

```
program DiffPosNeg;
```

```
var element_courant, nb_elements_positifs, nb_elements_negatifs : integer;
```

```
begin
```

```
  (* Initialisation *)
```

```
  nb_elements_positifs:=0;
```

```
  nb_elements_negatifs:=0;
```

```
  writeln ( 'Saisissez une suite d'entiers termines par -1' );
```

```
  (* Lecture du premier entier *)
```

```
  read(element_courant);
```

```
  (* Tant que l'élément courant n'est pas la marque *)
```

```
  while element_courant <> 0 do
```

```
  begin
```

```
    if element_courant < 0 then
```

```
      nb_elements_negatifs := nb_elements_negatifs + 1
```

```
    else
```

```
      nb_elements_positifs := nb_elements_positifs + 1;
```

```
    (* Lecture de l'entier suivant *)
```

```
    read(element_courant)
```

```
  end;
```

```
  (* Affichage du résultat *)
```

```
  write ( 'La difference positifs - negatifs est:' );
```

```
  writeln (nb_elements_positifs-nb_elements_negatifs)
```

```
end.
```

```
program Factoriel;  
var i, n, fac : integer;  
    (* compteur de boucle, entier donné, factoriel de n *)  
begin  
    (* Lecture de l'entier positif n *)  
    writeln('Saisissez un entier');  
    readln (n);  
  
    fac:=1;  
    (* Initialisation du factoriel à 1 : 1! *)  
    for i:=2 to n do  
        fac:=fac*i;  
  
    (* Affichage du résultat *)  
    writeln(n, '!=', fac)  
end.
```

```
program Fibonacci;
const nbNombres = 20;

var i,f1,f2,f3:integer;
    (* Compteur de boucle et les termes n-2, n-1 et n de la suite *)

begin
    (* Initialisation des termes 0 et 1 de la suite *)
    f1:=1;
    f2:=1;

    (* Affichage des termes 0 et 1 *)
    writeln('f( , 0:2, ')=' , f1);
    writeln('f( , 1:2, ')=' , f2);

    for i := 2 to nbNombres do
    begin
        (* Calcul du terme n *)
        f3:=f1+f2;

        (* Affichage du terme calculé *)
        writeln('f( , i:2, ')=' , f3);

        (* Décalage des termes *)
        f1:=f2;
        f2:=f3
    end
end.
```

```
program Motif1_1;
var h,l,i,j:integer;
  (* Paramètres du motif et compteurs de boucle *)
begin
  (* Lecture des paramètres *)
  writeln ('Tapez les parametres H et L');
  readln(h,l);

  (* Parcours des lignes *)
  for i:=1 to h do
  begin
    (* Parcours des colonnes *)
    for j:=1 to l do
      write('* ');

      (* Insertion d'un saut de ligne *)
      writeln;
    end
  end.
end.
```

```
program Motif1_2;
var h,l,i,j:integer;
  (* Paramètres du motif et compteurs de boucle *)

begin
  (* Lecture des paramètres *)
  writeln ('Tapez les parametres H et L');
  readln(h,l);

  (* Parcours des lignes *)
  for i:=1 to h do
  begin
    (* Selon le numéro de la ligne, traitement différent *)
    if odd(i) = 1 then
      for j:=1 to l do
        write('* ')
      else
        for j:=1 to l-1 do
          write('* ');
        (* Insertion d'un saut de ligne *)
        writeln;
      end
    end
  end.
```

```
program Motif2_1;
var t,i,j:integer;
  (* Paramètre du motif et compteurs de boucle *)
begin
  (* Lecture du paramètre *)
  writeln ('Tapez le parametre T');
  readln(t);

  (* Parcours des lignes *)
  for i:= t downto 1 do
  begin
    (* Parcours des colonnes (dont le nombre dépend de la ligne) *)
    for j:=1 to i do
      write('* ');

      (* Insertion d'un saut de ligne *)
      writeln;
    end
  end.
end.
```

```
program Motif2_2;
var t,i,j:integer;
  (* Paramètre du motif et compteurs de boucle *)
begin
  (* Lecture du paramètre *)
  writeln ('Tapez le parametre T');
  readln(t);

  (* Parcours des lignes *)
  for i:= 1 to t do
  begin
    (* Parcours des colonnes blanches *)
    for j := t-i downto 1 do
      write(' ');

    (* Parcours des colonnes avec * *)
    for j := 1 to i do
      write('* ');

    (* Insertion d'un saut de ligne *)
    writeln;
  end
end.
```

