Liste d'exercices pour les TD

I Windows

1 boutons de la souris

La souris a deux (ou trois) boutons sur lesquels on peut appuyer ('cliquer'). En général, quand on ne précise pas, on utilise le bouton de gauche, mais quelque fois on utilise le bouton de droite. On a trois façons de cliquer :

- Le clic simple : On se place sur l'objet que l'on veut sélectionner, puis on appuie sur le bouton de la souris et on le relache immédiatement.
- Le clic double : On se place sur l'objet que l'on veut activer, puis on fait deux clics simples très rapprochés sans déplacer la souris entre les deux clics. Il faut donc tenir la souris fermement pour qu'elle ne bouge pas entre les deux clics.
- Le clic glissé : On se place sur l'objet que l'on veut déplacer, puis on enfonce le bouton de la souris, puis on déplace la souris (et l'objet) sur un autre emplacement, et là on relache le bouton de la souris. On peut souvent sélectionner une partie d'un texte en faisant un clic glissé entre le début et la fin du morceau que l'on veut sélectionner.

2 changement de taille, fermeture et destruction des fenêtres

Les fenêtres de `Windows` sont en général dans un cadre qui comporte dans sa partie supérieure un bandeau bleu, qui comporte à son extrémité droite, trois boutons accollés sur lesquels on peut éventuellement cliquer. Ces boutons ont l'effet suivant

_ réduit la fenêtre. Elle n'est plus visible, mais son contenu n'est pas perdu. On peut la faire réapparaître de diverses manières selon les cas :
  . En cliquant sur l'icône qui l'a remplacée.
  . En cliquant tout en bas de l'écran sur son nom, dans la liste des fenêtres qui existent.
  . En allant dans un menu `fenêtre`

□ agrandit la fenêtre à sa taille maximale, ou la ramène à sa taille avant agrandissement.

`X` Détruit la fenêtre en perdant son contenu.

On peut aussi changer la taille d'une fenêtre en faisant un clic glissé sur un bord ou un coin de cette fenêtre. On peut déplacer une fenêtre en faisant un clic glissé sur son bandeau bleu.

Exercice 3 : création du dossier (ou répertoire) `Mes documents\DUPONT`

A partir du bureau vide, c'est-à-dire, sans aucune fenêtre ouverte, faites un double clic sur l'icône `Mes documents`. Dans la fenêtre qui s'ouvre, allez dans le menu `Fichier` : `Nouveau` : `Dossier`. Vous venez de créer un nouveau dossier qui s'appelle `nouveau dossier`. Changer son nom. Appelez le `DUPONT` si vous vous appeler Dupont (ou `MARTIN` si vous vous appelez Martin). Détruisez le fenêtre que vous venez d'ouvrir.

Exercice 4 : création du fichier `DUPONT\beau.txt` avec le bloc-notes (`notepad`)

Lancer le programme `notepad` (cela veut dire bloc-notes en anglais). Il y a plusieurs façon de faire :

- Allez dans le menu `Démarrer` : `Exécuter` et tapez la commande `notepad`
- Allez dans le menu `Démarrer` : `Programmes` : `Accessoires` : `bloc-notes`

Dans la fenêtre du bloc-notes, tapez le texte "Papa, il fait beau et chaud". Allez dans le menu `Fichier` : `Enregistrer`. Une fenêtre va apparaître, qui vous propose de sauver votre texte dans un

fichier pour le moment `Sans titre`, dans le dossier `C:\WINNT` (ou un autre). Changez de dossier. Montez d'abord jusque dans le poste de travail, puis descendez dans `Mes documents`, puis dans `DUPONT`. Changez le nom, remplacez le par `beau`. Après cela pouvez cliquer sur `OK`. Vous venez de créer le fichier `DUPONT\beau.txt` qui contient le texte "Papa, il fait beau et chaud".

On aurait pu s'éviter d'avoir à changer le nom et le dossier, en allant au début dans le menu `Démarrer` : `Exécuter` et en tapant la commande

```
notepad C:\Documents and Settings\etudiant\Mes documents\DUPONT\beau.txt
```

Exercice 5 : montrer les extensions des fichiers

Quand on regarde le contenu du dossier `DUPONT`, le fichier `beau.txt` apparaît sous le nom `beau`, si les extensions reconnues par Windows (comme `.txt`) sont cachées. Pour rendre les choses plus claires, il vaut mieux ne pas les cacher. Allez dans le menu

`Outils` : `Options des dossiers` : `Affichage`

Enlevez la marque devant "cacher les extensions connues".

Exercice 6 : création des fichiers `DUPONT\chaud.txt` et `DUPONT\chaud.doc` avec `Word`

Refaites l'exercice 4, en utilisant `Word` au lieu de `notepad` et en créant un fichier qui s'appelle `chaud` et qui contient le texte "Papa, il fait chaud et beau". On sauvegardera deux fois le fichier, une première fois comme un document Word. Cela donnera un gros fichier appelé `chaud.doc`, et une deuxième fois en choisissant d'écrire dans le fichier le texte brut. Cela donnera le fichier `chaud.txt`.

Exercice 7 : création du fichier `DUPONT\paschaud.txt` avec l'éditeur de `pascal`

Refaites l'exercice précédent, en utilisant `pascal` au lieu de `notepad` et en créant un fichier qui s'appelle `paschaud` et qui contient le texte "Il fait pas chaud et pas beau".

Exercice 8 : création du dossier (ou répertoire) `DUPONT\papa`

Créez un nouveau dossier appelé `papa` dans `DUPONT`.

Exercice 9 : copie des 4 fichiers de `DUPONT` dans `DUPONT\papa`

Ouvrez une fenêtre montrant le dossier `DUPONT`. Sélectionnez les 4 fichiers qui s'y trouvent,

- en tapant `crtl A`,
- ou par le menu `Editer` : `Selectionner tout`,
- ou encore en sélectionnant d'abord un premier fichier par un clic simple, puis chacun des autres, un par un, par un clic simple alors que la touche ctrl reste enfoncée.

