Llenguatge Algorìsmic
Tipus elementals i Objectes

Llenguatges de Programació

GRAM-2

Estructura general d'un algorisme

algorisme *nom_algorisme*
    const

    fconst
    tipus

    ftipus
    var

    fvar
    { estat inicial }
    accions
    { estat final }
falgorisme

accions i funcions

Program *nom_algorisme*
    const

    type

    var

begin
    { accions }
end.

{accions i funcions}

Exemple 1: area d'un triangle

```
Algorítme Area triangle
var
        base, altura, area: real
fvar

LlegirReal (base)
LlegirReal (altura)
area := base * altura / 2.0
EscriureReal (area)
fAlgorítme
```

… en Pascal

```
Program AreaTriangle;

var
    base, altura, area: real;
    letra: char;

begin
    readln (base);
    readln (altura);
    area := base * altura / 2;
    writeln (area);
end.
```

Exemple 2 (I)

- Equació de segon grau:
  - Per a:

$$ax^2 + bx + c = 0$$

  - Tenim:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Exemple 2 (II)

- Si el discriminant:

$$b^2 - 4ac$$

  - és < 0 → no té solució real
  - és > 0 → si té

Exemple 2 (III)

```
Algoritme arrels
var
    a,b,c : real
    discriminant,x1,x2 : real
fvar

escriure ("entra a"); LlegirReal(a)
escriure ("entra b"); LlegirReal(b)
escriure ("entra c"); LlegirReal(c)
discriminant := b*b-4*a*c
si (discriminant < 0)
    llavors escriure("no té solució")
    altrament    x1:=(– b + SQRT(discriminant)) / 2a
                 x2:=(– b – SQRT(discriminant)) / 2a
                 escriure ("X1: ", x1, " X2: ",x2)
fsi
falgorisme
```

Objectes (I)

- Emmagatzemen els valors que defineixen l'estat de l'algorisme en tot moment.
- Atributs dels objectes:
  - **Nom**: permet identificar unívocament un objecte.
  - **Tipus**: rang de valors possibles que pot prendre i operacions aplicables a l'objecte.
  - **Valor**: pot ser modificable o no.

Objectes (II)

- Donat un objecte, direm que és
  - **CONSTANT** si el seu valor no és modificable.
  - **VARIABLE** si el seu valor és modificable.

Tipus elementals (I)

- Rang de valors possibles per als objectes d'aquests tipus.
- Conjunt d'operacions aplicables sobre els objectes del tipus.
- Paraules reservades

Tipus elementals (II)

Nom tipus	enter	real	booleà	caràcter
Valors possibles	Nombres enters entre *min* i *Max*	Subconjunt dels nombres reals	{fals,cert}	{a,b,…,z,A,B,…,Z,+,-,!,…}
Sintaxi valors	–32, 1, 873, 0	0.0, -3.57, -1.3e12, 5.892E–2	fals, cert	'a', 'b', '+', '"', '!', 'A', '?', '>', ';'
Operacions internes	? : enter × enter → enter on ? pot ser +, –, *, div, mod – : enter → enter	? : real × real → real on ? pot ser +, –, *, / – : real → real	? : booleà × booleà → booleà on ? pot ser i, o no : booleà → booleà	
Operacions externes	? : enter × enter → booleà on ? pot ser <, >, ≤, ≥, =, ≠	? : real × real → booleà on ? pot ser <, >, ≤, ≥, =, ≠	? : booleà × booleà → booleà on ? pot ser <, >, ≤, ≥, =, ≠	? : caràcter × caràcter → booleà on ? pot ser <, >, ≤, ≥, =, ≠
Operacions internes no elementals		? : real → real on ? pot ser *arrel*, *abs*, *exp*, *ln*, *sin*, *cos*, …		
Conversió de tipus	*real* : enter → real *car* : enter → caràcter	*enter* : real → enter		*codi* : caràcter → enter

Declaració d'objectes

- Qualsevol objecte ha d'haver estat declarat abans de ser usat.
- Sintaxi:
  - Constants:

```
const
    nom_constant : nom_tipus = valor (expressió constant)
fconst
```

  - Variables

```
var
    nom_variable, ... : nom_tipus
fvar
```

Declaració d'objectes: Exemple

- Càlcul àrea cercle: πr^2
- Càlcul perímetre: $2\pi r$

```
const
        pi : real = 3.1415926
fconst
var
        r, àrea, perím : real
fvar

escriure("entra r")
LlegirReal(r)
àrea := pi * r * r
perím := 2 * pi * r
escriure ("àrea",àrea)
escriure ("Peímetre", perím)
```

Expressions (I)

- Sintaxi:
  - Un valor és una expressió.
  - Una variable és una expresió.
  - Una constant és una expresió.
  - Si **A** és una expresió, **(A)** és una expressió.

Expressions (II)

- Sintaxi:
  - Si **A** és una expressió i **?** un operador unari, **?A** és una expressió.
  - Si **A** i **B** son expressions i **?** un operador binari, **A ? B** és una expressió.
  - Si **A1**, **A2**, … **An** són expressions i **f** una funció, **f(A1, A2, …, An)** és una expressió.

Expressions (III)

- Semàntica
  - No tota expressió sintacticamente correcta te sentit.
  - Cal respectar la tipificació de variables, operadors i funcions.
- Resultat de una expresió
  - numérico: expresió aritmètica
  - logic: expresió de tipus logic
  - caràcter: expresió de caràrcters

Expressions (IV)

- Avaluació d'una expressió
  - Tota expressió en ser avaluada dóna lloc a un **VALOR** d'un cert **TIPUS**.
  - Regles per a avaluar una expressió:
    - Un valor s'avalua al valor que representa.
    - Una constant s'avalua al valor especificat en la seva definició.
    - Una variable s'avalua al valor que emmagatzema en el moment en que es produeix l'avaluació.

Expressions (V)

- Una funció s'avalua al valor calculat per la funció en funció dels valors dels paràmetres.
- Els operadors s'avaluen segons les següents regles de precedència i associativitat:
  - -, no
  - *, /, div, mod
  - +, -
  - <, >, <=, >=, =, <>
  - i, o
  - Associativitat d'esquerra a dreta

Expressions - Exemples

- **4 + 3 * 2 = 10**.
- **(B <= 25)** → donarà cert o fals segons **B**.
- **(A > (B >= 25))** → **A** ha de ser booleà.
- **(10 > B >= 25)** → no té cap sentit.
  S'ha de escriure **(10 > B) i (B >= 25)**
- **'A' < 'C'**→ donarà cert (ordre lexicogràfic)
- **('A' <= lletra) i (lletra <= 'Z')** ens dirà si **lletra** és majúscula.

Exercicis

- Avaluar les siguents expresions:
  - A + B **mod** C (A=5, B=25, C=10)
  - A > B i A > C
  - A + C <= B
  - (C + B + 1) **div** A