컴퓨터 알고리즘

2013335004 3B
컴퓨터 시스템
고명진

실행 결과

```
 algorithm l
.  ..  subject0.c  subject1.c
 algorithm gcc subject1.c
 algorithm l
.  ..  a.out  subject0.c  subject1.c
 algorithm ./a.out
10
재귀적 방법 결과 값: 3.628800e+06, 0.000005 초입니다.
반복적 방법 결과 값: 3.628800e+06, 0.000003 초입니다.
 algorithm ./a.out
50
재귀적 방법 결과 값: 3.041409e+64, 0.000008 초입니다.
반복적 방법 결과 값: 3.041409e+64, 0.000004 초입니다.
 algorithm ./a.out
100
재귀적 방법 결과 값: 9.332622e+157, 0.000008 초입니다.
반복적 방법 결과 값: 9.332622e+157, 0.000005 초입니다.
 algorithm ./a.out
150
재귀적 방법 결과 값: 5.713384e+262, 0.000009 초입니다.
반복적 방법 결과 값: 5.713384e+262, 0.000007 초입니다.
 algorithm ./a.out
200
재귀적 방법 결과 값: inf, 0.000028 초입니다.
반복적 방법 결과 값: inf, 0.000009 초입니다.
 algorithm
```

소스코드

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4
 5 double fec1(int n);
 6 double fec2(int n);
 7
 8 void main(void)
 9 {
10     int n;
11     scanf("%d", &n);
12
13     clock_t start, finish;
14     start = clock();
15
16     fec1(n); // 재귀적 방법
17
18     finish = clock();
19
20     double duration;
```

```
21     duration = (double)(finish - start)/CLOCKS_PER_SEC;
22
23     printf("재귀적 방법 결과 값: %e, %f 초입니다.\n", fec1(n), duration);
24     start = clock();
25
26     fec2(n); // 반복적 방법
27
28     finish = clock();
29
30     duration = (double)(finish - start)/CLOCKS_PER_SEC;
31
32     printf("반복적 방법 결과 값: %e, %f 초입니다.\n", fec2(n), duration);
33 }
34
35 double fec1(int n)
36 {
37     if(n > 1)
38         return n * fec1(n-1);
39     else
40         return 1;
41 }
42
43 double fec2(int n)
44 {
45     double a = 1.0;
46     for(int i = n; i > 1; i--)
47     {
48         a *= i;
49     }
50
51     return a;
52 }
53
```

분석내용

우선 주어진 과제인 ‘n!을 구하는 알고리즘을 n=10, 50,100, 150, 200 등의 값을 구하는 프로그래밍을 하세요. (실행시간, 값 계산)’의 재귀호출(fec1()), 반복적 방법(fec2())를 구성하였습니다.

두 함수 모두 ‘n’이라는 입력 값을 받아 fectorial 을 계산하여 결과 값을 return 합니다. 즉, 내부적으로는 다르나 동작은 같은 함수입니다.

또한, 큰 값을 출력해야 하기에 값을 저장하는 자료형과 출력 포맷의 경우는 double 과 상수(e)를 포맷으로 하여 값의 범위를 크게 하였습니다.

마지막으로 실행 속도의 경우 반복적 방법보다 재귀호출 방법이 더 느린 경향이 있었고 , 이는 함수를 호출하여 결과를 return 하는 것보다 for 과 같은 반복문을 이용해 구현하는 것이 효율적이라는 것을 느껴 프로그램을 구현함에 있어 재귀적 방법보다는 반복적 방법을 사용하리라 생각하게 되었습니다.