Allez dans le menu `Editer` : `Copier`, ou tapez ctrl X. Allez dans le dossier `DUPONT\papa`. Allez dans le menu `Editer` : `Coller`, ou tapez ctrl C.

Allez dans le menu `Affichage` : `Détail` et comparez les tailles des fichiers. Normalement, le `.doc` est beaucoup plus gros.

Exercice 10 : concaténation des 3 fichiers de texte dans `DUPONT\papa\contrep.txt`

En utilisant `Word` ou `pascal` créez un nouveau fichier inialement vide. Puis en allant dans le menu `Insertion` : `Fichier` vous pourrez insérer un par un les trois fichiers de texte du dossier `papa`.

Exercice 11 : copie des 5 fichiers de `DUPONT\papa` sur la disquette `A:`

Sélectionnez d'abord les cinq fichiers, puis

- faites un copier-coller,
- ou bien allez dans le menu `Fichier` : `Envoyer vers` : `Disquette`.
- Vous pouvez aussi atteindre ce dernier menu en cliquant sur le bouton de droite de la souris.

II premier programme

1 Affichage

Taper les programmes suivant et les exécuter.

```
program exemple;
begin
  writeln('Bonjour')
end.
```

```
program exemple;
begin
  writeln('Bonjour');
  writeln('re-bonjour'
end.
```

```
program exemple1;
begin
  writeln('Bonjour,');
  write('Il fait beau ');
  writeln('aujourd''hui')
end.
```

```
program exemple2;
begin
  write('Bonjour,');
  write('Il fait beau ');
  write('aujourd''hui')
end.
```

```
program entier;
var e,f:integer;
begin
  e:=100;
  writeln('Le carré de ',e,' est ',e*e);
  write('Donnez un nombre : ');
  readln(f);
  writeln('La somme de ',e,' et de ',f,' est ',e+f)
end.
```

2 variations

Ecrire des programmes qui affichent ce qui suit quand on les exécute. (Les nombres qui se trouvent en fin de ligne après ":" sont évidemment tapés par l'utilisateur du programme pendant son exécution. Ils sont donc lus par le programme par un `readln`.)

```
Donnez un nombre : 13
Le carré de 13 est 169
Le cube de 13 est 2197
```

```
Donnez deux nombres : 4 5
4+5=9
4*5=20
4-5=-1
```

```
note et coefficient : 12 2
note et coefficient : 9 3
La moyenne esr 10.20
```

```
Donnez les longueurs des trois cotés : 6 5 5
La surface du triangle est 12.00000
```

On rappelle que la surface d'un triangle dont les trois côtés ont pour longueur a, b et c est $\sqrt{(a+b+c)(a+b-c)(a-b+c)(-a+b+c)}/4$.

```
Quel est le rayon de la sphère ? 12
La surface de la sphère est 1809.557
Son volume est 7238.229
```

On rappelle que la surface d'une sphère de rayon r est $4\pi r^2$ et que son volume est $\frac{4}{3}\pi r^3$.

III `if then else`

1 syntaxe

En pascal une instruction conditionnelle a deux formes possibles (avec ou sans `else`) :

```
if <condition> then <instruction>
```

ou

```
if <condition> then <instruction1> else <instruction2>
```

La condition est une "expression booléenne" c'est-à-dire quelque chose qui vaut "vrai" ou "faux". Quand on exécute l'instruction conditionnelle, on évalue d'abord la condition (c'est-à-dire qu'on calcule sa valeur). Si elle vaut "vrai" on exécute l'instruction qui suit le `then` mais pas celle qui suit le `else`. Si elle vaut "faux" on n'exécute pas l'instruction qui suit le `then` mais celle qui suit le `else`. Par exemple `if 3=4 then i:=7 else j:=8` est équivalent à `j:=8` La condition peut être n'importe quelle expression booléenne :

- une constante booléenne, `true` ou `false`. Par exemple `if false then j:=3` ne fait rien.
- une variable booléenne. `b:=3=5; if b then x:=1 else x:=4` équivaut à `b:=false; x:=4`.
- une comparaison comme `i=4` ou `j<>k+3` ou `i<5` ou `j>=(i+3)*2`.
- la négation d'une expression booléenne, comme `not (i=5)` qui est équivalent à `i<>5`.
- la conjonction de deux expressions booléennes comme `(i>6) and (i<20)`.
- la disjonction de deux expressions booléennes comme `(j=2) or (j=5)`.

L'instruction qui suit le `then` (comme celle qui suit le `else`) peut être n'importe quelle instruction pascal, par exemple

- l'instruction vide : `if i=2 then else j:=3`
- une affectation : `if i<4 then j:=4`
- une instruction conditionnelle :

```
if i=3 then if j=4 then x:=3
                   else x:=4
       else x:=6
```

- une instruction composée, c'est-à-dire un bloc `begin end` :

```
if i>43 then begin j:=3; k:=4 end
        else j:=2
```

2 Vrai ou faux ?

Dans un programme commençant par

```
program vraioufaux;
var i,j:integer;
    s:string;
    a,b:boolean;
    c:char;
begin
  i:=4;
  s:='bonjour';
  j:=length(s);
  c:=s[4];
  a:=i<j;
  b:=not a;
```

que valent les expressions booléennes suivantes ?

```
i=j          i<>j+3      i<j+5       not a or not b
c='n'        odd(i)      odd(j)      odd(i) and (c<'x')
```

```
'petit'<'grand'          (s<'bonsoir') and (i<6)
('a'<=c) and (c<='z')    (c='o') or (c='O')
```

3 Pair ou impair ?

Ecrire un programme qui donne par exemple

```
Donnez un nombre : 56
56 est pair.
```

4 Equation du second degré

Ecrire un programme qui demande la saisie au clavier de trois nombres a, b et c, puis affiche les racines de l'équation $ax^2 + bx + c = 0$. Il y a plein de cas à distinguer. Par exemple si $a = 0$ l'équation se ramène à $bx + c = 0$ qui a pour racine $x = -c/b$ sauf si $b = 0$. Dans ce cas l'équation se ramène à $c = 0$ qui n'a pas de racine si c est non nul, mais dont tous les nombres réels sont solutions si c est nul.

