ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Universidade Vigo

Programación I

Tema 2

Grado en Ingeniería Informática

María José Lado Touriño - Xosé Antón Vila Sobrino

Tema 2. Algoritmos y Tipos de Datos

1. Tipos de Datos Básicos
   1.1. Identificadores
   1.2. Constantes
   1.3. Variables
   1.4. Expresiones
   1.5. Funciones
2. Tipos de Datos Definidos por el Usuario

Tema 2. Algoritmos y Tipos de Datos

3. Instrucciones de Decisión/Selección
   - 3.1. Estructuras Condicionales Simples
   - 3.2. Estructuras Condicionales Dobles
   - 3.3. Estructuras de Selección Múltiples
4. Algoritmos Iterativos (Instrucciones de Repetición)
   - 4.1. Mientras
   - 4.2. Hasta
   - 4.3. Desde

1. Tipos de Datos Básicos

Datos

- Información, expresión que describe los objetos con los que opera el algoritmo.
- Captados directamente por el ordenador.
- Se pueden dar en forma de grafismos (alfabéticos, numéricos, caracteres especiales).
- Pueden ser de varios tipos.

Tipos de Datos

Simples (básicos)

- Únicamente almacenan un valor.
- Predefinidos en la mayoría de los lenguajes de programación.
- Pueden ser de los siguientes tipos:
  - Numérico:
    - Entero.
    - Real.
  - Lógico.
  - Cadena.
  - Carácter.

Tipos de Datos Simples

Enteros

- Subconjunto finito de números enteros, cuyo tamaño depende del lenguaje de codificación y de la computadora utilizada.

Reales

- Subconjunto de números reales, limitado en tamaño y precisión.

Lógicos

- Conjunto formado por los valores VERDADERO y FALSO.

Caracteres

- Conjunto finito y ordenado de los caracteres reconocidos por la computadora.

Cadenas

- Serie finita de caracteres.

Todos los datos se representan como constantes, variables, expresiones o funciones.

1.1. Identificadores

- Nombres dados a las constantes (simbólicas), variables, procedimientos, funciones,...
- Reglas de construcción:
  - Deben ser significativos.
  - No pueden coincidir con palabras reservadas del lenguaje.
  - Máximo de 32 caracteres.
  - Comenzarán por carácter alfabético, seguido de más caracteres alfanuméricos.
  - Pueden ser en mayúsculas o minúsculas.
- Ejemplo: dato1, dato_1, Dato1, Dato_1, 1_dato, 1-Dato

1.2. Constantes

- Su valor no cambia durante todo el desarrollo del algoritmo.
- Tipos:
  - Numéricas:
    - Enteras: rango de los números enteros. Compuestas por (+,-) y dígitos (0..9).
    - Reales: compuestas por (+,-), dígitos (0..9) y punto decimal (.).
  - Lógicas: verdadero y falso.
  - Carácter: cualquier carácter reconocible entre ‘ ’.
  - Cadena: serie de caracteres simples entre ‘ ’

Constantes

- Su valor no cambia durante todo el desarrollo del algoritmo.
- Pueden ser:
  - Literales: valor de cualquier tipo.
  - Simbólicas: con nombre (las identifica) y valor asignado.
- Ejemplo:

```
CONSTANTES
   valor : Entero = 5
   letra : Caracter = 'a'
```

1.3. Variables

- Su valor puede cambiar durante el desarrollo del algoritmo.
- Identificadas por:
  - Nombre: identificar el lugar de la memoria donde se almacena.
  - Tipo: determinar conjunto de valores que puede tomar y operaciones que se pueden realizar.
- Necesario declararlas en la mayoría de los lenguajes.
- La declaración reserva espacio necesario en memoria.
- Ejemplo:

```
VARIABLES
   valor : Entero
   letra : Caracter
   decision : Logico
```

1.4. Expresiones

- Combinación de operadores y operandos.
- Operandos: constantes, variables, otras expresiones,...
- Operadores: de cadena, aritméticos, relacionales, lógicos.
- Según resultados que producen pueden ser:
  - Numéricas.
  - Alfanuméricas.
  - Booleanas.

Operadores

Aritméticos:
+, -,*, ^, /, mod, div

Relacionales:
<, >, <=, >=, =, <>

Lógicos:
no, y, o

Prioridad:

^	Exponenciación
no, -	Operadores unarios
*, /, mod, div, y	Operadores multiplicativos
+, -, o	Operadores aditivos
<, >, <=, >=, =, <>	Operadores relacionales

1.5. Funciones

- En los lenguajes de programación suelen existir funciones predefinidas o internas que aceptan argumentos y producen un determinado resultado.
- Se emplean escribiendo su nombre seguidos de los argumentos entre paréntesis.
- Ejemplo:

```
Entero FUNCIÓN Absoluto (E Entero: x)
```

2. Tipos de Datos Definidos por el Usuario

Enumerados

- Tipos de datos simples definidos por el usuario.
- Permiten definir nuevos tipos de datos simples, de cardinalidad (n) reducida.
- Son tipos ordinales (sus valores pertenecen a un conjunto ordenado y finito, y todos tienen un predecesor -excepto el primero- y un sucesor -excepto el último-).

