

Fonaments d'Informàtica Pràctiques

Curs 2001/02



Normativa per a la presentació dels treballs

Dins de l'àrea Informàtica s'imparteixen diverses assignatures durant tots els quadrimestres.

Cada assignatura es valora a través de les notes obtingudes en:

- § La presentació durant tot el quadrimestre de treballs i pràctiques.
- § Les proves de control que es realitzen durant el curs.
- § Notes de Classe.

El lliurament dels treballs necessita de certes dades que han d'estar incloses en la portada del treball per la ràpida classificació en les diferents assignatures de l'àrea.

Aquestes dades aniran a la portada del treball situat de la següent manera:

Nom de l'assignatura i especialitat
Títol del Treball
Nom professor al qui va adreçat el treball Nom alumnes que realitzen el treball Data del treball

Documentació. 10 consells importants.

1. Diagrames

La inclusió d'exemples o/i diagrames és vàlida per aclarir certs punts foscos en la interpretació del disseny i també per facilitar l'assimilació d'idees. No s'utilitzaran mai per la descripció d'algorismes

2. Comentaris:

El disseny i els seus comentaris no han de separar-se mai. No ha d'existir, per tant, el disseny algorísmic per un costat i l'argumentació d'aquest per l'altre.

3. Disseny per refinaments:

Quan el disseny és el suficientment extens i/o complicat es recomana realitzar la divisió d'aquest en petits blocs per realitzar un desenvolupament individualitzat. Els blocs han de tenir independència pròpia, és a dir, realitzar una acció o funció concreta per què així la seva correcció o execució pugui realitzar-se sense dificultat. En aquest punt és molt recomanable l'aplicació dels criteris bàsics sobre disseny descendent.

4. Documentació estructurada.

L'Estructuració de la documentació és necessària i permet una localització ràpida dels elements dins d'ella. S'ha d'evitar que la documentació es torni il·legible. Lògicament, la divisió de la mateixa en seccions i subseccions reflexa l'estructura del disseny descendent efectuat.

5. Noms

És fonamental que programes, constants, variables, tipus i accions, tinguin identificadors (noms) significatius. Són necessaris per una millor lectura del disseny i una bona interpretació del mateix.

S'ha d'escollir un criteri per crear els noms de tots els identificadors. És molt recomanable l'ús de verbs per designar accions; adjectius o combinacions amb el verb 'ser' per identificar funcions lògiques (booleanes) i substantius per la resta de funcions, variables i tipus. També és molt interessant:

- § Utilitzar majúscules per identificar constants (per ex. **MAXLISTA**),
- § Utilitzar el prefix 't_' per identificar Tipus de variables (per ex. **t_lista**)
- § Utilitzar el prefix 'f_' per identificar arxius o fitxers (per ex. **f_alumnos**)
- § Utilitzar minúscules per les variables simples
- § En cas d'utilitzar noms compostos o amb més d'una paraula (molt recomanable) aquestes es poden separar amb un guió de subratllat (per

ex., **mi_programa** o **busca_código**) o amb combinació de majúscules i minúscules (per ex., **MiPrograma** o **BuscaCódigo**)

Tot això son només recomanacions i cadascú pot establir el criteri que vulgui. En qualsevol cas, es demana que quan s'adopti un criteri, aquest es mantingui i sigui coherent durant tot el disseny i, molt recomanat, en dissenys posteriors.

6. Trivialitats

Les accions d'entrada de dades i sortida de resultats son problemes de mínima rellevància i no cal desenvolupar-los en el disseny (o, al menys, cal tractar-los d'una manera marginal). Per exemple,

@ **ACCIO** llegir mi_dato @

@ **ACCIO** llegir registre mi_registre de fitxer f_datos @

@ **ACCIO** escriure mi_dato @

@ **ACCIO** escriure registre mi_registre en fitxer f_datos @

són primitives o accions que no cal desenvolupar, ja que l'acció que realitzen ja és prou explícita. Igualment, quan un procediment o subprograma tingui un desenvolupament trivial, o que ja s'ha desenvolupat i està ben especificat, no cal tornar-lo a dissenyar.

7. Tractament d'errors.

De la mateixa manera que no es desenvolupen les accions d'entrada de dades i impressió de resultats, s'han d'evitar també en el disseny tot allò relatiu a la detecció i tractament d'errors que no siguin propis del programa. Això no vol dir que aquest aspecte no tingui importància. El tractament d'errors és una part molt important en el funcionament del programa, però és una funció pròpia de cada llenguatge, i per això no es pot incloure dins del desenvolupament del disseny, perquè complicant l'exposició i comprensió del disseny i no porten gaire més beneficis a l'aprenentatge de l'alumne.

8. Traducció o implementació del disseny a un llenguatge.

És possible que en la fase d'implementació sigui necessari la modificació del disseny original per adaptar-lo al llenguatge (limitacions del llenguatge, noves variables, tractament d'errors, etc.). En aquest cas, s'inclourà dins aquesta fase els comentaris oportuns. Aquí i només en aquesta secció es podrà realitzar el tractament d'errors que no siguin. Serà necessari també que el llistat complet de la implementació haurà d'estar escrit amb els seus sagnats o tabulacions corresponents, línies en blanc, comentaris, etc. que garanteixin la seva llegibilitat.

9. Conclusions

En resum, al disseny d'una aplicació s'han de seguir cronològicament aquests punts:

- § Anàlisi del problema a resoldre
- § Desenvolupament del disseny
- § Implementació en un llenguatge concret.

La realització d'aquests punts d'una forma inversa a la descrita, no justificarà de cap manera el desenvolupament del disseny de l'aplicació, ja que és una traducció literal de les línies de codi del programa en un text semblant a l'algorímic. De cap manera es considerarà això com un disseny.

Pràctica 1. Arquitectura del computador

Objectius de la pràctica

- Introducció a l'arquitectura i funcionament intern de l'ordinador.
- Coneixement dels diferents dispositius de la màquina.
- Comprensió dels termes: CPU, memòria, bus, perifèric, sistema operatiu, fitxer.
- Identificació i solució de petits problemes de hardware i software.

Exercicis Obligatoris

1.1 Escriviu a un full en blanc dos o tres termes relacionats amb el món de la informàtica que no arribeu a entendre bé. Durant cinc minuts, cerqueu informació a Internet o entre els companys sobre un d'aquests termes. A continuació feu una posada en comú dels resultats obtinguts, intentant resoldre els dubtes dels companys (i demanant el seu ajut i el del professor en els dubtes que encara tingueu).

1.2 Individualment, feu un dibuix o esquema de l'arquitectura d'un ordinador. A continuació, en grups de tres persones poseu en comú els vostres esborranys i realitzeu un esquema final conjunt, on ha de constar:

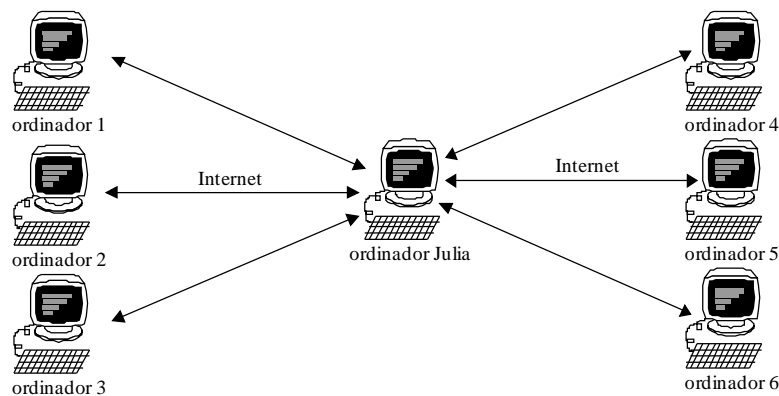
- Nom dels diferents components.
- Funció de cada component.
- Quins components són imprescindibles i quins no.

Per últim, obriu un o més ordinadors, i identifiqueu les parts que apareixen a l'esquema anterior.

1.3 Imagineu-vos que sou venedors/res d'una botiga d'informàtica. Aconselleu als següents cinc clients que arriben a la vostra botiga sobre quin és l'ordinador s'ajusta a les seves necessitats, és a dir, quins components i accessoris necessitaran, com han de ser aquests i el perquè. Com a llista de components tenim: microprocessador, memòria RAM, ratolí, teclat, joystick, disquetera, disc dur, CD-ROM, DVD, gravadora CD-ROM, escàner, monitor, impressora, mòdem, tarja de xarxa, tarja de vídeo, tarja de so, altaveus, caixa i s.a.i.

- Client 1: Na Júlia és una enginyera. A la feina necessita un ordinador per fer simulacions de dinàmiques de fluids (càlculs molt costosos i moltíssimes dades). La resta d'ordinadors de la feina accediran constantment a aquest

ordinador mitjançant la seva xarxa local o intranet per consultar els resultats de les simulacions. Imagineu-vos:



- Client 2: En Xavi és un noi que ja té un ordinador. L'utilitza sobretot per jugar: és un apassionat dels videojocs. El problema és que els últims jocs que ha comprat ja van una mica lents en el seu ordinador. Quins components del seu ordinador li cal canviar per millorar el rendiment d'aquest a l'hora de jugar amb els jocs? A més a més, en Xavi i la seva germana Eulàlia estan aprenent solfeig i volen formar un grup. Volen utilitzar l'ordinador com a assistent a l'hora de compondre música.
- Client 3: En Joan treballa a una oficina. Vol un ordinador per a poder escriure a casa informes i fer feina del treball a casa. Com és l'ordinador que li vendríeu?
- Client 4: I a tu, quin ordinador t'agradaria tenir? Per què?

Exercicis d'intensificació

1.4 Cerqueu informació i feu un breu resum sobre quins han estat els canvis més importants a la història dels microprocessadors de la família Intel 80x86, d'ençà el 8088 fins el Pentium 4.

1.5 Quins trets diferencien els sistemes operatius MS-DOS, Windows i Linux?

Pràctica 2. Codificació

Objectius de la pràctica

Coneixement i resolució de problemes en els temes següents:

- Canvi de base (binari, hexadecimal i decimal).
- Codificació de nombres enters amb signe.
- Aritmètica binària.
- Codificació de nombres reals en simple i doble precisió.
- Memòria i Precisió.
- Codis detectors i correctors d'errors.

