utel UNIVERSIDAD®

Dirección Académica

Programación estructurada

Ciclo Mayo 2022

CONTENIDO

REPASO CLASE ANTERIOR

Repaso clase anterior

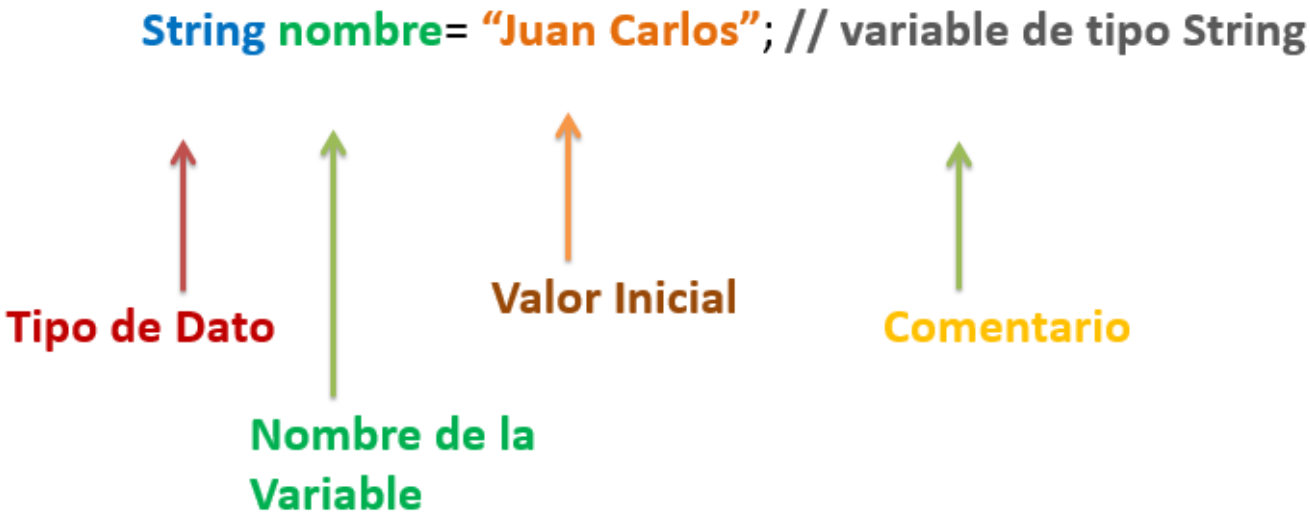
¿De qué manera los elementos básicos de un programa permiten la lectura y escritura de datos, así como las operaciones lógicas y matemáticas?
Lenguaje de programación C#
Capture y muestre mi nombre “Juan Carlos”
Tipo de dato a los datos que la computadora reconoce.
Numerico (0,1,2,0.5,8.9,3.1416), Caracteres (a,A,$,”), Cadenas(“aA$”)
Traducir a lenguaje de programación C# los tipos de datos:
Numericos->Integer (int) (0,2,389,1000), float (0.4,23.1,3.1416), double(3.14164578)
Caracteres -> char
Cadenas -> string

Repaso clase anterior

Tipos de Datos:

NOMBRE	CONJUNTO DE VALORES	OPERACIONES
Enteros	Negativos y positivos sin decimal	Sumar, restar, dividir, multiplicar, residuo
Reales	Negativos y positivos, con decimal	Sumar, restar, dividir, multiplicar
Lógicos	Verdadero o Falso(1 o 0)	And, Or, Not
Caracteres	Letras, números, especiales, juntos forman una cadena	Sumar carácter + entero restar, multiplicar por entero

Sintaxis en C# para usar o guardar datos
“Juan Carlos”
Variable



Repaso clase anterior

Tipos de Datos:

void

- Indica “nada”
- Es un variable sin tipo definido
- Cuando una función o programa no retorna ningún resultado final se dice que devuelve el tipo **void**.

Cuarto solo sin muebles, sin personas. No hay nada en ese cuarto.
Suma=2+2=4(tipo de dato entero) C#->int
Suma = void

