E.E.S.T. Nº 1 Lomas de Zamora, 6to 2da

Lenguajes Electrónicos (2024)

Profesor Nicolás Leguiza

Profesor Sebastián E. Millán

Unidad temática 1: Estructuras selectivas simples y dobles

1. Introducción

Las estructuras selectivas evalúan una condición, y de acuerdo con el resultado de esa evaluación eligen uno de entre varios caminos. Estas estructuras se utilizan cuando se desea que el algoritmo tome una decisión lógica.

Existen tres tipos de estructuras selectivas: **simples**, **dobles** y **múltiples**.

2. Estructuras selectivas simples

Las estructuras selectivas simples evalúan una única condición. Si el resultado de esa evaluación es verdadero se realizarán un conjunto de acciones, pero si el resultado es falso no se realizará nada.

A continuación, se muestra el diagrama de flujo y su sintaxis en C de una estructura selectiva simple:

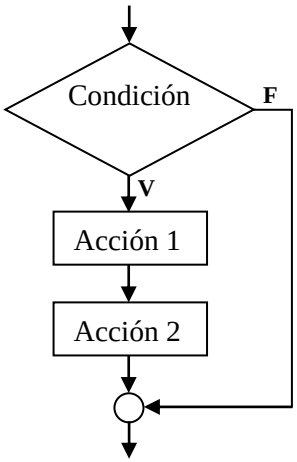


Diagrama de flujo de una estructura selectiva simple

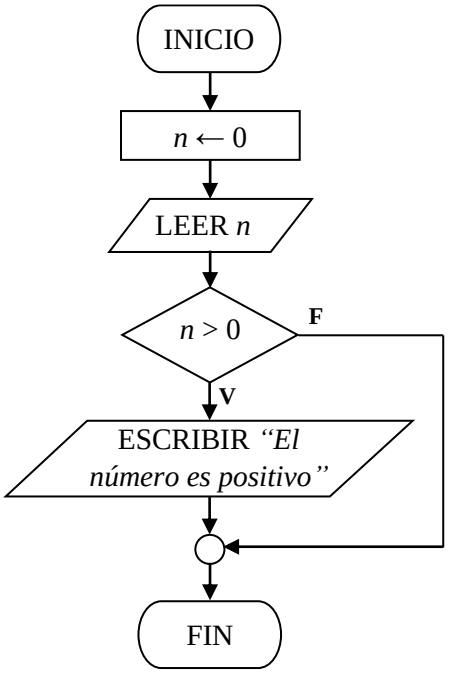
```
if(condicion)
{
  accion1;
  accion2;
}
```

Sintaxis en C en una estructura selectiva simple

Observar que, en C la estructura selectiva simple se realiza mediante la instrucción "*if*", la cual significa "*si*" en inglés. Para comprender como se aplica esta estructura para resolver problemas, se muestra a continuación un ejemplo:

Ejemplo: Se desea realizar un algoritmo que indique si un número es positivo. El valor del número se debe ingresar por teclado, y el mensaje se debe imprimir en pantalla. Realizar el diagrama de flujo y programa en C del algoritmo.

Solución: El algoritmo deberá, en primera instancia, declarar la variable que se va a utilizar (*n*). Luego, deberá leer el dato de entrada, realizar las comparaciones lógicas necesarias, y escribir el resultado si así correspondiera. El diagrama de flujo del algoritmo es el siguiente:



Inicia el programa.

Se declara la variable "*n*" y se le asigna el valor "0".

Se lee el valor de "*n*".

Se realiza la comparación lógica "*n* > 0" ("*n* mayor que cero") para determinar si el número es positivo, ya que los números positivos son mayores a cero.

Si "*n* > 0" es verdadero, se escribirá el mensaje "El número es positivo". Si "*n* > 0" es falso, no se efectuará ninguna acción.

Finaliza el programa.

Diagrama de flujo del algoritmo

Una vez realizado el diagrama de flujo del algoritmo, se deberá expresar en un lenguaje de programación. El lenguaje pedido por la consigna es el C:

```c
#include <stdio.h> /* Añade la librería "standard input-output
                      header" que contiene operaciones de
                      entrada/salida tales como "scanf" y "printf". */
main()     /* "main" es la función principal, donde está
              alojado el programa principal. */
{
   int n = 0; /* Declara la variable entera "n" y le asigna el valor
                 "0". */
   scanf("%d", &n); /* Ingresa un número por teclado y lo almacena en
                       la variable "n". */
   if(n>0)     /* Realiza la comparación lógica "n>0". */
   {           /* Si la comparación es verdadera, imprime el mensaje.*/
       printf("El numero es positivo.");
   }
}
```

Programa en C del algoritmo

3. Operadores relacionantes y Operadores lógicos

En el ejemplo anterior, la condición de la estructura selectiva fue una *comparación lógica*. La comparación *n* > 0 evalúa si *n* es mayor que 0. Si *n* es mayor que 0, el resultado es verdadero. Si *n* no es mayor que 0, el resultado es falso.

