Exemple de disseny descendent

Introducció

Josep Vilaplana

ETSEIB/UPC

Enunciat

Un col·lectiu d'aficionats als "récords", motivats per esbrinar quin projecte o obra de la construcció ha durat més temps en nombre de dies, volen un programa que els ajudi. Després de recollir la informació dels tres darrers segles, volen entrar la informació com una seqüència de caràcters que té la següent organització:

$P_1 DI_{d1} DI_{m1} DI_{a1} DF_{d1} DF_{m1} DF_{a1} P_2 DI_{d2} DI_{m2} DI_{a2} DF_{d2} DF_{m2} DF_{a2} \ldots$

$P_i DI_{di} DI_{mi} DI_{ai} DF_{di} DF_{mi} DF_{ai} \ldots P_n DI_{dn} DI_{mn} DI_{an} DF_{dn} DF_{mn} DF_{an} F$

On

P_i *és el nom del projecte* i *de la seqüència, i consisteix d'un seguit de caràcters alfanumèrics (com a molt 30) acabat en un o més espais,*

DI_{di} DI_{mi} DI_{ai} *són enters que respectivament representen el dia, mes i any de la data d'inici del projecte* i *i estan separats per un o més espais,*

Enunciat

DF_{di} DF_{mi} DF_{ai} són enters que respectivament representen el dia, mes i any de la data de fi del projecte i i estan separats per un o més espais

FI representa el final de la seqüència i consisteix dels caràcters 'FI' seguit d'un espai.

Es demana dissenyar un algorisme, que a partir de la seqüència descrita ens escrigui a la sortida, el nom del projecte més llarg en durada amb les seves dates inicials i finals.

Enunciat

Notes*:*

1. *En cas d'empat es decidirà escriure el primer que apareix a la seqüència.*
2. *Una informació que ens pot ser valuosa al fer el disseny és la següent després d'haver consultat una enciclopèdia:*
   En el calendari gregorià, es considera que,
   Anys divisibles per 4 son anys de traspàs;
   excepte que anys divisibles per 100 no son anys de traspàs;
   excepte que anys divisible per 400 son anys de traspàs;
   excepte que anys divisible per 4000 no son anys de traspàs.
3. *En el cas de que la seqüència no tingui cap projecte, el programa no escriurà res.*
4. *No hi ha cap projecte que s'anomeni 'FI'.*

Especificació

{**Pre:** A l'entrada es té una seqüència de caràcters s, i $s = S$
i està estructurada com $S = Pr_1 Pr_2 \ldots Pr_i \ldots Pr_N F$,
on cada Pr_i representa la durada del projecte i identificat
per un nom P_i, amb les seves dates d'inici (DI_i) i final (DF_i),
i F conté els caràcters 'FI' i $N \geq 0$ (nombre de Pr_i de la seqüència)}
$projectes$
{**Post:** A la sortida es té una seqüència de caràcters
t, i $t = T$, i en el cas que $N \neq 0, T = P_{llarg} DI_{llarg} DF_{llarg}$
on P_{llarg} és el nom P_j del projecte Pr_j més
llarg de durada en dies de tots els Pr_i presents en
S i DI_{llarg} i DF_{llarg} són les dates inicials (DI_j)
i finals DF_j del projecte P_j. En el cas que $N = 0, T$ és buida.}

Algorisme

algorisme $projectes$;
  var
    $\ldots$
    $\ldots$
  fvar
  $IniciTractament$
  $ObtenirPrimerElement$
  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte : tProjecte$

$\ldots$

fvar

$IniciTractament$

$ObtenirPrimerElement$

mentre $\neg darrerElement$ **fer**

$tractarElement$

$ObtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte : tProjecte$

$\ldots$

fvar

$IniciTractament$

$llegirProjecte(projecte)$;

mentre $\neg darrerProjecte(projecte)$ **fer**

$tractarElement$

$ObtenirSeguentElement$

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerProjecte(projecte)$ **fer**

    $tractarElement$

    $llegirProjecte(projecte)$;
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte : tProjecte$

$\ldots$

fvar

$IniciTractament$

$llegirProjecte(projecte)$;

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte)$;

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP$;

$projecteLlarg := projecte$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte)$;

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$IniciTractament$

$llegirProjecte(projecte)$;

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte)$;

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP;$
$projecteLlarg := projecte$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte)$;

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$nDiesProjLlarg := -1;$

$llegirProjecte(projecte);$

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte);$

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP;$
$projecteLlarg := projecte$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte);$