2.1 Canvis de base.

Completar la següent taula:

Binari	Hexadecimal	Decimal
		128.75
10011101.11001		
	BE.A7	

2.2 Codificació de nombres enters amb signe.

Representar els nombres 223 i -223 en binari en els diferents codis per a la representació d'enters amb signe.

		Binari
Binari natural	223	
	-223	
Signe i magnitud	223	
	-223	
Complement a 2	223	
	-223	
Excés	223	
	-223	

Calcular el valor decimal del nombre binari 10100111_2 representat en els diferents codis.

		Decimal
Binari Natural	10100111 ₂	
Signe i Magnitud	10100111 ₂	
Complement a 2	10100111 ₂	
Excés	10100111 ₂	

2.3 Aritmètica binària.

Realitzar les següents operacions en 8 bits i complement a 2, donant el valor del resultat en decimal.

Sumar

$26 + 24 =$

$-15 + 82 =$

$84 + 69 =$

$-46 + (-10) =$

2.4 Codificació de Nombres Reals

Representar 123.12 segons norma IEEE 754 en simple precisió. Utilitzar truncat o arrodoniment si fos necessari.

2.5 Memòria i precisió

En un microprocessador de 16 bits desitgem executar un programa que ocupa 20000 posicions de memòria precisant a més: 10000 dades enteres (16 bits), 5000 dades reals de simple precisió (32 bits), 2000 dades reals de doble precisió (64 bits) i 1000 alarmes binàries (1 bit). Indicar la capacitat mínima de la memòria necessària en posicions de memòria, bits, bytes i quilobytes.

2.6 Codi ASCII. Codi detector d'errors. Codis de Paritat.

Volem enviar per una línia telefònica una informació en paquets de 8+1 bits. Aplicar un codi de paritat parella per enviar la cadena de caràcters "Alumne" (sense les cometes) en codi ASCII de 8 bits. Discutir la possibilitat de detectar transmissions errònies.

Aplicar també un codi de paritat senar a la mateixa cadena ASCII.

Pràctica 3. Entorn Turbo Pascal

Objectius de la pràctica

- Introducció a la utilització de l'entorn de programació Turbo Pascal
- Procediments d'entrada/sortida
- Expressions aritmètiques
- Precisió en la representació de les quantitats numèriques

Introducció a l'entorn Turbo Pascal (TP)

Per posar en marxa l'entorn de TP cal seguir els següents passos:

1. Engagar l'ordinador
2. Accedir al directori de l'entorn: `c:\cd tp` (per a la versió 7.0 del TP)
3. Invocar l'entorn: `c:\tp> turbo`

Les funcions més representatives de l'entorn TP són:

Tecla	Menú	Funció
F10		Permet accedir al menú principal
F3	File → Load	Obre un arxiu existent
F10,F,N	File → New	Obre un arxiu nou amb el nom NONAME.PAS
F2	File → Save	Guarda un arxiu amb el nom que té
F10,F,a	File → Save as	Guarda un arxiu amb un nom diferent
Alt + F9	Compile → Compile	Compila el programa
Ctrl + F9	Run → Run	Executa el programa
F7	Run → Trace into	Executa el programa pas per pas
F4	Run → Go to cursor	Executa el programa fins la posició del cursor
Ctrl + F7	Debug → Add watch	Obre una finestra per visualitzar el valor d'una o més variables durant una execució pas per pas
Ctrl + F4	Debug → Evaluate	Obre momentàniament una finestra per visualitzar i/o canviar el valor d'una variable durant una execució pas per pas
Alt + F5	Debug → User screen	Canvia de la pantalla d'edició a la pantalla de resultats
F6	Window → Next	Canvia de finestra, quan hi han vàries obertes
F1		Ajuda
Maj + F1	Help → Index	Mostra ajuda sobre la paraula o tema que cerquem
Alt + X	File → Exit	Surt de l'entorn Turbo Pascal

Exercicis amb l'entorn TP. Precisió amb les operacions aritmètiques

3.1 Suma d'enters

L'editor amb el que treballem té les opcions de treball amb blocs. Això permet definir un bloc com una part del text amb el que treballem, i sobre el que podem realitzar un conjunt d'accions: moure, copiar, esborrar, guardar, recuperar.

En l'ajuda (F1) tenim la referència de les opcions de treball amb l'editor.

En el següent exemple podem marcar com un bloc les línies 5 i 6. Si escollim l'opció de copiar bloc, ja no ens caldrà copiar completament la 7 i la 8. L'únic que haurem de fer serà modificar-les.

1. Copieu el següent programa a l'entorn TP

```
PROGRAM Suma_enters;  
VAR i, j, k : INTEGER;  
BEGIN  
    WRITELN('Suma de dos enters.');
```

WRITE('Primer nombre: ');
 READLN(i);
 WRITE('Segon nombre: ');
 READLN(j);
 k := i + j;
 WRITELN('Resultat = ', k);
END.

2. Després d'haver copiat el programa guardeu-lo en disc amb el nom A3_1.pas

3. Executeu el programa:

- a) Primer en realitzarem la compilació
- b) Quan la compilació no doni cap error el farem funcionar amb l'opció RUN

4. Proveu el programa com a mínim amb els següents valors:

200 + 350
250 – 300
20000 + 30000
40000 – 10

5. Comproveu el resultat i esbrineu el que ha succeït.

3.2 Suma de Reals

1. Utilitzeu el programa anterior modificant-lo de forma que quedi:

```
PROGRAM Suma_reals;  
VAR i, j, k : REAL;  
BEGIN  
  WRITELN('Suma de dos reals.');
```

WRITE('Primer nombre: ');
READLN(i);
WRITE('Segon nombre:');
READLN(j);
k := i + j;
WRITELN('Resultat = ', k);

```
END.
```

2. Després d'haver modificat el programa salveu-lo en disc amb el nom A3_2.pas

3. Proveu el programa com a mínim amb els següents valors:

250.30	+	300.50
1E12	+	1.0
3333333333333333.333	+	6.667
987654321004	+	-987654321002
123456789876543	+	0

Comproveu el resultat i esbrineu el resultat que dona. Són correctes?
Perquè?

3.3 Execució pas per pas i traça de variables.

1. Obriu a l'entorn del Turbo Pascal qualsevol dels dos programes anteriors.
2. Amb Ctrl+F7 obriu una finestra per visualitzar el valor de la variable *i*. Repetiu aquest procés per visualitzar també el valor de les variables *j* i *k*.
3. Canvieu la mida de la finestra del programa per tal que totes dues finestres, la del programa i la dels valors de les variables, siguin visibles alhora. Ho aconseguireu arrossegant amb el ratolí la cantonada inferior dreta de la finestra del programa, o a l'opció Size/Move del menú Window amb la combinació de tecles Maj+fletxes.
4. Proveu el programa executant pas per pas (tecla F7) amb els valors proporcionats pel corresponent exercici anterior, tot fixant-vos en quins moments es produeixen desbordaments i pèrdues de precisió. Amb la combinació de tecles Alt+F5 podreu veure, quan ho necessiteu, la pantalla dels resultats.

Petit annex I: entrada i sortida en mode text a Turbo Pascal

Turbo Pascal 7 disposa d'instruccions d'entrada i sortida per pantalla que permeten donar als nostres programes una aparença més professional i, fins i tot, divertida. Cal tenir en compte, però, que aquestes instruccions no pertanyen al llenguatge Pascal estàndard, sinó tan sols a l'entorn de programació Turbo Pascal.

Per fer servir aquestes instruccions, cal indicar-ho escrivint sota el nom del programa el nom de la unitat on estan contingudes:

```
Uses crt;
```

Algunes de les instruccions més interessants són:

- | `ClrScr`: esborra el contingut de la pantalla.
- | `GotoXY(x, y)`: posiciona el cursor a les coordenades especificades.
- | `TextColor(color)`: canvia el color del text.
- | `TextBackground(color)`: canvia el color del fons.
- | `NormVideo`: restaura els colors originals.
- | `KeyPressed`: retorna cert o fals segons s'hagi premut o no una tecla.
- | `ReadKey`: retorna un únic caràcter llegit pel teclat.
- | `Delay(temp)`: espera el nombre de mil·lisegons especificat.
- | `Sound(hertzis)`: reproduïx un só per l'altaveu fins que s'executa `NoSound`.

Exemple:

```
Program prova_pantalla;
uses Crt;

var c: char;

begin
  ClrScr; (* Esborra pantalla *)
  GotoXY(5,10); (* columna 5 i línia 10 *)
  Write('Hola! Prem una tecla '); (* Escriu una salutació *)
  c := Readkey; (* Llegeix una tecla *)
  GotoXY(1,15); (* columna 1 i línia 15 *)
  TextColor(Green+Blink); (* Text verd pampallugant *)
  TextBackground(LightGray); (* Fons gris brillant *)
  Writeln('Has premut ', c); (* Escriu la tecla *)
  Writeln('El seu valor ASCII és ', Ord(c));
  Sound(1000); (* Xiula *)
  Delay(200); (* Espera 200 miliseg. *)
  NoSound; (* Deixa de xiular *)
  NormVideo; (* Restaura els atributs *)
  Writeln('Adeu i gràcies!') (* Escriu una despedida *)
```

end.

Petit annex II: entrada i sortida en mode gràfic a Turbo Pascal

Turbo Pascal 7 disposa d'instruccions de dibuix que permeten inserir gràfics als nostres programes. Cal tenir en compte, però, que aquestes instruccions no pertanyen al llenguatge Pascal estàndard, sinó tan sols a l'entorn de programació Turbo Pascal.

Per fer servir aquestes instruccions, cal indicar-ho escrivint sota el nom del programa el nom de la unitat on estan contingudes:

```
Uses graph;
```

També cal iniciar i finalitzar el funcionament en mode gràfic amb les instruccions:

```
InitGraph(GraphDriver, GraphMode, 'camí als controladors');  
...  
CloseGraph;
```

Algunes de les instruccions més interessants són:

- | `SetColor(color)`: canvia el color de dibuix.
- | `SetBkColor(color)`: canvia el color del fons.
- | `SetLineStyle(estil, patró, ample)`: estableix el tipus de línia de dibuix.
- | `SetFillStyle(patró, color)`: estableix el tipus de farciment.
- | `GetMaxX` i `GetMaxY`: obtenen les coordenades màximes referenciables.
- | `GetX` i `GetY`: obtenen les coordenades actuals.
- | `MoveTo(x, y)`: desplaça les coordenades actuals al punt (*x*, *y*).
- | `PutPixel(x, y, color)`: dibuixa un punt del color especificat.
- | `Line(x1, y1, x2, y2)`: dibuixa una línia del punt (*x*₁, *y*₁) al punt (*x*₂, *y*₂).
- | `LineTo(x, y)`: dibuixa una línia de les coordenades actuals al punt (*x*, *y*).
- | `Circle(x, y, radi)`: dibuixa un cercle a les coordenades especificades.
- | `Rectangle(x1, y1, x2, y2)`: dibuixa un rectangle delimitat per les coordenades (*x*₁, *y*₁) i (*x*₂, *y*₂).
- | `FloodFill(x, y, color_frontera)`: omple una àrea delimitada per un color determinat.
- | `OutTextXY(x, y, text)`: escriu un text a les coordenades especificades.

