



Computación I

Curso 2024

Facultad de Ingeniería
Universidad de la República

Construcción de Programas

■ Dos etapas

□ Programación

- Análisis, planificación, diseño y construcción del programa
 - Se definen las acciones a realizar.

□ Ejecución

- Puesta en práctica del mismo.
 - Las acciones se ejecutan.

Construcción de Programas

■ Proceso de construcción

□ Definición del problema

- Definirlo en términos sencillos.
- Determinar entradas y salidas.
- Identificar si tiene una solución conocida.

□ Diseño de la solución

- Empleando algún método y herramientas: diagramas, lenguaje natural o pseudocódigo.

□ Codificación

- Escritura de la solución en algún lenguaje de programación.

Construcción de Programas

□ **Compilación**

- Traducción del programa escrito en el lenguaje de programación elegido, al lenguaje de máquina.

□ **Ejecución**

- Si no hay errores de sintaxis detectados en la Compilación

□ **Codificación**

- Nuevamente si hay errores de sintaxis detectados en la Compilación

□ **Diseño y Codificación**

- Nuevamente si hay errores en la solución diseñada que se detectan en la Ejecución del programa.

Objetivos de la programación

■ **Exactitud** en la realización de la tarea

- Tiene que satisfacer la especificación exactamente.
 - Simplicidad. Elegir el algoritmo o técnica más simple disponible.

■ **Eficiencia:**

- Tiempo de ejecución
- Uso de memoria RAM
- Uso de otros recursos (gráficos, acceso a disco, de comunicación, etc.)

Objetivos de la programación

■ Claridad del código fuente

- Un programa es necesariamente tan complejo como el algoritmo que describe.
- Se logra a través de:
 - Separación lógica en partes comprensibles que reflejen la distinción entre los temas que describen, y su presentación en una secuencia lógica que refleje las relaciones entre ellas.
 - Selección de las características del lenguaje.
 - Selección de las palabras usadas para denotar los objetos y conceptos involucrados
 - Inclusión de comentarios

Objetivos de la programación

- Un programador podría ser tan diligente en el uso de éstas técnicas para lograr claridad como el autor de cualquier texto.
- En muchos casos, la utilidad de un programa es determinada tanto por la claridad de su texto como por las cualidades del algoritmo que describe.

Programa Informático

- Un programa se compone de
 - Datos
 - Son la representación dentro de la computadora de aspectos de la realidad.
 - Servirán para ser procesados y producir resultados.
 - Instrucciones.
 - Son la parte central del programa.
 - Manipulan los datos, realizan cálculos, muestran los resultados, etc.

Datos

■ Tipos de datos

- Enteros
- Reales
- Caracteres
- Palabras (Strings)
- Vectores
- Booleans (verdadero, falso)
- Otros

Datos

■ Constantes

- Representan un dato que se mantiene sin cambio alguno, es decir son invariables.

■ Variables

- Representan un valor posible y que puede variar tantas veces como se desea.

Constantes

- Son datos que tienen un significado fijo y conocido.
- Su significado no cambia durante la ejecución del programa.
- Son introducidas mediante declaraciones de la forma:

`<constante> = <valor>;`

- Ejemplo: `pi = 3.1416;`

Constantes

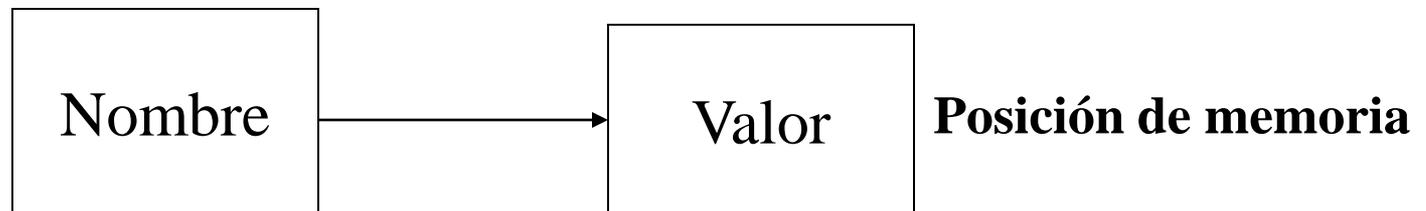
- Declarar una constante tiene como objetivo denotar mediante un nombre, el significado o uso de un cierto valor (fijo) que va a ser manipulado en el programa.
- Usar constantes ayuda a que los programas sean mas fáciles de mantener y modificar.

Variables

- Son datos cuyo valor asociado o significado puede cambiar durante la ejecución del programa.
- Su cambio de valor no es arbitrario, sino producto de la ejecución de ciertas sentencias en el programa.
- Son posiciones de memoria a las que asignamos un nombre y a través de las cuales podremos almacenar y recuperar datos.

Variables

- Por cada variable se reserva una posición en la memoria donde se aloja su valor corriente.
- Tal posición es solo accedida a través del nombre de la variable.



Variables

- Son introducidas mediante declaraciones de la forma:

`<variable> = <valor>;`

- Ejemplo: `x = 3.1416;`

Condiciones

- Los algoritmos con frecuencia presentan situaciones en las que se deben proporcionar acciones alternativas que pueden o no realizarse, dependiendo de los datos de entrada, reflejándose el cumplimiento o no de una determinada condición.

