Introduction à l'algorithmique

Informatique et Création Numérique

Introduction à l'algorithmique

Informatique et Création Numérique

Informatique

Science du traitement rationnel de l'information
par des machines automatiques

Philippe Dreyfus, 1962

Introduction à l'algorithmique

Informatique et Création Numérique

Algorithme et organigramme

- Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème.
- Un organigramme est une représentation d'une programmation sous forme d'un schéma.
- Un programme est une implémentation d'un algorithme ou d'un organigramme.

Introduction à l'algorithmique

Informatique et Création Numérique

Exemples d'algorithmes

Briques de LEGO → suite de dessins → Camion de pompiers

Meuble en kit → notice de montage → Cuisine équipée

Farine, œufs, ... → recette → gâteau

Introduction à l'algorithmique

Informatique et Création Numérique

Niklaus Wirth*

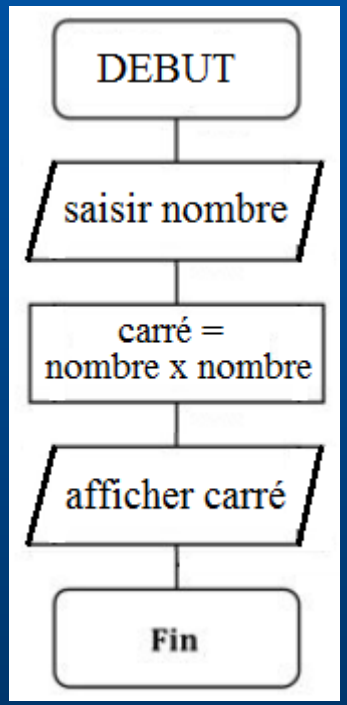
Langage Pascal

* professeur d'informatique Suisse, prix Turing 1984

Introduction à l'algorithmique

Informatique et Création Numérique

Séquence linéaire



Algorithme EleveAuCarre

Début
  Saisie
  Traitement
  Affichage
Fin

Introduction à l'algorithmique

Informatique et Création Numérique

Un premier algorithme

```
Algorithme EleveAuCarre
{ Cet algorithme calcule le carré du nombre que lui fournit l'utilisateur }
Variables
        unNombre, sonCarre : entier ;          { déclarations des variables }
Début
        { préparation du traitement }
        unNombre := saisir("Quel nombre voulez-vous élever au carré ?") ;
        { traitement : calcul du carré }
        sonCarre := unNombre × unNombre ;
        { présentation du résultat }
        afficher("Le carré est : ", sonCarre) ;
Fin
```

Introduction à l'algorithmique

Informatique et Création Numérique

Un premier algorithme

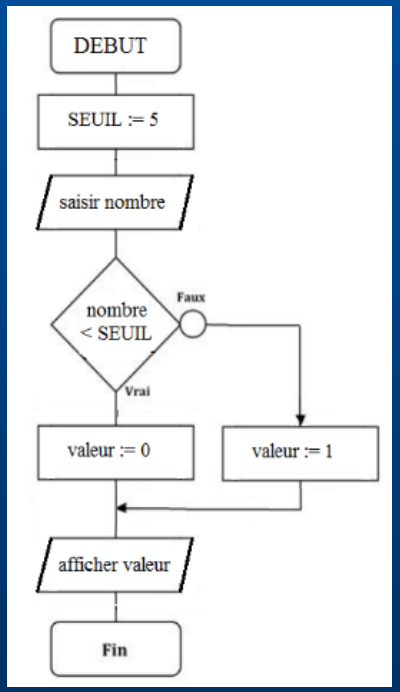
```
program EleveAuCarre;
{ Cet algorithme calcule le carré du nombre que lui fournit l'utilisateur }

var
  unNombre, sonCarre : integer;          { déclarations des variables }

begin
     { préparation du traitement }
     write('Quel nombre voulez-vous élever au carré ? ');
     readln(unNombre);   { Lecture }

     { traitement : calcul du carré }
     sonCarre := unNombre * unNombre;

     { présentation du résultat }
     writeln('Le carré est : ', sonCarre);
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Un deuxième algorithme

