2023

Programación II

Estructuras Básicas

CONCEPTOS BÁSICOS

En todo programa que se desarrolla no solo se debe buscar cumplir con los requerimientos del momento, también se deberá pensar a futuro, es decir tener en cuenta que todo software durante la etapa de desarrollo como así también luego de finalizado, siempre es susceptible de sufrir modificaciones. Estas modificaciones pueden tener su raíz en cuestiones tales como: nuevos requerimientos o cambios a los ya existentes, mejoras en el algoritmo haciéndolo mas eficiente en el uso de los recursos (ej. tiempo, memoria), etc. Estos factores por lo tanto obligan a ampliar y/o retocar el código fuente.
Transcurrido un período considerable desde la finalización del programa, toda modificación obligará a la reinterpretación del código, el cual durante su generación resultaba prácticamente obvio para el desarrollador por encontrarse inmerso en su resolución desde un tiempo prolongado. Dicha reinterpretación aumentara en complejidad si el código había sido elaborado por otro individuo, con su propia lógica y costumbres de programación (herramientas, técnicas, etc.).
En consecuencia, en la etapa de desarrollo se buscara que el algoritmo cumpla con los siguientes requerimientos: *claridad, legibilidad y modificabilidad*. Los cuales facilitaran el entendimiento del código generado, permitiendo de esta manera reducir los tiempos de desarrollo y la generación de programas más robustos.

a) ***Claridad:*** implica que la resolución algorítmica sea sencilla, que esté correctamente estructurada, resultando un algoritmo de fácil compresión.

b) ***Legibilidad:*** significa que en la codificación se utilizaron ***nombres adecuados*** en lo que respecta a variables, funciones, procedimientos, etc., ***comentarios aclaratorios*** y una correcta ***diagramación*** del texto.

c) ***Modificabilidad:*** cualquier cambio que sea necesario incorporar en alguna de sus partes, no obligue a realizar cambios y/o controles en todo el programa, sino limitarlo solo al módulo interesado.

TIPOS DE DATOS

Tipo	Descripción	Bits	Rango de valores	Alias
SByte	Bytes con signo	8	[-128, 127]	**sbyte**
Byte	Bytes sin signo	8	[0, 255]	**byte**
Int16	Enteros cortos con signo	16	[-32.768, 32.767]	**short**
UInt16	Enteros cortos sin signo	16	[0, 65.535]	**ushort**
Int32	Enteros	32	[-2.147.483.648, 2.147.483.647]	**int**
UInt32	Enteros sin signo	32	[0, 4.294.967.295]	**uint**
Int64	Enteros largos	64	[-9.223.372.036.854.775.808, 9.223.372.036.854.775.807]	**long**
UInt64	Enteros largos sin signo	64	[0-18.446.744.073.709.551.615]	**ulong**
Single	Reales con 7 dígitos de precisión	32	$[1,5\times10^{-45} - 3,4\times10^{38}]$	**float**
Double	Reales de 15-16 dígitos de precisión	64	$[5,0\times10^{-324} - 1,7\times10^{308}]$	**double**
Decimal	Reales de 28-29 dígitos de precisión	128	$[1,0\times10^{-28} - 7,9\times10^{28}]$	**decimal**
Boolean	Valores lógicos	32	true, false	**bool**
Char	Caracteres Unicode	16	['\u0000', '\uFFFF']	**char**
String	Cadenas de caracteres	Variable	El permitido por la memoria	**string**
Object	Cualquier objeto	Variable	Cualquier objeto	**object**
DateTime	Representa un instante de tiempo	64	01/01/0001 00:00:00, 31/12/9999 23:59:59	**DateTime**

OPERADORES

Operador Matemáticos	Significado
+	Suma
-	Resta
*	Producto
/	División
%	resto (módulo)
++	Incremento
--	Decremento

Operador Comparación	Significado
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
==	Igual que
!=	Distinto de

Operador Lógicos	Significado
&&	AND
||	OR
!	NOT

ESTRUCTURAS DE CONTROL

Estructura	Sintaxis	Ejemplo

Condicional	if (Condicion)
	{
	Instrucciones a ejecutar cuando la condicion resulta verdadera
	}
	else
	{
	Instrucciones a ejecutar cuando la condicion resulta falsa
	}
		if (Num1 > Num2)
		{
		Mayor = Num1;
		}
		else
		{
		Mayor = Num2;
		}