- | `SetTextStyle(font, direcció, mida)`: canvia el tipus de lletra, la mida i la direcció vertical o horitzontal.

Exemple:

```
Program prova_grafics;

uses Graph;

var Driver, Mode, angle: Integer;

begin
  DetectGraph(Driver, Mode);           (* Detecta mode gràfic *)
  InitGraph(Driver, Mode, 'c:\tp\bgi'); (* Inicia mode gràfic *)
  SetBkColor(Black);                  (* Color de fons negre *)

  SetColor(White);                    (* Color de dibuix blanc *)
  Line(0, 0, GetMaxX, GetMaxY);       (* diagonal a tota pantalla *)

  SetColor(Red);                      (* Color de dibuix vermell *)
  SetLineStyle(DashedLn, SolidFill, 3); (* Línia a ratlles gruixuda *)
  Rectangle(50, 350, 450, 450);      (* Rectangle 400 x 100 *)

  (* Dibuixa la funció sinus en groc dins el rectangle *)
  FOR angle := 0 TO 359 DO
    PutPixel(70+angle, 400+Round(40*Sin(angle*PI/180)), Yellow);

  SetColor(Blue);                     (* Color de dibuix blau *)
  SetLineStyle(SolidLn, SolidFill, 1); (* Línia sòlida fina *)
  Circle(400, 100, 50);               (* Cercle de radi 50 *)
  SetFillStyle(SolidFill, Green);     (* Farciment verd *)
  FloodFill(400, 100, Blue);          (* Omplim el cercle *)

  SetTextStyle(SansSerifFont, VertDir, 1); (* Nou estil de text *)
  OutTextXY(380, 80, 'prova');        (* Text dins cercle *)

  Readln;                              (* Espera que l'usuari premi INTRO *)
  CloseGraph;                           (* Tanca mode gràfic *)
end.
```

Pràctica 4. Seqüencials

Objectius de la pràctica

- Expressions aritmètiques
- Expressions lògiques
- Assignacions
- Procediments d'entrada i sortida

Exercicis Obligatoris

4.1 Observeu el següent programa en Pascal, que implementa el càlcul de l'àrea i el perímetre d'un cercle, donat un radi r .

Equació de l'àrea d'un cercle: $A = \pi * r^2$

Equació del perímetre d'un cercle: $P = 2 * \pi * r$

Responen les següents qüestions:

- Quines variables són d'entrada, quines auxiliars i quines de sortida?
- Com és que podem fer servir PI, si no ha estat declarat?
- Per què posem un missatge (WRITE) davant l'entrada de dades (READ)? És obligatori fer-ho així? Què passaria si no?

Programa en Pascal

```
PROGRAM A4_1;

VAR radi, area, perimetre : REAL;

BEGIN
  (* Demanem el radi *)
  WRITE('Quin és el radi ? ');
  READLN(radi);

  (* Calculem l'àrea i el perímetre *)
  perimetre := 2 * PI * radi;
  area := PI * radi * radi;

  (* Visualitzem els resultats *)
  WRITELN('El perímetre és ', perimetre:4:2);
  WRITELN('L'àrea és ', area:4:2);
END.
```

4.2 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que permetin trobar el punt mig de dos punts de l'espai bidimensional, segons la fórmula:

$$\text{Siguin els punts } \vec{a} = (a_x, a_y) \text{ i } \vec{b} = (b_x, b_y) \text{ llavors } \vec{m} = \vec{a} + \vec{b} = \left(\frac{a_x + b_x}{2}, \frac{a_y + b_y}{2} \right)$$

Exercicis d'intensificació

4.3 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que, donat un nombre enter que designa un període de temps expressat en segons, retornin l'equivalent en dies, hores, minuts i segons.

Per exemple: 24000 segons seran 0 dies, 6 hores, 40 minuts i 0 segons.

Per exemple: 7400 segons seran 0 dies, 2 hores, 3 minuts i 20 segons.

Pràctica 5. Estructures alternatives

Objectius de la pràctica

- Utilització de l'estructura alternativa simple, doble i múltiple

Exercicis Obligatoris

5.1 Siguin A , B i C tres variables enteres que representen les vendes de tres productes A , B i C , respectivament. Utilitzant aquestes variables, escriviu les expressions que representin les següents afirmacions:

- Les vendes del producte A són les més elevades.
- Cap producte té unes vendes inferiors a 200.
- Algún producte té unes vendes superiors a 400.
- La mitjana de vendes és superior a 500.
- El producte B no és el més venut.
- El total de vendes està entre 500 i 1000.

5.2 Donada una variable c de tipus caràcter, trobeu les expressions que valguin cert si i només si:

- c és una vocal.
- c és una lletra minúscula.
- c és un símbol de l'alfabet.

5.3 Observeu el següent programa en Pascal que, introduïts tres nombres qualsevol pel teclat, calcula el mínim i el màxim dels tres nombres i els mostra per pantalla.

Responen les següents qüestions:

- Per què s'utilitzen estructures alternatives niuades?
- Fa servir comparacions repetides o innecessàries? Es podria fer d'una altra manera amb menys comparacions?

Programa en Pascal

```
PROGRAM A5_3;

VAR min, max : REAL; (* Aquí guardarem el mínim i el màxim *)
    a, b, c : REAL; (* Aquí guardarem el valors introduït *)

BEGIN
  WRITE('Introdueix el primer valor ');
  READLN(a);
  min := a;
  max := a;

  WRITE('Introdueix el segon valor ');
  READLN(b);
  IF b > max THEN
    max := b
  ELSE
    min := b;

  WRITE('Introdueix el tercer valor ');
  READLN(c);
  IF c > max THEN
    max := c
  ELSE
    IF c < min THEN
      min := c;

  WRITELN('El mínim és ', min);
  WRITELN('El màxim és ', max);
END.
```

5.4 Observeu el següent programa en Pascal que, a partir del número del dia de la setmana introduït (de 1 a 7), si aquest és laboral escriu el nom del dia corresponent per la pantalla, i si no escriu festiu.

Responen les següents qüestions:

- Quan es pot emprar l'alternativa múltiple a Pascal i quan no?
- Es podria escriure aquest programa amb estructures alternatives IF niuades, enlloc de l'alternativa múltiple CASE?
- Quines avantatges té llavors el CASE sobre l'IF?

Programa en Pascal

```
PROGRAM A5_4;

VAR dia : 1..7;

BEGIN
  WRITE('Introdueix un dia de la setmana (entre 1 i 7) : ');
  READLN(dia);

  WRITE(' El dia és ... ');

  CASE dia OF
```

```
1: WRITELN('dilluns');
2: WRITELN('dimarts');
3: WRITELN('dimecres');
4: WRITELN('dijous');
5: WRITELN('divendres');
6,7: WRITELN('festiu');
ELSE
  WRITELN('incorrecte');
END;
```

5.5 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal tal que, introduïts dos nombres qualsevol a i b per teclat, trobin la solució a l'equació de primer grau:

$$a x + b = 0$$

Tingueu en compte que hi ha tres possibles solucions:

- Quan $a \neq 0$ existeix la solució única $x = -b/a$.
- Quan $a = 0$ i $b \neq 0$ no existeix solució.
- Quan $a = 0$ i $b = 0$ existeixen infinites solucions.

5.6 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal tal que, introduïda la mida d'un cargol pel teclat mostrin per pantalla el text corresponent a la mida, segons la següent taula:

De 1 cm. (inclòs) fins a 3 cm. (no inclòs)	Petit
De 3 cm. (inclòs) fins a 5 cm. (no inclòs)	Mitjà
De 5 cm. (inclòs) fins a 6.5 cm. (no inclòs)	Gran
De 6.5 cm. (inclòs) fins a 8.5 cm. (no inclòs)	Molt gran

5.7 Amplieu l'exercici 4.2 per tal que, a més a més de trobar el punt mig dels dos punts de l'espai bidimensional, escrigui per pantalla a quin quadrant del pla pertany aquest punt resultant.

Exercicis d'intensificació

5.8 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que ens permetin resoldre l'equació de segon grau $a x^2 + b x + c = 0$. La formula matemàtica que resol aquesta equació és la següent:

$$x = \frac{b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2}$$

Cal tenir en compte els casos 'especials' en la seva resolució:

- L'equació és de primer grau ($a = 0$), però es pot calcular el resultat fent servir l'algorisme de l'exercici 5.5.
- Les arrels són imaginàries ($b^2 - 4ac < 0$), però es pot mostrar el resultat separant la part real de la imaginària.

En Pascal, per calcular l'arrel quadrada podem utilitzar $\text{SQRT}(x)$ i per elevar al quadrat $\text{SQR}(x)$.

5.9 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que ens permetin calcular l'arcsinus d'un determinat valor x a partir de la funció arctangent. Cal tenir en compte que la funció $\text{ArcSin}(x)$ no existeix en el Pascal i el que farem és utilitzar l'equivalència trigonomètrica. Els valors 1 i -1 reben un tracte especial. Al final, el valor resultant en radians s'haurà de passar a graus.

$$\arcsin(x) = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right) \quad \forall x \in]-1, 1[$$
$$\arcsin(-1) = -\frac{\pi}{2}$$
$$\arcsin(1) = \frac{\pi}{2}$$

5.10 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que, donada una data (dia, mes i any), determini si la data correspon a un valor vàlid. Les dates seran tres dades de tipus enter que correspondran a un dia, un mes i un any (dd, mm, aa).

