405 TEXTOS DOCENTS

INTRODUCCIÓ A LA INFORMÀTICA EXERCICIS

Jaume Baixeries (coord.)
Natàlia Pallarès
Enrique Romero

Departament d'Econometria, Estadística i Economia Aplicada

UNIVERSITAT DE BARCELONA

INTRODUCCIÓ A LA INFORMÀTICA
EXERCICIS

Jaume Baixeries (coord.)
Natàlia Pallarès
Enrique Romero

Departament d'Econometria, Estadística
i Economia Aplicada

405 TEXTOS DOCENTS

UNIVERSITAT DE BARCELONA
Edicions

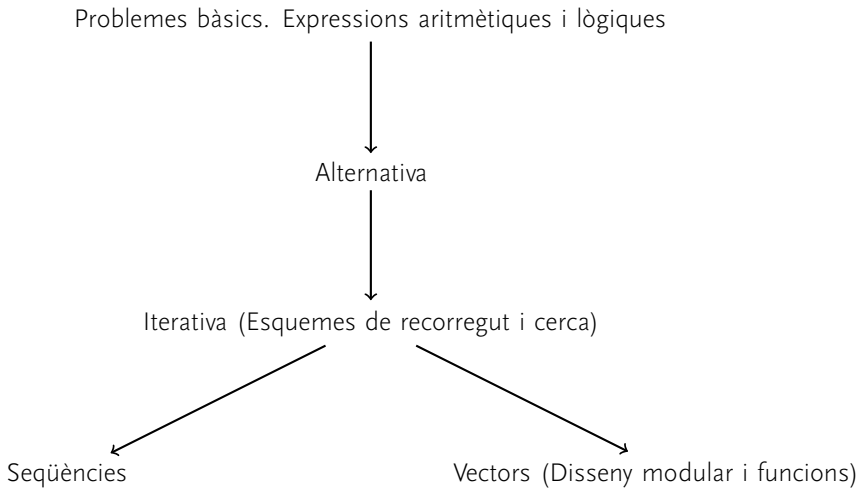
Índex

1. INTRODUCCIÓ

Aquest llibre és una selecció de problemes que s'han fet servir a l'assignatura d'Iniciació a la Programació al grau d'Estadística que s'imparteix, de manera conjunta, per la Universitat de Barcelona i la Universitat Politècnica de Catalunya. Els problemes que conté s'han presentat als estudiants d'aquesta assignatura, durant 5 anys, a les classes de problemes, al laboratori i als exàmens.

Els problemes que hi ha cobreixen el material que es fa a la majoria d'assignatures de programació en estudis que no són específicament d'informàtica, normalment a les enginyeries o a estudis de tipus tècnic, com estadística, matemàtiques o física.

Al principi de cada capítol hi ha una breu explicació dels coneixements que cal per a poder resoldre els problemes. Els capítols estan ordenats de manera seqüencial, de menor a major complexitat. La relació de prerequisit en els capítols d'aquesta col·lecció es pot veure així:



A cada capítol hi hem afegit algunes matèries que en són requeriment. Per exemple, al capítol dedicat a la instrucció iterativa cal assumir que l'estudiant sap quins són els esquemes de cerca i recorregut, que, a més, són requisit per als dos capítols que segueixen: seqüències i vectors.

Alguns problemes es van redactar quan el llenguatge de programació de l'assignatura era el C++; alguns altres, quan el llenguatge de programació era l'R . La presència de la sintaxi d'aquests dos llenguatges en aquesta col·lecció és residual, i, en tot cas, si n'hi ha, es comenta al principi del capítol.

Aquesta col·lecció de problemes es pot utilitzar en qualsevol curs d'iniciació a la programació en estudis fora de l'àmbit de la informàtica, independentment del llenguatge de programació que es faci servir.

2. PROBLEMES BÀSICS

En aquest capítol veiem la instrucció d'assignació, la de lectura i escriptura pels canals estàndards d'entrada i sortida, i també l'avaluació d'expressions.