```
Algorithme ToutOuRien
{ affiche 0 si une valeur saisie est inférieure à un seuil donné sinon affiche 1 }
constante (SEUIL : entier) := 5 ;          { seuil à 5 V }
variable    val : réel ;                   { valeur analogique }
début
        val := saisir("Donnez un nombre :") ;
        si val < SEUIL
        alors
                val := 0 ;
        sinon
                val := 1 ;
        afficher("La valeur finale est : ", val) ;
fin
```



Introduction à l'algorithmique

Informatique et Création Numérique

Un deuxième algorithme

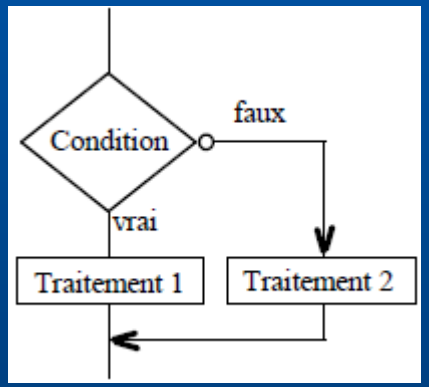
```
program ToutOuRien;
{ affiche 0 si une valeur saisie est inférieure à un seuil donné sinon affiche 1 }

const
  SEUIL = 5;     { seuil à 5 V }
var
  val : real;    { valeur analogique }

begin
     write('Donnez un nombre : ');
     readln(val);

     if val < SEUIL
     then
         val := 0
     else
         val := 1;

     writeln('La valeur finale est : ', val);
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

L'instruction conditionnelle

```
si <expression logique (vraie)>
alors
        début
                Instructions ;
        fin
sinon
        début
                Instructions ;
        fin
```

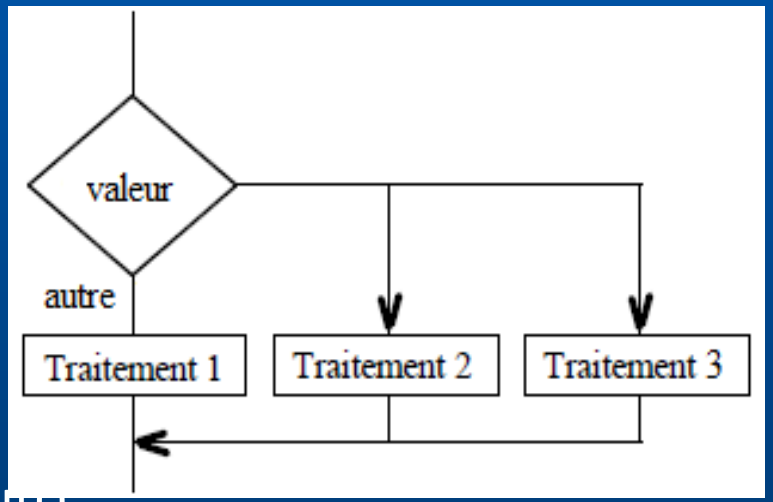


Introduction à l'algorithmique

Informatique et Création Numérique

Sélection conditionnelle

```
Selon <identificateur> Faire
        (Liste de) valeur :
                Traitement() ;
                FinFaire

        Sinon :
                Traitement_par_defaut() ;
finselon
```

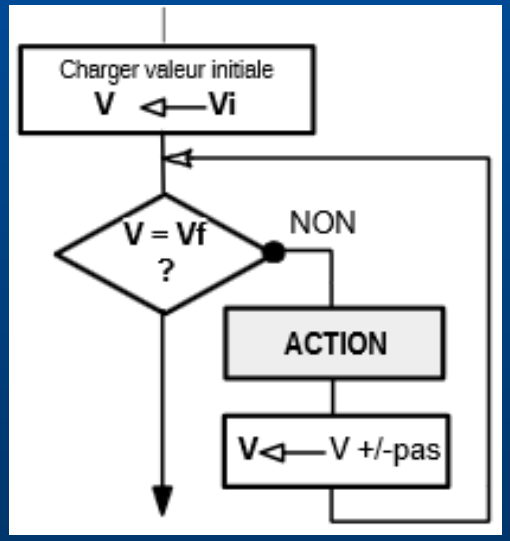


Introduction à l'algorithmique

Informatique et Création Numérique

Les instructions itératives

Pour <variable> de <initiale> à <finale> par pas de <n> Faire
Début
Action(s)
Fin