S'ha de tenir present el valor dels dies en funció dels mesos i dels anys. És a dir:

- Els mesos 1, 3, 5, 7, 8, 10 i 12 tenen 31 dies.
- Els mesos 4, 6, 9 i 11 tenen 30 dies.
- El mes 2 te 28 dies, excepte quan l'any és divisible per 4, que té 29 dies.

5.11 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que ens permeti escollir entre les següents opcions:

1. Calcular el sinus d'un nombre a .
2. Calcular el logaritme neperià d'un nombre a .
3. Calcular el logaritme en base b d'un nombre a .

Cal observar que no existeix en Pascal una funció que permeti directament el logaritme en una base qualsevol. En el seu lloc podem fer servir la següent fórmula: $\log_b a = \ln a / \ln b$

Pràctica 6. Estructures iteratives

Objectius de la pràctica

- Estudi de les estructures iteratives de condició inicial, de condició final i repetitiva

Exercicis obligatoris

6.1 Observeu el següent programa en Pascal que calcula la mitjana entre una sèrie de valors que l'usuari introduirà per teclat fins que no entri el valor 0.

Responen les següents qüestions:

- Què fan les instruccions del tipus `variable := variable + expressió aritmètica`?
- Com es podria haver realitzat amb una estructura iterativa de condició inicial (`WHILE`)?
- Com es podria haver realitzat amb una estructura iterativa repetitiva (`FOR`)?
- En quin cas especial l'algorisme i el programa no funcionen? Com ho podríem solucionar?

Programa en Pascal

```
PROGRAM A6_1;

VAR x, mitjana, sumatori: REAL;
    terme: INTEGER;

BEGIN
  WRITELN('Càlcul de la mitjana d''una sèrie de dades');
  WRITELN('-----');

  (* Inicialitzem les variables *)
  sumatori:=0;
  terme:=0;

  (* Llegim dades i les processem *)
  (* fins que introdueixin un zero. *)
  REPEAT
    WRITE('Introdueix una dada (0 per acabar) : ');
    READLN(x);
    sumatori := sumatori + x;
    terme := terme + 1;
  UNTIL x=0;

  (* Com hem contat el zero com a terme, restem 1 al *)
```

```
(* nombre de termes a l'hora de calcular la mitjana.*)  
mitjana := sumatori / (terme-1);  
  
(* Donem els resultats *)  
WRITELN(La mitjana dels elements és ', mitjana:8:2);  
END.
```

6.2 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que presentin els elements de la següent sèrie i també la seva suma:

$$Sèrie = \sum_{i=1}^n \frac{1}{i^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$$

Dissenyeu l'algorisme en les tres estructures iteratives (condició inicial, final i repetitiva) i decidiu quina estructura és la més adient per aquest cas.

6.3 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que comprovi el valor d'una dada entera i després calculi el factorial de la dada.

- Aquesta dada ha d'ésser més gran que zero i més petita que vint.
- Si la dada no és correcta, l'ha de tornar a demanar tantes vegades com sigui necessari.

Recordem que el factorial d'un nombre enter és aquest nombre multiplicat per tots els seus antecessors:

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1$$

Vigileu amb els tipus de variables en Pascal, i el seu valor màxim de representació.

6.4 Amplieu l'exercici 4.2 per tal que permeti calcular el centre de gravetat (la mitja) dels punts de l'espai bidimensional que l'ordinador demanarà per teclat fins que l'usuari introdueixi el punt origen de coordenades (0,0). Per aconseguir això, s'haurà d'acumular el valor de les coordenades i al mateix temps comptar el nombre de valors introduïts per l'usuari.

Exercicis d'intensificació

6.5 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal en els que l'usuari introdueixi nombres enters fins que endevini el nombre aleatori entre 0 i 100 generat inicialment per l'ordinador. El programa avisarà de que el nombre introduït per l'usuari és més gran o més petit que el nombre generat aleatòriament.

La instrucció de Pascal que ens permet generar un nombre aleatori és la següent:

`RANDOM(n)` genera un nombre enter aleatori entre 0 i $n-1$, ambdós inclosos.

La instrucció de Pascal que ens permet que els nombres aleatoris siguin diferents a cada nova execució del programa és la següent:

`RANDOMIZE` inicialitza el generador de nombres aleatoris.

6.6 A la botiga dels germans Pastafullada és tradició presentar les llaunes de conserva apilades triangularment. En el primer pis n'hi ha una, en el segon dues, en el tercer tres i així successivament. Per exemple, sis llaunes es posen així:

```
  *
 * *
* * *
```

Els germans tenen grans problemes per fer les comandes de llaunes. Fixeu-vos que no tot nombre de llaunes pot apilar-se triangularment: per exemple, 8 llaunes no poden apilar-se. Escriviu un algorisme en pseudocodi i el corresponent programa en Pascal tal que, donat un nombre natural, comprova si aquest és adequat per muntar piles.

6.7 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que calculi i visualitzi els elements de la sèrie de Fibonacci. Aquesta sèrie es defineix pels seus paràmetres:

`Fibonacci(0) = 1`
`Fibonacci(1) = 1`
`Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)`

L'usuari haurà d'entrar, només, el nombre d'elements que vol visualitzar.

6.8 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que, introduïts dos nombres enters a i b per teclat, trobi el resultat de realitzar la seva multiplicació a partir de sumes. És a dir:

$a \times b = a + a + a + \dots + a$ (a sumat b vegades)

Tingueu en compte que tant a com b poden ser nombres negatius.

6.9 A un ordinador primitiu construït a inici dels 80 cal fer el càlcul de la funció $f(x) = \sin(x)$ per alguns valors de x . A tal efecte cal utilitzar el desenvolupament en sèrie de Taylor

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

aturant el càlcul serà quan el terme calculat, en valor absolut, sigui més petit o igual que un valor d'error ε donat per l'usuari o fixat pel programa ($\varepsilon \geq |x^n / n!|$). Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que efectui el càlcul. Tingueu en compte que les unitats de l'angle les haurem de passar a radians per poder realitzar les operacions.

6.10 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que calculi l'arrel quadrada d'un nombre real positiu a qualsevol mitjançant el desenvolupament en sèrie

$$x_1 = a$$
$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right)$$

i que aturi el càlcul quan dues aproximacions successives difereixin en menys d'un valor ε donat per l'usuari o fixat pel programa.

Pràctica 7. Variables estructurades homogènies

Objectius de la pràctica

- Introducció al concepte de variable estructurada homogènia
- Declaració de variables estructurades homogènies
- Referència directa i indirecta (indexada) dels elements
- Introducció i visualització de dades
- Operacions i ordenació de les dades

Exercicis obligatoris

7.1 Sense ajuda de l'ordinador, determineu el valor dels vectors a i b a cada pas de l'execució de les següents instruccions:

```
VAR a, b: ARRAY [1..3] OF INTEGER;  
    i: INTEGER;  
  
BEGIN  
    a[1] := 2;  
    a[2] := 4;  
    a[3] := a[1] + a[2];  
    b[2 + 1] := a[1] - 1;  
    b[a[1]] := 2*a[2] - 1;  
    b[a[3]/(a[1] + a[2])] := b[1] + 1;  
    FOR i := 1 TO 3 DO b[i] := b[i] + i;  
    b[i+1] := 9;  
END.
```

Descobriu l'error semàntic i els dos errors de concepte que amaga el programa.

7.2 Observeu el següent programa en Pascal, que primer llegeix un vector de deu elements numèrics i a continuació calcula el seu mòdul, segons la fórmula:

$$\text{sigui } \vec{v} = (\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n) \text{ llavors } |\vec{v}| = \sqrt{\vec{v}_1^2 + \vec{v}_2^2 + \dots + \vec{v}_n^2}$$

Responen les següents qüestions:

- Quina diferència hi ha entre una constant (CONST), un tipus de dades (TYPE) i una variable (VAR)?
- Quina és l'avantatge de declarar constants?
- Quina és l'avantatge de declarar nous tipus de dades?

Compileu el programa i realitzeu una execució pas per pas (tecla F7) amb traça de variables (tecles CTRL+F7), per tal de visualitzar en tot moment l'estat dels vectors.

Programa en Pascal

```
PROGRAM A7_2;

CONST
  MAX_ELEMENTS = 10;

TYPE
  t_vector = ARRAY [1..MAX_ELEMENTS] OF REAL;

VAR
  v : t_vector;
  i : INTEGER;
  suma : REAL;

BEGIN
  WRITELN('Càlcul del mòdul d'un vector');
  WRITELN('-----');
  WRITELN;

  (* Omplim el vector *)
  FOR i := 1 TO MAX_ELEMENTS DO
    BEGIN
      WRITE('Introdueix l'element ', i, ' : ');
      READLN(v[i]);
    END;

  (* Calculem la suma dels quadrats *)
  suma := 0;
  FOR i := 1 TO MAX_ELEMENTS DO
    BEGIN
      suma := suma + v[i]*v[i];
    END;

  (* Imprimim el mòdul *)
  WRITELN('El mòdul del vector és ', SQRT(suma));
END.
```

7.3 Dissenyau un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que emmagatzemi en un vector la nota dels alumnes d'un grup de pràctiques i posteriorment calculi i visualitzi el nombre de notes que apareixen dins dels següents intervals:

[0 , 5[Insuficient
[5 , 7[Aprovat
[7 , 9[Notable
[9 , 10]	Excel·lent

Tingueu en compte que, encara que els grups de pràctiques tenen un màxim de vint alumnes, cada grup pot tenir un nombre d'alumnes diferent. El programa ha de servir per qualsevol grup.

7.4 Amplieu l'exercici 6.4 per tal que permeti trobar el centre de gravetat d'un conjunt de punts de l'espai bidimensional emmagatzemats a un vector, segons la fórmula:

Sigui el vector de punts $\vec{v} = ((v_{1x}, v_{1y}), \dots, (v_{nx}, v_{ny}))$ llavors $\vec{m} = \sum_{i=1}^n \vec{v}_i = (\frac{\vec{v}_{1x} + \dots + \vec{v}_{nx}}{n}, \frac{\vec{v}_{1y} + \dots + \vec{v}_{ny}}{n})$

L'usuari indicarà a inici de programa quants punts vol introduir.

7.5 Dissenyau un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que sigui capaç de llegir els valors d'una matriu 3x3 i calculi el valor del determinant.