Assumirem que hi ha els diferents tipus bàsics: enter o numèric, booleà o lògic i cadena de caràcters. La implementació de la solució dependrà de la sintaxi de cada llenguatge. Així, en R no hi ha declaració explícita del tipus, sinò que el tipus s'assigna a una variable quan se li assigna un valor. En canvi, en altres llenguatges imperatius compilats, el tipus es declara de manera explícita quan es declara la variable. Això no afecta de cap manera la resolució dels problemes.

Finalment, en aquest capítol hi ha uns pocs exercicis que contenen instruccions i operadors propis del llenguatge de programació R. Els esmentem i en donem les equivalències en altres llenguatges de programació per mor de generalitat.

Instrucció	R	***C++***	**Python**
Assignació	`var <- EXPR`	`var = EXPR`	`var = EXPR`
Escriptura al canal de sortida	`cat`	`cout <<`	`print`
Lectura del canal d'entrada	`scan`	`cin >>`	`input`
Operador lògic `I`	`&&`	`and`	`and`
Operador lògic `O`	`||`	`or`	`or`

Un dels aspectes més importants a l'hora d'avaluar una expressió és assegurar-nos que és correcta sintàcticament i semànticament. Una expressió és sintàcticament correcta quan la seqüència de símbols es pot analitzar correctament segons la definició que donem a cada operador. Per exemple, una expressió ben formada sintàcticament seria:

```
((10 + 33) * 2) / 33
```

En canvi, una expressió sintàcticament incorrecta seria:

```
+ 32 10
```

Òbviament, si volíem sumar 32 i 10 cal escriure `32 + 10`. Aquí el problema és sintàctic, perquè hem escrit l'expressió en un ordre incorrecte. Si el canviéssim, el problema estaria resolt:

```
32 + 10
```

Fixeu-vos que fem servir els parèntesis per a assegurar la manera d'avaluar (calcular) l'expressió. Ara, mirem l'expressió següent, que té un problema semàntic:

```
32 + "hola"
```

Aquí, per molt que reordenem els elements de què es compon l'expressió, o per molt que n'eliminem algun, l'expressió seguirà sent incorrecta. El problema és que li estem dema-

nant a l'operador + de suma que sumi un nombre (32) i una cadena de caràcters (`"Hola"`), cosa que no pot fer, perquè l'operador de suma *no està definit* en les cadenes de caràcters. Per tant, el problema és semàntic (no estem fent servir l'operador amb els operands del tipus que cal) i no pas sintàctic.

Per tant, un error sintàctic es pot corregir modificant l'ordre dels elements de l'expressió, mentre que un error semàntic, normalment, té a veure amb un ús incorrecte dels tipus.

2.1. Primers problemes

Problema 1. *Preguntes bàsiques: suma.*

Feu un programa que demani 3 valors per l'entrada estàndard i n'escrigui la suma per pantalla.

Problema 2. *Preguntes bàsiques: mitjana.*

Feu un programa que demani 3 valors per l'entrada estàndard i n'escrigui la mitjana per pantalla.

Problema 3. *Preguntes bàsiques: programa.*

Què escriu per la sortida estàndard aquest programa?

```
x <- 20
y <- 30
z <- (x + y) / 2
cat ("el resultat entre ",x," i ",y," és ",x+y,"\n")
```

Problema 4. *Preguntes bàsiques.*

Què escriu per la sortida estàndard aquest programa quan hi entrem 4 per l'entrada estàndard? I si hi entrem 23?

```
x <- scan(n=1,quiet=TRUE)
x <- x %% 2
cat (x,"\n")
```

Problema 5. *Preguntes bàsiques.*

Què escriu per la sortida estàndard aquest programa quan entrem 20 i 30 per l'entrada estàndard?

```
x <- scan(n=1,quiet=TRUE)
y <- scan(n=1,quiet=TRUE)
cat (x > y && x > 0,"\n")
```

