

Data Structure

Practical Class - Week 02

Exercise01 – Problem 01 (Skeleton)

```
1  #pragma warning(disable: 4996)
2  #include<stdio.h>
3
4  int main(void)
5  {
6      char ch = 'c';
7      char *chptr = &ch;
8
9      int i = 20;
10     int *intptr = &i;
11
12     char *ptr = "I am a string";
13
14     //무엇이 출력될 지 예측해보세요
15     printf("\n [%c], [%d], [%c], [%llu], [%llu], [%c], [%s]\n",
16           *chptr, *intptr, *ptr, ptr, ptr+5, *(ptr+5), ptr);
17
18     getchar();
19
20     return 0;
21 }
```

Output>

Exercise01 - Problem 01 (Answer)

```
1  #pragma warning(disable: 4996)
2  #include<stdio.h>
3
4  int main(void)
5  {
6      char ch = 'c';
7      char *chptr = &ch;
8
9      int i = 20;
10     int *intptr = &i;
11
12     char *ptr = "I am a string";
13
14     //무엇이 출력될 지 예측해보세요
15     printf("\n [%c], [%d], [%c], [%llu], [%llu], [%c], [%s]\n",
16           *chptr, *intptr, *ptr, ptr, ptr+5, *(ptr+5), ptr);
17
18     getchar();
19
20     return 0;
21 }
```

Output> [c], [20], [I], [11163736], [11163741], [a], [I am a string]

ptr[0]의 주소
(다를 수 있음)

5byte

ptr[5]의 주소

Exercise01 – Problem 02 (Skeleton)

```
1  #pragma warning(disable: 4996)
2  #include<stdio.h>
3
4  struct st {
5      int a;
6      char ch;
7  };
8
9  int main()
10 {
11     printf("학번 201821234 이름 홍길동\n\n");
12
13     struct st obj;
14     struct st *stobj = &obj;
15
16     //stobj 변수를 사용해서
17     //obj.a 를 5로 변경해보세요
18     //obj.ch를 'a'로 변경해보세요
19
20     (1)
21
22
23     printf("\n [%d] [%c]\n", stobj->a, stobj->ch);
24
25     getchar();
26
27     return 0;
28 }
```

Exercise02 - Introduction

여러 함수들은 재귀식이나 일반항으로 표현될 수 있다.

팩토리얼 함수의 재귀적 표현

$$\begin{cases} F(n) = n \times F(n-1) & \text{if } n \geq 2 \\ F(n) = 1 & \text{if } n = 1 \end{cases}$$

팩토리얼 함수의 일반항 공식

$$F(n) = \prod_{i=1}^n i = 1 \times 2 \times \cdots \times (n-1) \times (n)$$

```
long long facto_iter(int n)
{ //팩토리얼 일반항의 구현
    int i;
    long long result = 1;
    for (i = 1; i <= n; i++)
    {
        result *= i;
    }
    return result;
}
```

Exercise02 - Introduction

일반항이 '잘' 정의 된 함수는 비교적 간단하게 반복문 등을 통해 구현할 수 있다.

물론, 재귀적으로 정의 된 함수도 반복문 등으로 구현될 수 있지만 그렇지 않은 경우 재귀함수를 이용해 구현해야 한다.

* 팩토리얼, 피보나치 함수 등은 반복문으로도 간단히 구현할 수 있다.

문제 해결을 위한 재귀함수를 구현 할 때에는 ...

- 함수 간의 순환 호출이 일어나지 않도록 재귀적 관계를 정의한다.
- 파라미터에 대하여 함수의 역할과 의미를 명확히 정의한다.
- 사전에 정의한대로 정확히 코드를 구현한다.

Exercise02 – Main Function

```
1  #pragma warning(disable: 4996)
2  #include<stdio.h>
3
4  long long fact(int);
5
6  int main()
7  {
8      int n;
9      long long result;
10     printf("학번 201821234 이름 홍길동\n\n");
11
12     //n을 입력받는다
13     printf("Input 'N'\n> ");
14     scanf("%d", &n);
15
16     //fact(n)을 호출해 수열의 n번째 값을 전달받는다.
17     result = fact(n);
18
19     //결과를 출력한다.
20     printf("%lld\n", result);
21
22     getchar();
23     return 0;
24 }
25
```