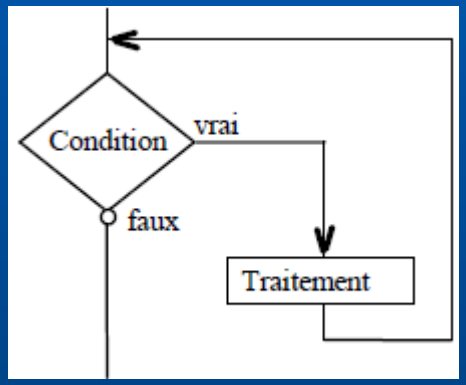


Introduction à l'algorithmique

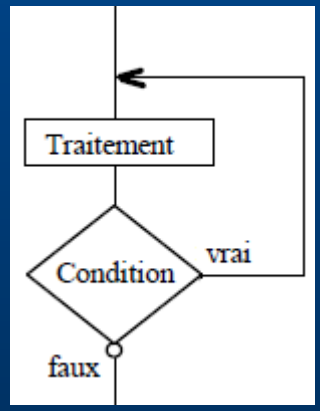
Informatique et Création Numérique

Les instructions itératives

1. tanque <expression logique (vraie)> faire
   début
   Instructions ;
   Fin



2. répéter
   Instructions ;
   jusqu'à <expression logique (fausse)> ;



Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 1 : le distributeur de boisson

Un distributeur propose de 2 types de boissons : eau et soda.
Le stock initial de chaque boisson est égal à 20.
Quand les stocks sont vides, le système doit avertir la maintenance et se mettre Hors Service.
Sinon, le système doit demander la boisson désirée.

- Le bouton 1 sélectionne une bouteille d'eau
- Le bouton 2 sélectionne une canette de soda

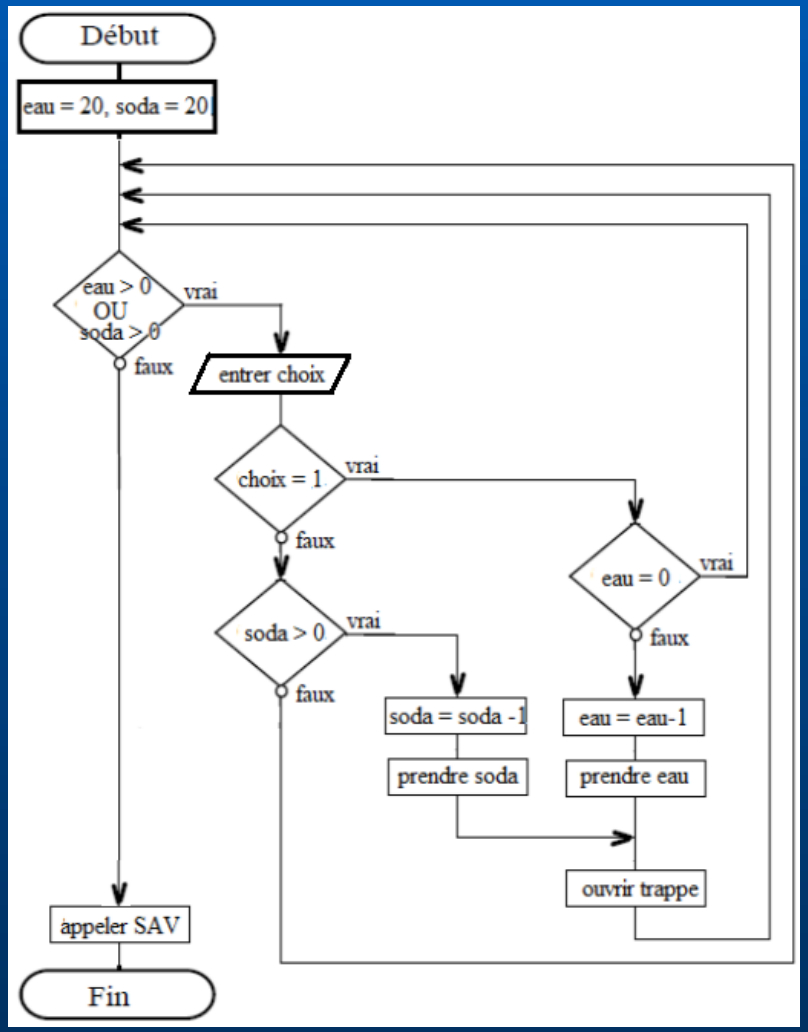
Une fois la sélection faite, si le stock de la boisson sélectionnée n'est pas vide, le système met à jour le stock, sélectionne la boisson demandée et ouvre la trappe d'accès à la boisson.

