INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Unidad II: Estructuras de Control y Aplicaciones con Arreglos.

Nombre: Arturo Matamoros Balderas

2019640534

Grupo: 1MV1

Materia: Introducción a la Programación

Introducción

Las sentencias de control que existen son:

- Secuencia
- Selección
- Repetición

Las sentencias de selección nos permiten ejecutar cierta parte del código siempre y cuando se cumpla una condición, existen dos estructuras de selección en c++ y switch-case.

If

Si la condición evaluada en la sentencia es verdadera se ejecutaba la acción, y si es falsa se ejecuta la acción.

Switch

La sentencia Switch case, Es una sentencia de control útil para seleccionar una de entre múltiples alternativas. El selector puede ser tipo int o char.

For

La sentencia for sirve para ejecutar un bloque de sentencias un número fijo de veces.

Do-While

La sentencia Do-While: se utiliza para especificar un bucle condicional que se ejecutaba al menos una vez.

Arreglos o vectores

Un vector es una secuencia de datos del mismo tipo. Los tipos de datos pueden ser de cualquier tipo, como: int, float, char, estructuras, etc.

Tiene un limite inferior que es 0 y un limite superior que es n-1, donde n es el numero de elementos almacenados en el vector.

Arreglos multidimensionales

Los arreglos multidimensionales son aquellas que tienen más de una dimensión y por lo tanto más de un índice.

Lo mas usuales son los de dos dimensiones conocidos como: **tablas** o **matrices.**

Objetivos

- Desarrollo de programas utilizando las sentencias de control.
- Desarrollar programas aplicando

Desarrollo

Bucle While.

El lenguaje C++ tiene varias estructuras de control para bucles y bifurcaciones condicionales. Empezaremos por el bucle WHILE. Este bucle continuará ejecutándose mientras (while significa mientras, en inglés) sea cierta la condición impuesta. Cuando esta condición no se cumpla, el bucle se parará. El nombre en sí ya es una buena descripción.

```
#include <iostream.h>       // Este es un ejemplo del bucle "while"

main()

{

int contador;

contador = 0;

while (contador < 6) {

cout <<"El valor del contador es " << contador << "\n";

contador = contador + 1;

}

}
```

Empezamos con un comentario y el nombre del programa. Tras esto procedemos a definir la variable entera “contador” en el cuerpo del programa. La variable tiene como valor inicial 0. Y ahora, vamos al bucle WHILE. La sintaxis es tal y como se describe ahora: la palabra reservada WHILE es seguida por una expresión o por algo entre paréntesis, seguido a su vez por un mandato compuesto, encerrado entre llaves. Tantas veces como sea cierta la condición, tantas veces se ejecutará este mandato. En este caso, la variable contadora irá incrementando su valor en 1 y, cuando alcance 6, el bucle dejará de ejecutarse. El control del programa pasará a la primera instrucción tras el mandato o mandatos ejecutados por el bucle.

Debemos destacar que:

1) Si la variable “contador” fuera mayor que 5, el bucle podría no ejecutarse jamás, (Ej.: si empezara con 6), ya que la comparación se realiza al iniciar el bucle.

2) Si la variable no se incrementa, el bucle no acabará de ejecutarse jamás.

3) Si el mandato dependiente del bucle es uno solo, no es necesario incluirlo entre llaves.

Bucle Do While

Es casi idéntico al anterior, excepto en que el bucle comienza con la palabra reservada “DO “, seguida por un bloque de mandatos encerrados entre llaves, luego la palabra reservada WHILE y, finalmente la expresión entre paréntesis. El bloque de comandos entre llaves se ejecutará mientras la condición entre paréntesis se cumpla, Cuando la condición deje de ser cierta, el control pasará al primer mandato tras el bloque de mandatos encerrados entre llaves.

```
// Este es un ejemplo del bucle do-while

main()

{

int i;

i = 0;

do {

cout <<"El valor de i es ahora " << i << "\n";

i = i + 1;

} while (i <= 5);

}
```

Destacamos que:

1) Como la comprobación se realiza al final del bucle, el bloque de mandatos entre llaves se ejecutará al menos 1 vez.

2) Si “i” no se cambiara dentro del bucle, el programa se ejecutaría eternamente.

3) Igual que con el bucle WHILE, si el bloque de mandatos se reduce a uno, no son necesarias las llaves.

Estos bucles pueden anidarse. Esto es, un bucle dentro de otro, y el número de anidamientos es ilimitado.

Bucle For

El bucle for no es nada nuevo. Es una nueva manera de describir el bucle while. La estructura del bucle for consiste en la palabra reservada for, seguida de determinado número de expresiones entre paréntesis. Estas expresiones son dos o tres campos separados por punto y coma.

```
#include <iostream.h>

// Este es un ejemplo de bucle for

main()

{

int indice;

for(indice = 0;indice < 6;indice = indice + 1)

cout <<"El valor de indice es " <<indice<< "\n";

}
```

En nuestro ejemplo, el primer campo contiene la expresión “índice= 0”, y es un campo de inicialización. Algunas expresiones se ejecutan antes del primer paso, a través del bucle. No hay límite para estas expresiones, pero un buen estilo de programación recomienda no excederse. Varios mandatos de inicialización pueden ubicarse en este campo, separados por punto y coma.

El segundo campo, en este caso conteniendo “índice < 6”, es el test que se hace al principio de cada vuelta del bucle. Esta expresión se puede evaluar verdadera o falsa.

La expresión contenida en el 3er. campo se ejecuta cada vez que se ejecuta el bucle, esto no ocurre hasta que todos los mandatos contenidos en el bucle se ejecutan en su totalidad. Este campo, al igual que el primero, puede contener varios operandos separados por;

Siguiendo a la expresión for (), algún mandato simple, o un bloque que será ejecutado en el cuerpo del bucle.

Condicional IF

En el siguiente programa, vemos que hay un bucle for con un mandato que contiene 2 mandatos “if”. Es un ejemplo de anidación. Debe quedar claro que, en el ejemplo, cada if se ejecutará 10 veces.

```
#include <iostream.h>       // Este es un ejemplo de los mandatos if
main()
{
int dato;
for(dato = 0;dato < 10;dato = dato + 1) {
if (dato == 2)
cout <<"Dato es ahora igual a " <<dato<< "\n";
if (dato < 5)
cout <<"Dato es ahora " << dato << ", el cual es menor de 5\n";
else
cout <<"Dato es ahora " << dato << ", el cual es mayor de 4\n";
}  // Fin del bucle
}
```

El condicional empieza con la palabra reservada if, seguida de una expresión entre paréntesis. Si la expresión se evalúa y resulta cierta, el mandato simple que sigue a if se ejecutará, pero si es falsa, el control del programa pasará a la siguiente instrucción. En lugar del comando simple, podemos poner un bloque de instrucciones encerrado por llaves.

La expresión “dato == 2″, está simplemente preguntando si el valor de “dato” es igual a 2, es decir que “== “significa comparación, en cambio si la expresión fuera “= “la función sería de asignación.

Conclusiones

Las sentencias de control nos pueden ayudar para crear partes de un código y condicionarlas de tal manera que el código puede tener más opciones de uso.

Esto nos ayuda para crear vagamente dicho una ramificaciones de variaos caminos para un programa dependiendo de pocos parámetros el programa se hará multifuncional.

Bibliografía

http://ejercicioscpp.blogspot.com/2012/11/estructuras-de-control-en-c.html

http://www.lcc.uma.es/~valverde/lp1t3.pdf