Dans le cas où $a \neq 0$, on a bien une équation du second degré. On calcule alors le discriminant $\Delta = b^2 - 4ac$. Il n'y a pas de racine réelle si $\Delta < 0$. Il y a une racine double réelle égale à $-b/2a$ si $\Delta = 0$. Il y a deux racines réelles distinctes si $\Delta > 0$. Dans ce cas pour avoir la meilleure précision possible, il faut calculer la racine de plus grande valeur absolue à partir de $\sqrt{\Delta}$ et l'autre en utilisant le fait que c/a est le produit des deux racines. On calcule donc $x_1 = (-b - \sqrt{\Delta})/2a$ si $b > 0$ et $x_1 = (-b + \sqrt{\Delta})/2a$ sinon. Puis on calcule $x_2 = (c/a)/x_1$.

On essaiera le programme avec les équations $0x^2 + 0x + 0 = 0$, $0x^2 + 0x + 3 = 0$, $0x^2 + 2x + 6 = 0$, $2x^2 + 5x + 3 = 0$, $2x^2 + 5x + 4 = 0$ et $2x^2 + 6x + 18 = 0$.

5 Tri de trois nombres

Ecrire un programme qui lit trois nombres, puis les affiche dans l'ordre croissant. On doit voir, par exemple

```
Donnez trois nombres : 23 7  13
7 < 13 < 23
```

6 Un système expert en mobilier

Ecrire un programme qui sache distinguer entre une table, une chaise, un fauteuil, un tabouret et un tabouret de bar en posant successivement les questions :

Combien y a-t-il de pieds ?

Quelle est la hauteur des pieds ?

Y a-t-il des accoudoirs ?

La troisième question ne devra être posée que si elle est nécessaire. On admettra qu'un tabouret a toujours 3 pieds et que seuls le tabouret de bar et la table ont des pieds de plus de 80 cm.

IV L'affectation

Exercice 1 **:**

Qu'affichent les programmes suivants ? (On tape `1 2 3 4 5` pendant l'exécution de `ex1`)

```
program ex1;
var a,b,c,d:integer;
begin
  a:=3; b:=4;
  a:=b; a:=a; b:=a; b:=b;
  writeln('a:',a,' b:',b);
  a:=a+1; b:=a+1; a:=a+1; b:=a+1;
  writeln('a:',a,' b:',b);
  a:=1; b:=3; c:=5; d:=7;
  readln(b,a,d,b,a);
  b:=1; b:=c;
  writeln('a:',a,' b:',b,
         ' c:',c,' d:',d)
end.
```

```
program ex2;
var a,b,c,d,e,x,i,n:integer;
begin
  a:=1;b:=2;c:=3;d:=4;e:=5;
  n:=4;
  for i:=1 to n do
  begin
    x:=a;a:=b;b:=c;c:=d;d:=e;e:=x
  end;
  writeln('a:',a,' b:',b,' c:',c,
                 ' d:',d,' e:',e)
end.
```

Que donne le programme `ex2` si on remplace l'instruction `n:=4` par `n:=34` ? par `n:=33` ?

2 Echange de 2 valeurs

On suppose avoir un programme qui déclare 2 variables `x` et `y` de même type. Ecrire quelques lignes de Pascal pour échanger la valeur des 2 variables `x` et `y`.

Exercice 3 **:**

Que font les programmes suivants ? On tape `3` pendant l'exécution d'`ex3` et `6 7 8 9` pendant l'exécution d'`ex4`.

```
program ex3;
var a,b,c,i,n:integer;
begin
  a:=0;b:=1;c:=1;
  write('n ? ');
  readln(n);
  for i:=1 to n do
  begin
    a:=a+i;b:=b*i;c:=c+c;
  end;
  writeln('a:',a,' b:',b,' c:',c)
end.
```

```
program ex4;
var a,b,c,d,i:integer;
begin
  for i:=1 to 4 do
  begin
    d:=c;c:=b;b:=a;
    read(a)
  end;
  writeln('a:',a,' b:',b,' c:',c,' d:',d)
end.
```

4 Tri de 4 valeurs

Ecrire un programme qui demande au clavier quatre valeurs entières, stocke ces entiers dans les variables `a`, `b`, `c` et `d`, puis trie les variables de telle sorte que le plus petit entier soit la valeur de `a`, le deuxième plus petit la valeur de `b`, etc... Bref, un `writeln(a,'<',b,'<',c,'<',d)` devra afficher les valeurs dans l'ordre.

On pourra d'abord échanger `a` et `b` s'il ne sont pas dans le bon ordre, puis recommencer avec `c` et `d`, puis avec `a` et `c`, puis avec `b` et `d` et enfin avec `b` et `c`.

V Les boucles : `for`, `while` et `repeat`

1 Forme des boucles

Les boucles sont utilisées pour répéter un nombre déterminé ou indeterminé de fois une partie de programme, appelée "corps de la boucle" et formé d'une ou plusieurs instructions.