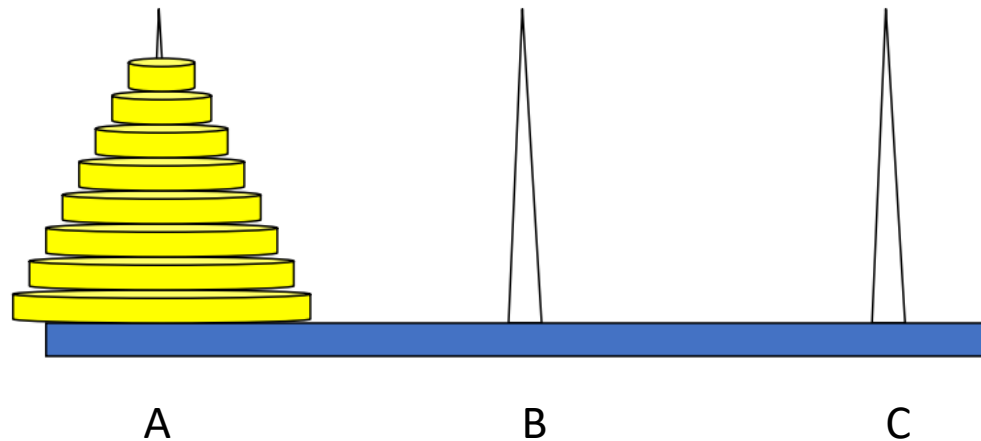
Exercise02 - Factorial Function (Skeleton)

```
27  /**
28   * @brief
29   * 항의 번호 n을 입력받아서 팩토리얼 수열의 n번째 항을 반환해주는 함수
30   * 재귀적으로 구현해야 한다
31   * 가장 먼저 호출된 항의 번
32   *
33   * @param n 계산하고자 하는 팩토리얼 수열 항의 번호
34   * @return long long 팩토리얼 수열의 n번째 항
35   */
36  long long fact(int n)
37  {
38      long long value;
39
40      if (n <= 1)
41      {
42          printf("fact(1) called!\n");
43          printf("fact(1) returned!!\n");
44
45          (1)
46
47      }
48      else {
49          printf("fact(%d) called!! \n", n);
50
51          (2)
52
53          printf("fact(%d) returned value: %lld !!\n", n, value);
54          return value;
55      }
56  }
```


Exercise03 - Introduction

A타워에 놓인 N개의 원반을 규칙을 만족하며 모두 c로 이동시켜야 한다.

- 최소한의 횟수만 움직여야 한다.
- 더 큰 원반이 자신보다 작은 원반위로 올 수 없다.
- 동시에 하나의 원반만 다른 기둥으로 옮길 수 있다.



Exercise03 - Introduction

Hint>

세 기둥 s, t, w 가 있다고 하자. (기둥 순서와는 관련 없다.)

- s 번 기둥을 제외하고는 모두 비어있다.

현재 s 번 기둥에 N 개의 원반이 있고 이를 모두 t 번 기둥으로 옮기는 최적의 방법

- s 번 기둥에서 가장 큰 원반을 제외한 $(N-1)$ 개의 원반을 w 로 옮긴다.
- s 번 기둥에 남은 가장 큰 원반을 t 번 기둥으로 옮긴다.
- w 에 있는 $(N-1)$ 개의 원반을 마저 t 번 기둥으로 옮긴다.

Exercise03 – Main Function

```
1  #pragma warning(disable: 4996)
2  #include <stdio.h>
3
4  /**
5   * @brief
6   * 'start' 기둥에 놓여있는 n개의 원반을 하노이 타워의 규칙을 만족하면서
7   * 'target' 기둥으로 모두 옮기는 함수
8   * # 'start'번 기둥에는 1번~n번 원반이 차례로 놓여있다.
9   *
10  * @param n      현재 'start'기둥에 있는 원반의 수
11  * @param start   현재 n개의 원반들이 놓여 있는 기둥
12  * @param work    세 기둥 중 출발점과 목적지가 아닌 남은 기둥
13  * @param target  원반들을 옮겨야 할 목적지 기둥
14  */
15  void hanoi(int n, char start, char work, char target);
16
17  int main()
18  {
19      int n;
20      printf("학번 201821234 이름 홍길동\n\n");
21      printf("Number of disk. > ");
22      scanf("%d", &n);
23
24      //n개의 원반을 'A'기둥 에서 'c'기둥 으로 모두 옮긴다.
25      hanoi(n, 'A', 'B', 'C');
26
27      getchar();
28      return 0;
29  }
```

Exercise03 – Hanoi Function (Skeleton)

```
33 void hanoi(int n, char start, char work, char target)
34 {
35     /**
36      * @brief 옮겨야 할 원반이 하나 뿐이라면?
37      */
38     if (n == 1)
39     {
40         (1)
41     }
42     /**
43      * @brief 두 개 이상의 원반을 옮겨야 한다면?
44      */
45     else
46     {
47         (2)
48     }
49 }
50
51
52
53
54
55 }
```