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$nDiesProjLlarg := -1;$

$llegirProjecte(projecte);$

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte);$

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP;$
$projecteLlarg := projecte$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte);$

fmentre;

si $nDiesProjLlarg \geq 0 \rightarrow escriureProjecte(projecteLlarg)$

$\square\ nDiesProjLlarg < 0 \rightarrow$

fsi

falgorisme

Nivell Projectes

acció $llegirProjecte($**sor** $projecte : tProjecte);$

{**Pre:** A l'entrada es té una seqüència de caracters s. La part dreta de s, PD, no és buida i la part esquerra de s és PE, i Pr_i és el primer projecte de PD}

{**Post:** projecte representa Pr_i, i Pr_i està després de PE a la part esquerra de s}

funció $darrerProjecte(projecte : tProjecte)$: **booleà**;

{**Pre:** projecte=P}

{**Post:** darrerProjecte(projecte) és cert si P és el projecte sentinella i fals en cas contrari}

acció $escriureProjecte($**ent** $projecteLlarg : tProjecte)$

{**Pre:** projecteLlarg=PLL}

{**Post:** A la sortida es té el nom de PLL, la data inicial de PLL, i la data final de PLL}

funció $nDiesProjecte(projecte : tProjecte)$ **retorna enter**;

{**Pre:** $projecte = P$}

{**Post:** $nDiesProjecte(projecte)$ retorna el nombre de dies que ha durat el projecte P}

Nivell Projecte

Constructores:

Nivell Projecte

Constructores: *llegirProjecte*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores:

Nivell Projecte

Constructores: *llegirProjecte*
Consultores: *nDiesProjecte,*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores: *nDiesProjecte, darrerProjecte,*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores: *nDiesProjecte, darrerProjecte, escriureProjecte,*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores: *nDiesProjecte, darrerProjecte, escriureProjecte,*

tipus

$tProjecte =$ **tupla**

$nomProjecte : tNom;$

$dataInici : tData;$

$dataFi : tData$

ftupla

ftipus

Nivell Projecte

{Pr_i és una subseqüència formada per un nom P_i,
una data DI_i i una data DF_i. El darrer element només
té el nom 'FI' i no té cap data a continuació}

acció $llegirProjecte($**sor** $p : tProjecte);$
  $llegirNom(p.nomProjecte);$
  si $\neg nomFI(p.nomProjecte)$
    $\rightarrow llegirData(p.dataInici);$
      $llegirData(p.dataFi);$
  $\square\ nomFI(p.nomProjecte) \rightarrow$
  fsi

facció { llegirProjecte }

funció $darrerProjecte(projecte : tProjecte)$ **retorna booleà**;
  retorna $nomFI(p.nomProjecte)$

ffunció; { darrerProjecte }

Nivell Projecte

acció $escriureProjecte($**ent** $projecte : tProjecte);$
$escriureNom(projecte.nomProjecte);$
$escriureData(projecte.dataInici);$
$escriureData(projecte.dataFi);$
facció { escriureProjecte }

funció $nDiesProjecte(projecte : tProjecte)$ **retorna enter**;
retorna $nDiesEntre(projecte.dataInici, projecte.dataFi)$
ffunció;

Especificació nivell nom

acció $llegirNom($**sor** $nom : tNom);$

{**Pre:** A l'entrada es té una seqüència de caracters

s. La part dreta de s, PD, no és buida i la part esquerra de

s és PE, i P_i és un nom i és el primer element de PD}
{**Post:** nom representa P_i, i P_i està

a continuació de PE a la part esquerra de s}

funció $nomFI(nom : tNom)$ **retorna booleà**;

{**Pre:** nom=NOM}

{**Post:** nomFI(nom) és cert si NOM és 'FI'. En cas contrari és fals}

acció $escriureNom($**ent** $nom : tNom);$

{**Pre:** nom=NOM}

{**Post:** A la sortida es té una seqüència de caràcters

corresponents als que que hi ha a nom, seguit d'un caràcter

espai.}

Especificacio nivell Data

acció $llegirData($**sor** $data : tData);$

{**Pre:** A l'entrada es té una seqüència de caràcters s. La part dreta de s, PD, no és buida i la part esquerra de s és PE, i D_i és una data i és el primer element de PD}
{**Post:** data representa D_i, i D_i està a continuació de PE a la part esquerra de s}

acció $escriureData($**ent** $d : tData);$

{**Pre:** d=D}

{**Post:** A la sortida es té una seqüència de caràcters que representen la data D seguit d'un caràcter espai.}

funció $nDiesEntre(d1 : tData;\ d2 : tData)$ **retorna enter**;

{**Pre:** d1=D1 i d2=D2 i D1 és anterior a D2}

{**Post:** nDiesEntre(d1,d2) retorna el nombre de dies que hi ha entre la data D1 i la data D2.}

Nivell Data

tipus

$tData =$ **tupla**

$dia, mes, any : enter$

ftupla

ftipus

Nivell Data

acció $llegirData(\textbf{sor}\ data : tData);$
$\quad llegirEnter(data.dia);$
$\quad llegirEnter(data.mes);$
$\quad llegirEnter(data.any);$
facció { llegirData }