7.6 Dissenyau un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que ens permeti realitzar la suma de dues matrius de dimensions $N \times N$ (utilitzarem un màxim de 10×10). La suma ens ve donada per la següent fórmula:

$$R[i, j] = M_1[i, j] + M_2[i, j]$$

Un cop escrit i compilat el programa, realitzeu una execució pas per pas amb traça de les variables, per veure el funcionament dels bucles FOR niats.

Què ens cal canviar a l'anterior algorisme o programa per tal que en lloc de sumar matrius les multipliqui? La multiplicació ens ve donada per la següent fórmula:

$$R[i, j] = \sum_{k=1}^n M_1[i, k] \times M_2[k, j]$$

Exercicis d'intensificació

7.7 Dissenyau un algorisme en pseudocodi i el corresponent programa en Pascal que donat un vector de 50 elements enters, el descompongui en dos, un format pels valors senars i un altre pels valors parells. Cal remarcar que en aquests dos vectors resultants els valors es posaran correlativament (un després de l'altre).

7.8 Dissenyau un algorisme en pseudocodi i el corresponent programa en Pascal que donat un vector de 15 elements amb valors aleatoris introduïts per l'usuari, sigui capaç d'ordenar el vector i treure el resultat per pantalla.

7.9 Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que donats dos vectors ordenats realitzi la fusió d'ambdós per obtenir un tercer vector també ordenat. Cada vector conté cinc elements.

7.10 Suposem que disposem d'una taula amb distàncies quilomètriques:

	Barcelona	Girona	Lleida	Tarragona	Saragossa	Terol
Barcelona		100	156	98	296	409
Girona			256	198	396	509
Lleida				91	140	319
Tarragona					231	311
Saragossa						181
Terol						

Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que ens permeti calcular:

- | La distància entre dues poblacions, el nom de les quals serà introduït per l'usuari.
- | Les dues ciutats més allunyades entre si i la distància que les separa.
- | La distància total recorreguda en l'itinerari circular que passa per totes les ciutats en l'ordre següent: primera, segona, tercera, ..., última i primera de nou.

7.11 Un/a alumne d'enginyeria desitja realitzar una estadística de les hores d'estudi mensuals dedicades a cadascuna de les seves assignatures. Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que ens permeti calcular:

- | El total anual d'hores dedicades a cada assignatura.
- | El total mensual d'hores dedicades a estudiar.
- | El nom i total d'hores de l'assignatura més estudiada.

	Gener	Febrer	...	Decembre	Total
Assignatura 1					
...					
Assignatura 5					
Total					

7.12 Una matriu quasi-nul·la és una matriu amb un alt percentatge d'elements nuls. Una matriu quasi-nul·la amb k elements no nuls se sol representar

emmagatzemant els elements no nuls en una matriu de k files i tres columnes, contenint cada columna d'aquesta matriu la fila, la columna i el valor dels elements no nuls, respectivament. Per exemple:

$$\text{la matriu quasi-nul·la } \begin{pmatrix} 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \text{ es pot representar per } \begin{pmatrix} 1 & 3 & 3 \\ 3 & 2 & 6 \\ 3 & 3 & 1 \\ 5 & 5 & 1 \end{pmatrix}$$

Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal per tal de convertir una matriu quasi-nul·la en representació normal a la nova representació més compacta.

7.13 Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que donat un nombre introduït en una base qualsevol b_1 sigui capaç de convertir-lo a una altra base qualsevol b_2 . Com que ens veiem limitats a l'hora de treballar amb bases pel nombre de símbols de que disposem, emprarem els símbols 0, 1, ..., 9, A, B, ..., Z podent treballar així fins amb bases fins a la base 36. El procediment pot ser el següent:

1. Llegim les dues bases b_1 i b_2 .
2. Llegim el nombre en base b_1 convertint-lo a base 10 mitjançant el mètode de les potències successives.
3. Convertim el nombre a base b_2 mitjançant el mètode de les divisions successives.

Caldrà saber treballar amb strings que, de fet, són vectors de caràcters.

Pràctica 8. Variables estructurades heterogènies

Objectius de la pràctica

- Introducció al concepte de variable estructurada heterogènia
- Definició de variables estructurades heterogènies
- Accés als elements de les variables

Exercicis obligatoris

8.1 Dissenyeu un tipus de dades per representar cadascuna de les entitats següents:

- Un interval a la recta real.
- Un punt de l'espai.
- Una matriu en els reals de dimensions màximes 5×5.
- Un nombre complex.
- Una data.
- Una persona: nom, data de naixement i telèfon de contacte.
- Una agenda amb capacitat per guardar 100 persones.

Quan les dades que componen un nou tipus estructurat són totes del mateix tipus (per ex. el punt de l'espai), quan és millor utilitzar un tipus estructurat homogeni (ARRAY) i quan un tipus estructurat heterogeni (RECORD)?

8.2 Volem construir una aplicació que ens permeti emmagatzemar les dades personals dels alumnes amb un màxim de 20. De cada un d'ells guardarem el nom, el cognom i les notes obtingudes en cinc pràctiques. El programa calcularà i guardarà la nota final com la mitjana de les cinc anteriors.

L'aplicació demanarà a l'usuari el nombre d'alumnes amb que vol treballar i les dades de cada un d'ells. A continuació, calcularà la nota final de tots els alumnes i visualitzarà el nom i la nota final de tots els alumnes.

Programa en Pascal

```
PROGRAM A8_2;  
  
CONST MAX_AL = 20;  
      MAX_NOTES = 5;  
  
TYPE t_alumne = RECORD  
    nom          : STRING;  
    cognom      : STRING;  
    notes       : ARRAY [1..MAX_NOTES] OF REAL;
```

```
        nota_total : REAL;
        END;

        t_classe = RECORD
            alumnes : ARRAY [1..MAX_AL] OF t_alumne;
            num_al   : 1..MAX_AL;
            END;

VAR classe : t_classe;
    i, j : INTEGER;

BEGIN

    (* Demanar el nombre d'alumnes i les seves dades *)
    REPEAT
        WRITE('Quants alumnes tenim (màxim ' , MAX_AL , ') ? ');
        READLN(classe.num_al);
    UNTIL (classe.num_al >= 1) AND (classe.num_al <= MAX_AL);

    FOR i := 1 TO classe.num_al DO
        WITH classe.alumnes[i] DO
            BEGIN
                WRITELN;
                WRITE('Introdueixi el nom de l'alumne ', i, ' : ');
                READLN(nom);
                WRITE('Introdueixi el cognom de l'alumne ', i, ' : ');
                READLN(cognom);
                FOR j := 1 TO MAX_NOTES DO
                    REPEAT
                        WRITE('Introdueixi la nota ', j,
                            ' de l'alumne ', i, ' : ');
                        READLN(notes[j]);
                    UNTIL (notes[j] >= 0) AND (notes[j] <= 10);
                END;

            (* Calcular la nota final dels alumnes *)
            FOR i := 1 TO classe.num_al DO
                WITH classe.alumnes[i] DO
                    BEGIN
                        nota_total := 0;
                        FOR j := 1 TO MAX_NOTES DO
                            nota_total := nota_total + notes[j];
                        nota_total := nota_total / MAX_NOTES;
                    END;

            (* Escriure el resultat de la nota final de cada alumne *)
            WRITELN;
            WRITELN('Informe de notes finals');
            WRITELN('-----');
            FOR i := 1 TO classe.num_al DO
                WITH classe.alumnes[i] DO
                    WRITELN(nom , ' ', cognom, ' : ', nota_total:4:2);
            END.
END.
```

8.3 Amplieu l'exercici 7.6 per tal que permeti fer multiplicacions de matrius no quadrades, utilitzant per emmagatzemar la informació de cada matriu un registre que contindrà els següents camps:

- Una matriu de 10×10 elements.
- Una variable que indiqui el nombre de files reals de la matriu.
- Una variable que indiqui el nombre de columnes reals de la matriu.

```
TYPE t_matriu = RECORD
    valors : ARRAY [1..10,1..10] OF REAL;
    fil : 1..10;
    col : 1..10;
END;
```

Les matrius poden tenir un nombre de files diferent al de les columnes, amb una dimensió màxima de 10 files per 10 columnes. Tingueu present, però, que per poder multiplicar dues matrius, *A* i *B*, s'ha de complir la següent condició:

nombre de columnes d' *A* = nombre de files de *B*

i que la matriu resultant, *C*, compleix:

nombre de files de *C* = nombre de files d' *A*

nombre de columnes de *C* = nombre de columnes de *B*

8.4 Amplieu l'exercici 7.4 per tal que permeti trobar el centre de gravetat d'un conjunt de punts de l'espai bidimensional emmagatzemats en un vector.

Un punt estarà format per un nom que l'identifica i les seves coordenades:

t_punt	id	x	y
--------	----	---	---

Exercicis d'intensificació

8.5 Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que ens permeti sumar i multiplicar dos polinomis. Els polinomis els implementarem com un registre on guardarem:

- ┆ El grau del polinomi. Limitarem el grau màxim a 10.
- ┆ Els coeficients del polinomi.

t_polinomi	grau	coeficients										
		0	1	2	3	4	5	6	7	8	9	10

Cal recordar les normes de suma i multiplicació de polinomis. Es recomana fer a mà un exemple de cada per veure la mecànica de treball, per exemple:

Polinomi 1: $2x^3 + 3x^2 - 2$	3	-2	0	3	2							
-------------------------------	---	----	---	---	---	--	--	--	--	--	--	--

Polinomi 2: $-2x^2 + x - 3$

2	-3	1	-2								
---	----	---	----	--	--	--	--	--	--	--	--

Suma: $2x^3 + x^2 + x - 5$

3	-5	1	1	2							
---	----	---	---	---	--	--	--	--	--	--	--

Mult: $-4x^5 - 4x^4 - 3x^3 - 5x^2 - 2x + 6$

5	6	-2	-5	-3	-4	-4					
---	---	----	----	----	----	----	--	--	--	--	--

Pràctica 9. Disseny d'una aplicació. Subprogrames i mòduls.

Objectius de la pràctica

- Definició i conceptes de subprogrames.
- Utilització del pas de paràmetres.
- Disseny descendent d'una aplicació.