El operador “>” utilizado en la comparación es un *operador relacionante*, y no es el único que existe. A continuación, se detallan todos los operadores relacionantes que se pueden utilizar:

Operadores relacionantes			
Operador	**Relación**	**Ejemplo**	**Resultado**
<	Menor	*a* < *b*	Si *a* es menor que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.
>	Mayor	*a* > *b*	Si *a* es mayor que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.
<=	Menor o igual	*a* <= *b*	Si *a* es menor o igual que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.
>=	Mayor o igual	*a* >= *b*	Si *a* es mayor o igual que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.
==	Igual	*a* == *b*	Si *a* es igual que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.
!=	Distinto	*a* != *b*	Si *a* es distinto que *b*, el resultado de la comparación es *verdadero*. Caso contrario, el resultado es *falso*.

Cabe mencionar, que un resultado verdadero en C equivale a un valor distinto de 0, mientras que un resultado falso equivale a un valor igual a 0. También, en C se pueden representar a los estados verdadero y falso con las palabras clave **true** y **false** respectivamente.

Además de los operadores relacionantes, también existen los *operadores lógicos*. Estos operan con estados lógicos, y entregan como resultado otro estado lógico. A continuación, se detallan todos los operadores lógicos que se pueden utilizar:

Operadores lógicos			
Operador	**Acción**	**Ejemplo**	**Resultado**
&&	AND lógico	*a* && *b*	El resultado es verdadero si *a* y *b* son verdaderos.
||	OR lógico	*a* || *b*	El resultado es verdadero si *a* ó *b* son verdaderos.
!	NOT lógico	!*a*	El resultado es la negación lógica de *a*.

4. Estructuras selectivas dobles

Las estructuras selectivas dobles, también evalúan una única condición. Si el resultado de esa evaluación es verdadero se realizarán un conjunto de acciones, y si el resultado es falso se realizará otro conjunto de acciones.

A continuación, se muestra el diagrama de flujo y su sintaxis en C de una estructura selectiva doble:

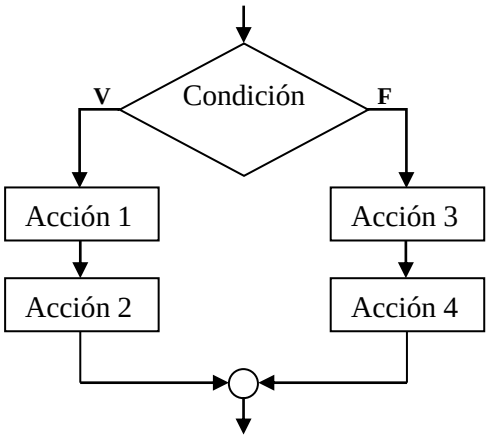


Diagrama de flujo de una estructura selectiva doble

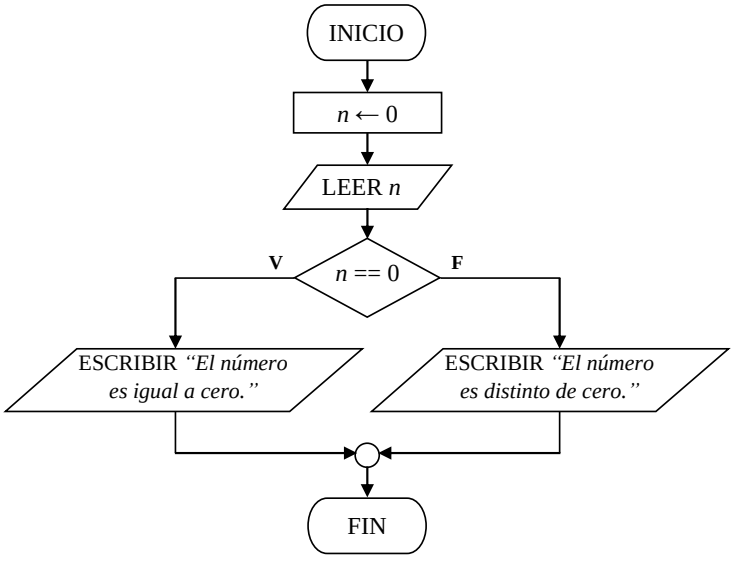
```
if(condicion)
{
  accion1;
  accion2;
}
else
{
  accion3;
  accion4;
}
```