Enumerados

- No tienen operadores específicos.
- Mejoran la legibilidad.
- Pueden tomar un valor uno de los incluidos dentro de un conjunto ordenado de valores definido por la persona usuaria.
- Sintaxis:

```
enumerado <nombreTipo> (<constante1> [= <valor1>],
                        <constante2> [= <valor2>],
                        ……..
                        <constanten> [= <valorn>]
```

Enumerados

- Ejemplo: semáforo

```
TIPOS
   semaforo = (rojo, amarillo, verde)
VARIABLES
   miSemaforo : semaforo
```

- El primer valor del conjunto representa un entero, empezando por 0, y se incrementan de 1 en 1.
- Se puede variar el valor del entero de comienzo:

```
semaforo = (rojo = -2, amarillo, verde)
```

Tipos de Datos Simples

- Estándar
  - Numéricos
    - Enteros
    - Reales
  - Carácter
  - Lógico
- Definidos por el programador
  - Enumerado

3. Instrucciones de Decisión/Selección

- Estructuras secuenciales: cada instrucción se ejecuta a continuación de la anterior. El orden coincide con el orden físico en el que se han colocado las instrucciones. No suficientes a veces.
- Decisión/selección: se ejecuta un conjunto de instrucciones según se verifique o no una determinada condición (una expresión lógica adecuada tomará el valor VERDADERO o FALSO).
- Pueden ser:
  - Simples.
  - Dobles.
  - Múltiples: elección entre varios casos.

3.1. Estructuras Condicionales Simples

- Se evalúa condición.
  - Si se cumple, se ejecuta un conjunto de instrucciones.
- Construcción:

```
SI <condición> ENTONCES
   INSTRUCCIÓN
FIN_SI
```

- Ejemplo pseudocódigo:

```
SI (a = b) ENTONCES
   IMPRIMIR(a)
FIN_SI
```

Estructuras condicionales dobles

- Se evalúa condición.
  - Si se cumple, se ejecuta un conjunto de instrucciones.
  - Si no se cumple, se ejecuta otro conjunto de instrucciones.

- Construcción:

```
SI <condición> ENTONCES
    INSTRUCCIÓN 1
SI_NO
    INSTRUCCIÓN 2
FIN_SI
```

- Ejemplo pseudocódigo:

```
SI (a < b) ENTONCES
    IMPRIMIR (a)
SI_NO
    IMPRIMIR (b)
FIN_SI
```

3.2. Estructuras Condicionales Múltiples

- Permite implementar condiciones más complejas.
- Se encadenan unas condiciones en otras.

- Construcción:

```
SI <condición 1> ENTONCES
    INSTRUCCIÓN 1
SI_NO <condición 2> ENTONCES
    INSTRUCCIÓN 2
SI_NO
    INSTRUCCIÓN 3
FIN_SI
```

- Ejemplo pseudocódigo:

```
SI (a < b) ENTONCES
    IMPRIMIR (a)
SI_NO (a > b) ENTONCES
    IMPRIMIR (b)
SI_NO
    IMPRIMIR(a, b)
FIN_SI
```

3.3. Estructuras de Selección Múltiples

- Selección entre varios caminos posibles, en función del valor de una determinada instrucción.

- Construcción:

```
CASO <expresión> SEA
    <lista 1>: < INSTRUCCIONES 1>
    <lista 2>: < INSTRUCCIONES 2>
    ..........
FIN_CASO
```

- Ejemplo pseudocódigo:

```
CASO v SEA
    1: v <- v + 1
    2: v <- v * 2
    3: v  <- v - 1
FIN_CASO
```

4. Algoritmos Iterativos (Instrucciones de Repetición)

- Permiten la repetición de un conjunto de instrucciones un número determinado de veces.
- Pueden ser:
  - Mientras.
  - Hasta.
  - Desde.

4.1. Mientras

- Se pregunta por una condición al principio.
- Instrucciones del cuerpo del bucle se ejecutan mientras se verifique una determinada condición.

- Construcción:

```
MIENTRAS <expresión_lógica> HACER
   <INSTRUCCIONES>
FIN_MIENTRAS
```

- Ejemplo pseudocódigo:

```
MIENTRAS (x < 100) HACER
   y <- y + x
   x <- x + 1
FIN_MIENTRAS
```

4.2. Hasta

- Se pregunta por una condición al final.
- Instrucciones del cuerpo del bucle se ejecutan mientras se verifique una determinada condición.

- Construcción:

```
REPETIR
   <INSTRUCCIONES>
HASTA_QUE < expresión_lógica>
```

- Ejemplo pseudocódigo:

```
REPETIR
   y <- y + x
   x <- x + 1
HASTA (x < 100)
```

4.3. Desde

- Se pregunta por una condición al principio.
- Instrucciones del cuerpo del bucle se ejecutan mientras se verifique una determinada condición.

- Construcción:

```
DESDE v vi HASTA vf HACER
   <INSTRUCCIONES>
FIN_DESDE
```

- Ejemplo pseudocódigo:

```
DESDE x <- 1 HASTA 100
   y <- y + x
FIN_DESDE
```

Referencias

Joyanes Aguilar, L.; Rodríguez Baena, L.; Fernández Azuela, M. (2008). Fundamentos de programación. McGraw-Hill.

García-Bermejo Giner, J.R. (2008). Programación estructurada en C, Pearson Prentice Hall.

Joyanes Aguilar, L.; Zahonero Martínez, I. (2007). Programación en C : metodología, algoritmos y estructura de datos.