Repaso clase anterior

Tipos de Datos:

int
Tipo de dato entero
Valores positivos y negativos

float
Tipo de dato real
Precisión simple
Valores positivos y negativos

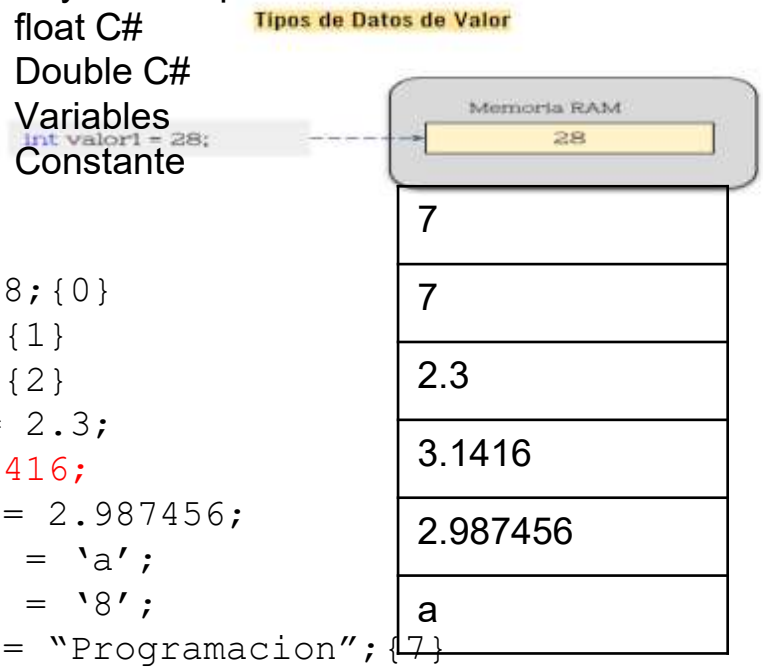
double
Tipo de dato real
Precisión doble
Valores positivos y negativos

char
Almacena cualquier carácter “normal”
Ocupa un byte

Repaso clase anterior

Decimales Español
Real Ingles
Keywords o palabras reservada
float C#
Double C#
Variables
Constante

Tipos de Datos:



0

```
program(){
        int valor1 = 28;{0}
        int numero1=7;{1}
        int numero2=7;{2}
        float numero3= 2.3;
        float pi = 3.1416;
        double numero4= 2.987456;
        char caracter1 = 'a';
        char caracter2 = '8';
        string cadena = "Programacion";{7}
        Console.WriteLine("{0},{1},{2},{3},{4},{5},{6},{7}" +
        numero1);
        }
```

Repaso clase anterior

Identificadores

Son palabras clave o reservadas en el lenguaje de programación o el programa que se esté laborando.

Variables o Constantes:

Puntos al usar identificadores:

Las mayúsculas y minúsculas las trata de forma diferentes.

No debe usar mas de 32 caracteres.

No debe usar palabras reservadas de la biblioteca del programa.

El primer carácter de un identificador debe ser una letra o subrayado.

Repaso clase anterior

-127	-3	-2	-1	RAM	0	1	2	308

Declaración de datos:

-7895647

7895647

Todos los tipos de datos (variables o constantes) se deben declarar antes de ser usadas por el programa, con el fin reservar su espacio en memoria. Considerando los siguientes puntos:

Tipo	Rango	descripción
Carácter → char	-128 a 127	Guarda caracteres del código ASCII
Entero → int	-32,768 a 32,767	Usa cantidades enteras positivas o negativas no fraccionales.
Punto flotante→float	-3.4E-38 a 3.4E38	Usa cantidades fraccionarias positivas y negativas. Con hasta 38 dígitos después del punto decimal.
Doble punto flotante → double	1.TE-308 a 1.7E308	Usa cantidades fraccionarias positivas y negativas. Con hasta 308 dígitos después del punto decimal.
Sin valor→ void	Sin valor	Ayuda a mejorar la comprobación de tipos.

Repaso clase anterior

Identificadores

Ejemplos de identificadores y los que no lo son:

Correcto	Incorrecto (en rojo)
Contador	3Contador
Prueba23	Hola!tu
Alto_balance	Alto…balance
pago_por_ hora	Pago*hora
Nombre_del_alumno	Nombre del alumno

Ejemplo de repaso

Identificadores - Variables

Son el espacio para guardar información en un espacio de memoria reservado para el programa realizado, donde se puede grabar un valor, usarlo durante su ejecución y recuperar ese valor más tarde.

.

Sintaxis:

```
tipo_de dato  variable  (1 o más identificadores);
```

Ejemplo:

```
int num1,num2,num3;
char nom_alum;
float precio;
```

Ejemplo de repaso

Identificadores - Constantes

Son valores fijos que el programa no puede alterar, pero que son utilizados con esos mismo valores durante todo el programa (Rentería, 2015).

Sintaxis:

```
tipo_de dato constante =  asignación del valor  (1 o más identificadores);
```

Ejemplo:

```
int num1=4,num2=67;
char afirmacion=´s´, negación=´n´;
float Pi=3.141652, IVA=15%;
```

Ejemplo de repaso

Actividad:

Piensa y reflexiona la siguiente pregunta:

¿Puedo programar todo lo que imagine?