Problema 6. *Preguntes bàsiques.*

Què escriu per la sortida estàndard aquest programa quan entrem 2 i 20 per l'entrada estàndard?

```
x <- scan(n=1,quiet=TRUE)
y <- scan(n=1,quiet=TRUE)
ex <- x > 0 && x < 21 && y %% 2 == 0
cat (ex,"\n")
```

Problema 7. *Preguntes bàsiques.*

Què escriu per la sortida estàndard aquest programa?

```
x <- 20
y <- 20
cat (x,y,x+y,"\n")
```

Problema 8. *Preguntes bàsiques.*

Què escriu per la sortida estàndard aquest programa quan hi entrem 4 per l'entrada estàndard? I si hi entrem 23?

```
cat("Escriu un valor entre 1 i 20")
x <- scan(n=1,quiet=TRUE)
cat (x %% 20,"\n")
```

Problema 9. *Instruccions.*

Què escriu per la sortida estàndard aquest programa quan entrem 20 i 30 per l'entrada estàndard?

```
x <- scan(n=1,quiet=TRUE)
x <- scan(n=1,quiet=TRUE)
cat (x,"\n")
```

Problema 10. *Intercanvi de variables.*

Donats dos enters, feu un programa que intercanviï el valor de les dues variables.

2.2. Expressions aritmètiques

Problema 11. *Parell més a prop.*

Feu un programa que, donat un enter, escrigui el nombre parell que té més a prop seu. Exemple: si entra el 23, escriu el 24 (o el 22). Si entra el 22, escriu el 22.

Problema 12. *Segons.*

Donat un nombre de segons, escriu els nombre d'hores, minuts i segons que representen per la sortida estàndard.

Problema 13. *Hora.*

Donat un nombre d'hores, un de minuts i un de segons, escriu per la sortida estàndard el nombre de segons que representen.

Problema 14. *Canvi.*

Donat un import en euros, feu un programa que escrigui la descomposició en bitllets de 100, 50, 20, 10, 5 i monedes d'1 euro.

2.3. Expressions lògiques

Problema 15. *Seqüència.*

Donats 3 enters, feu `TRUE` per la sortida estàndard si i només si els enters fan una seqüència creixent o decreixent.

Problema 16. *Doble.*

Donats 2 enters, feu un programa que escrigui `TRUE` per la sortida estàndard si i només si un dels dos enters és més gran que el doble de l'altre.

Problema 17. *Divisors.*

Donat un enter, feu un programa que escrigui `TRUE` si i només si té almenys un divisor senar més petit que 10.

Problema 18. *Sobre un nombre.*

Donat un enter, feu un programa que escrigui `TRUE` si i només si l'enter és múltiple de 3 i més gran que 30.

Problema 19. *Múltiples?*

Donats 2 enters, feu un programa que escrigui `TRUE` si i només si un dels dos enters és múltiple de l'altre.

Problema 20. *Pesos.*

Donat un enter més gran que 10, feu un programa que escrigui `TRUE` si i només si la segona xifra (començant per la dreta) és el número 7.

Problema 21. *Dígits múltiples.*

Donat un enter més gran que 100 i menor que 1000, feu un programa que escrigui `TRUE` si i només si la suma de tots 3 dígits és múltiple de 5.

Problema 22. *Alarmes.*

Siguin 3 variables booleanes que representen tres alarmes diferents. Se suposa que hi ha alarma perillosa quan **només una** de les alarmes està activada (és `TRUE`). Feu un programa que, donades les 3 variables, escrigui `TRUE` si i només si hi ha una alarma perillosa.

Problema 23. *Classificació.*

Per a 2 equips de futbol, tenim els punts, els gols a favor i els gols en contra. També tenim el resultat dels dos partits que els han enfrontat a la lliga. Feu un programa que escrigui `TRUE` si i només si el primer equip va abans que el segon a la classificació.