Trois formes de boucles existent en Pascal :

- `for` <variable>`:=`<expr> `to` <expr> `do` <instruction>
- `while` <condition> `do` <instruction>
- `repeat` <suite d'instructions séparées par des points-virgules> `until` <condition>

Dans un `repeat` ... `until`, le corps de la boucle peut être constitué de plusieurs instructions séparées par des points-virgules. Par contre dans un `for` ou un `while` le corps de la boucle est une seule instruction. Si on veut répéter plusieurs instructions, il faut les transformer en une seule en les mettant dans un bloc `begin` ... `end`. Par exemple

```
for i:=1 to 10 do writeln(i)                    est équivalent à
for i:=1 to 10 do begin write(i); writeln end    mais pas à
for i:=1 to 10 do write(i); writeln              où le writeln n'est pas répété.
```

Les deux premières versions sont aussi équivalentes à

```
i:=1;
while i<=10 do
begin
  writeln(i);
  i:=i+1
end
```

```
i:=1;
repeat
  writeln(i);
  i:=i+1
until i>10
```

Noter que dans le `while` et le `repeat` il faut mettre explicitement l'instruction `i:=i+1` alors que dans le `for` il ne faut pas la mettre.

En général on utilise une boucle `for` quand on sait exactement combien de fois on veut exécuter la boucle. Par exemple `for i:=1 to 10 do write('*')` écrit dix étoiles. Sinon on utilise une boucle `repeat` quand on veut que la boucle s'exécute au moins une fois, et une boucle `while` quand on veut qu'elle puisse éventuellement s'exécuter zéro fois.

2 One two three ...

Ecrire un programme qui demande un entier n à l'utilisateur et affiche trois fois les nombres de 1 à n. La première fois avec une boucle `for`, la deuxième fois avec une boucle `while` et la troisième fois avec une boucle `repeat until`. Pendant l'exécution on verra sur l'écran :

```
Jusqu'où voulez-vous compter ? 13
for    1 2 3 4 5 6 7 8 9 10 11 12 13
while  1 2 3 4 5 6 7 8 9 10 11 12 13
repeat 1 2 3 4 5 6 7 8 9 10 11 12 13
```

3 Compte à rebours

Ecrire un programme affichant `10 9 8 7 6 5 4 3 2 1 0`. Il devra y avoir environ une seconde d'attente avant l'affichage de chaque nombre. Il faut utiliser un `for` ... `downto`.

4 Premiers nombres pairs

Ecrire un programme affichant les n premiers nombres pairs, puis les nombres pairs inférieurs à n. La valeur de n doit être demandée à l'utilisateur.

5 Calculer $\sum_{i=1}^{1000} \frac{1}{i}$, $\prod_{i=1}^{100} \left(1 + \sum_{j=1}^{80} \frac{1}{i^2+j^2}\right)$ et $\prod_{i=1}^{1000} \left(\frac{3}{10} + \sum_{j=i}^{2i} \frac{1}{j}\right)^{10}$

6 Moyenne

Ecrire un programme qui calcule la moyenne, le plus petit et le grand de n nombres. Le nombre n ainsi que les n nombres doivent être saisis au clavier.

Etendre ce programme pour qu'après affichage de la moyenne, il recommence à redemander n puis n nombres jusqu'à ce que n soit égal à 0.

Calculer la moyenne des moyennes, la plus petite et la plus grande des moyennes.

7 Calcul de suites

Ecrire un programme qui affiche les premiers termes de la suite $u_n = 2u_{n-1} + 4$ avec $u_0 = 1$. On demandera d'abord à l'utilisateur combien de termes il veut.

Même question avec la suite $v_n = 3v_{n-1} - 3v_{n-2} + v_{n-3}$ avec $v_2 = 4$, $v_1 = 1$ et $v_0 = 0$

8 Suite de Fibonacci

$u_n = u_{n-1} + u_{n-2}$ avec $u_0 = 0$ et $u_1 = 1$. Ecrire un programme qui affiche tous les termes de cette suite qui sont inférieurs à un nombre demandé à l'utilisateur.

9 Suite de Syracuse

Ecrire un programme qui affiche les termes de la suite d'entiers naturels définie par $u_{n+1} = u_n/2$ si u_n est pair et $u_{n+1} = 3u_n + 1$ sinon. Le premier terme est demandé à l'utilisateur. La suite s'arrête quand on tombe sur 1.

```
D'où part-on ? 7
7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

10 Table de multiplication

Ecrire un programme affichant proprement la table de multiplications pour les entiers de 1 à `n` (`n` demandé à l'utilisateur et inférieur à 15)

11 Recherche dichotomique

Ecrire un programme qui devine un nombre choisi par l'utilisateur :

```
Choisissez un nombre entier entre 1 et 100.
Votre nombre est-il égal, inférieur ou supérieur à 50 ? (=,<,>) >
Votre nombre est-il égal, inférieur ou supérieur à 75 ? (=,<,>) <
Votre nombre est-il égal, inférieur ou supérieur à 62 ? (=,<,>) <
Votre nombre est-il égal, inférieur ou supérieur à 56 ? (=,<,>) >
Votre nombre est-il égal, inférieur ou supérieur à 59 ? (=,<,>) =
```

Après avoir posé les 3 premières questions l'ordinateur sait que le nombre cherché x vérifie $x \in [1, 100]$, $x > 50$, $x < 75$ et $x < 62$. Donc $x \in [51, 61]$. C'est pourquoi il propose le nombre 56 qui est le milieu de l'intervalle $[51, 61]$, ce qui permettra d'avoir un intervalle deux fois plus court à l'itération suivante. (C'est pourquoi on appelle ça une dichotomie)

Le programme a donc besoin de deux variables `a` et `b` qui représentent la plus petite et la plus grande valeur possible pour x. Il propose à chaque fois la moyenne de `a` et `b` et suivant que la réponse est `'<'` ou `'>'` il change la valeur de `b` ou de `a`.

Combien de questions posera-t-il au maximum ?

Ecrire un programme qui échange le rôle des deux joueurs.

12 Extraction de racine carrée par la méthode de Newton

Soit $a > 0$. La suite récurrente définie par $u_0 = (a + 1)/2$ et $u_{n+1} = (u_n + a/u_n)/2$ est décroissante et converge vers $\sqrt{a}$. Utiliser cette suite pour calculer la racine carrée d'un nombre demandé à l'utilisateur. On sait qu'on a atteint la plus grande précision possible, quand, à cause des erreurs d'arrondi, la suite cesse de décroître strictement.

V Procédure et Fonction

1 Procédures

Ecrire et exécuter les programmes suivants.

