Universidad Autónoma deBaja California

Facultad de Ingeniería, Arquitectura yDiseño

Ingeniero en Computación

Asignatura:

Programación Estructurada

Actividad 2:

Estructuras de control de Selección

(Evaluar Optimización de código)

Brayan Arturo Rocha Meneses

Matricula: 371049

Ensenada Baja California 10 de Septiembre del 2023

Introducción:

Las estructuras son esenciales para crear programas más flexibles y dinámicos, ya que permiten que un programa tome diferentes caminos o realice acciones específicas según se cumplan o no ciertas condiciones.

Las declaraciones "if", "else", "elif" y "switch" en diferentes lenguajes de programación, permiten tomar decisiones en el flujo del programa. También examinaremos la importancia de evaluar y optimizar el código que utiliza estas estructuras para garantizar un rendimiento óptimo en nuestras aplicaciones.

Competencia:

Diseñar programas de cómputo, aplicando las estructuras de control de iteración, para proporcionar soluciones óptimas a problemas del área de ingeniería, de manera innovadora y ordenada.

Fundamentos:

En este capítulo se revisan los distintos métodos con los que C controla el flujo lógico de un programa.

Los operadores relaciones binarios que se usan son:

< Menor que

<= Menor Igualque

> Mayor que

>= Mayor igual que

== Exactamente Igual

!= Diferente

Los operadores lógicos binarios:

|| OR

&& AND

y el operador lógico unario de negación !, que sólo toma un argumento.

! NOT

Los operadores anterior son usados con las siguientes estructuras que se muestran.

La sentencia if

Las tres formas como se puede emplear la sentencia if son:

if (condicion)

sentencia;

...o

if (condicion)

sentencia1;

else

sentencia2;

...o

```
if (condicion1)



sentencia1;

else

if (condicion2)

sentencia2;

...

else

sentencian;
```

El flujo lógico de esta estructura es de arriba hacia abajo. La primera sentencia se ejecutará y se saldrá de la estructura if si la primera condición es verdadera. Si la primera condición fue falsa, y existe otra condición, se evalúa, y si la condición es verdadera, entonces se ejecuta la sentencia asociada.

Si existen más condiciones dentro de la estructura if, se van evaluando éstas, siempre y cuando las condiciones que le preceden sean falsas.

La sentencia que está asociada a la palabra reservada else, se ejecuta si todas las condiciones de la estructura if fueron falsas.

Por ejemplo:

```
int main()

{

int x, y, w;

.....

if (x>0)

{

z=w;

.....
```

```
}

else

{

z=y;

.....

}

}
```

La sentencia switch

Aunque con la estructura if ... else if se pueden realizar comprobaciones múltiples, en ocasiones no es muy elegante, ya que el código puede ser difícil de seguir y puede confundir incluso al autor transcurrido un tiempo.

Por lo anterior, C tiene incorporada una sentencia de bifurcación múltiple llamada switch. Con esta sentencia, la computadora comprueba una variable sucesivamente frente a una lista de constantes enteras o de carácter. Después de encontrar una coincidencia, la computadora ejecuta la sentencia o bloque de sentencias que se asocian con la constante.

La forma general de la sentencia switch es:

```
switch (variable) {

case constante1:


secuencia de sentencias

break;

case constante2:


secuencia de sentencias

break;

case constante3:


secuencia de sentencias
```

```
break;


...

default:

secuencia de sentencias

}
```

Donde la computadora ejecuta la sentencia default si no coincide ninguna constante con la variable, esta última es opcional. Cuando se encuentra una coincidencia, la computadora ejecuta las sentencias asociadas con el case hasta encontrar la sentencia break con lo que sale de la estructura switch.

Las limitaciones que tiene la sentencia switch ... case respecto a la estructura if son:

● Sólo se tiene posibilidad de revisar una sola variable.

● Con switch sólo se puede comprobar por igualdad, mientras que con if puede ser con cualquier operador relacional.

● No se puede probar más de una constante por case.

La forma como se puede simular el último punto, es no teniendo sentencias asociados a un case, es decir, teniendo

una sentencia nula donde sólo se pone el caso, con lo que se permite que el flujo del programa caiga al omitir las

Sentencias, como se muestra a continuación:

```
Switch (letra)

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

numvocales++;

break;
```

```
case ' ':
numesp++;
break;
default:
numotras++;
```

Procedimiento:

Las estructuras de control de selección son elementos fundamentales en la programación que permiten tomar decisiones y dirigir el flujo de un programa en función de ciertas condiciones.

1.- Programa en C que lea 3 calificaciones calcule el promedio del alumno y desplegar:

Si prom < 30 Repetir

Si prom >=30 y prom <60 extraordinario

Si prom >=60 y prom <70 suficiente

Si prom >=70 y prom <80 Regular

Si prom >=80 y prom <90 bien

Si prom >=90 y prom <98 muy bien

Si prom >=98 y prom <=100 excelente

Si prom >100 Error en promedio

(OPTIMIZADO)

2.- Programa en C que sirva para el juego del CHINCHAMPU (Piedra, Papel, Tijera) para 1 jugador y la computadora,

(usar condición anidada)

3.- Programa en C que sirva para el juego del CHINCHAMPU (Piedra, Papel, Tijera) para 1 jugador y la computadora,

(usar selección múltiple)

4.- Programa en C que lea 3 números y desplegar cuál número es el mayor (usar AND o OR)

5.- Programa en C que lea 3 números y desplegar el número del medio (usar AND o OR)

6.- Programa en C que lea 3 números y despegarlos en forma ascendente (usar AND o OR)

7.- Función en C que pida el mes y día de nacimiento de una persona y el programa le despliega el signo del zodiaco que le corresponde y su correspondiente horóscopo del Dia.

Conclusiones:

Las estructuras de control de selección son elementos cruciales en la programación que nos permiten tomar decisiones y direccionar el flujo de un programa de manera efectiva. Estas estructuras, como las declaraciones "if", "else", "elif" y "switch", son fundamentales para crear programas flexibles y adaptativos que puedan responder a diferentes condiciones y situaciones

Comprender y utilizar adecuadamente las estructuras de control de selección y aplicar técnicas de optimización de código son habilidades esenciales para cualquier programador, ya que permiten crear software más eficiente y efectivo en términos de recursos y rendimiento.

Anexo:

https://drive.google.com/file/d/1kOk_lQu34NjG9nRrF6LvAQepTEwx2GT/view?usp=sharing