

## Práctica 4. Estructuras de control iterativas

### Objetivos de la práctica:

- Adquirir la capacidad de diseñar algoritmos para la resolución de problemas básicos.
- Conocer cómo se traduce un algoritmo escrito en pseudocódigo en un lenguaje de programación.

### 1. Conceptos fundamentales

En las prácticas anteriores vimos qué etapas teníamos que seguir cuando programamos en lenguaje C:

- En papel, diseñar el algoritmo que resuelve el problema que queremos resolver.
- Traducir el algoritmo a lenguaje C. Para ello usaremos un editor de textos como por ejemplo el *gedit*.
- Compilar el programa para obtener un ejecutable. Vamos a utilizar el compilador *gcc*. El comando a utilizar sería: `$gcc -o ejecutable fuente.c`
- Ejecutar el programa y probar que su funcionamiento es el deseado. Emplearemos el comando `$/ejecutable`

### 2. Operaciones

En prácticas anteriores hemos visto que se puede operar con las variables:

- `a=b+c;` //Guarda en `a` la suma de `b` y `c`
- `a=b-c;` //Guarda en `a` la resta de `b` y `c`
- `a=b*c;` //Guarda en `a` el producto de `b` y `c`
- `a=b/c;` //Guarda en `a` la división de `b` y `c`
- `a=b%c;` //Guarda en `a` el resto de la división entera de `b` entre `c`

Cuando el operando es el mismo que el destino, se puede abreviar de la siguiente forma:

- `a=a+b;`
- `a+=b;`

Ambas expresiones son equivalentes.

Otras expresiones abreviadas que se pueden realizar son las siguientes:

- `a++`; // Equivale a `a=a+1`;
- `a--`; // Equivale a `a=a-1`;
- `b=a++`; // Equivale a `b=a`; `a=a+1`;
- `b=++a`; // Equivale a `a=a+1`; `b=a`;

### 3. Estructuras de control iterativas: bucles

Permiten definir fragmentos de un programa que se repiten (bucles). Hay que controlar que no sea un bucle infinito:

- Controlando una condición en cada iteración.
- Contando el número de iteraciones (uso de contadores).

La estructura de los bucles puede ser:

- Mientras se cumpla una condición, haz unas instrucciones.
- Haz unas instrucciones mientras se cumpla una condición.
- Haz unas instrucciones desde una condición inicial hasta que se alcance una condición final.

#### 3.1. Bucle while

La estructura sería:

```
MIENTRAS condición HACER
    instrucciones
FINMIENTRAS
```

Y en C corresponde con:

```
while (condición){
    //instrucciones
}
```

Las instrucciones se ejecutan 0 o más veces.

Un ejemplo sería el siguiente:

```
int n=3;
while (n>0){
    printf("Iteración%d", n);
    n--;
}
```

Se ejecuta 3 veces el bucle, para `n` con valor 3, 2 y 1.

### 3.2. Bucle do-while

La estructura sería:

```
HACER
    instrucciones
MIENTRAS condición
```

Y en C corresponde con:

```
do{
    //instrucciones
}while (condición);
```

Las instrucciones se ejecutan 1 o más veces.

Un ejemplo sería el siguiente:

```
int n=3;
do{
    printf("Iteración%d", n);
    n--;
}while (n>0);
```

Se ejecuta 3 veces el bucle, para n con valor 3, 2 y 1.

### 3.3. Bucle for

La estructura sería:

```
PARA condición_inicial MIENTRAS condición HACER
    instrucciones
FINPARA
```

Y en C corresponde con:

```
for(condición_inicial; condición; actualización_contadores){
    //instrucciones
}
```

Las instrucciones se ejecutan 0 o más veces.

Un ejemplo sería el siguiente:

```
int n;
for(n=3;n>0;n--){
    printf("Iteración%d", n);
}
```

Se ejecuta 3 veces el bucle, para n con valor 3, 2 y 1.

## 4. Algoritmos

### 4.1. Algoritmo Suma\_n\_primeros\_nums\_v1

El algoritmo siguiente calcula la suma de los  $n$  primeros números enteros:

ALGORITMO Suma\_n\_primeros\_nums\_v1

ENTRADAS:

Num: Entero ; Números a sumar

SALIDAS:

Total: Entero ; Suma de los n primeros números

```
VARIABLES:
  Num: Entero
  Total: Entero
  i: Entero ; Contador
INICIO
  ESCRIBA "Escribe cuantos números quieres sumar: "
  LEA Num
  SI  $\text{Num} \geq 1$  ENTONCES
    Total  $\leftarrow 0$ 
    i  $\leftarrow 1$ 
    MIENTRAS  $i \leq \text{Num}$  HACER
      Total  $\leftarrow \text{Total} + i$ 
      i  $\leftarrow i + 1$ 
    FINMIENTRAS
    ESCRIBA "La suma es: "
    ESCRIBA Total
  SINO
    ESCRIBA "El número ha de ser mayor o igual a 1"
  FINSI
FIN
```

