Licenciatura en Sistemas
Programación de computadoras

Equipo docente: Jorge Golfieri, Natalia Romero, Romina Masilla y Nicolás Perez
Mails: jgolfieri@hotmail.com , nataliab_romero@yahoo.com.ar , romina.e.mansilla@gmail.com, nperez_dcao_smn@outlook.com
Facebook: https://www.facebook.com/groups/171510736842353
Git: ***http://***github.com/UNLASistemasProgramacion/Programacion-de-Computadoras

---

Unidad 2:

Estructuras de control: bifurcación y selección (if, switch). Funciones: tipos de parámetros y retorno; argumentos y valores de retorno. Problemas: implementación y testeo.

Bibliografía citada:

Fundamentos de programación: libro de problemas. Algoritmos, estructuras de datos y objetos (2a. ed.) Luis Joyanes Aguilar, Luis Rodríguez Baena y Matilde Fernández Azuela. McGraw-Hill España. Capítulo 3-4.

C algoritmos, programación y estructuras de datos. Luis Joyanes Aguilar, Andrés Castillo Sanz, y Lucas Sánchez García. Capítulo 3-5-7.

Unidad 2: Estructuras de control

Hasta ahora hemos visto que la programación se basa en realizar operaciones en algún determinado orden, por ejemplo, ingresar un número, luego ingresar otro, luego sumarlos y luego mostrar el resultado de dicha suma. Esta secuencia de pasos ha sido muy clara y se bifurca en un único camino hasta el momento. Pero la realidad es que no siempre el orden en que uno realiza las operaciones en tan claro y tan poco dependiente de la situación.

Ahora estudiaremos las estructuras selectivas que se utilizan para controlar el orden en que se ejecutan las sentencias de un programa, en particular llegamos a ver la sentencia IF (o si, en castellano).

El IF es una estructura selectiva, es decir se selecciona que operación realizar si se cumplen determinadas condiciones.

Un ejemplo clásico para explicar la estructura del IF es la solución de una ecuación lineal de primer grado.

Es decir a.x + b = 0, claramente si a y b son distintas de cero, el resultado de la x es –b/a.

En pseudocódigo se lo podría expresar así:

```
algoritmo RESOL1
var
  real : a, b, x
inicio
  leer (a, b)
  si a <> 0 entonces
     x ← -b/a
     escribir(x)
  si_no
     si b <> 0 entonces
         escribir ('solución imposible')
     si_no
         escribir ('solución indeterminada')
     fin_si
  fin_si
fin
```

En programación, la estructura de selección es un tipo de estructura de control. También llamada estructura de decisión o estructura selectiva.

En una estructura de selección/decisión, el algoritmo al ser ejecutado toma una decisión, ejecutar o no ciertas instrucciones si se cumplen o no ciertas condiciones. Las condiciones devuelven un valor, verdadero o falso, determinado así la secuencia a seguir.

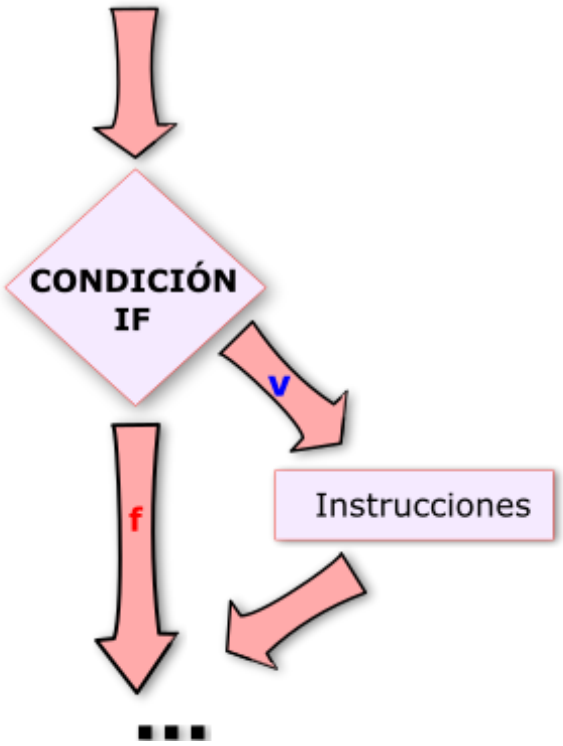
Básicamente hay tres tipos de estructuras de selección:

- Estructura de selección simple: if
- Estructura de selección doble: if-else
- Estructura de selección múltiple: case o switch

Por lo general los lenguajes de programación disponen de dos estructuras de este tipo: estructura de decisión simple (if), y estructura de decisión múltiple (CASE, SWITCH).

Los otros dos tipos de estructuras de control son: estructura de secuencia, y estructura de repetición.

Ejemplo de Estructura de selección simple IF:



Estructura de selección simple: IF

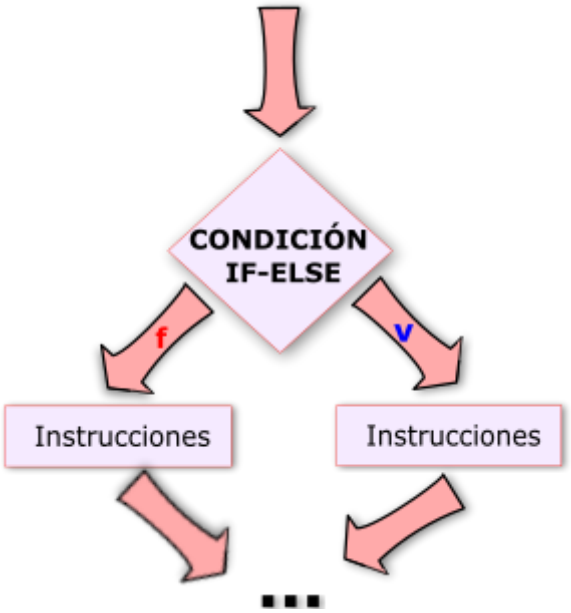
IF (edad > 18)

{

mostrar "Es mayor de edad"

}

Ejemplo de Estructura de selección doble IF-ELSE



Estructura de selección doble: IF-ELSE

IF (edad > 18)

{

mostrar "Es mayor de edad"

}

ELSE

{

mostrar "Es menor de edad"

}

Ahora bien, este ejemplo puede resultar poco ilustrativo, así que veamos algunos ejemplos como los que estuvimos trabajando con las PCs.

Ejercicio 1: Este programa es para determinar si un número ingresado por teclado es par o impar. Es decir SI la división por DOS es CERO, ENTONCES es un número par, SINO es impar.

```
#include <stdio.h>
#include <stdlib.h>
// Determinar si un numero es par o impar
int main() {
   int num; int respuesta;
   printf("Programa para determinar naturaleza par o impar de un numero\n\n");
   printf ("Introduzca un numero entero: ");
   scanf ("%d", &num);
   //El comando %2 se queda con el resto de la división por 2
   //Es decir si es 5 dividido 2, nos da el 1
   res = num%2;
   if (res==0) {  //Si el resto es cero, entonces es par
      printf ("El numero es par\n");
   } else { //de no ser asi es impar
      printf ("El numero es impar\n");
   }

   //Recuerden que las { } no son necesarias en caso de usar una sola sentencia en el //if
   return 0;
```

```
}
```

Ejercicio 2: Un programa que pida 3 números y los muestre de menor a mayor.