Condiciones

- Diseñar un algoritmo para calcular el salario semanal de un empleado que trabaja por horas.
- La empresa paga una tasa de 1.5 la tasa normal por todas las horas trabajadas mayores a 40.

```
leer(tasa)
leer(horas)
si horas > 40
entonces
    paga <- tasa * 40 + 1.5 * tasa *(horas - 40)
sino
    paga < tasa * horas
fin_si
```

Condiciones

- Las condiciones nos permiten hacer **preguntas** sobre el estado de los **datos** empleados en un programa
- De acuerdo a si las respuestas son **satisfactorias** o no, pueden **tomarse decisiones** sobre las acciones a desarrollarse.
- Las respuestas posibles a las preguntas sobre el estado de los datos pueden ser solo dos:
 - **verdadero** (true), o **falso** (false)
 - **1** o **0**

Condiciones

Expresiones Lógicas

- Son expresiones que solamente pueden tomar uno de dos valores, verdadero o falso
- Los operandos de una expresión lógica pueden ser:
 - **Expresiones relacionales:** que comparan dos valores utilizando operadores relacionales y determinan si existe o no una cierta relación entre ellos.
 - **Expresiones lógicas:** que se relacionan entre si mediante operadores lógicos.

Condiciones

Expresiones Lógicas

■ Expresiones relacionales

- Forma general
 - Exp_1 **operador_relacional** Exp_2
 - Exp_1 debe ser de igual tipo que Exp_2
 - Numéricos
 - Caracteres
- Operadores relacionales
 - Menor que <
 - Menor o igual que <=
 - Mayor que >
 - Mayor o igual que >=
 - Distinto que != ~=
 - Igual que ==

Condiciones

Expresiones Lógicas

Operadores lógicos:

NOT

AND

OR

Condiciones

Expresiones Lógicas

■ Simples

- $2+2==4$
- $5>6$
- 3 es múltiplo de 8

■ Complejas

- $(5>3)$ AND $(3>1)$
- (5 es múltiplo de 2) AND (4 es múltiplo de 2)
- $(4>5)$ OR $(5>4)$
- $(5!=6)$ OR $(5==7)$

Condiciones

- Expresiones lógicas:
 - Operandos
 - Variables y constantes
 - Operadores
 - Operadores relacionales
 - Operadores lógicos,

Condiciones

Expresiones Lógicas

(tablas de verdad:

AND y OR)

OP1	OP2	AND
V	V	V
V	F	F
F	V	F
F	F	F

OP1	OP2	OR
V	V	V
V	F	V
F	V	V
F	F	F

Condiciones

Expresiones Lógicas

(tablas de verdad: NOT)

OP1	NOT
V	F
F	V

Orden de precedencia en la evaluación:

1°	2°	3°
NOT	AND	OR
-	X	+

El orden se puede alterar usando () igual que en expresiones matemáticas

Expresiones lógicas

■ Evaluación estricta

- Se evalúan todas las expresiones para obtener el resultado.

■ Evaluación lazy o circuito corto

- Se evalúan las expresiones de izquierda a derecha.
- Si en algún momento se sabe el resultado definitivo, no se evalúan el resto de las expresiones.

Expresiones lógicas

■ OR:

- evaluación estricta: |
- evaluación lazy: ||

■ AND:

- evaluación estricta: &
- evaluación lazy: &&

■ NOT:

- ! o ~

Instrucciones

- **Asignación**
 - Asignan valores a variables.
- **Estructuras de Control**
 - Organizan el curso (flujo) de ejecución.
 - Pueden basarse en Condiciones

Asignación

- El objetivo de una sentencia de asignación es cambiar el valor almacenado en una variable.

`<variable> = <expresión>`

- Ejemplos:

`x = 3 + 4`

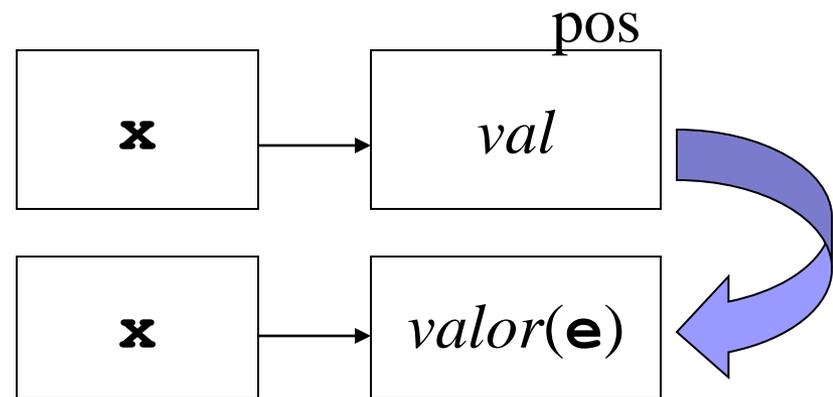
`x = x + (3.0 / 5.9)`

`c = 'a'`

Asignación

- Una asignación $x = e$ es ejecutada siguiendo estos pasos:

- 1.- Se evalúa la expresión e
- 2.- Se reemplaza el valor almacenado en la posición de memoria pos , correspondiente a la variable x , por el valor de e .



Asignación

- La ocurrencia de una variable en el lado izquierdo de una asignación denota la posición de memoria donde almacenar el valor resultante de evaluar la expresión en el lado derecho.

$$x = 1;$$

- La ocurrencia de una variable en el lado derecho de una asignación denota su valor actual.

$$y = x + 1;$$

Asignación

- Una misma variable puede aparecer en la parte izquierda y derecha de una asignación.

$$**x = x + 1**$$

- Esto NO debe interpretarse como una ecuación matemática!
- Sólo significa que estamos usando el valor actual de la variable x para calcular su nuevo valor.