Todo lo que ves….PENSAMIENTOS, representaciones de cosas externas

¡¡¡Los pensamientos ayudan al hombre a explicar el mundo!!!

Presentación

La pregunta sobre la que reflexionarás en esta unidad es:

¿Cómo las sentencias de decisión y repetición minimizan el uso de instrucciones secuenciales?

En esta unidad aprenderás la función de las estructuras de condición simple y múltiple, y su aplicación en problemas reales que involucren la elección de un camino entre más de una opción. De igual forma, comprenderás el uso de las tres diferentes instrucciones de repetición que utilizaremos en el curso y la forma en que ayudan a optimizar la escritura de código en el desarrollo de programas.

Objetivo

- Utilizar las instrucciones de selección **“if”** e “**if..else”** para elegir una de varias acciones alternativas.

Les gusta el futbol? SI me gusta
NO me gusta

If (gusta el futbol) Si
else (gusta el basquetbol) Si
if else (gusta voleyball) Si
No

- Conocer la sintaxis de C# para las instrucciones condicionales simples, dobles y múltiples, así como también la utilidad en la programación.

- Aprender a utilizar la estructura y sintaxis del switch-case para la evaluación de condiciones múltiples

Resultados de aprendizaje

- Distinguir la sintaxis de las diferentes sentencias de control de la programación estructurada.
- Elaborar programas basados en sentencias de control de condición múltiple.
- Diseñar soluciones que utilicen instrucciones repetitivas.

Competencias

- Diseñar programas que requieran la aplicación de sentencias condicionales y repetitivas para resolver situaciones que no se solucionan con controles secuenciales.

DESARROLLO

Introducción

Por lo general, las instrucciones en una aplicación se ejecutan una después de la otra, en el orden en que se escriben.

A este proceso se le conoce como ejecución secuencial.

Varias instrucciones de C# le permiten especificar que la siguiente instrucción a ejecutar no es necesariamente la siguiente en la secuencia.

A este se le conoce como transferencia de control.

Operadores Lógicos

Operadores Lógicos

Los operadores lógicos producen un resultado booleano (verdadero o falso) y sus operandos son también valores lógicos. Nos permiten formular condiciones complejas a partir de condiciones simples.

A continuación se muestra una tabla con las tres compuertas lógicas básicas que nos servirán como operadores lógicos:

OPERADOR	C#	SINTAXIS	COMENTARIO
AND	&&	Exp_Lógica && Exp_Lógica	Devuelve verdaderos si se cumplen ambas condiciones.
OR	||	Exp_Lógica || Exp_Lógica	Devuelve verdaderos si se cumple al menos una de las condiciones.
NOT	!	!	Niega la condición.

Juegas futbol && corres
Juega futbol || corres
! juegas futbol Verdadero
No corres
Falso

Operadores Lógicos

Operadores Lógicos

Nota: La operación(x&&y)corresponde a la operación(xandy).

La operación(x||y)corresponde a la operación(xory).

OPERADOR	C#	SINTAXIS	COMENTARIO
AND	&&	Exp_Lógica && Exp_Lógica	Devuelve verdaderos si se cumplen ambas condiciones.
OR	||	Exp_Lógica || Exp_Lógica	Devuelve verdaderos si se cumple al menos una de las condiciones.
NOT	!	!	Niega la condición.

Estructuras de Selección en C#

C# cuenta con tres tipos de estructuras de selección, que de aquí en adelante denominaremos instrucciones de selección.

La **instrucción if** realiza (selecciona) una acción si una condición es verdadera, o ignora la acción si la condición es falsa.

a. La **instrucción if...else** realiza una acción si una condición es verdadera o realiza una acción distinta si la condición es falsa.

b. La **instrucción switch** realiza una de varias acciones distintas, dependiendo del valor de una expresión (expresión de control).
¿Que deporte te gusta dame una opcion? Opcion=9
Switch (opcion):
Case 1 : Futbol
Case 2 :Basquetbol
Case 3: Volleybol
Default: Ninguna

Estructuras de Selección en C#

C# cuenta con tres tipos de estructuras de selección, que de aquí en adelante denominaremos instrucciones de selección.

c. A la **instrucción if** se le llama instrucción de selección simple , debido a que selecciona o ignora una acción individual.

d. A la **instrucción if... else** se le llama instrucción de selección doble, debido a que selecciona una de dos acciones distintas (o grupos de acciones).

e. A la **instrucción switch** se le llama instrucción de selección múltiple , debido a que selecciona una de varias acciones distintas (o grupo de acciones)

Sintaxis de las Instrucciones Condicionales

Una instrucción if simple responde a la siguiente sintaxis:

if (expresión booleana) {

Instrucción(es) de condición verdadera.