```
Program essai1;
var a,b:integer;

procedure afficher(x,y:integer);
begin
  writeln(x,' ',y)
end;
begin
  afficher(2,3);
  a:=1;
  b:=3;
  afficher(a,b)
end.
```

```
Program essai2;
var a,b:integer;

procedure afficher(x,y:integer);
begin
  writeln(x,' ',y)
end;
procedure incrementer(x:integer; var y:integer);
begin
  x:= x + 1;
  y:= y + 1
end;

begin
  a:=1;
  b:=5;
  incrementer(a,b);
  afficher(a,b)
end.
```

```
Program essai3;
var a,b:integer;

procedure afficher(x,y:integer);
begin
  writeln(x,' ',y)
end;
procedure incrementer(x:integer;var y:integer);
var a:integer;
begin
  writeln('Valeur d''incrementation : ');
  readln(a);
  x:= x + a;
  y:= y + a
end;

begin
  a:=1;
  b:=5;
  incrementer(a,b);
  afficher(a,b);
end.
```

2 tri

Décrire ce que fait chacune des procédures suivantes et compter les comparaisons d'entiers qu'elle effectue.

```
program tri2345;
const bavard=true;
procedure echange(var a,b:integer);
var c:integer;
begin
  c:=a;a:=b;b:=c
```

```
end;
procedure tri2(var a,b:integer);
begin
  if a>b then echange(a,b)
end;
procedure tri3(var a,b,c:integer);
begin
  tri2(a,b);
  if c<b then begin echange(b,c); tri2(a,b) end
end;
procedure insere3(var a,b,c,d:integer);
begin
  if bavard then write('(',a,' ',b,' ',d,')+',c);
  if b<c then tri2(c,d)
         else begin echange(b,c);tri2(a,b) end;
  if bavard then writeln('=(',a,' ',b,' ',c,' ',d,')')
end;
procedure tri4(var a,b,c,d:integer);
begin
  tri3(a,b,d); insere3(a,b,c,d)
end;
procedure tri5(var a,b,c,d,e:integer);
begin
  tri2(a,b); tri2(c,e);
  if b>e then begin echange(a,c);echange(b,e) end;
  insere3(a,b,d,e);
  insere3(a,b,c,d)
end;
var a,b,c,d,e:integer;
begin
  readln(a,b,c    ); tri3(a,b,c    ); writeln(a:6,b:5,c:6);
  readln(a,b,c,d  ); tri4(a,b,c,d  ); writeln(a:6,b:5,c:6,d:6);
  readln(a,b,c,d,e); tri5(a,b,c,d,e); writeln(a:6,b:5,c:6,d:6,e:6)
end.
```

Qu'écrit ce programme si on l'exécute en lui faisant lire

```
4 6 2
6 2 3 5
7 3 4 1 6
```

Qu'écrit-il si on remplace `bavard=true` par `bavard=false` ?

Qu'écrit-il si on remplace `procedure echange(var a,b:integer);` par `procedure echange(a,b:integer);` ?

Compléter le programme suivant :

```
program fonctions;
type real=extended;
     fonc=function(x:real):real;
const ga=2.0;
function min(a,b:real):real;
begin
  if a<b then min:=a
         else min:=b
end;
function max(a,b:real):real; ...
function min3(a,b,c:real):real; begin min3:=min(min(a,b),c) end;
function med3(a,b,c:real):real; ...   function max3(a,b,c:real):real; ...
function moyarit(a,b:real):real; (* =(a+b)/2 *) ...
function moygeom(a,b:real):real; (* =√ab *) ...
function moyharm(a,b:real):real; (* =2ab/(a+b) *) ...
function surface_triangle(a,b,c:real):real; ...
function volume_sphere(rayon:real):real; ...
function surface_sphere(rayon:real):real; ...
procedure aritgeom(a,b:real;var m,s:real);
var x:real;
begin
  x:=1/2;     s:=0;      m:=(a+b)/2;
  repeat
    s:=s+x*sqr(a-b);     //  u_0 = a     u_{n+1} = (u_n + v_n)/2     m:= lim_{n→∞} u_n
    x:=x*2;
    b:=sqrt(a*b);        //  v_0 = b     v_{n+1} = √(u_n v_n)        s:= Σ_{n=0}^{∞} 2^{n-1}(u_n - v_n)^2
    a:=m;
    m:=(a+b)/2
  until m=a
end;
function longueur_ellipse(a,b:real):real;
var s,m:real;
begin
  aritgeom(a,b,m,s);
  longueur_ellipse:=pi*(a*a+b*b-s)/m
end;
function surface_ellipse(a,b:real):real; (* πab *) ...
function int_trapeze(a,b:real;f:fonc;n:integer):real;
var s,h:real;
    i:integer;  // h( (f(a)+f(b))/2 + Σ_{i=1}^{n-1} f(a+ih) ) ≈ ∫_a^b f(x)dx
begin
  h:=(b-a)/n; ...
function int_rectangle(a,b:real;f:fonc;n:integer):real;(* h Σ_{i=1}^{n} f(a+(i-1/2)h) *)...
function int_simpson  (a,b:real;f:fonc;n:integer):real;
begin
  int_simpson:=(2*int_rectangle(a,b,f,n)+int_trapeze(a,b,f,n))/3
```