Sintaxis en C de una estructura selectiva doble

Observar que, en C la estructura selectiva doble se realiza con las palabras "*if*" y "*else*", las cuales significan "*si*" y "*si no*" en inglés. Para comprender como se aplica esta estructura para resolver problemas, se muestra a continuación un ejemplo:

Ejemplo: Se desea realizar un algoritmo que indique si un número es igual o distinto de cero. El valor del número se debe ingresar por teclado, y el mensaje se debe imprimir en pantalla. Realizar el diagrama de flujo y programa en C del algoritmo.

Solución: El algoritmo deberá, en primera instancia, declarar las variables que se van a utilizar (*n*). Luego, deberá leer el dato de entrada, realizar las comparaciones lógicas necesarias, y escribir el resultado que corresponda. El diagrama de flujo del algoritmo es el siguiente:



Inicia el programa.

Se declara la variable "*n*" y se le asigna el valor "0".

Se lee el valor de "*n*".

Se realiza la comparación lógica "*n* == 0", para determinar si el número es igual o distinto de cero.

Si "*n* == 0" es verdadero, se escribirá el mensaje "El número es igual a cero". Si "*n* == 0" es falso, se escribirá el mensaje "El número es distinto de cero".

Finaliza el programa.

Diagrama de flujo del algoritmo

Una vez realizado el diagrama de flujo del algoritmo, se deberá expresar en un lenguaje de programación. El lenguaje pedido por la consigna es el C:

```c
#include <stdio.h> /* Añade la librería "standard input-output
                      header" que contiene operaciones de
                      entrada/salida tales como "scanf" y "printf". */
main()     /* "main" es la función principal, donde está
                      alojado el programa principal. */
{
   int n = 0; /* Declara la variable entera "n" y le asigna el valor
                 "0". */
   scanf("%d", &n); /* Ingresa un número por teclado y lo almacena en
                       la variable "n". */
   if(n==0)    /* Realiza la comparación lógica "n==0". */
   {           /* Si la comparación es verdadera, imprime el mensaje:*/
       printf("El numero es igual a cero.");
   }
   else
   {           /* Si la comparación es falsa, imprime el mensaje:*/
       printf("El numero es distinto de cero.");
   }
}
```

Programa en C del algoritmo

No se debe dejar de tener en cuenta que la sintaxis es determinante para que el programa pueda ser compilado. La condición debe ir entre paréntesis, y las acciones deben ir entre llaves.

Ejercicios

1) Diseñar un algoritmo que, a partir de la edad de una persona, determine si es o no mayor de edad. La edad de la persona se debe ingresar por teclado y el resultado se debe imprimir en pantalla. Dibujar el diagrama de flujo y escribir el programa en C del algoritmo.

2) Diseñar un algoritmo que realice la división de dos números decimales. Los números se deben ingresar por teclado y el resultado se debe imprimir en pantalla. En caso de que el divisor sea igual a cero, se deberá imprimir en pantalla el mensaje "Los datos ingresados no son válidos" (ya que en matemática no está definida la división por cero). Dibujar el diagrama de flujo y escribir el programa en C del algoritmo.

   Ayuda: Para dividir dos números se puede utilizar el *operador aritmético* "/". A continuación, se muestra una tabla con algunos de los operadores aritméticos que existen en C:

Operadores aritméticos			
Operador	**Acción**	**Ejemplo**	**Resultado**
+	Suma	x = 5 + 3	x vale 8
-	Resta	x = 5 - 3	x vale 2

*	Multiplicación	x = 5 * 3	x vale 15
/	División	x = 12 / 3	x vale 4
%	Resto	x = 11 % 3	x vale 2
++	Incremento	x = 1; x++	x vale 2
--	Decremento	x = 1; x--	x vale 0

3) Diseñar un algoritmo que, a partir de la calificación de un alumno, determine si está "aprobado", "desaprobado" o "aplazado". Tener en cuenta que las calificaciones están comprendidas entre 0 y 10, según la siguiente tabla:

Nota del alumno	**Condición del alumno**
0 a 3	Aplazado
4 a 6	Desaprobado
7 a 10	Aprobado

La calificación del alumno se debe ingresar por teclado y el resultado se debe imprimir en pantalla. Dibujar el diagrama de flujo y escribir el programa en C del algoritmo.

