Data Structure

Practical Class - Week 02

Exercise01 – Problem 01 (Skeleton)

```c
#pragma warning(disable: 4996)
#include<stdio.h>


int main(void)
{
    char ch = 'c';
    char *chptr = &ch;

    int i = 20;
    int *intptr = &i;

    char *ptr = "I am a string";

    //무엇이 출력될 지 예측해보세요
    printf("\n [%c], [%d], [%c], [%d], [%d], [%c], [%s]\n",
                *chptr, *intptr, *ptr, ptr, ptr+5, *(ptr+5), ptr);

    getchar();

    return 0;
}
```

Output>

Exercise01 – Problem 02 (Skeleton)

```c
#pragma warning(disable: 4996)
#include<stdio.h>

struct st {
    int a;
    char ch;
};

int main()
{
    printf("학번 201821234 이름 홍길동\n\n");

    struct st obj;
    struct st *stobj = &obj;

    //stobj 변수를 사용해서
    //obj.a 를 5로 변경해보세요
    //obj.ch를 'a'로 변경해보세요

    (1)

    printf("\n [%d] [%c]\n", stobj->a, stobj->ch);

    getchar();

    return 0;
}
```

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.

* 펙토리얼, 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

```c
#pragma warning(disable: 4996)
#include<stdio.h>

long long fact(int);

int main()
{
    int n;
    long long result;
    printf("학번 201821234 이름 홍길동\n\n");

    //n을 입력받는다
    printf("Input 'N'\n> ");
    scanf("%d", &n);

    //fact(n)을 호출해 수열의 n번째 값을 전달받는다.
    result = fact(n);

    //결과를 출력한다.
    printf("%lld\n", result);

    getchar();
    return 0;
}

```

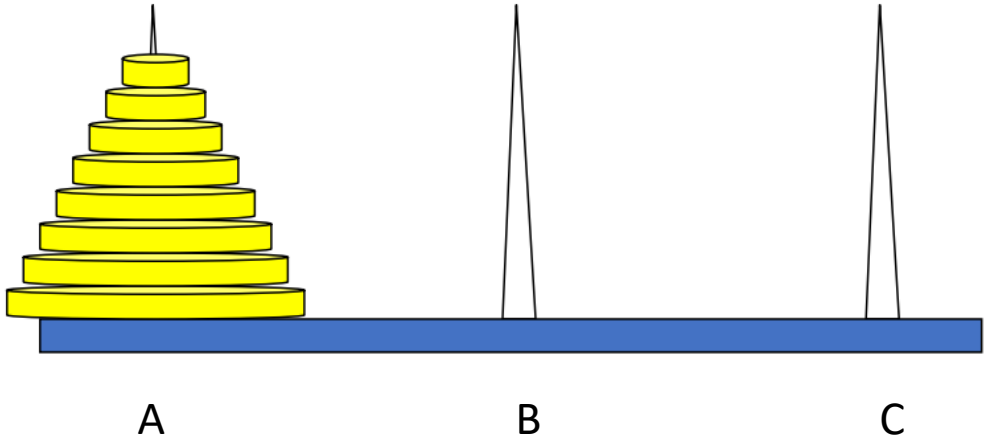
Exercise02 – Factorial Function (Skeleton)

```
27  /**
28   * @brief
29   * 항의 번호 n을 입력받아서 펙토리얼 수열의 n번째 항을 반환해주는 함수
30   * 재귀적으로 구현해야 한다
31   * 가장 먼저 호출된 항의 번
32   *
33   * @param n  계산하고자 하는 펙토리얼 수열 항의 번호
34   * @return long long 펙토리열 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, W가 있다고 하자. (기둥 순서와는 관련 없다.)

- S번 기둥을 제외하고는 모두 비어있다.

현재 S번 기둥에 N개의 원반이 있고 이를 모두 T번 기둥으로 옮기는 최적의 방법

- S번 기둥에서 가장 큰 원반을 제외한 (N-1)개의 원반을 W로 옮긴다.
- S번 기둥에 남은 가장 큰 원반을 T번 기둥으로 옮긴다.
- W에 있는 (N-1)개의 원반을 마저 T번 기둥으로 옮긴다.

Exercise03 – Main Function