Exercicis obligatoris

9.1 Observeu el programa i responeu les següents qüestions:

- Quin és el valor de les variables *x* i *y* després de cridar al procediment *prova*?
- Quina diferència hi ha entre pas de paràmetres per valor i per referència? Quin seria el valor de les variables *x* i *y* després de cridar al procediment *prova* si canviéssim el seu encapçalament per `PROCEDURE prova(VAR n1:INTEGER; n2:INTEGER)?`
- Per què és convenient que l'entrada i sortida a un subprograma es realitzi mitjançant paràmetres d'entrada, paràmetres de sortida i el valor de retorn (si es tracta d'una funció), i no mitjançant `READ()` i `WRITE()`?
- Per què no és convenient fer servir variables globals dins una funció o procediment? Quin seria el valor de les variables *x* i *y* després de cridar al procediment *prova* si no existís dins seu la declaració local `VAR x:INTEGER?`

```
PROGRAM A9_1;

VAR x, y: INTEGER;

PROCEDURE prova(n1:INTEGER; VAR n2:INTEGER);
VAR x: INTEGER;
BEGIN
  x := n2;
  y := 5*x*x;
  n1 := y;
  n2 := 4;
END;

BEGIN
  x := 25;
  y := 2;
  WRITELN('Abans      x = ', x, '      y = ', y);
  prova(x, y);
```

```
WRITELN('Després      x = ', x, '      y = ', y);  
END.
```

9.2 Escriviu els subprogrames corresponents a les següents especificacions, considerant a cada cas si és més adient emprar funció o procediment.

- Un subprograma tal que, donats dos reals a i b , calcula el logaritme en base b d' a segons la següent fórmula: $\log_b a = (\ln a) / (\ln b)$.
- Un subprograma tal que, donat un enter, calcula el signe de l'enter i ens torna -1 , 0 o 1 segons si l'enter és negatiu, zero o positiu.
- Un subprograma tal que, donat un caràcter, ens calcula si el caràcter en qüestió és una lletra o no.
- Un subprograma tal que, donats tres reals, obté el mínim interval tancat que conté tots tres.
- Un subprograma tal que, donats dos vectors de 10 elements, retorna el vector resultat de sumar els dos anteriors.
- Un subprograma tal que, donat un polinomi, retorna la seva derivada. Feu servir el tipus polinomi definit a l'exercici 8.5.
- (Opcional) Un subprograma tal que, donat un enter, calcula recursivament el seu factorial (aquest nombre enter multiplicat per tots els seus antecessors):

$$n! = n \times (n-1) \times (n-2) \times \dots \times 1 = n \times (n-1)! \text{ si } n > 1 \text{ i } 1 \text{ si } n=1 \text{ ó } n=0$$

Quina diferència hi ha entre funció i procediment? Com es pot convertir una funció en procediment i viceversa?

9.3 Realitzeu el disseny modular de l'exercici 8.3 (multiplicació de matrius). La definició de les matrius és la mateixa. Només cal definir tres subprogrames:

- Un procediment que llegeixi per teclat els elements d'una estructura de tipus `t_matriu`, passada com a paràmetre:

```
PROCEDURE LlegirMatriu(VAR M : t_matriu);
```

- Un procediment per visualitzar per pantalla els elements d'una estructura de tipus `t_matriu`, passada com a paràmetre:

```
PROCEDURE VeureMatriu(VAR M : t_matriu);
```

- Una funció que calculi el producte de dues matrius passades com a paràmetre, i deixi el resultat en una tercera matriu també passada com a paràmetre, en el cas que això sigui possible.

```
FUNCTION MultMatrius(VAR M1, M2, R : t_matriu):BOOLEAN;
```

La funció per multiplicar matrius intentarà calcular $R = M_1 \times M_2$ i, si el càlcul és possible, ens retornarà en R el valor de la matriu producte i la funció retornarà

el valor booleà `TRUE`. Si el càlcul no és possible la funció retornarà el valor booleà `FALSE`.

9.4 Amplieu l'exercici 8.4 per tal que permeti introduir noms i coordenades de punts de l'espai bidimensional i posteriorment calcular el centre de gravetat d'aquests punts.

Un punt estarà format per un nom que l'identifica i les seves coordenades:

t_punt	id	x	y
--------	----	---	---

En iniciar el programa les opcions que ens apareixeran seran:

1. Introduir punts. 2. Visualitzar punts. 3. Modificar un punt cercant pel nom. 4. Calcular el centre de gravetat. 5. Finalitzar.

El disseny de l'algorisme ha de ser modular. Feu servir funcions i subrutines per implementar les diferents opcions del menú.

Exercicis d'intensificació

9.5 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal tal que ens permetin realitzar la resolució de sistemes d'equacions lineals 3×3 . Per resoldre aquests sistemes utilitzarem el mètode de Kramer. Aquest mètode utilitza el càlcul de determinants, que es pot realitzar mitjançant la utilització de la fórmula de càlcul de determinants 3×3 .

Aprofiteu l'algorisme de l'exercici 7.5 per resoldre l'exercici.

Quina és la millor manera de passar una matriu com a paràmetre d'un subprograma: per valor o per referència? Per què?

És interessant l'ampliació d'aquest exercici al càlcul de sistemes de $N \times N$.

9.6 Redissenyeu l'aplicació 5.10 utilitzant els criteris de programació modular. Per això implementareu els següents subprogrames en Pascal:

a) `FUNCTION RangDada (min, max : INTEGER) : INTEGER ;`

Funció que retorna una dada entrada per l'usuari pel teclat i que es troba dins dels límits marcats pels valors *min* i *max*. Feu servir l'algorisme de l'exercici 6.3.

b) `FUNCTION DataOK (dia, mes, any : INTEGER) : BOOLEAN ;`

Funció que donats tres valors enters que suposadament formen una data (dia-mes-any), retorna el valor `TRUE` si la data és correcta i `FALSE` en cas contrari. Feu servir l'algorisme de l'exercici 5.10.

c) `PROCEDURE Pausa;`

Procediment que genera una pausa en el programa. No se surt del procediment fins que no es premi la tecla `<ENTER>`.

El procediment `RangDada` serà utilitzat a l'hora de demanar el dia, mes i any; el procediment `DataOK` per comprovar si la data és correcta; i el procediment `Pausa` pot ser utilitzat a l'hora de parar la pantalla al final del programa.

9.7 Donat el següent programa en Pascal:

```
PROGRAM A9_7;

USES vector;

VAR a : ARRAY [1..5] OF INTEGER;
    i : INTEGER;

BEGIN
  FOR i := 1 TO 5 DO a[i] := 2*i;
  WRITELN( sumatori(a) );
  WRITELN( mitjana(a) );
END.
```

Creeu la unitat `vector` que implementi els següents procediments i funcions:

- Un procediment que inicialitza a zero els elements d'un vector de qualsevol mida:

```
PROCEDURE Inicialitzar(VAR taula : ARRAY OF INTEGER);
```

- Una funció que retorna el resultat de calcular la suma de tots els elements d'un vector de qualsevol mida:

```
FUNCTION Sumatori(VAR taula : ARRAY OF INTEGER):LONGINT;
```

- Una funció que retorna el resultat de calcular la mitjana de tots els elements d'un vector de qualsevol mida:

```
FUNCTION Mitjana(VAR taula : ARRAY OF INTEGER):REAL;
```

Pascal disposa de dues funcions que permeten consultar quins són els límits inferior i superior d'un vector:

- `LOW(nom_vector)` ens retorna l'índex inferior del vector.
- `HIGH(nom_vector)` ens retorna l'índex superior del vector.

9.8 Sigui n un nombre enter de quatre xifres diferents. Definim les funcions:

- | $gran(n)$ com el nombre més gran que es pot formar amb les xifres de n .
- | $petit(n)$ com el nombre més petit que es pot formar amb les xifres de n .
- | $dif(n) = gran(n) - petit(n)$.

Per exemple, si tenim $n = 1984$, aleshores $gran(n) = 9841$, $petit(n) = 1489$ i $dif(n) = 8352$.

Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que, donat un enter n , comprovi que aquest té quatre xifres diferents i calculi i escrigui els valors de la següent successió fins que trobi un terme tal que $dif(k) = k$:

$$dif(n)$$

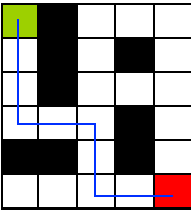
$$dif(dif(n))$$

$$dif(dif(dif(n)))$$

9.9 Imagineu un laberint format per una matriu quadrada de dimensions $n \times m$, amb k cel·les marcades que no es poden travessar. Dissenyeu un algorisme en pseudocodi i el corresponent programa en Pascal que ens permeti trobar el camí més curt per viatjar de la cantonada $(1,1)$ a la cantonada (n,m) sense passar per les cel·les marcades.

L'usuari introduirà els valors de n , de m i de k , així com les coordenades de les k cel·les marcades. La sortida serà les coordenades de les cel·les que formen el camí (tingueu en compte que pot no haver camí o haver més d'un).

Per exemple:

Laberint	Entrada	Sortida
	6 5 8 1 2 2 2 2 4 3 2 4 4 5 1 5 2 5 4	1 1 2 1 3 1 4 1 4 2 4 3 5 3 6 3 6 4 6 5

Pràctica 10. Fitxers.

Objectius de la pràctica

- Familiaritzar-se amb el tipus de dada estructurada Fitxer.
- Conèixer les instruccions bàsiques per treballar amb fitxers.

Exercicis obligatoris

10.1 Dissenyau un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que demani que introduïm noms d'alumne i la seva nota corresponent per teclat, i els guardi al fitxer 'alumnes.fxt'. Un cop haguem acabat d'introduir les dades, obri el fitxer en mode de lectura i imprimeixi per pantalla tots els noms i notes que conté.

