
Fundamentos de la Programación
www.profmatiasgarcia.com.ar

El Software

- Las operaciones que debe realizar el hardware son especificadas con una lista de instrucciones, llamadas programas o software.
- Dos grandes grupos de software
  - Software del Sistema
    - Indispensable para que la máquina funcione y poder escribir programas de aplicación
  - Software de Aplicación
    - Realizan tareas concretas que tienen utilidad para ciertos usuarios

Lenguajes de Programación

- Lenguajes utilizados para escribir programas de computadoras que puedan ser entendidos por ellas
- Se clasifican en tres grandes categorías
  - lenguajes de máquina
    - instrucciones directamente entendibles por la computadora (lenguaje binario)
  - lenguajes de bajo nivel
    - Proveen un juego de instrucciones más comprensibles por los humanos
  - lenguajes de alto nivel

Lenguajes de Programación (2/2)

- Lenguajes de alto nivel
  - Utilizan instrucciones escritas con palabras similares a los lenguajes humanos
  - Son independientes de la máquina en la que se ejecutan
  - Necesitan ser traducidos a instrucciones en lenguaje máquina (Compilación)
- Existen diversos tipos
  - Estructurados
  - Orientados a Objetos
  - Declarativos
  - Funcionales

Resolución de problemas con computadora

- El proceso de diseñar un programa es, esencialmente, un proceso creativo.
- Sin embargo, hay una serie de pasos comunes a seguir:
  - Análisis del problema
  - Diseño del algoritmo solución
  - Codificación
  - Compilación y Ejecución
  - Verificación
  - Depuración
  - Documentación

Entorno de Programación

- También conocidos como IDEs
- Herramienta esencial a la hora de desarrollar software
- Incluye
  - Editor
  - Intérprete o Compilador
  - Depurador
  - Ayuda en línea

Tipos de Datos

- Datos: piezas de información con las que un programa trabaja
- Cada dato tiene asociado un único *Tipo*
- El Tipo de Dato determina la naturaleza del conjunto de valores que un dato puede tomar
- Ejemplos:
  - Número Entero → int/integer
  - Número Real o coma flotante → float/double
  - Cadena de Caracteres → string
  - Valor Lógico (Verdadero o Falso) → boolean

Variables y Constantes

- Existen dos grupos principales de datos
  - Constantes: su valor no puede cambiar durante la ejecución de un programa
  - Variables: su valor puede cambiar durante la ejecución de un programa
- Ambas tienen un nombre y un valor
- Ambas permiten representar mediante un nombre a una posición de memoria que contiene el valor

Variables y Constantes en C#

- La instrucción const se utiliza para declarar una constante y establecer su valor. Al declarar una constante, puede asignar un nombre significativo a un valor. Una vez que se declara una constante, no se puede modificar ni se le puede asignar un nuevo valor
  - Se declara: const <tipo dato> <nombre de constante> = <valor>
    - Ejemplo: const float IVA = 0,21;
- Una variable se declara para especificar su nombre y sus características. Esta puede cambiar o no su valor durante la ejecución del programa.
  - Se declara: <tipo dato> <nombre de variable>;
    - Ejemplo: int numero;
    - Ejemplo para declarar e inicializar: float valor = 30,8;

Sentencias

- Describen acciones algorítmicas que pueden ser ejecutadas
- Se clasifican en
  - Ejecutables / No ejecutables
  - Simples / Estructuradas

Operadores y Expresiones (1/2)

- Sirven para procesar variables y constantes
- Una expresión es un conjunto de datos unidos por operadores que tiene un único resultado
  - Expresiones aritméticas
    - El resultado es un número
    - *a = ((2+6) / 8) * 3*
  - Expresiones lógicas
    - El resultado es un valor verdadero o falso
    - *(a < 10) && (b > 50)*

Operadores y Expresiones (2/2)

- Operadores

Lógicos

Operador	Descripción
&&	Las 2 expresiones deben ser verdaderas
||	Alguna de las 2 expresiones es verdadera
!	Negación del resultado de la expresión
^	Si 1 y sólo 1 de las expresiones es verdadera

Asignación

Operador	Descripción
=	Asigna el valor indicado
+= -=	Incrementa o decrementa y asigna
*= /=	Multiplica o divide y asigna el valor indicado

Aritméticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Residuo (resto de la división entera)
+	Concatenación de Cadenas
++	Incremento
- -	Decremento

Comparación

Operador	Descripción
==	Igualdad
!=	Desigualdad
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

Estructuras de Control

- El orden de ejecución de las sentencias de un programa determina su flujo de control
- Las estructuras de control permiten alterar el orden del flujo de control
- Existen dos tipos básicos
  - De Selección o condicionales
  - De Repetición o Iteración

Estructuras de Control Selectivas (1/2)

