Informática I
Guía de Práctico

Martin Nievas
mnievas@frc.utn.edu.ar

20 de agosto de 2019

Funciones 2^{da} Parte

Ejercicio 1

Implementar la función `fibonacci` en forma recursiva.

```
int fibonacci(int);
```

Ejercicio 2

El rompecabezas de la Torre de Hanoi fue inventado por el matemático francés Edouard Lucas en 1883. Se inspiró en una leyenda acerca de un templo hindú donde el rompecabezas fue presentado a los jóvenes sacerdotes. Al principio de los tiempos, a los sacerdotes se les dieron tres postes y una pila de 64 discos de oro, cada disco un poco más pequeño que el de debajo.

Su misión era transferir los 64 discos de uno de los tres postes a otro, con dos limitaciones importantes. Sólo podían mover un disco a la vez, y nunca podían colocar un disco más grande encima de uno más pequeño. Los sacerdotes trabajaban muy eficientemente, día y noche, moviendo un disco cada segundo. Cuando terminaran su trabajo, dice la leyenda, el templo se desmenuzaría en polvo y el mundo se desvanecería.

En la Figura 1 Se puede ver la representación gráfica del juego.

Supongamos que los sacerdotes intentan mover los discos de la clavija 1 a la clavija 3. Deseamos desarrollar un algoritmo que imprima la secuencia precisa de transferencias de clavija de disco a disco. Si tuviéramos que abordar este problema con métodos convencionales, nos encontraríamos rápidamente anudados en la gestión de los discos. En cambio, si atacamos el problema con la recursión en mente, inmediatamente se vuelve manejable. Mover n discos se puede ver en términos de mover solo n - 1 discos (y, por lo tanto, la recursividad) de la siguiente manera:

1. Mueva n - 1 discos de la clavija 1 a la clavija 2, utilizando la clavija 3 como área de retención temporal.

2. Mueva el último disco (el más grande) de la clavija 1 a la clavija 3.

3. Mueva los discos n - 1 de la clavija 2 a la clavija 3, utilizando la clavija 1 como área de retención temporal.

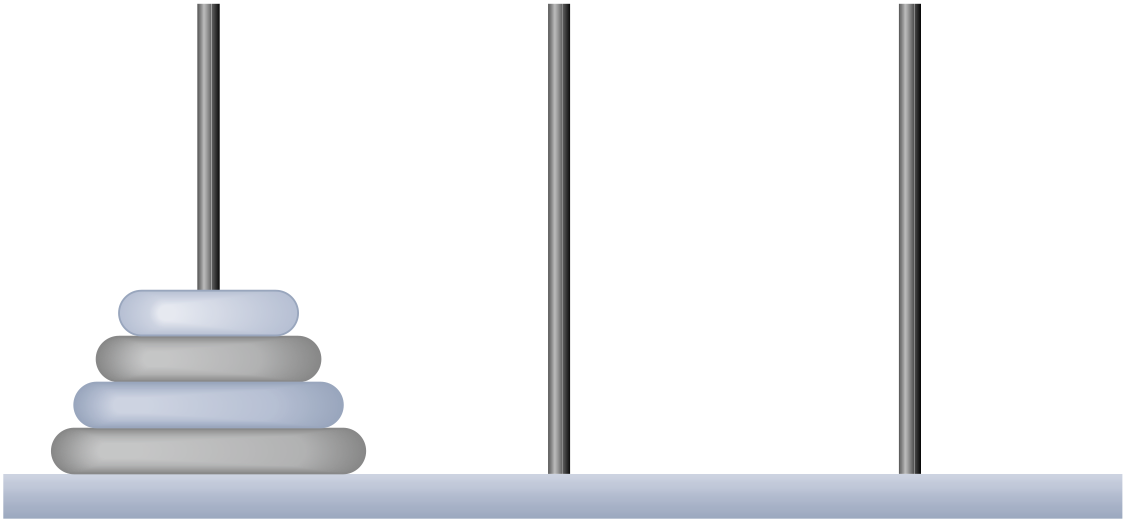

Figura 1: Torres de Hanoi
Fuente: Deitel&Deitel

El proceso finaliza cuando la última tarea implica mover n = 1 disco, es decir, el caso base. Esto se logra moviendo trivialmente el disco sin la necesidad de un área de retención temporal. Escribe un programa para resolver el problema de Towers of Hanoi. Use una función recursiva con cuatro parámetros:

1. El número de discos a mover
2. La clavija en la que se enroscan estos discos inicialmente
3. La clavija a la que se debe mover esta pila de discos
4. La clavija que se utilizará como área de retención temporal.

Su programa debe imprimir las instrucciones precisas que necesitará para mover los discos desde la clavija de inicio a la clavija de destino. Por ejemplo, para mover una pila de tres discos de la clavija 1 a la clavija 3, su programa debe imprimir la siguiente serie de movimientos:

```
1 -> 3
1 -> 2
3 -> 2
1 -> 3
2 -> 1
2 -> 3
1 -> 3
```

Ejercicio 3

Modificar el siguiente programa, para que los valores del arreglo sean ingresados con la función **carga**.

```c
#include <stdio.h>

#define TAM 500

int main(void)
{
  int i, n;
  int arreglo[TAM];

  printf("Ingrese la cantidad de elementos: ");
  scanf("%d", &n);

  for (i = 0; i < n; i++)
  {
    printf("Ingrese el elemento [%d]: ", i);
    scanf("%d", &arreglo[i]);
  }

  for (i = 0; i < n; i++)
    printf("%d\n", arreglo[i]);

  return 0;
}
```

Prototipo de la función **carga**

```c
void carga(int a[], int n);
```

Ejercicio 4

Modificar el programa anterior para que la impresión del arreglo se realice con la función con prototipo:

```c
void imprime(int a[], int n);
```

Ejercicio 5

Modificar el programa anterior para que el arreglo ingresado se ordene de mayor a menor con la función con prototipo:

```c
void ordenar(int a[], int n);
```

Ejercicio 6

Modificar el programa del Ejercicio 4, para que se imprima la cantidad de números primos en el arreglo. Utilizar los prototipos:

```c
int es_primo(int num);
int contar_primos(int a[], int n);
```

Ejercicio 7

Modificar el programa del Ejercicio 4, para que se imprima el mayor y el menor de los números en el arreglo, utilizando los prototipos:

```
int mayor(int a[], int n);
int menor(int a[], int n);
```