Programa en PASCAL

```
PROGRAM A10_1;

TYPE t_alumne = RECORD
    nom : STRING[15];
    nota : REAL;
END;

VAR fitxer : FILE OF t_alumne;
    entrada : t_alumne;

BEGIN
    (* La variable fitxer treballarà amb el fitxer alumne.txt *)
    ASSIGN(fitxer, 'alumnes.fxt');

    (* Creem el fitxer d'alumnes *)
    REWRITE(fitxer);

    (* Demana noms i notes i els grava al fitxer *)
    (* mentre no s'introdueix ENTER en el camp del nom *)
    WRITE('Introdueix un nom o ENTER per acabar ... ');
    READLN(entrada.nom);
    WHILE entrada.nom <> '' DO
        BEGIN
            WRITE('Introdueix la nota corresponent a aquest alumne ');
            READLN(entrada.nota);
            WRITE(fitxer, entrada);          (* Escriu l'alumne al fitxer *)
            WRITE('Introdueix un nom o ENTER per acabar : ');
            READLN(entrada.nom);
        END;

    (* Tanca el fitxer *)
    CLOSE(fitxer);
```

```
(* Obre el fitxer en mode de només lectura *)
RESET(fitxer);

(* Mentre no s'arribi al final del fitxer *)
(* llegeix alumnes i els imprimeix *)
WRITELN;
WRITELN('Els alumnes que hi han al fitxer són:');
WRITELN('-----');
WHILE NOT EOF(fitxer) DO
  BEGIN
    READ(fitxer, entrada);      (* Llegeix un alumne del fitxer *)
    WRITELN(entrada.nom, ' ', entrada.nota:4:1);
  END;

(* Tanca el fitxer *)
CLOSE(fitxer);

READLN;
END.
```

10.2 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que permetin, al fitxer d'alumnes creat a l'exercici 10.1, cercar un alumne pel seu camp nom i visualitzar per pantalla la seva nota.

Programa en PASCAL

```
PROGRAM A10_2;

TYPE t_alumne = RECORD
  nom   : STRING[15];
  nota  : REAL;
END;

VAR fitxer      : FILE OF t_alumne;
    entrada     : t_alumne;
    nom_cercat  : STRING[15];
    trobat      : BOOLEAN;

BEGIN
  (* La variable fitxer treballarà amb el fitxer alumne.ftx *)
  ASSIGN(fitxer, 'alumnes.ftx');

  (* Demana el nom de l'alumne a buscar al fitxer *)
  WRITE('Introdueix el nom de l'alumne a cercar : ');
  READLN(nom_cercat);

  (* Mentre no s'arribi al final del fitxer, llegeix *)
  (* alumnes i compara el seu nom amb el nom cercat *)
  trobat := FALSE;
  RESET(fitxer);
  WHILE (NOT EOF(fitxer)) AND (NOT trobat) DO
    BEGIN
      READ(fitxer, entrada);
      trobat := (nom_cercat = entrada.nom);
    END;
  END;
```

```
(* Comprova si l'ha trobat *)
IF trobat THEN
  WRITELN(entrada.nom, ' ', entrada.nota:4:1)
ELSE
  WRITELN('Aquest alumne no es troba dins el fitxer');

(* Tanca el fitxer *)
CLOSE(fitxer);

READLN;
END.
```

10.3 Dissenyeu un algorisme en pseudocodi i realitzeu el corresponent programa en Pascal que permetin, al fitxer d'alumnes creat a l'exercici 10.1, cercar un alumne per la seva posició dins el fitxer i visualitzar per pantalla les seves dades.

Suposarem que el primer alumne del fitxer és el número 1.

Feu servir el procediment `SEEK(fitxer, posició)` i la funció `FILESIZE(fitxer)` de Turbo Pascal.

Programa en PASCAL

```
PROGRAM A10_3;

TYPE t_alumne = RECORD
  nom   : STRING[15];
  nota  : REAL;
END;

VAR fitxer      : FILE OF t_alumne;
    entrada     : t_alumne;
    num_alumne  : LONGINT;

BEGIN
  (* La variable fitxer treballarà amb el fitxer alumne.ftx *)
  ASSIGN(fitxer, 'alumnes.ftx');
  RESET(fitxer);

  (* Demana el número d'alumne a recuperar del fitxer. *)
  (* Suposem que el primer alumne del fitxer és el 1. *)
  REPEAT
    WRITE('Introdueix el número de l'alumne a llistar : ');
    READLN(num_alumne);
    IF (num_alumne < 1) OR (num_alumne > FILESIZE(fitxer)) THEN
      WRITELN('Número d'alumne incorrecte. ');
  UNTIL (num_alumne >= 1) AND (num_alumne <= FILESIZE(fitxer));

  (* Es posiciona al fitxer i llegeix l'alumne. *)
  SEEK(fitxer, num_alumne-1);
  READ(fitxer, entrada);
  (* També es pot fer, encara que de manera més ineficient, *)
  (* llegint tots els registres fins arribar al que volem *)
  (* (suposem que tinguem declarada la variable 'i') : *)
  (* FOR i := 1 TO num_alumne DO READ(fitxer, entrada); *)

```

```
(* Escriu l'alumne per pantalla *)  
WRITELN(entrada.nom, ' ', entrada.nota:4:1);  
  
(* Tanca el fitxer *)  
CLOSE(fitxer);  
  
READLN;  
END.
```

10.4 Amplieu l'exercici 9.4 per tal que permeti crear un fitxer que contindrà noms i coordenades de punts de l'espai bidimensional, i que permetrà posteriorment calcular el centre de gravetat d'aquests punts.

Un punt estarà format per un nom que l'identifica i les seves coordenades:

t_punt	id	x	y
--------	----	---	---

En iniciar el programa les opcions que ens apareixeran seran:

- | |
|---|
| <ol style="list-style-type: none">1. Afegir punts a un fitxer.2. Llistar un fitxer de punts.3. Modificar un punt d'un fitxer.4. Calcular el centre de gravetat.5. Finalitzar. |
|---|

- | La primera opció permetrà obrir un fitxer de punts (o crear un fitxer nou de punts, en cas que el fitxer que ens indiqui l'usuari no existeixi) i introduir nous punts.
- | La segona opció ens permetrà obrir un fitxer de punts existent i llistar tots els punts que conté.
- | La tercera opció ens permetrà obrir un fitxer de punts existent i realitzar una cerca d'un determinat punt pel camp de l'identificador. Un cop trobat, podrem modificar les seves coordenades i tornar a escriure-ho al fitxer.
- | La quarta opció ens permetrà obrir un fitxer de punts existent i realitzar el càlcul del centre de gravetat.
- | La cinquena opció és per sortir del programa.
- | I podeu afegir tot de noves opcions al menú que penseu convenients: càlcul de la recta de regressió dels punts, càlcul de les cotes dels punts, càlcul del cercle de radi mínim que conté els punts, ordenar els punts pel camp identificador, dibuixar els punts per pantalla, etc.

El disseny de l'algorisme ha de ser modular. Feu servir funcions i subrutines per implementar les diferents opcions del menú.

Petit annex III: Tractament dels fitxers en Pascal

Definició

Definim els següents tipus

```
Type t_dades = Record
    Nom      : string[20];
    Cognoms  : String[20];
    Telefon  : longint;
end;

t_fitxer = FILE of t_dades;

Var F : t_fitxer;
```

Assignació

```
ASSIGN(F, 'nom_fitxer');
```

F és una variable de tipus `t_fitxer` i `nom_fitxer` és el nom d'un arxiu físic de disc. `nom_fitxer` ha d'indicar la localització exacta de l'arxiu que conté les dades. Per exemple:

```
ASSIGN(F, 'c:\tp\dades.dat');
```

Obertura

Reset(F): obre el fitxer F .

F és una variable de tipus fitxer associat a un fitxer extern mitjançant `ASSIGN`. `Reset` obre el fitxer existent amb el nom assignat a F . Es produeix un error si no existeix un fitxer extern amb el nom donat. Si F ja està obert, primer es tanca i després es torna a obrir. L'apuntador de registre es col·loca a la primera posició.

Rewrite(F): crea i obre un fitxer nou.

F és una variable de tipus fitxer associat a un fitxer extern mitjançant `ASSIGN`. `Rewrite` crea un nou fitxer extern amb el nom assignat a F . Si ja existeix un fitxer extern amb el mateix nom, l'esborra i crea un de nou. Si F ja està obert, primer el tanca i després el torna a crear. `Rewrite` elimina tota la informació existent al fitxer F . L'apuntador de registre es col·loca a la primera posició.

Lectura/escriptura en fitxers

Write: escriu un registre al fitxer, a la posició que es troba l'apuntador.

Restricció: el fitxer ha d'estar obert per realitzar l'escriptura de dades. Sintaxi:

```
Write(F , P1 [, P2, ..., Pn ]);
```

Read: llegeix el contingut del registre de la posició en la que es troba l'apuntador. Una vegada llegit el registre, l'apuntador passa a la posició següent.

Restricció: el fitxer ha d'estar obert per realitzar la lectura de dades. Sintaxi:

```
Read ( F, V1 [, V2, ..., Vn ]);
```

Comprovació de l'estat de les operacions d'e/s i gestió d'errors

Commutador de comprovació d'entrada/sortida \$I: activa o desactiva la generació automàtica de codi que comprova el resultat d'una crida a un procediment d' E/S.

Sintaxi: {\$I+} ó {\$I-}

Valor por Defecte: {\$I+}

Observacions:

Si un procediment d'e/s retorna un valor diferent de zero quan el commutador \$I està activat {\$I+}, el programa termina, mostrant un missatge d'error en temps d'execució. Altrament, si el commutador \$I està desactivat {\$I-}, el programador gestiona els errors consultant la variable interna `IOResult` per comprovar si el procediment d'e/s ha estat correcte.

Si utilitzem la directiva {\$I-}, la variable interna `IOResult` retornarà un valor 0 si l'operació ha estat satisfactòria, i el codi d'error (diferent de zero) si no ho ha estat.

Variable interna `IOResult`: retorna l'estat de la darrera operació d' E/S.

Valor retornat: 0 si l'operació ha estat satisfactòria. Observacions: les comprovacions d'e/s han d'estar desactivades {\$I-} per detectar errors d'e/s mitjançant `IOResult`. Si es produeix un error d'e/s i la comprovació d'e/s està

desactivada, les següents operacions d'e/s s'ignoren fins que es faci una crida a la funció `IOResult`. Una crida a `IOResult` allibera el senyal d'error intern.