- Dirigen el flujo de ejecución según el resultado de evaluación de expresiones
- IF

si expresion_logica *entonces*
    hacer acción A
sino
    hacer acción B
fin_si

En C#:

```
If (expresion_logica) {
        acción verdadero
} else {
        acción falso
}
```

ejemplo

```
Int edad;
If (edad >= 21) {
        label1.text = “es mayor de edad”;
} else {
        label1.text = “es menor de edad”;
}
```

Estructuras de Control Selectivas (2/2)

- SWITCH

según_sea selector *hacer*
C11,C12,...
sentencia 1
C21 to C22,...
sentencia 2
.....
[*sino*
sentencia x]
fin_según

En C#:

```
switch (selector) {
    case <expresión>:
        Hacer sentencia1;
    break;
    case <expresión>:
        Hacer sentencia2;
                .....
    break;
    [default:
        Hacer sentenciaX]
    break;
}
```

Ejemplo Switch

ejemplo

```
int edad;
edad = TextBox1.Text;
switch (edad) {
    case 0:  case 1:  case 2: case 3:
        label1.text = "es un bebe";
            break;
    case 4: goto case 6;
    case 5: goto case 6;
    case 6:
        label1.text = "es un niño";
            break;
    case int n when (n >= 7 && n <= 12):
        label1.text = "esta en primaria";
            break;
    case var s when new[ ] {13, 14, 15, 16,
            17, 18}.contains(s):
        Label1.text = "esta en secundaria";
            break;
    case 19: case 20:
        label1.text = "no sabe que hacer";
            break;
    default:
        label1.text = "adulto";
            break;
}
```

Declaro la variable edad
Guardo en la variable edad lo que el usuario escribió en el textbox1
Utilizo el switch para determinar de acuerdo a lo guardado en la variable edad.
Si esta entre 0 y 3 escribo en el label1 “es un bebe”, si son los números 4, 5 o 6 escribo “es un niño”.

Si se encuentra entre 7 y 12, escribo “esta en primaria”, si esta entre 13 y 18 “esta en secundaria.

Si es 19 o 20 escribo “no sabe que hacer” y en cualquier otro caso (para este particular mayor que 20) escribo en el label1 “adulto”.

Finaliza el switch

Estructuras de Control Repetitivas (1/3)

- Permiten ejecutar un conjunto de sentencias repetidamente una cierta cantidad de veces o hasta que se cumpla una determinada condición
- El conjunto de sentencias se denomina bucle
- Cada repetición del cuerpo del bucle se denomina iteración

Estructuras de Control Repetitivas (2/3)

- **WHILE**

mientras condición ***hacer***

sentencia/s

.....

fin_mientras

En C#:

while (condición**) {**

sentencia/s

.....

}

ejemplo

```
int nro, calculo;
nro = 1;
calculo = 2;
while (nro < 5) {
        calculo = calculo * nro;


     nro++;
}
Label1.text = calculo
```

Declaro las variables nro y calculo
Le doy valor 1 a la variable nro
Le doy valor 2 a la variable calculo
Mientras nro sea menor a 5
Guardo en la variable calculo la multiplicación del valor actual de calculo x el valor actual de nro.
En nro guardo el valor de la suma del valor actual de nro + 1
Finaliza el mientras

Muestro en label1 el valor actual de calculo

Estructuras de Control Repetitivas (3/3)

- **FOR**

desde variable ← valor_inicial *hasta* valor_final *hacer*

sentencia/s

.....

fin_desde

En C#:

```
for (valor_inicial; condicion_corte; iteracion ) {
    sentencia/s
      .....
}
```

ejemplo

```
Int nro, calculo;
calculo = 0;
for (nro = 1; nro <= 5; nro++) {
        calculo += nro * 10;
}
Label1.Text = calculo;
```

Declaro las variables nro y calculo
Le doy valor 0 a la variable calculo
Desde nro comenzando en 1 hasta 5
Sumo y guardo en la variable calculo la multiplicación del valor actual de nro x 10 + el valor actual de calculo.
Muestro en el Label1 el resultado.

Procedimientos y Funciones (1/4)

- Descomposición en subprogramas: estrategia para resolver problemas complejos
- Los subprogramas se implementan a través de procedimientos y funciones
  - Compuestos por un grupo de sentencias
  - Se les asigna un nombre
  - Pueden invocarse entre sí utilizando ese nombre
  - Constituyen una unidad de programa

Procedimientos y Funciones (2/4)

- Los procedimientos y funciones se comunican con su invocador a través de parámetros.
- Los parámetros son un medio para pasar información, implementados a través de variables con valor.
- Tipos de parámetro
  - De Entrada: su valor es proporcionado por el invocador antes de llamar al subprograma
  - De Salida: su valor es calculado dentro de un subprograma y devuelto a su invocador

Procedimientos y Funciones (3/4)