}

Sintaxis de las Instrucciones Condicionales

La instrucción if solo garantiza la ejecución de determinadas acciones basándose en una condición verdadera (true). Se ejecutan o no se ejecutan.

Para controlar tanto la condición true como la falsa (false) es necesario utilizar la instrucción if...else.

Su sintaxis en la siguiente:

```
if (expresión booleana){
Instrucción(es) de condición verdadera.
} else {
        Instrucción(es) de condición falsa.
}
```

Sintaxis de las Instrucciones Condicionales

Instrucción Switch

Cuando hay muchas condiciones a evaluar, la instrucción if...else puede resultar demasiado compleja de manejar. Una solución mucho más limpia en estos casos consiste en usar la instrucción switch. La instrucción switch permite cualquier valor entero o de cadena con múltiples valores. Cuando se produce una coincidencia se ejecutan todas las instrucciones asociadas con ella. Esta estructura es muy utilizada a la hora de trabajar con menú dentro de las aplicaciones.

Sintaxis de las Instrucciones Condicionales

Instrucción Switch

La sintaxis es la siguiente:

Switch (expresión de control){

case<literal-1>:Instrucción(es)

break;..

case<literal-n>:Instrucción(es)

break; default:Instrucción(es)

}

Ejemplos

Diseñar un programa en C# que nos permita saber cuál es el número mayor entre 2 números.

```
class Program
   {
      static void Main(string[] args)
      {
         Console.Title="El amyor de dos numeros";
         int x;
         int y;
         Console.WriteLine("Digita el primer numero a comparar");
         Console.WriteLine("Entre el 1 y el 100");
         x = int.Parse(Console.ReadLine());
         Console.WriteLine("Digita el segundo numero a comparar");
         Console.WriteLine("Entre el 1 y el 100");
         y = int.Parse(Console.ReadLine());

         if (x > y) {
            Console.WriteLine("\nEl numero {0} es mayor que {1}", x, y);
            } else {
               Console.WriteLine("\nEl numero {0} es mayor que {1}", y, x);
            }
          Console.WriteLine("\n\n");
         Console.WriteLine("\n----->Fin del programa");
         Console.ReadKey();
      }
   }
```

Ejemplos

Diseñar un programa en C# que nos permita saber cuál es el número mayor entre 2 números.

No. Corrida	Datos entradas	Resultado
1	x=100 y=87	
2	x=150 y=245	
3	x=200 y=135	

Ejemplos

Se necesita un programa que muestre un menú con las siguientes opciones:

1.Suma.

2.Resta.

3.Multiplicación.

4.División.

ACTIVIDADES DE REFORZAMIENTO

Actividades de reforzamiento

Ejercicio 1. Operaciones básicas

A partir de los recursos revisados, programa las operaciones básicas de matemáticas con dos valores:

Actividades de reforzamiento

```
using System;

namespace Programacion {

public class Ejemplo 2{

public static void Main() {
   double Num1, Num2;
   Num1 = Convert.ToDouble(TxtNum1.Text);
   Num2 = Convert.ToDouble(TxtNum2.Text);
   //Suma de dos números en C#
      double Suma;
      Suma = Num1 + Num2;
      Console.WriteLine("El Resultado de la suma es: " + Suma);
   //Resta de dos números en C#
      double Resta;
      Resta = Num1 – Num2;
      Console.WriteLine("El Resultado de la Resta es: " + Resta);
   //Multiplicación de dos números en C#
      double Multiplicacion;
      Multiplicacion = Num1 * Num2;
      Console.WriteLine("El Resultado de la Multiplicación es: " + Multiplicacion);
   //División de dos números en C#
      double Division;
      Division = Num1 / Num2;
      Console.WriteLine("El Resultado de la División es: " + Division);
   }
  }
}
```

Conclusión


Puede ser leído en secuencia.
Más fáciles de entender
Reducción del esfuerzo en las pruebas.
Los errores se pueden detectar y corregir.
Programas + sencillos y rápidos.
PROGRAMACIÓN ESTRUCTURADA
Se busca que el programador elabore programas sencillos y fáciles de entender
Secuencial
Las instrucciones de un programa se ejecutan una después de la otra
Plantea la selección entre dos alternativas
Selectiva
Ejecución repetida de una instrucción mientras que se cumple una determinada condición.
Repetitiva (Iterativa)
Estructuras básicas de control.