Exercise04 - Introduction

피보나치 함수의 재귀적 표현

$$\begin{cases} F(n) = F(n-1) + F(n-2) & \text{if } n \geq 1 \\ F(n) = 0 & \text{if } n = 0 \end{cases}$$

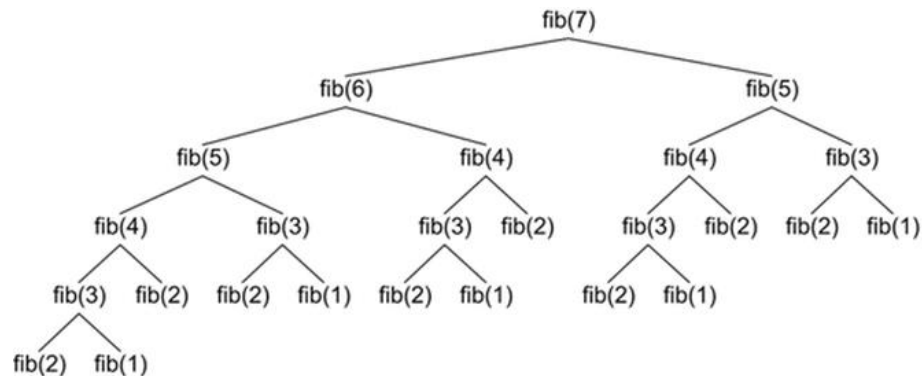
팩토리얼 함수의 일반항 공식

은 간단하지 않다... (궁금하면 구글링 해보세요)

하지만 재귀식이 간단하므로 반복문으로도 구현할 수 있다.

Exercise04 - Introduction

두 개 이상의 항을 재귀식에 포함하는 함수는 시간 복잡도가 기하급수적으로 증가한다.



재귀적으로 구현된 피보나치 함수 $F(n)$ 과 $F(n + 1)$ 은 연산량 차이가 몇 배 일지 고민해보자

Exercise04 – Main Function

```
1  #pragma warning(disable: 4996)
2  #include<stdio.h>
3  #include<time.h>
4  #define MAX_LENGTH (int)(61)
5
6  unsigned long long recursive_fibo(unsigned int n);
7  unsigned long long iterative_fibo(unsigned int n);
8
9  int main()
10 {
11     unsigned int length;
12     double begin, end;
13     double t1, t2;
14     unsigned long long val;
15     printf("학번 201821234 이름 홍길동\n\n");
16     /**
17      * @brief
18      * [1, MAX_LENGTH]의 범위에 대한 각 n에 대하여
19      * n번째 항을 계산하는데 소요되는 시간을 각각 구하여 출력한다
20      */
21     for (length = 1; length <= MAX_LENGTH; length += 5)
22     {
23         //recursive method
24         begin = clock();
25         val = recursive_fibo(length);
26         end = clock();
27         t1 = (end - begin) / CLOCKS_PER_SEC;
28
29         //iterative method
30         begin = clock();
31         val = iterative_fibo(length);
32         end = clock();
33         t2 = (end - begin) / CLOCKS_PER_SEC;
34
35         //print result
36         printf("[Length: %3d] Recursive : %.3lf sec , Iterative : %.3lf sec\n", length, t1, t2);
37     }
38     getchar();
39     return 0;
40 }
```

Exercise04 – Fibonacci Functions (Skeleton)

```
46  /**
47   * @brief 피보나치 수열의 n번째 항을 재귀적으로 계산하는 함수
48   * <Note> 내부에서 별도의 반복문을 사용하지 마세요!
49   *
50   * @param n 항 번호
51   * @return unsigned long long 피보나치 수열의 n번째 항의 값
52   */
53  unsigned long long recursive_fibo(unsigned int n)
54  {
55      (1)
56  }
57
58
59
60  /**
61   * @brief 피보나치 수열의 n번째 항을 반복적으로 계산하는 함수
62   * <Note> 내부에서 별도의 함수를 사용하지 마세요!
63   *
64   * @param n 항 번호
65   * @return unsigned long long 피보나치 수열의 n번째 항을
66   */
67  unsigned long long iterative_fibo(unsigned int n)
68  {
69      unsigned long long previous = 0;
70      unsigned long long current = 1;
71      unsigned long long next=1;
72      (2)
73
74
75
76
77
78
79
80
81
82 }
```