```
end;
function inv(x:real):real; (* =1/x *) ...
function f1(x:real):real; (* =1/(1+x^2) *) ...
function f2(x:real):real; (* =sqrt(sin^2 x + ga^2 cos^2 x) *) ...
function f3(x:real):real; (* =sqrt(1-(x/ga)^2) *) ...
function puissance(x:real;n:integer):real; (* =x^n *) ...
function factorielle(n:integer):longint; (* =n! *) ...
function expo(x:real;n:integer):real;              (* =sum_{i=0}^{n} x^i/i! *) ...
function pgcd(a,b:longint):longint; ...
function harm(n:integer):real;      (* =sum_{i=1}^{n} 1/i *) ...
var i:integer;   a,b,c,d:real;
begin
  for i:=2 to 4 do
  writeln('La sphère de rayon ',i,' a pour surface ',
  surface_sphere(i):0:5,' et pour volume ',volume_sphere(i):0:5);
  writeln('L''ellipse de demi-grand axe ',ga:0:5,' et de demi-petit axe 1 a',
    ' pour longueur ',longueur_ellipse(ga,1):0:20,' = ',
    4*int_rectangle(0,pi/2,@f2,1000):0:20);
  writeln('Sa surface est ',surface_ellipse(1,ga):0:10,
          ' = ',4*int_rectangle(0,ga,@f3,1000):0:10,
          ' = ',4*int_trapeze  (0,ga,@f3,1000):0:10,
          ' = ',4*int_simpson  (0,ga,@f3,1000):0:10);
  a:=37;b:=13;c:=30;
  writeln('Le triangle dont les cotés ont pour longueur ',a:0:5,', ',
          b:0:5,' et ',c:0:5,' a pour surface ',
          surface_triangle(a,b,c):0:5,' et pour périmètre ',a+b+c:0:5);
  writeln(min3(a,b,c):0:5,'<',med3(a,b,c):0:5,'<',max3(a,b,c):0:5);
  writeln(a:0:5,' et ',b:0:5,
          ' ont pour moyenne arithmétique ',moyarit(a,b):0:5,
             ',pour moyenne géométrique ',moygeom(a,b):0:5,
           ' et pour moyenne harmonique ',moyharm(a,b):0:5);
  writeln('ln 2=',ln(2):0:20);
  writeln('    =',int_rectangle(1,2,@inv,1000):0:20);
  writeln('    =',int_trapeze(1,2,@inv,1000):0:20);
  writeln('    =',int_simpson(1,2,@inv,1000):0:20);
  writeln('pi/4=',pi/4:0:20);
  writeln('    =',int_rectangle(0,1,@f1,1000):0:20);
  writeln('    =',int_trapeze(0,1,@f1,1000):0:20);
  writeln('    =',int_simpson(0,1,@f1,1000):0:20);
  aritgeom( 5, 3,a,b);aritgeom( 5,4,c,d);writeln(4*a*c/( 5* 5-b-d):0:20);
  aritgeom(13,12,a,b);aritgeom(13,5,c,d);writeln(4*a*c/(13*13-b-d):0:20);
  aritgeom(sqrt(2),1,a,b);              writeln(2*a*a/(1-b):0:20)
end.
```

VII Tableaux à une dimension et tris

Compléter le programme :

```
program tableaux;
(*$r+*)
const n=10;
type tabi=array[1..n] of integer;

procedure aff(a:tabi);
var i:integer;
begin
  for i:=1 to n do write(a[i]:7)
end;

procedure affln(a:tabi); begin aff(a); writeln end;

procedure hasard(var t:tabi;a,b:integer); ...
function somme(a:tabi):integer;                (* =∑_{i=1}^{n} a[i] *) ...
function max(a:tabi):integer; (* =max_{i=1}^{n} a[i] *) ...
function min(a:tabi):integer;                                  (* =min_{i=1}^{n} a[i] *) ...
procedure add(a,b:tabi;var c:tabi); (* c:=a+b *) ...
function present(a:tabi;x:integer):boolean; ...
function present_trie(a:tabi;x:integer):boolean; ...
procedure retourne(var a:tabi); ...
// remplace a[1], a[2], ... a[n] par a[n], a[n-1], ... a[1]

procedure tri_bulle(var a:tabi);
var i,j:integer;
begin
  for i:=1 to n-1 do
  for j:=1 to n-i do
  if a[j]>a[j+1] then // les échanger
end;

procedure tri_insertion(var a:tabi);
var i:integer;
begin
  for i:=2 to n do
  // inserer a[i] à sa place parmi a[1], a[2], ... a[i-1] qui sont déjà triés
end;

procedure tri_selection(var a:tabi);
var i:integer;
begin
  for i:=n downto 2 do
  // chercher le plus grand parmi a[1], a[2], ... a[i] puis l'échanger avec a[i]
end;
var a,b,c:tabi;
begin
```

