b, 2) - 4 * a * c < 0) {
		std::cout << "not a real solution" << std::endl;
	}
	//판별식이 음수일 경우 실근이 없으므로 not a real solution 문자열 출력.
	else {
		std::cout << (-b + sqrt(pow(b, 2) - 4 * a * c)) / 2 * a << ", " << (-b -
sqrt(pow(b, 2) - 4 * a * c)) / 2 * a << std::endl;
		//판별식이 양수일 경우 근의 공식을 구현하여 두 개의 근을 출력.
		if (pow(b, 2) - 4 * a * c == 0) {
			std::cout << "multiple root" << std::endl;
		}
		//판별식이 0일 경우 multiple root 문자열 추가 출력.
	}
	return 0;
	//반환값 설정.
}
```