Valors típics d' `IOResult`:

0	No hi ha error
1	Número de funció no vàlida
2	Fitxer no trobat
3	Camí no trobat
4	Massa arxius oberts
5	Accés denegat
6	Operació amb fitxer no vàlida
12	Codi d'accés invàlid
15	Numero de dispositiu invàlid
16	No es pot esborrar el directori actual
17	Canvi de nom no vàlid
101	Error d'escriptura en disc
102	Fitxer no assignat
103	Fitxer no obert
104	Fitxer no obert per entrada
105	Fitxer no obert per escriptura
106	Format numèric Invàlid
150	Disc protegit contra escriptura
152	Unitat no disponible
154	Error CRC en dades
156	Error en disc
157	Medi de comunicació desconegut
158	Sector no trobat
159	Impressora sense paper
160	Error d'escriptura
161	Error de lectura
162	Error físic
200	Divisió per zero
201	Error de rang
205	Desbordament en coma flotant
207	Operacions de coma flotant invàlida

Exemple 1: demana el nom del fitxer de dades i comprova si existeix. Si no el troba, el torna a demanar de nou.

```
REPEAT
  WRITE('Nom de fitxer: ');
  READLN(NomFitx);
  ASSIGN (F, NomFitx);
  {$I-}           B Desactivat el control d'errors d'e/s
  RESET(F);
  Error := IOResult;
  {$I+}           B Activat el control d'errors d'e/s
  CASE Error OF
    2, 3: Writeln('Fitxer no trobat. Error en l''especificació de
l''arxiu');
```

```
150, 156, 157: Writeln('Error en disc o no hi ha disc a
l''unitat');
END;
UNTIL Error = 0;
.....
```

Exemple 2: comprova si l'arxiu a obrir existeix. Si existeix l'obre amb RESET, sinó el crea de nou amb REWRITE.

```
WRITE('Nom de fitxer: ');
READLN(NomFitx);
ASSIGN(F, NomFitx);
{$I-}          B Desactivat el control d'errors d'e/s
RESET(F);
IF IOResult <> 0 THEN REWRITE(F);
{$I+}          B Activat el control d'errors d'e/s
.....
```

Exemple 3: comprova si l'impressora està en línia (el fitxer d'impressora es diu LST i es crea cridant a la unitat Printer).

```
PROGRAM Un_programa;
USES printer;
.....
REPEAT
  {$I-}          B Desactivat el control d'errors d'e/s
  Writeln(lst,#27);
  (* LST és el fitxer de la impressora *)
  (* 27 és el Codi ASCII d'ESCAPE *)
  {$I+}          B Activat el control d'errors d'e/s
  control := IoResult;
  IF Control<>0 THEN
  BEGIN
    Write('Impressora no disponible:');
    Write('Reintentar(S/N)?');
    seguir := Ucase(Readkey);
  END;
  UNTIL (Control=0) OR (seguir='N');
..... .
```

Altres funcions associades

- Close(F)** : Tanca el fitxer *F*.
- Eof(F)** : Retorna True si l'apuntador està al final del fitxer *F*.
- FilePos(F)** : Retorna la posició de l'apuntador en el fitxer *F*.
- Filesize(F)** : Retorna la longitud del fitxer *F*.
- Seek(F; N)** : Col·loca l'apuntador a la posició *N* en el fitxer *F*.
- Truncate(F)** : Esborra el fitxer *F* a partir de la posició actual fins el final.

Erase(F) : Esborra el fitxer extern associat a *F*.

Rename(F, N) : Renombra amb nou nom *N* el fitxer extern associat a *F*.

Pràctica 11. Exercicis finals.

Objectius de la pràctica

- Disseny d'una aplicació.
- Integrar en un únic exercici els conceptes assolits fins ara.

Exercicis

11.1 Volem tenir una base de dades amb tots els alumnes de l'assignatura Fonaments d'Informàtica d'un trimestre, on constaran els noms, grup, notes, etc. Aquesta informació es guardarà a un fitxer i en tot moment la podrem consultar o modificar. Es tracta de realitzar un algorisme en pseudocodi i el corresponent programa en Pascal que ens permetin gestionar aquestes dades.

Les fitxes dels alumnes contindran la següent informació:

```
CONST MAX_NOTES = 5;

TYPE t_alumne = RECORD
    Especialitat: STRING[15];
    Grup: STRING[4];
    Nom: STRING[30];
    Notes: ARRAY [1..MAX_NOTES] OF REAL;
END;
```

El programa o algorisme mostrarà un menú amb les següents opcions:

- | |
|--|
| <ol style="list-style-type: none">1. Obrir un fitxer d'alumnes.2. Altes de nous alumnes.3. Baixes d'alumnes.4. Modificacions de les dades d'un alumne.5. Llistat dels alumnes del fitxer.6. Ordenació dels alumnes pel camp nom.0. Finalitzar. |
|--|

i el programa no acabarà fins que l'opció escollida per l'usuari sigui finalitzar.

Podrem tenir diferents fitxers d'alumnes. Això ens permetrà guardar en fitxers diferents les dades dels alumnes de cada trimestre i any. L'usuari escollirà amb la primera opció del menú el nom del fitxer amb el que vol treballar. Si ja s'estava treballant amb un fitxer, tancarem aquest i obrirem el nou fitxer escollit.

El disseny del programa i algorisme haurà de ser modular. Serà necessari definir els subprogrames que realitzin les següents accions:

- 1. Obrir un fitxer, donat el seu nom passat com a paràmetre. Si no existeix el fitxer, crear-lo.

- | Visualitzar per pantalla les dades d'un alumne passat com a paràmetre, incloent la seva mitjana de notes, que no hi és dins el fitxer (s'haurà de calcular).
- | Cercar un alumne dins el fitxer, donat el seu nom passat com a paràmetre, i retornar la seva posició. Si no ha estat trobat, retornar `-1`.
- | Donar d'alta nous alumnes, escrivint-los dins el fitxer. Si ha estat possible, retornar el valor booleà `TRUE`, si no ha estat possible (l'alumne ja existia o error d'entrada/sortida al fitxer) retornar el valor booleà `FALSE`.
- | Donar de baixa alumnes existents, esborrant-los del fitxer i reorganitzant el fitxer de nou. Si ha estat possible, retornar el valor booleà `TRUE`, si no ha estat possible (l'alumne no existia o error d'entrada/sortida al fitxer) retornar el valor booleà `FALSE`.
- | Modificar les dades dels alumnes existents, escrivint les noves dades al fitxer. Si ha estat possible, retornar el valor booleà `TRUE`, si no ha estat possible (l'alumne no existia o error d'entrada/sortida al fitxer) retornar el valor booleà `FALSE`.
- | Llistat dels alumnes del fitxer, utilitzant el subprograma anterior per visualitzar un alumne per pantalla.
- | Ordenació dels alumnes del fitxer pel camp nom.

11.2 Simulacions mitjançant un autòmat cel·lular: "el joc de la vida".

Un laboratori d'investigació distribueix una colònia de bacteris en un cultiu, que es pot considerar una superfície quadriculada de N files i M columnes. Cada casella d'aquesta superfície pot estar buida o contenir un bacteri. A partir d'una configuració inicial de bacteris en caselles, la colònia evoluciona generació a generació segons unes lleis genètiques determinades que depenen del nombre de veïns que tingui cada casella:

- Naixement: a cada casella buida que tingui exactament tres caselles veïnes ocupades per bacteris hi naixerà un nou bacteri.
- Mort per aïllament: tot bacteri que ocupi una casella amb cap o un veí mor per aïllament.
- Mort per asfíxia: tot bacteri que ocupi una casella amb més de tres veïns mor per asfíxia.
- Només sobreviuen els bacteris que tinguin dos o tres veïns.

La casella que ocupa la fila i i la columna j s'identifica mitjançant (i, j) , on $1 \leq i \leq N$, $1 \leq j \leq M$. Els veïns d'una casella (i, j) són les posicions adjacents $(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$ i $(i+1, j+1)$ que estan compreses dins la superfície i que estan ocupades per un bacteri.

Els bacteris que neixen o moren no afecten fins que s'ha completat un cicle evolutiu, entenent per aquest un cicle en el que s'ha decidit la supervivència o mort dels insectes (vius en començar el cicle) d'acord a les lleis genètiques mencionades.

Així, a una superfície 4x4, la colònia de l'esquerra de la següent figura evoluciona a les dues pròximes generacions tal i com es mostra:

```

. . * .      . * * .      . * . *      . . . .
. * * .      . * * *      . . . .      . . . .
. . * .      . . * *      . * . *      . . . .
. . * .      . . . .      . . . .      . . . .

```

Es demana simular l'evolució d'una colònia inicial de bacteris durant un cert nombre de transicions. Els passos a seguir seran els següents:

1. Demanar la configuració del tauler. Aquesta es pot introduir per teclat o residir en un fitxer. La configuració té el següent format:
 - Els dos primers elements són dos nombres enters que indiquen el nombre de files i columnes del tauler. Cal comprovar que aquest dos valors no excedeixin les dimensions màximes del tauler.
 - Els següents elements seran parelles de nombres que indicaran on hi ha una posició del tauler ocupada. Cal comprovar que els seus valors no surtin fora del tauler definit pels dos valors anteriors.
2. Simular generacions i visualitzar-les per pantalla fins que l'usuari premi una tecla especial, per exemple ESC, per finalitzar. Per exemple

<pre> EL JOC DE LA VIDA Premi F per llegir una configuració d'un fitxer. Premi T per introduir una configuració per teclat. T 5 5 2 2 3 2 3 3 3 4 </pre>	<pre> Generació 0 * * * * Premi: F per guardar en fitxer C per la següent generació ESC per finalitzar </pre>	<pre> Generació 1 * * * * Premi: F per guardar en fitxer C per la següent generació ESC per finalitzar </pre>
---	---	---

La colònia de bacteris serà una matriu d'elements del tipus cel·la. Les seves dimensions màximes venen definides per les constants MAXFIL i MAXCOL que definim a l'inici de l'algorisme.

El tipus cel·la serà un registre format per una variable que indicarà si està ocupada o buida i un altre variable que guardarà el nombre de veïns que té. D'aquesta manera, per calcular la següent generació podrem recórrer la matriu cel·la a cel·la i comptar el nombre de veïns que té. A continuació tornem a

recórrer la matriu marcant com a buides les cel·les dels elements que moren i marcant com a ocupades les cel·les dels nous elements que neixen.

És aconsellable (i obligatori) la utilització de subrutines en la realització de la pràctica. Per exemple:

- Una funció que retorni un nombre enter llegit pel teclat comprovant que es troba entre dos valors passats com a paràmetres de la funció. Es farà servir per llegir les dimensions de la colònia i les caselles ocupades.
- Un procediment que llegeixi la colònia d'un fitxer i un altre que la guardi.
- Un procediment que visualitzi la colònia per pantalla.
- Una funció que retorni el nombre de veïns d'una cel·la.
- Un procediment que calculi la següent generació.
- I totes les funcions i procediments que es creguin convenientes per afavorir la simplicitat i llegibilitat del programa i la reutilització del codi.