```c
#pragma warning(disable: 4996)
#include <stdio.h>

/**
* @brief
* 'start' 기둥에 놓여있는 n개의 원반을 하노이 타워의 규칙을 만족하면서
* 'target' 기둥으로 모두 옮기는 함수
* # 'start'번 기둥에는 1번~n번 원반이 차례로 놓여있다.
*
* @param n      현재 'start'기둥에 있는 원반의 수
* @param start  현재 n개의 원반들이 놓여 있는 기둥
* @param work   세 기둥 중 출발점과 목적지가 아닌 남은 기둥
* @param target 원반들을 옮겨야 할 목적지 기둥
*/
void hanoi(int n, char start, char work, char target);

int main()
{
    int n;
    printf("학번 201821234 이름 홍길동\n\n");
    printf("Number of disk. > ");
    scanf("%d", &n);

    //n개의 원반을 'A'기둥 에서 'C'기둥 으로 모두 옮긴다.
    hanoi(n, 'A', 'B', 'C');

    getchar();
    return 0;
}
```

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
41          (1)
42
43      }
44      /**
45       * @brief 두 개 이상의 원반을 옮겨야 한다면?
46       */
47      else
48      {
49
50
51          (2)
52
53
54      }
55  }
```

Exercise04 - Introduction

피보나치 함수의 재귀적 표현

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

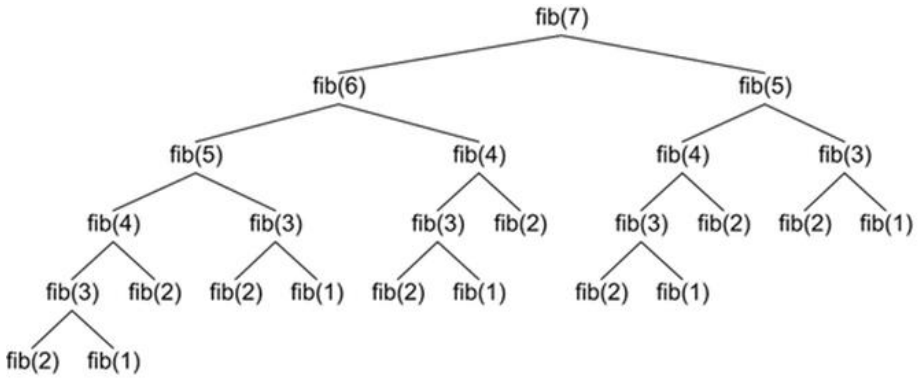
팩토리얼 함수의 일반항 공식

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

```c
#pragma warning(disable: 4996)
#include<stdio.h>
#include<time.h>
#define MAX_LENGTH (int)(61)

unsigned long long recursive_fibo(unsigned int n);
unsigned long long iterative_fibo(unsigned int n);

int main()
{
    unsigned int length;
    double begin, end;
    double t1, t2;
    unsigned long long val;
    printf("학번 201821234 이름 홍길동\n\n");
    /**
     * @brief
     * [1, MAX_LENGTH]의 범위에 대한 각 n에 대하여
     * n번째 항을 계산하는데 소요되는 시간을 각각 구하여 출력한다
     */
    for (length = 1; length <= MAX_LENGTH; length += 5)
    {
        //resulsive method
        begin = clock();
        val = recursive_fibo(length);
        end = clock();
        t1 = (end - begin) / CLOCKS_PER_SEC;

        //iterative method
        begin = clock();
        val = iterative_fibo(length);
        end = clock();
        t2 = (end - begin) / CLOCKS_PER_SEC;

        //print result
        printf("[Length: %3d] Reculsive : %.3lf sec , Iterative : %.3lf sec\n", length, t1, t2);
    }
    getchar();
    return 0;
}
```

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
56      (1)
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
73
74
75
76      (2)
77
78
79
80
81
82  }
```