Clase 4: Estructuras de control

Mauricio Gómez García

Índice

Introducción **2**

Definición **3**

Estructuras de control **4**

`if` 4

`else` 5

`else if` 6

`switch` 7

`while` 8

`for` 9

Conclusión **10**

Introducción

En la sesión pasada se dio a conocer el tema de las palabras claves. Dentro de las palabras clave, existe un tema importante para la elaboración de código llamado **estructuras de control**. Éstas nos sirven para indicar el flujo de las instrucciones que se realizarán dependiendo de los valores que utilizemos.

Es importante conocer el funcionamiento de las estructuras de control para poder realizar las implementaciones de la manera correcta y que sean más útiles al programador. De igual modo conocer su sintaxis y los métodos en los cuales operan ayudan para comprender el código a mayor profundidad.

El objetivo de la sesión es introducir al concepto de las estructuras de control, cómo definirlos y cómo utilizarlos dentro de la elaboración de código. De igual modo, se empezarán a ver las partes elementales de la lógica de programación, lo cual será súmamente útil para crear programas funcionales que cumplan los objetivos del programador.

Definición

Las estructuras de control son herramientas de código que nos ayudan para indicarle al programa el orden o secuencia en la que ciertas instrucciones se realizarán, y cuales instrucciones se llevarán a cabo según ciertas **variables de control**.

C cuenta con distintas estructuras de control, incluyendo (pero no limitado a) las siguientes:

- `if`: Realizar un bloque de código si una condición se cumple.
- `switch`: Ejecutar un bloque de código que coincida con una condición.
- `while`: Ejecutar un bloque de código mientras que una condición se cumpla.
- `for`: Ejecutar un bloque de código una cantidad de veces definida o calculada.

Como se puede observar, estas estructuras nos permiten realizar distintas operaciones o acciones dependiendo de ciertas variables o **condiciones** que se establecen en el programa.

Como se ha visto anteriormente, se pueden definir variables en nuestro código para utilizarse en el transcurso del programa. Estas variables se pueden utilizar también dentro de las estructuras de control para corroborar datos o condiciones sobre las cuales se realizarán distintas acciones o flujos de instrucciones.

Estructuras de control

En este apartado se verán las estructuras de control mencionadas préviamente, junto con un ejemplo de código por cada estructura. Los ejemplos se pueden compilar y ejecutar para observar su funcionamiento, y se proveerá la salida esperada por el programa con cada ejemplo.

if

Los bloques `if` se ejecutan si se cumple una condición, como su nombre lo indica. Seguido de la palabra clave `if` se debe de tener una declaración entre paréntesis `()` que contenga la condicional a checar, seguido de un bloque de código entre llaves `{}` con el código a ejecutar en caso que se cumpla la condicion.

```c
#include <stdio.h>

int main() {
    int x = 10;

    // declaracion de estructura if
    if (x==10) { // si x==10
        printf("x es igual a 10\n");
    }
    return 0;
}
```

En este ejemplo, se define una variable `x` con valor `10`. Seguido de esto, se declara una estructura `if` que valida si el valor de `x` es igual a `10` con la condicional `(x==10)` y si la condición se cumple, se imprime texto a la consola.

La salida esperada de este programa es la siguiente:

```
x es igual a 10
```

else

Opcionalmente, las estructuras `if` pueden tener cláusulas para ejecutar código en caso que **no** se cumpla la condicional. Esto se realiza con la palabra clave `else`. Después de esta palabra se inicia otro bloque con llaves `{}` con el código a ejecutar en caso que no se cumpla la condicional.

```c
#include <stdio.h>

int main() {
    int x = 15;

    // declaracion de estructura if
    if (x==10) { // si x==10
        printf("x es igual a 10\n");
    } else { // en cualquier otro caso
        printf("x no es igual a 10\n");
    }
    return 0;
}
```

Este código es muy similar a la prévia, pero en este ejemplo se checa si el valor de `x` es igual a `10` con la condicional `x==10`. Dependiendo del valor de la condicional (verdadero o falso) se ejecutan bloques de código distintos.

La salida de este programa es la siguiente:

```
x no es igual a 10
```

else if

Adicionalmente, pueden haber cláusulas para realizar múltiples comprobaciones una tras otra con la estructura de los **else if**:

```c
#include <stdio.h>

int main() {
    int x = 15;

    // declaracion de estructura if
    if (x==10) { // si x==10
        printf("x es igual a 10\n");
    } else if (x==15) { // si x==15
        printf("x es igual a 15\n");
    } else { // en cualquier otro caso
        printf("x no es igual a 10 ni 15\n");
    }

    return 0;
}
```

Igual que en los ejemplos prévios, tomamos una variable `x` y comprobamos su valor. Dependiendo del valor de esta variables se ejecutarán distintos bloques de código:

- Si `x` es igual a `10` imprimir `"x es igual a 10\n"`
- Si `x` es igual a `15` imprimir `"x es igual a 15\n"`
- En cualquier otro caso, imprimir `"x no es igual a 10 ni 15\n"`