Para resolver esto, lo que realizamos son IF dentro de otros IF, lo que se llaman IF anidados. El código quedaría algo así:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   int num1,num2,num3;

   printf("Introduzca número 1:");
   scanf("%d",&num1);

   printf("Introduzca número 2:");
   scanf("%d",&num2);

   printf("Introduzca número 3:");
   scanf("%d",&num3);

   if (num1<num2 && num1<num3)
   {
      if (num2<num3)
      {
```

```
        printf("%d",num1);
        printf("%d",num2);
        printf("%d",num3);
      }
     else
      {
        printf("%d",num1);
        printf("%d",num3);
        printf("%d",num2);
      }
   }
  else if (num2<num1 && num2<num3)
   {
     if (num1<num3)
      {
        printf("%d",num2);
        printf("%d",num1);
        printf("%d",num3);
      }
     else
      {
        printf("%d",num2);
        printf("%d",num3);
        printf("%d",num1);
      }
   }
  else if (num3<num1 && num3<num2)
```

```
    {
        if (num1<num2)
        {
         printf("%d",num3);
         printf("%d",num1);
         printf("%d",num2);
        }
```

Ejercicio 2.b: Los invitamos a que lo hagan con 4 números y de mayor a menor ahora.

Ejercicio 3: Que pida un número del 1 al 7 y diga el día de la semana correspondiente.

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
            int i;
            printf("Introduzca número del 1 al 7:");
                scanf("%d",&i);

if (i==1) printf ("Es Lunes");

if (i==2) printf ("Es Martes");

if (i==3) printf ("Es Miercoles");
```

```
if (i==4) printf ("Es Jueves");

if (i==5) printf ("Es Viernes");

if (i==6) printf ("Es Sabado");

if (i==7) printf ("Es Domingo");



    system("PAUSE");
    return 0;
}
```

Vemos que usando los IF, la estructura queda un tanto repetitiva y con demasiadas condiciones, esto nos sirve para abrir las puertas a una nueva estructura selectiva llamada ***SWITCH CASE***. Donde separamos en casos según el ingreso por teclado. El código quedaría algo así:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        int i;
        printf("Introduzca número del 1 al 7:");
    scanf("%d",&i);
```

```
    switch(i){
            case 1:
                printf ("Lunes\n");
                break;
            case 2:
                printf ("Martes\n");
                break;
            case 3:
                printf ("Miércoles\n");
                break;
            case 4:
                printf ("Jueves\n");
                break;
            case 5:
                printf ("Viernes\n");
                break;
            case 6:
                printf ("Sábado\n");
                break;
            case 7:
                printf ("Domingo\n");
                break;
            default:
                printf ("Opción no válida\n");
                break;
    }
```

```
    system("PAUSE");
    return 0;
}
```

Funciones y procedimientos

Ya podemos solucionar problemas bastante complejos, pero esta solución puede ser muy extensa y difícil de comprender y seguir por un lector de nuestro código.

Para lograr que los programas sean más simples, entendibles, más fáciles de modificar, más fácil la cooperación entre distintos programadores y sobre todas las cosas para que los códigos sean reutilizables en futuros códigos, aparece el concepto de FUNCION, donde cada función se utiliza para solucionar pequeños problemas y así desgranar un gran problema en suma de pequeños problemas. Es decir, cuando tengamos que resolver un ejercicio debemos pensar dicho ejercicio como pequeños sub ejercicios e irlos atacando uno por uno hasta lograr el objetivo final.

Como vimos, C tiene como bloque básico la función `main()` , también hemos visto la sentencia `printf()` que es otra función, y de igual forma hay muchas más funciones predefinidas, pero nosotros mismos también podemos definir nuestras propias funciones. De hecho, es fundamental hacerlo.

Podemos definir una función cualquiera de la misma manera en que definimos la función `main()`. Basta con poner su tipo, su nombre, sus argumentos entre paréntesis y luego, entre llaves, su código:

```
/* Inclusión de archivos */
#include <stdio.h>
```

```
void holamundo(void) /* Función donde se ejecuta la lógica del programa */
{
        printf("Hola Mundo\n"); /* imprime la cadena */
        return; /* sale de la función */
}

int main(void) /* Función principal del programa */
{
        holamundo(); /* llamada a la función holamundo */
        return 0; /* sale del programa con código 0 (correcto) */
}
```

¿Cómo es el formato de una función/procedimiento?

Las funciones dan como resultado un único valor, ese valor es el denominado RETORNO, el retorno puede ser int, char, float, doublé, char[], un vector, una matriz o incluso estructuras más complejas que aún no hemos visto.

Lo primero que se pone para construir una función es SU RETORNO, es decir si se pone int, el retorno o resultado de nuestra función será un entero.

Luego de declarar el tipo de retorno hay que dar el NOMBRE de la función, podemos llamarla como queramos, siempre y cuando ese nombre no esté en uso y respete los standares de la programación. Se recomienda poner siempre un nombre que defina lo que hace la función.

Por último, y entre paréntesis, hay que establecer con los tipos de datos que va a trabajar esta función, es decir los ARGUMENTOS.

Entonces, por ejemplo, la función int CalcularEdad (int nacimiento, int anio_actual), es una función que RETORNA UN VALOR ENTERO, es decir nos da como resultado la edad. La función SE LLAMA CalcularEdad; y opera con las variables de entrada enteras nacimiento y anio_actual. Esto quiere decir que usando el nacimiento y el año actual, se opera internamente para dar como resultado nuestra edad.

Un último comentario por demás necesario es aclarar que las funciones de declaran arriba del segmento del main y afuera del mismo.

La función antes mencionada quedaría así:

<librerías>

```
int CalcularEdad(int nacimiento, int anio_actual){
int edad = 0;
edad = anio_actual – nacimiento;
return edad
}

int main(){

//llamo a la función
int miedades = CalcularEdad(1990, 2018);
//también podríamos:
//int a = 1990;
```

```
//int b = 2018;

//int miedades = CalcularEdad(a,b);


}
```

Notar que podemos ingresar a las funciones con valores definidos como el 2018 o el 1990, también ver que los nombres de las variables internas al main no necesariamente deben ser los mismos que dentro de las funciones.

Ejercicio: Resuelto – Se piden dos números, desarrollar una función que compare dichos números y devuelva el mayor de ellos.

Respuesta:

```
#include <stdio.h>

int compara( int a, int b )     /* Metemos los parámetros a y b a la función */
{
        int mayor;                      /* Esta función define su propia variable;
                                           esta variable sólo se puede usar aquí */
        if ( a>b )
                mayor = a;
        else
                mayor = b;

        return mayor;
}


main()
{
```

```
        int num1, num2;
        int resultado;

        printf( "Introduzca dos números: " );
        scanf( "%i %i", num1, num2 );

        resultado = compara( num1, num2 );  /* Almacenamos en resultado el valor
                                               que devuelve la función */

        printf( "El mayor de los dos es %i\n", resultado );
}
```

Practica Unidad 2:

1- Escribir un programa que solicite el ingreso de la temperatura en grados, si la temperatura está por encima de 100 grados desplegar el mensaje “arriba del punto de ebullición del agua”, de lo contrario desplegar el mensaje “abajo del punto de ebullición del agua”.

2- Escribir un programa que escriba la calificación correspondiente a una nota, de acuerdo con el siguiente criterio:

Condición	Mensaje
0 a < 5.0	Suspenso
5 a < 6.5	Aprobado
6.5 a < 8.5	Notable
8.5 a < 10	Sobresaliente
10	Matrícula de honor

3- Escribir un programa que simule un calculador simple. Lee dos enteros y un carácter. Si el carácter es un +, se imprime la suma; si es un -, se imprime la diferencia; si es un *, se imprime el producto; si es un /, se imprime el cociente.

4- Realizar nuevamente el ejercicio 3, pero utilizando un procedimiento de este tipo: **void calculadora(int numero1, int numero2, char símbolo)**

5- Realizar una función que nos diga cuantos segundos pasaron en un determinado día, dada la hora de ese día, la función debe ser: **int calcularSegundos (int hora, int minutos, int segundos)**