Nivell Data

acció $escriureData($**ent** $d : tData);$
  $escriureEnter(d.dia);\ escriureCaracter(\text{`` ''})$
  $escriureEnter(d.mes);\ escriureCaracter(\text{`` ''})$
  $escriureEnter(d.any);\ escriureCaracter(\text{`` ''})$
facció { escriureData }

funció $nDiesEntre(d1 : tData;\ d2 : tData)$ **retorna enter**;
  retorna $nData(d2) - nData(d1)$
ffunció; { nDiesEntre }

Nivell Data

funció $nData(d : tData)$ **retorna enter**;

var $n : enter$; **fvar**

$n := d.dia + nDiesAnyIniMes(d.mes)$

$+(d.any - 1) * 365 + (d.any$ **div** $4) - (d.any$ **div** $100) + (d.any$ **div** $400)$;

si $anyTraspas(d.any) \wedge (d.mes \leq 2) \rightarrow n := n - 1$

$\square \neg anyTraspas(d.any) \vee (d.mes > 2) \rightarrow$

fsi;

retorna n;

ffunció; { nData }

Nivell Data

{Pre: $(mes = MES) \wedge (MES \geq 1) \wedge (MES \leq 12)$}
{Post: $nDiesAnyIniMes(mes)$ és el nombre de dies d'un any de 365 dies de tots els mesos anteriors al MES}
funció $nDiesAnyIniMes(mes$: **enter**$)$ **retorna enter**;
  var $dies$: **enter fvar**;
  si $mes = 1 \rightarrow dies := 0$ { Gener }
  □ $mes = 2 \rightarrow dies := 31$ { Febrer }
  □ $mes = 3 \rightarrow dies := 59$ { Març }
  □ $mes = 4 \rightarrow dies := 90$ { Abril }
  □ $mes = 5 \rightarrow dies := 120$ { Maig }
  □ $mes = 6 \rightarrow dies := 151$ { Juny }
  □ $mes = 7 \rightarrow dies := 181$ { Juliol }
  □ $mes = 8 \rightarrow dies := 212$ { Agost }
  □ $mes = 9 \rightarrow dies := 243$ { Setembre }
  □ $mes = 10 \rightarrow dies := 273$ { Octubre }
  □ $mes = 11 \rightarrow dies := 304$ { Novembre }
  □ $mes = 12 \rightarrow dies := 334$ { Desembre }
  fsi
  retorna $dies$
ffunció; { diesMes }

Nivell Data

funció $anyTraspas(any : enter)$: **booleà**;
  retorna $((any \ \textbf{mod} \ 4 = 0) \wedge (any \ \textbf{mod} \ 100 \neq 0)) \vee (any \ \textbf{mod} \ 400 = 0)$
ffunció

Nivell Nom

const $MaxCaracters$: **enter** $= 30$ **fconst**;
tipus
  $tNom =$ **tupla**
    $long : enter;$
    c : **taula** $[1..MaxCaracters]$ **de caràcter**
  ftupla
ftipus

Nivell Nom

acció $llegirNom($**sor** $n : tNom);$
  var c : **caràcter**; i : **enter**; **fvar**
  $i := 0;\ llegirCaracter(c);$
  mentre $c =$ ‘ ’ **fer** $llegirCaracter(c)$ **fmentre**;
  mentre $c \neq$ ‘ ’ **fer**
    $i := i + 1;\ n.c[i] := c;$
    $llegirCaracter(c)$
  fmentre;
  $n.long := i;$
facció { llegirNom }

Nivell Nom

acció $escriureNom($**ent** $n : tNom);$
  var
    i : **enter**;
  fvar
  $i := 1;$
  mentre $i \leq n.long$ **fer**
    $EscriuCaracter(n.c[i]);$
    $i := i + 1;$
  fmentre;
  $escriureCaracter(\text{`}\ \text{'})$
facció { escriureNom }

funció $nomFI(nom : tNom)$ **retorna booleà**;
  retorna $(nom.long = 2) \wedge (nom.c[1] = \text{`F'}) \wedge (nom.c[2] = \text{`I'})$
ffunció; { nomFI }

bre l'abstracció de dades: Una altre enunc

... Imaginem que l'enunciat és idèntic a l'anterior excepte que on diu

> Es demana dissenyar un algorisme, que a partir de la seqüència descrita ens escrigui a la sortida, el nom del projecte més llarg en durada amb les seves dates inicials i finals.

diu

> Es demana dissenyar un algorisme, que a partir de la seqüència descrita ens escrigui a la sortida, el nombre de dies del projecte més llarg en durada de la seqüència.