A la classificació tenim en compte els punts. En cas d'empat a punts, qui tingui una diferència de gols més gran, i en cas d'empat, qui tingui més gols en el global de tots dos partits que han jugat els equips.

Problema 24. *Capicua.*

Donat un enter més gran que 99 i menor que 1000, feu un programa que escrigui `TRUE` si i només si és un número capicua.

Problema 25. *Expressions.*

Digueu el valor de les expressions següents, tenint en compte els valors inicials de les variables.

```
x <- TRUE
y <- TRUE
z <- TRUE

a <- 44
b <- 10
```

```
d <- 14
```

1. `((x && y) || (z && !y)) == (x == y)`
2. `((a + 1) > 4) && (b %/% 2 > 1)`
3. `((a + b + c) %% 5) > 3`
4. `x || z || ((a + 1) > 4)`

3. ALTERNATIVA

Aquest capítol està dedicat a la instrucció alternativa amb les diferents variants que normalment hi ha en els llenguatges de programació. També en aquest cas donem les correspondències d'aquesta instrucció en els llenguatges de programació R , C++ i Python.

R i C++	Python
`if (EXPRESSIO_1) {`	`if EXPRESSIO_1:`
`  INSTRUCCIO_1`	`  INSTRUCCIO_1`
`} else if (EXPRESSIO_2) {`	`elif EXPRESSIO_2:`
`  INSTRUCCIO_2`	`  INSTRUCCIO_2`
`...`	`...`
`} else {`	`else:`
`  INSTRUCCIO_N`	`  INSTRUCCIO_N`

Problema 26. *Operació.*

Feu un programa que, donats un caràcter (corresponent a un dels caràcters: `+`, `-`, `*`, `/`) i 2 enters, calculi el resultat de fer l'operació demanada entre ells.

Problema 27. *Equació de 2n grau.*

Feu un programa que avaluï una equació de segon grau. Els paràmetres són els coeficients de l'equació.

Problema 28. *Valor absolut.*

Feu un programa que calculi el valor absolut d'un enter que entreu pel teclat.

Problema 29. *Màxim de 3.*

Feu un programa que calculi el màxim de 3 valors que entreu pel teclat.

Problema 30. *Per sobre la mitjana.*

Feu un programa que, donats 4 valors que entreu pel teclat, digui quants d'aquests valors són més grans que la mitjana de tots quatre.

Problema 31. *En ordre.*

Feu un programa que, donats 3 valors per teclat, els imprimeixi en ordre per la pantalla.

Problema 32. *Freqüència.*

Feu un programa que, donades 3 paraules, digui quantes vegades apareix la més freqüent (òbviament, poden ser repetides).

4. ITERACIONS

En aquest capítol, utilitzem la instrucció iterativa, que inclou la instrucció `while` i també la instrucció `for`.

La instrucció `while` té la sintaxi següent:

```
...

instruccio A

while (COND)
{
    Instruccions
}

instruccio B

...
```

i funciona de la manera següent:

1. avalua `COND`.

2. si `COND` avalua `TRUE`:
   (a) executa `Instruccions`
   (b) avalua `COND`.
   (c) si `COND` avalua `TRUE`, torna al pas (a).
   (d) si `COND` avalua `FALSE`, va a `instrucció B`
3. si `COND` avalua `FALSE`, va a `instrucció B`

Vist d'una altra manera: mentre `COND` avaluï `TRUE`, es va executant `Instruccions`.

Una altra variant de la instrucció `while` és la instrucció `for`, que té la sintaxi següent:

```
for (var in SEQ)
{
    Instruccions
}
```

Aquí cal tenir en compte que `var` és una variable qualsevol i que `SEQ` és una **seqüència** d'elements.[1] La variable pren el valor de tots els elements de la seqüència, i per cada valor, executa `Instruccions`.

1 De les seqüències, se'n parla al capítol següent.