Selección Múltiple	switch (Expresion)
	{
	case Valor1:
	{
	Instrucciones a ejecutar cuando la Expresion coincida con Valor1
	break;
	}
	case ValorN:
	{
	Instrucciones a ejecutar cuando la Expresion coincida con ValorN
	break;
	}
	default:
	{
	Instrucciones a ejecutar cuando la Expresion
	no coincida con ningun Case
	break;
	}
	}
		switch (Presion)
		{
		case 15:
		{
		Coef = 3;
		break;
		}
		case 17:
		{
		Coef = 5;
		break;
		}
		case 20:
		{
		Coef = 7;
		break;
		}
		default:
		{
		Coef = 0;
		break;
		}
		}

Para	for (Inicio;Condicion;Incremento)
	{
	Instrucciones a ejecutar en cada ciclo
	}
		for (k = 1; k <= 10; k++)
		{
		Suma = Suma + k;
		}

Mientras	while (Condicion)
	{
	Instrucciones a ejecutar en cada ciclo
	}
		while (Cuenta <= 10)
		{
		Cuenta = Cuenta + 1;
		}

Repetir	do
	{
	  Instrucciones a ejecutar en cada ciclo
	}
	while (Condicion);
		do
		{
		  Cuenta = Cuenta + 1;
		}
		while (Cuenta <= 10);

Para Cada	foreach (Tipo Objeto in Arreglo/Coleccion)
	{
	  Instrucciones a ejecutar en cada ciclo
	}
		foreach (int k in vector)
		{
		  Suma = Suma + k;
		}

DECLARACIÓN DE VARIABLES

Objetivo	Sintaxis
Declarar una variable.	TipoDeDato NombreDeVariable;
Declarar un conjunto de variables de un mismo tipo de dato.	TipoDeDato NombreDeVariable1,NombreDeVariable2,NombreDeVariable3;
Declarar una variable asignandole un valor inicial.	TipoDeDato NombreDeVariable = ValorInicial;
Declarar un vector sin especificar la cantidad de filas a contener.	TipoDeDato[] NombreDeVariable;
Declarar un vector especificando la cantidad de filas a contener.	TipoDeDato[] NombreDeVariable = new TipoDeDato[CantFilas];
Declarar un vector especificando el contenido del mismo.	TipoDeDato[] NombreDeVariable = {Valor1,Valor2,Valor3};
Declarar una matriz de 2 dimensiones sin especificar la cantidad de filas y columnas a contener.	TipoDeDato[,] NombreDeVariable;
Declarar una matriz de 2 dimensiones especificando la cantidad de filas y columnas a contener.	TipoDeDato[,] NombreDeVariable = new TipoDeDato[CantFilas,CantCols];
Declarar una matriz de 2 dimensiones de 3x2 especificando el contenido de la misma.	TipoDeDato[,] NombreDeVariable = {{Valor11,Valor12},{Valor21,Valor22},{Valor31,Valor32}};

MODIFICADOR CONST

Las variables declaradas con la palabra **const** delante del tipo de datos, indican que son de sólo lectura. Es decir, constantes. Las constantes no pueden cambiar de valor, el valor que se asigne en la declaración será el que permanezca (es obligatorio asignar un valor en la declaración). Ejemplo: const float PI=3.141592.

COMENTAR LÍNEAS DE CÓDIGO

Con frecuencia es útil incorporar dentro del código fuente comentarios aclaratorios sobre determinadas situaciones a tener en cuenta, recordatorios o temporalmente deshabilitar ciertas líneas de código que no desea que sean compiladas. Para ello existen 2 alternativas:

1. Comentar una única línea: al comienzo de la misma se deben tipear dos barras inclinadas **//** Ejemplo:
   //es un comentario y no será compilado

2. Comentar un conjunto de líneas consecutivas (bloque): al comienzo del mismo se debe tipear **/*** y al finalizar ***/** Ejemplo:
   /*es un comentario de múltiples líneas
   y no será compilado*/