La salida de este programa es la siguiente:

```
x es igual a 15
```

switch

Los bloques `switch` cumplen una función similar a la estructura de los bloques `if {} else if {}`. Estos toman un valor de entrada, y según una lista definida de casos se ejecutan bloques de código que coincidan con el valor de entrada. Cada caso se define con la palabra clave `case` seguido de dos puntos `:` y código a ejecutarse. Es importante notar que el bloque no se encapsula con llaves `{}` sino que se debe de terminar con la palabra clave `break`, el cual sirve para detener el flujo de bucles y ciertas estructuras de control.

```c
#include <stdio.h>

int main() {
    int x = 10;

    // declaracion de estructura switch
    switch (x){
        case 10: // si (x==10)
            printf("x es igual a 10\n");
            break;
        case 15: // si (x==15)
            printf("x es igual a 15\n");
            break;
        default: // cualquier otro caso
            printf("x no es igual a 10 ni 15\n");
            break;
    }

    return 0;
}
```

En este ejemplo se toma el valor de la variable `x`, y según su valor se ejecutan bloques de código distinto. Como se puede observar, se cumple una función similar a los bloques `if {} else if {}`, pero la sintaxis de esta estructura es más intuitiva para las personas. En general, ninguno de los dos es más rápido que el otro, y se pueden utilizar de manera intercambiable si se implementan de la manera correcta.

while

Las estructuras `while` nos sirven para definir bucles, los cuales son bloques de código que se repetiran una cantidad de veces. Existen dos tipos de bucles, y los bucles `while` se conocen como bucles infinitos, ya que es posible que dichos se ejecuten sin detenerse (o hasta matar el programa externamente).

Este tipo de bucle se utiliza cuando no se puede saber con anticipación cuándo queremos que termine el bucle, como al esperar que un usuario ingrese una tecla al programa o esperando a que un servidor responda con un código específico.

Los bucles `while` toman una condicional de entrada, y siempre y cuando el valor de la condicional sea verdadero se ejecutará el bloque de código repetidamente.

```c
#include <stdio.h>

int main() {
    int x = 0;

    // declaracion de bucle while
    while (x<5) {
        printf("x es menor a 5\n");
        x++
    }

    return 0;
}
```

En este programa, se define una variable `x` con valor `0`. Después se define una estructura de bucle `while`, en donde siempre y cuando el valor de `x` sea menor a `5` se imprimirá el texto `"x es menor a 5\n"`.

Igualmente, en cada iteración del bucle se aumenta el valor de `x` con la declaración `x++`. Esto permite que el bucle eventualmente termine, de otro modo se ejecutarían indefinidamente.

for

Los bucles `for` son el otro tipo de bucle, conocido como bucle finito. Este tipo de bucle se utiliza cuando se puede saber con anterioridad la cantidad de veces en que se repetirá el código del bucle. Su estructura es un poco más compleja que la del bucle `while`. Los bucles `for` se deben de declarar con un “iterador”, el cual tiene tres componentes:

- Inicializador: Una declaración que inicializa una, o menos comunmente varias variables para su uso en el iterador
- Condicional: Una condición a comprobarse en cada iteración, para verificar si el bucle continuará
- Acción: Una acción a tomarse en cada iteración, que servirá para continuar el bucle y eventualmente salir de la condicional

Un ejemplo básico utilizando un bucle `for` es el siguiente:

```c
#include <stdio.h>

int main() {
    // declaracion de bucle for
    for (int i=0; i<10; i++) {
        printf("Hola, Mundo!\n");
    }

    return 0;
}
```

En este caso, tenemos un bucle `for` que imprime el texto `"Hola, Mundo!\n"` 10 veces. Observemos cada parte del iterador:

- Inicializador (`int i=0`): Se inicializa una variable `int` de nombre `i` con valor `0`.
- Condicional (`i<10`): En cada iteración se comprueba si `i` es menor a `10`
- Acción (`i++`): En cada iteración el valor de `i` aumenta.

Observando las declaraciones y la función de cada parte del iterador, podemos seguir la lógica de cómo funciona para conocer que se busca realizar el mismo bloque de código 10 veces, segun la definición que hemos realizado.

Conclusión

Existen varias estructuras de control, y existen algunas más que no se han mencionado debido a que no son tan comunmente utilizados. Estas nos ayudan a controlar el flujo de nuestro programa, tomando y ejecutando distintas acciones según nuestras necesidades y definiciones. Conocer cada una de estas estructuras es fundamental para empezar a implementar la lógica de la programación y crear herramientas de software que sean flexibles a las necesidades del usuario.

Debido a los distintos usos que pueden tener las estructuras de control, es importante conocer cuándo es apropiado utilizar cada uno, para asi aprovechar al máximo el lenguaje y evitar crear programas demasiado complejos.

Por último, las estructuras de control son parte fundamental de todo programa y poder implementarlos de manera correcta asegura poder realizar código que siga el flujo que el programador requiera para alcanzar su fin.

Tarea opcional
Se recomienda practicar el utilizar las estructuras de control creando algún programa que los implemente. Abajo se dejan algunas ideas de programas que utilizen las estructuras vistas en la sesión, las cuales pueden servir como una base para practicar utilizar estas estructuras.

- Programa que cuente desde algún número hasta cero
- Programa que imprima si una edad (valor entero) es de mayoria de edad o no
- Programa que imprima la opción de un menu según el número que se asigne a cada entrada del menu
- Programa que imprima un valor numérico que aumente en cada iteración indefinidamente