INGENIERIA EN COMPUTACION

Materia: Ingeniería en computación

Alumno: Ayon Dimas Andres

Matricula: 366845

Profesor: Pedro Nunes Yepiz

Tema: Estructuras de control de selección

Ensenada B.C 10 de septiembre del 2023

INTRODUCCION

Aquí podremos ver lo que son las estructuras de control de selección, pero en esta ocasión lo haremos con funciones, poniendo todos los programas solicitados dentro de uno mismo usando las funciones(void) lo que nos ayuda a poder hacer una selección de una de estas funciones la cual cada función representa un programa. De esta manera nosotros podemos seleccionar uno para que se ejecute y poner los datos que nos solicita.

COMPETENCIA

Aprenderemos a utilizar lo que son las funciones y del por qué son tan importantes utilizarlas en C#, ya que nos facilita y nos ayuda en la eficiencia de memoria, procesador y también del espacio de almacenamiento. Así mismo veremos la optimización de código para que estos mismos sean más eficientes y eficaces.

FUNDAMENTOS

Permiten modular los programas, en otras palabras, permite dividir un programa en secciones más pequeñas (módulos) que realizan una tarea específica. Son equivalentes a las funciones (function) del Pascal.

Cuando se usa como un tipo de valor devuelto de función, la palabra clave **void** especifica que la función no devuelve ningún valor. Cuando se usa para la lista de parámetros de una función, **void** especifica que la función no toma ningún parámetro. Cuando se usa en la declaración de un puntero, **void** especifica que el puntero es "universal".

Si el tipo de puntero es void*, el puntero puede apuntar a cualquier variable que no se declare con la palabra clave **const** o **volatile**. Un puntero void* no se puede des referenciar a menos que se convierta en otro tipo. Un puntero void* se puede convertir en cualquier otro tipo de puntero de datos.

En C++, un puntero **void** puede apuntar a una función libre (una función que no es miembro de una clase) o a una función miembro estática, pero no a una función miembro no estática.

No se puede declarar una variable de tipo **void**.

Como cuestión de estilo, las directrices principales de C++ recomiendan no usar **void** para especificar una lista de parámetros formales vacía.

PROCEDIMIENTO

```
public class MiClase
{
  public MiClase (Puedes poner parametros, si quieres ya que no es obligatorio)
  {
    //Este es el constructor de la clase, aquí
    //puedes poner código que se ejecutará al instanciar a la clase
  }

  public void MetodoVacio (Puedes poner parametros, si quieres ya que no es
obligatorio)
  {
    //Este es un método que NO retorna valor alguno, puedes poner
    //código en su cuerpo o no, pero nunca podrá retornar valores;
    //curiosamente, si puedes usar return dentro de este tipo de método
    //pero NO podrás indicar un valor que retornar

    return;
  }

  public int MetodoEntero (Puedes poner parametros, si quieres ya que no es
obligatorio)
  {
     //Este es un método que retorna un valor entero, puedes poner
     //código en su cuerpo o no, pero siempre tendrá que retornar
     //un valor entero, en este caso he puesto 27

     return 27;
  }

  public string MetodoString(Puedes poner parametros, si quieres ya que no es
obligatorio)
  {
     //Este es un método que retorna una cadena de texto, puedes poner
     //código en su cuerpo o no, pero siempre tendrá que retornar
     //un valor string, en este caso he puesto Hola Mundo

     return "Hola Mundo";
  }

  public char MetodoCaracter (Puedes poner parametros, si quieres ya que no es
obligatorio)
  {
     //Este es un método que retorna un caracter, puedes poner
     //código en su cuerpo o no, pero siempre tendrá que retornar
     //un caracter, en este caso he puesto el caracter A
```

```
    return 'A';
  }

  //Y así sucesivamente

}
```

RESULTADOS Y CONCLUSIONES

Lo aprendido es muy útil ya que esto nos ayuda a reducir el espacio de memoria y eficiencia del programa, también podemos ver la importancia a la hora de la optimización de programa ya que esta es de suma importancia a la hora de correr el programa ya que entre mas procesos haga el programa, este se volverá más lento a la hora de que lo corran. También pudimos ver un comando para C# en Visual Studio Code el cual es el siguiente (Shift+Alt+f) ¿y por qué es importante este comando? Este comando nos ayudara acomodar las llaves sin importar que, con este comando podremos acomodarlas cual sea su posición y nos ayudara a que el programa se vea aún más legible y que pueda ser entendido por otros programadores. Con esto aprendido no podremos ayudar a realizar programas con una mejor eficiencia en todo sentido y a su vez que sea más legible.