```
  randomize;
  hasard(a,12,34); // met dans a des nombres tirés au hasard entre 12 et 34
  hasard(b,-3,3);
  add(a,b,c);
  affln(a);
  retourne(a);
  write('Les éléments extrèmes de '); aff(a);
  writeln(' sont ',min(a),' et ',max(a));
  affln(b);
  write(a[4]);
  if present(c,a[4]) then write(' est ')
                     else write(' n''est pas ');
  write(' présent dans ');
  affln(c);
  writeln(somme(a),'+',somme(b),'=',somme(c));
  tri_bulle(a);
  tri_insertion(b);
  tri_selection(c);
  affln(c); affln(b); affln(a);
  writeln(somme(a),'+',somme(b),'=',somme(c));
  // insérer un test de present_trie
end.
```

Dans un premier temps on pourra taper le programme tel quel, en mettant un corps vide, c'est-à-dire réduit à `begin end;` à toutes les procédures et fonctions qui doivent être complétées. De cette façon le programme pourra se compiler et s'exécuter. Ensuite, une par une, chacune des procédures devra être complétée et testée immédiatement.

La fonction `present_trie` fait la même chose que `present` mais elle suppose que le tableau est trié dans l'ordre croissant, donc elle n'est pas obligée de comparer le nombre cherché avec chacun des éléments du tableau, mais elle procède par dichotomie.

Dans la procédure `tri_bulle`, remplacer la boucle `for` externe, par une boucle `repeat until` qui s'arrête dès que le tableau est trié, c'est-à-dire, quand la boucle interne n'a fait aucun échange. Il faut donc les compter ou utiliser un booléen qui indique si on en a déjà fait.

On peut aussi remarquer que les plus grands éléments sont d'abord mis à leur place en haut du tableau, et qu'après cela ils ne bougent plus. On peut donc considérer qu'on a à trier seulement le début du tableau, sur une longueur `l`, qui est la position du dernier échange lors de l'itération précédente, c'est-à-dire la dernière valeur de `j` pour laquelle on a échangé `a[j]` et `a[j+1]`. La boucle interne, le `for`, se fait donc jusqu'à `l-1` au lieu de `n-i`. La boucle externe se fait tant que `l>1`.

Quand toutes les procédures et fonctions marcheront, et seulement à ce moment-là, faire une copie du programme sous un autre nom, et dans cette copie modifier toutes les procédures et fonctions de façon qu'elles ne s'appliquent plus à tout un tableau mais seulement au début de ce tableau. Par exemple, la fonction `somme` aura pour entête `function somme(a:tabi;p:integer):integer;` et `somme(b,4)` sera égal à `b[1]+b[2]+b[3]+b[4]`.

1 fichiers text

```
program f1;
var f:text;
    i,j:integer;
    s,t:string[3];
    c,d:char;
begin
  assign(f,'truc.t');
  rewrite(f);
  for i:=1 to 20 do
  writeln(f,i:3,i*i:3);
  close(f);
  reset(f);
  while not eof(f) do
  begin
    readln(f,i,j);
    writeln(i:10,j:10)
  end;
  close(f);
  readln
end.
```

```
program f2;
var f:text;
    nom:string;
    c,d:char;
    i:integer;
begin
  write('Quel caractère ? ');
  readln(c);
  repeat
    write('Dans quel fichier ? ');
    readln(nom);
    (*$i-*)
    assign(f,nom);
    reset(f) (*$i+*)
  until ioresult=0;
  i:=0;
  while not eof(f) do
  begin
    read(f,d);
    if c=d then i:=i+1
  end;
  write('Il y en a ',i);
  readln
end.
```

1–Faire tourner à la main le programme `p1` et en déduire ce qu'il affiche sur l'écran. On pourra le faire exécuter par l'ordinateur pour vérifier. On pourra aussi regarder le contenu du fichier `truc.t`.

2–Même exercice en remplaçant les deux lignes

```
  readln(f,i,j);
  writeln(i:10,j:10)
```

par

```
  read(f,i,c,s); writeln(i:10,c,c,s)
```

3–Que fait le programme `p2`

4–Le modifier pour qu'il compte toutes les lettres. On pourra utiliser

```
var t:array['a'..'z'] of integer;
  for c:='a' to 'z' do if t[c]<>0 then
```

5–Le modifier pour qu'il compte tous les caractères. On pourra utiliser

```
var t:array[char] of integer;
  for c:=#0 to #255 do if t[c]<>0 then
```

On rappelle que `char` est le même type que `#0..#255`, de même que `boolean` est le même type que `false..true` ou que `integer` est le même type que `-32768..32767`.

6–Ecrire un programme qui à partir d'un fichier de texte, crée un autre fichier de texte, qui est une copie du premier dans laquelle on a remplacé les groupes de blancs consécutifs par un seul blanc.

2 fichiers d'entiers

Exécuter à la main le programme :

```
var f:file of integer;
    i,j,k:integer;
begin
  assign(f,'truc.int');rewrite(f);
  for i:=1 to 20 do
  begin j:=i*(20-i); write(f,j) end;
  for i:=1 to 10 do
  begin
    seek(f,i*5 mod 7); read(f,j); read(f,k);
    writeln(j:4,'+',k:4,'=',j+k:4);
    j:=j+k; write(f,j)
  end;
  for i:=19 downto 0 do
  begin seek(f,i); read(f,j); write(j:4) end;
  readln
end.
```

IX records

1 nombres complexes

Un `record` est un objet qui regroupe plusieurs données en une seule. Il faut déclarer le type de chacune de ses composantes (on les appelle des champs). Par exemple, un nombre complexe est formé de 2 nombres réels, à savoir sa partie réelle et sa partie imaginaire, et donc on déclare le type `complexe` comme un `record` contenant deux champs `re` et `im` de type `real` :

```
type complexe = record re,im:real end;
```

Voici un opérateur qui additionne 2 complexes :

```
operator+(a,b:complexe)c:complexe; // c:=a+b
begin
  c.re:=a.re+b.re;
  c.im:=a.im+b.im
end;
```

On remarquera l'utilisation du "." pour accéder aux champs de `a`, `b` et `c`.

Compléter les fonctions, opérateurs et procédure suivants.