- **Ejemplo:**
  - **Definición**

procedimiento CalcularSuma(parámetro1 entero, parámetro2 entero) devuelve entero

devolver parámetro1 + parámetro2

fin_procedimiento

  - **Invocación desde el programa principal u otro subprograma**

número entero a = 2

número entero b = 3

número entero c = CalcularSuma(a,b)

carácter d = CalcularSuma(a,b) → ERROR

Procedimientos y Funciones (4/4)

- Ventajas de utilizar procedimientos
  - Facilita el diseño descendiente y modular
  - Promueven la reutilización de código
  - Facilita la división de tareas
  - Pueden comprobarse individualmente
  - Pueden encapsularse en bibliotecas independientes

Visibilidad de Variables

- Variable Local:
  - Declarada en un subprograma
  - Sólo está disponible durante el funcionamiento del subprograma
  - Su valor se pierde una vez que el subprograma termina
- Variable Global:
  - Declarada en el programa principal
  - Está disponible en el programa principal y en todos los subprogramas
  - Su valor se pierde una vez que el programa principal termina

Bibliotecas

- Archivo independiente que contiene un conjunto de subprogramas
- Pueden ser incluidas y referenciadas en el desarrollo de múltiples programas
- Facilitan la modularización de un programa
- Desarrollo → Programa Fuente
- Compilación → Programa Objeto
- Link-Edición → Programa Ejecutable

Arrays (Arreglos) (1/3)

- Son estructuras de datos en las que se almacenan un conjunto de datos finitos del mismo tipo
  - Almacenan sus elementos en posiciones de memoria contiguas
  - Tienen un único nombre de variable que representa a todos los elementos
  - Permiten acceso directo o aleatorio a sus elementos individuales
- Los arrays se clasifican en unidimensionales y multidimensionales.

Arrays (Arreglos) (2/3)

- Arrays unidimensionales (Vectores)
  - Número finito de elementos
  - Tamaño Fijo
  - Elementos Homogéneos
  - Se accede a los elementos utilizando el nombre del array y el subíndice específico
- Ejemplo:
  - ***salarios(3) Reales*** → Nombre del array, de 3 posiciones que contendrán número reales
  - ***salarios[1] = 23,4*** → Asignación de un valor al primer elemento del array

Arrays (Arreglos) (3/3)

- Arrays multidimensionales
  - Arrays bidimensionales (Matrices o Tablas)
    - Tienen dos índices, uno para filas y otro para columnas
    - Ejemplo:

tabla(3,3) enteros → Declaración de una matriz de 3 por 3

tabla [1][1] = 2 → Elemento de la primer fila y primer columna

tabla [2][3] = 5 → Elemento de la segunda fila y la tercer columna

Ejemplos Arrays

```
string[] nombre = new string [3];

      nombre[0] = "Matias";
      nombre[1] = "Angie";
      nombre[2] = "Brianna";
```

Creo un array llamado nombre de tres posiciones para guardar strings (cadenas de texto)
En la primer posición del vector (índice 0) guardo "Matias", en la siguiente posición (índice 1) guardo "Angie" y en la tercer posición (índice 2) guardo "Brianna".

```
double[] ventas = new double [6];
int i;
double suma;
ventas[0] = TextBox1.Text;
ventas[1] = TextBox2.Text;
ventas[2] = TextBox3.Text;
Ventas[3] = TextBox4.Text;
ventas[4] = TextBox5.Text;
ventas[5] = TextBox6.Text;
suma = 0;
for (i = 0; i <= 5; i++) {
      suma = suma + ventas[i];
}
lblventas.Text = suma;
```

Creo un array llamado ventas de 6 posiciones de tipo double.
Declaro las variables i de tipo entero y suma de tipo double.
En cada posición del array ventas guardo lo ingresado por el usuario en el textbox correspondiente a un bimestre de las ventas de la empresa. Son 6 bimestres.

Guardo en la variable suma el valor 0
Utilizo un bucle FOR para poder recorrer el array en cada posición y sumar el valor de cada una en la variable suma. Por eso el FOR va desde 0 a 5 y es utilizado con la variable i para indicar el índice de posición (ventas (i))
Escribo en la lblventas el valor resultado de suma.

El estilo de Programación

- Una de las características más importantes de un buen programador
- Un buen estilo facilita la comprensión, corrección y mantenimiento de un programa
- Algunos puntos a tener en cuenta
  - Comentarios
  - Elección de nombres significativos
  - Identación
  - Espacios y Líneas en Blanco
  - Validación usando datos de prueba

Webgrafía y Licencia:

- Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos.
- Este documento se encuentra bajo Licencia Creative Commons 2.5 Argentina (BY-NC-SA), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del ***Prof. Matías E. García*** y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.
- Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada

www.profmatiasgarcia.com.ar

info@profmatiasgarcia.com.ar