## 4.2. Algoritmo Potencia

El algoritmo siguiente calcula la potencia de un número elevado a un exponente:

```
ALGORITMO Potencia
  ENTRADAS:
    Base: Entero ; Número leído (base)
    Exp: Entero ; Número leído (exponente)
  SALIDAS:
    Pot: Entero ; Potencia (Base elevado a Exp)
  VARIABLES:
    Base: Entero
    Exp: Entero
    Pot: Entero
    i: Entero
  INICIO
    ESCRIBA "Escribe un número (base): "
    LEA Base
    ESCRIBA "Escribe un número (exponente): "
    LEA Exp
    SI  $\text{Exp} \geq 1$  Y  $\text{Base} \geq 1$  ENTONCES
      Pot  $\leftarrow 1$ 
      i  $\leftarrow 1$ 
      MIENTRAS  $\text{Exp} \geq 1$  HACER
        Pot  $\leftarrow \text{Pot} * \text{Base}$ 
        Exp  $\leftarrow \text{Exp} - 1$ 
      FINMIENTRAS
    ESCRIBA "La potencia es: "
    ESCRIBA Pot
```

```
SINO
    ESCRIBA "La base y el exponente han de ser mayores o iguales
a 1"
FINSI
FIN
```

## 5. Traducción a C

### 5.1. Programa Suma\_n\_primeros\_nums\_v1

Si traducimos el algoritmo Suma\_n\_primeros\_nums\_v1 a lenguaje C nos quedaría el programa siguiente:

```
//Librería que contiene las funciones scanf y printf
#include <stdio.h>

//Función principal del programa
int main ()
{
    // Este programa calcula la suma de los primeros n números

    // Declaro las variables de mi función
    int Num, Total, i;

    //Sustituyo la función ESCRIBA "cadena" por printf
    printf("Escribe cuantos números quieres sumar: ");

    //Sustituyo la función LEA Num por scanf ("%d", &variableEntera);
    scanf("%d", &Num); //Guarda el número leído en la variable Num

    //Inicializo las variables
    Total = 0;
    i = 1;

    //Compruebo si el número introducido es mayor o igual que 1
    if (Num >= 1) {
        //Hago el bucle para sumar hasta que i valga n
        while (i <= Num){
            //Calculo la suma parcial
            Total = Total + i;
            i = i + 1;
        }

        //Sustituyo la función ESCRIBA "cadena" por printf
        printf("La suma es:%d \n", Total);
    }
    else{
        //Sustituyo la función ESCRIBA "cadena" por printf
        printf("El número ha de ser mayor o igual a 1");
    }
}
```

```
    }  
    //Fin del programa  
    return 0;  
}
```

## 5.2. Programa Potencia

La traducción del algoritmo Potencia a lenguaje C sería la siguiente:

```
//Librería que contiene las funciones scanf y printf  
#include <stdio.h>  
  
//Función principal del programa  
int main ()  
{  
    // Este programa calcula la potencia de un número  
  
    // Declaro las variables de mi función  
    int Base, Exp, Pot, i;  
  
    //Sustituyo la función ESCRIBA "cadena" por printf  
    printf("Escribe un número (base): ");  
  
    //Sustituyo la función LEA Base por scanf ("%d", &variableEntera);  
    scanf("%d", &Base); //Guarda el número leído en la variable Base  
  
    //Sustituyo la función ESCRIBA "cadena" por printf  
    printf("Escribe un número (exponente): ");  
  
    //Sustituyo la función LEA Exp por scanf ("%d", &variableEntera);  
    scanf("%d", &Exp); //Guarda el número leído en la variable Exp  
  
    //Compruebo que la base y el exponente sean mayores que 1  
    if ((Base >= 1) && (Exp >= 1)){  
        //Inicializo las variables  
        Pot = 1;  
        i = 1;  
        while (Exp >= 1){  
            Pot = Pot * Base;  
            Exp = Exp - 1;  
        }  
        //Sustituyo la función ESCRIBA por printf  
        printf("La potencia es:%d", Pot);  
    }  
    else{  
        //Sustituyo la función ESCRIBA "cadena" por printf  
        printf("La base y el exponente han de ser mayores o iguales  
a 1");  
    }  
}
```

```
    }

    //Fin del programa
    return 0;
}
```

## 6. Ejercicios propuestos

### 6.1. Ejercicio 1

- Escribe en un archivo con extensión `.c` el programa `Suma_n_primeros_nums_v1`. Compíllalo con el compilador `gcc`. Después, ejecútalo para comprobar su funcionamiento. ¿Cómo sería el código usando una estructura de tipo *do-while*? ¿Y un bucle *for*?

### 6.2. Ejercicio 2

- Escribe en un archivo con extensión `.c` el programa `Potencia`. Compíllalo con el compilador `gcc`. Después, ejecútalo para comprobar su funcionamiento. ¿Cómo sería el código usando una estructura de tipo *do-while*? ¿Y un bucle *for*?