```
function module(a:complexe):real;  // |a|
function c_lire:complexe;          // read(a)
procedure c_ecrire(a:complexe);    // write(a)
operator-(a,b:complexe)c:complexe; // c:=a-b
operator*(a,b:complexe)c:complexe; // c:=a*b
operator/(a,b:complexe)c:complexe; // c:=a/b
operator*(a:complexe;b:real)c:complexe; // c:=a*b
operator/(a:complexe;b:real)c:complexe; // c:=a/b
function c_sqrt(a:complexe):complexe; // √a
```

Si $a = X + iY$ et $c = x + iy$ les équations $c^2 = a$ et $|c|^2 = |a|$ se réécrivent

$$\begin{aligned} x^2 - y^2 &= X \\ 2xy &= Y \\ x^2 + y^2 &= |a| = \sqrt{X^2 + Y^2} \end{aligned} \qquad \text{et donc} \qquad \begin{aligned} x &= \pm\sqrt{\frac{|a| + X}{2}} \\ y &= \pm\sqrt{\frac{|a| - X}{2}} \end{aligned}$$

En fait pour avoir la meilleure précision possible, il faut calculer $x = \sqrt{(|a| + X)/2}$ puis $y = Y/2x$ si $X > 0$ et $y = \sqrt{(|a| - X)/2}$ puis $x = Y/2y$ sinon. Ces deux calculs se terminant par $0/0$ si $a = 0$, il faut traiter ce cas à part.

Le programme principal devra résoudre l'équation $Ax^2 + Bx + C = 0$, en lisant d'abord les 3 coefficients complexes A, B et C, puis il affichera les 2 racines. Les calculs sont plus simples si on utilise $B2 = B/-2$ au lieu de B.

2 football

On définit les types suivants :

```
type equipe=record
              nom:string;
              gagne,perdu,nul:integer;
              marque,encaisse:integer;
            end;
     championnat=array[1..20] of equipe;
```

Ecrire une fonction `point` qui prend en argument une `equipe` et qui retourne son nombre de points (sachant qu'un match nul compte pour 1 point, un gagné pour 3 points et un perdu pour 0 point).

Ecrire une fonction `premier` qui prend en argument un `championnat` et qui retourne le nom de l'équipe en tête du championnat. Si 2 équipes ont le même nombre de points, c'est celle qui a la plus grande différence entre buts marqués et buts encaissés qui l'emporte.

Ecrire une fonction `init` qui demande le nom de 20 équipes et qui initialise un `championnat` avec ces noms. L'en-tête sera

```
procedure init (var c:championnat);
```

3 étudiants

Définir un type `etudiant` pour représenter la donnée d'un étudiant avec son nom, som prénom et un tableau de 12 notes.

Ecrire une procédure d'affichage des informations de type `étudiant`.

4 polynômes

Compléter le programme :

```
const degmax=30; (*$r+*)
type real=extended;
     poly=record
            coeff:array[0..degmax] of real;
            degre:integer
          end;
procedure majdegre(var a:poly);
begin while (a.degre>=0) and (a.coeff[a.degre]=0) do dec(a.degre) end;
procedure p_ecr(a:poly); // write(a)
var i:integer; x:real;
begin
  for i:=a.degre downto 0 do if a.coeff[i]<>0 then
  begin
    if ... write('+')
           write('-');
    x:=abs(a.coeff[i]);
    if (i=0) or (x<>1) then if x=int(x) then write(x:0:0)
                                        else write(x:0:5);
    case i of
      0:;
      1:write('x');
      2:write('x²');
      else ...
    end
  end;
  if ... then write(0)
end;
procedure p_add(a,b:poly;var c:poly); // c:=a+b
procedure p_sub(a,b:poly;var c:poly); // c:=a-b
procedure p_mul(a,b:poly;var c:poly); // c:=a*b
procedure p_div(a,b:poly;var q,r:poly);
```

```
var i,j:integer;
begin
  if b.degre=-1 then begin writeln('raté');halt end;
  q.degre:=a.degre-b-degre;
  if q.degre<0 then q.degre:=-1;
  for i:=q.degre downto 0 do
  begin
    q.coeff[i]=a.coeff[i+b.degre]/b.coeff[b.degre];
    for j:=0 to b.degre do ... // a:=a-(q.coeff[i] X^i)b
    a.degre:=a.degre-1
  end;
  majdegre(a);
  r:=a
end;
procedure p_gcd(a,b:poly;var c:poly);
var q:poly;
begin
  while b.degre<>-1 do ...
  c:=a
end;
function p_val(a:poly;x:real):real; // p(x)
  ...
  for i:= ... do v:=v*x+a.coeff[i];
  ...
procedure derive(p:poly;var d:poly); // d=p'
function newton(p:poly;x:real);
var dx:real; d:poly;
begin
  ... // d=p'
  repeat
    ... // dx:=p(x)/p'(x)
    x:=x-dx
  until abs(dx)<1e-10;
  newton:=x
end;
```

La fonction `p_val` doit utiliser la méthode de Horner : Par exemple $5x^3 + 2x^2 + 4x + 7 = (((0x + 5)x + 2)x + 4)x + 7$. Ecrire un programme principal qui calcule et affiche le polynôme $A = \prod_{i=1}^{6}(X^2 + iX + 1) - \prod_{i=1}^{6}(X^2 + iX - 1)$ sa dérivée A' et $\text{pgcd}(X^{30} + X^{15} + 1, X^{24} + X^{12} + 1)$ puis les nombres $A(-1)$, $A(2)$ et les nombres `newton(A,i)` pour `i`=3, 2, 1, 0, -1, ..., -7, -8.

5 polynômes sans record

Refaire le programme précédent en changeant la définition des polynômes :

```
program polynomes2; (*$r+*)
const degmax=30;
type real=extended;
     poly=array[0..degmax] of real;
function degre(a:poly):integer;
procedure p_ecr(a:poly); // write(a)
operator-(a,b:poly)c:poly;
operator*(a,b:poly)c:poly;
procedure p_div(a,b:poly;var q,r:poly);
function p_gcd(a,b:poly):poly;
```