Ayuda: Si es necesario evaluar más de una condición, se pueden recurrir a las *estructuras anidadas*. Las estructuras anidadas, son estructuras que dentro tienen otra estructura. A continuación, se muestra el diagrama de flujo y la sintaxis de una estructura anidada con dos estructuras selectivas dobles, una dentro de la otra:

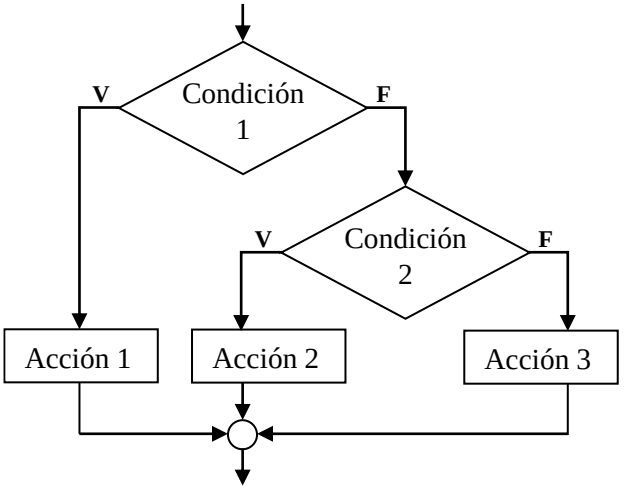


Diagrama de flujo de dos estructuras selectivas dobles anidadas

```
if(condicion1)
{
  accion1;
}
else
{
  if(condicion2)
  {
    accion2;
  }
  else
  {
    accion3;
  }
}
```

Sintaxis en C++ de dos estructuras selectivas dobles anidadas

4) Diseñar un algoritmo que, a partir de una letra, determine si es o no una vocal. La letra se debe ingresar por teclado y el resultado se debe imprimir en pantalla. Dibujar el diagrama de flujo y escribir el programa en C del algoritmo.

Ayuda 1: Recordar que las letras son caracteres, y los mismos se almacenan en variables "char". Las variables "char" de un único carácter se declaran de la siguiente forma:

```
char letra; /* Declara la variable de tipo caracter "letra". */
```

Ayuda 2: Para implementar las instrucciones "scanf" y "printf" con variables de un único carácter, se utiliza el "*%c*", como se muestra a continuación:

```
scanf("%c", &letra); /* Esta instrucción permite ingresar por teclado un único caracter y lo almacena en la variable "letra". */

printf("La letra ingresada es: %c", letra); /* Esta instrucción imprime en pantalla el mensaje "La letra ingresada es: " seguido del caracter almacenado en la variable "letra". */
```

Ayuda 3: Para comparar una variable con un caracter en una condición, mediante algún operador, el caracter debe estar entre comillas simples. A continuación, se muestra un ejemplo de cómo debe ser la sintaxis dentro de la condición de una estructura selectiva para comprar caracteres:

```
if(letra=='a')  /* Realiza una comparación lógica entre el carácter*/ {      /*almacenado en la variable "letra" con el carácter 'a'. */
  printf("La letra ingresada es la 'a'.");
}
```

Ayuda 4 (opcional): A veces, se pueden simplificar los algoritmos mediante la implementación de operadores lógicos. En el siguiente ejemplo se ilustra la ventaja de utilizar operadores lógicos en las condiciones de las estructuras selectivas.

El siguiente programa imprime en pantalla un mensaje cuando un número ingresado por teclado es "0" o "1":

```
if(numero==0)
{
  printf("El numero ingresado es 0 o 1.");
}
if(numero==1)
{
  printf("El numero ingresado es 0 o 1.");
}
```

El programa evalúa dos condiciones, si el número es 0, y si el número es 1. En caso de que se cumpla cualquiera de las dos condiciones, se imprimirá en pantalla el mismo mensaje. Pero utilizando el operador lógico "||" (OR lógico), se puede resolver el mismo problema evaluando una única condición, de la siguiente manera:

```
if(numero==0 || numero==1)
```

```
{
  printf("El numero ingresado es 0 o 1.”);
}
```

El resultado de evaluar la condición de la estructura selectiva será verdadero, cuando sea verdadera cualquiera de las condiciones “numero==0” o “numero==1”. De esta forma, se obtiene el mismo resultado, pero simplificando sustancialmente la elaboración del programa.

5) Diseñar un algoritmo que ordene de menor a mayor tres números. Los números se deben ingresar por teclado y el resultado se debe imprimir en pantalla. Dibujar el diagrama de flujo y escribir el programa en C del algoritmo.