Especificació

{**Pre:** A l'entrada es té una seqüència de caràcters s, i $s = S$
i està estructurada com $S = Pr_1 Pr_2 \ldots Pr_i \ldots Pr_N F$,
on cada Pr_i representa la durada del projecte i identificat
per un nom P_i, amb les seves dates d'inici (DI_i) i final (DF_i),
i F conté els caràcters 'Fl' i $N \geq 0$ (nombre de Pr_i de la seqüència)}
$projectes$
{**Post:** A la sortida es té una seqüència de caràcters
t, i $t = T$, i en el cas que $N \neq 0$, $T = nD$ on nD representa el nombre
de dies del projecte més llarg de la seqüència S. En el cas que
$N = 0$, T és buida.}

Algorisme

algorisme $projectes$;
  var
    $\ldots$
    $\ldots$
  fvar
  $IniciTractament$
  $ObtenirPrimerElement$
  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $ObtenirPrimerElement$
  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerElement$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerProjecte(projecte)$ **fer**

    $tractarElement$

    $ObtenirSeguentElement$
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerProjecte(projecte)$ **fer**

    $tractarElement$

    $llegirProjecte(projecte)$;
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;
  var
    $projecte : tProjecte$
    $\ldots$
  fvar
  $IniciTractament$
  $llegirProjecte(projecte)$;
  mentre $\neg darrerProjecte(projecte)$ **fer**
    $nDiesP := nDiesProjecte(projecte)$;
    si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP$;
    $\square$ $nDiesP \leq nDiesProjLlarg \rightarrow$
    fsi;
    $llegirProjecte(projecte)$;
  fmentre;

  $tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$IniciTractament$

$llegirProjecte(projecte)$;

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte)$;

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP$;

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte)$;

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$nDiesProjLlarg := -1;$

$llegirProjecte(projecte);$

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte);$

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP;$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte);$

fmentre;

$tractamentFinal$

falgorisme

Algorisme

algorisme $projectes$;

var

$projecte, projecteLlarg : tProjecte$

$nDiesProjLlarg, nDiesP : enter$

fvar

$nDiesProjLlarg := -1;$

$llegirProjecte(projecte);$

mentre $\neg darrerProjecte(projecte)$ **fer**

$nDiesP := nDiesProjecte(projecte);$

si $nDiesP > nDiesProjLlarg \rightarrow nDiesProjLlarg := nDiesP;$

$\square\ nDiesP \leq nDiesProjLlarg \rightarrow$

fsi;

$llegirProjecte(projecte);$

fmentre;

si $nDiesProjLlarg \geq 0 \rightarrow escriureEnter(nDiesProjLlarg)$

$\square\ nDiesProjLlarg < 0 \rightarrow$

fsi

falgorisme

Nivell Projectes

acció $llegirProjecte($**sor** $projecte : tProjecte);$

{**Pre:** A l'entrada es té una seqüència de caracters s. La part dreta de s, PD, no és buida i la part esquerra de s és PE, i Pr_i és el primer projecte de PD}

{**Post:** projecte representa Pr_i, i Pr_i està després de PE a la part esquerra de s}

funció $darrerProjecte(projecte : tProjecte)$: **booleà**;

{**Pre:** projecte=P}

{**Post:** darrerProjecte(projecte) és cert si P és el projecte sentinella i fals en cas contrari}

funció $nDiesProjecte(projecte : tProjecte)$ **retorna enter**;

{**Pre:** $projecte = P$}

{**Post:** $nDiesProjecte(projecte)$ retorna el nombre de dies que ha durat el projecte P}

Nivell Projecte

Constructores:

Nivell Projecte

Constructores: *llegirProjecte*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores:

Nivell Projecte

Constructores: *llegirProjecte*
Consultores: *nDiesProjecte,*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores: *nDiesProjecte*, *darrerProjecte*

Nivell Projecte

Constructores: *llegirProjecte*

Consultores: *nDiesProjecte, darrerProjecte*

tipus

$tProjecte =$ **tupla**

$final : boolea;$

$nDies$: **enter**

ftupla

ftipus

Nivell Projecte

acció $llegirProjecte($**sor** $p : tProjecte);$

var

$nom : tNom$

$dataInici, dataFi : tData$

fvar

$llegirNom(nom);$

si $\neg nomFI(nom)$

$\rightarrow p.final :=$ **fals**

$llegirData(dataInici)$

$llegirData(dataFi)$

$p.nDies := nDiesEntre(dataInici, dataFi)$

$\square\ nomFI(p.nomProjecte) \rightarrow p.final :=$ **cert**

fsi

facció { llegirProjecte }

Nivell Projecte

funció $darrerProjecte(projecte : tProjecte)$ **retorna booleà**;
  retorna $projecte.final$
ffunció; { darrerProjecte }

funció $nDiesProjecte(projecte : tProjecte)$ **retorna enter**;
  retorna $projecte.nDies$
ffunció;