program Moyenne;

var somme_elements, nb_elements, element_courant : *integer*;

begin

(initialisation : aucun élément n'est lu donc la somme et le
* le nombre lu sont nuls *)*

somme_elements := 0; nb_elements := 0;

writeln ('Saisissez une suite d'entiers terminés par -1');

read(element_courant); *(* lecture du premier entier *)*

while element_courant <> -1 **do**

begin

(Ajout de l'élément courant à la somme *)*

somme_elements := somme_elements + element_courant;

(Incrément du nombre d'éléments lus *)*

nb_elements := nb_elements + 1;

(lecture de l'entier suivant *)*

read(element_courant)

end;

(Affichage du résultat *)*

if nb_elements <> 0 **then**

writeln ('la moyenne est ', (somme_elements / nb_elements):2:3)

else

writeln ('Aucun element')

end.

```
program PartieEntiere;  
var i,n:integer;  
  
begin  
  writeln ('Saisissez un entier positif');  
  readln(n);  
  
  (* Initialisation de i *)  
  i := 1;  
  
  (* Tant que la condition d'arret n'est pas atteinte, on incrémente i *)  
  while (i*i) <= n do  
    i := i+1;  
  
  (* Affichage du résultat, on sort de la boucle pour i qui est juste  
  * supérieur à sqrt(n) *)  
  writeln ('sqrt(', n:2, ')=' , i-1)  
end.
```

program SuiteCroissante;

var crois : *boolean*;

(crois est vrai si la suite est croissante *)*

element_courant, element_precedent : *integer*;

begin

(Initialisation de crois et element_precedent : puisque aucune valeur*

** n'est encore lue, la suite est supposée croissante, pour que le*

** premier élément ne rende pas la suite non croissante, il faut un*

** élément précédent qui soit toujours plus petit (c'est le cas de -1) *)*

crois := **true**;

element_precedent := -1;

writeln ('Saisissez une suite d'entiers terminés par -1');

(Lecture du premier entier *)*

read(element_courant);

(Tant que l'élément courant n'est pas la marque et que on a pas*

** trouvé un élément qui rend la suite non croissante *)*

while (element_courant <> -1) **and** crois **do**

begin

if element_courant < element_precedent **then**

(La suite n'est pas croissante : on passe le booléen crois*

** à faux. Il prouvera la sortie de la boucle *)*

crois := **false**

else

(Lecture de l'entier suivant uniquement si la suite est*

** toujours croissante, dans le cas contraire, c'est inutile*

** puisque la boucle se termine sur le booléen crois à faux *)*

begin

element_precedent := element_courant;

read(element_courant)

end

end;

(Affichage du résultat *)*

if crois **then**

writeln ('La suite est croissante')

else

writeln ('La suite n'est pas croissante')

end.

```
program SuiteGeometrique;  
  
const nbTermes = 15;  
  
var i,a:integer;  
    (* Compteur de boucle, élément courant de la suite *)  
  
begin  
    (* Lecture du premier terme de la suite géométrique *)  
    writeln ( 'Saisissez le premier terme de la suite geometrique' );  
    readln(a);  
  
    for i:=1 to nbTermes do  
        begin  
            a:=a*2;  
            writeln(i:2, ':', a)  
        end  
    end  
end.
```