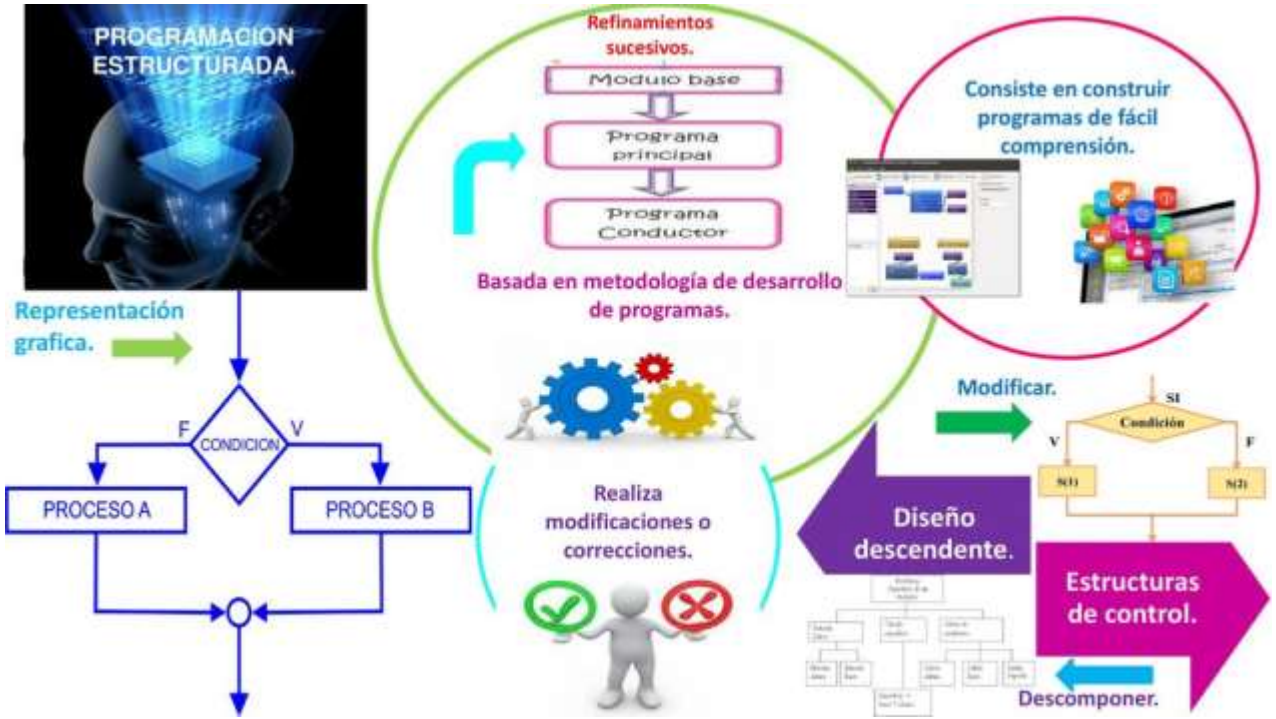
PREGUNTA DE INVESTIGACION

Pregunta de investigación

¿Cómo las sentencias de repetición minimizan el uso de instrucciones secuenciales?

CIERRE

Cierre


PROGRAMACION ESTRUCTURADA.
Representación grafica.
F
CONDICION
V
PROCESO A
PROCESO B
Refinamientos sucesivos.
Modulo base
Programa principal
Programa Conductor
Basada en metodología de desarrollo de programas.
Consiste en construir programas de fácil comprensión.
Realiza modificaciones o correcciones.
Modificar.
SI
Condición
V
F
S(1)
S(2)
Diseño descendente.
Estructuras de control.
Descomponer.

BIBLIOGRAFÍA COMPLEMENTARIA

http://www.msdn.microsoft.com/net/ecma.

“A programmer’s introduction to C#” escrito por Eric Gunnerson y publicado por Apress en 2000.

C# and the .NET Framework”, escrito por Andrew Troelsen y publicado por Apress en 2001

“C# Essentials”, escrito por Beb Albahari, Peter Drayton y Brand Merril y publicado por O’Reilly en 2000.

“C# Programming with the Public Beta”, escrito por Burton Harvey, Simon Robinson, Julian Templeman y Karli Watson y publicado por Wrox Press en 2000.

“Inside C#”, escrito por Tom Archer y publicado por Microsoft en 2000 • “Presenting C#”, escrito por Christoph Wille y publicado por Sams Publishing en 2000.

ÍNDICE

DUDAS