Introduction à l'algorithmique

Informatique et Création Numérique

Notions abordées

- Opérateur logique
- Conditions imbriquées



Introduction à l'algorithmique

Informatique et Création Numérique

Le distributeur de boisson

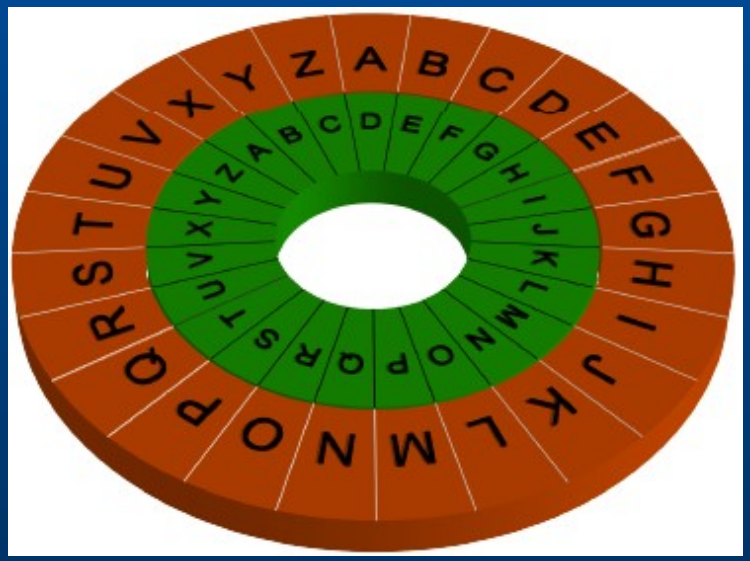
```
program Distributeur;
{ gestion distributeur de boissons }

const
  stock = 10;
var
  eau, soda, choix : integer;

begin
     eau  := stock;
     soda := stock;

     repeat
        writeln('eau : ',eau,' soda : ',soda);
        write('choix boisson (1 : eau, 2 : soda) : ');
        readln(choix);

       if choix = 1
       then
           begin
              if eau > 0
              then
                  begin
                       eau := eau -1;
                       writeln('prendre bouteille eau');
                  end
           end
          else
              begin
              if choix = 2
              then
                  if  soda > 0
                  then
                      begin
                          soda := soda - 1;
                          writeln('prendre bouteille soda');
                      end
              end
     until (eau = 0) and (soda = 0);

     writeln('Appel SAV');
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 2 : le chiffre de Caesar*

Jules César, dans ses correspondances secrètes, codait le texte en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet.



* http://fr.wikipedia.org/wiki/Chiffrement_par_décalage

Introduction à l'algorithmique

Informatique et Création Numérique

Le chiffre de Caesar

mod : modulo d'un nombre
ord(c) : numéro d'ordre dans la table ascii
chr(a) : caractère dont le code ascii est a

Les chaînes de caractères commencent à l'indice 1

```
program Caesar;
{ implémentation du chiffrement de caesar }

const
  modulo = 256; { caractères dans la table ASCII }
  clef   = 2;   { clef de chiffrement }
var
  i, num : integer;
  mot, crypte : string[20];

begin
     crypte := '';

     write('mot à coder : ');
     readln(mot);

     for i := 1 to length(mot) do
     begin
       num := (ord(mot[i]) + clef) mod modulo;
       crypte := crypte + chr(num);
     end;

     writeln(crypte);
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 3 : le mot de passe

Réaliser l'algorigramme d'un programme qui demande à un utilisateur de définir un mot de passe.

- Le mot de passe ne doit pas être inférieur à 5 caractères.
- Le mot de passe de doit pas dépasser 10 caractères.
- Tant que le mot de passe est incorrect, on doit demander le mot de passe.
- Si le mot de passe est correct, on fait appel à un sous programme de chiffrement*, puis on enregistre le mot de passe chiffré dans un fichier.

* avec un algorithme de chiffrement par décalage

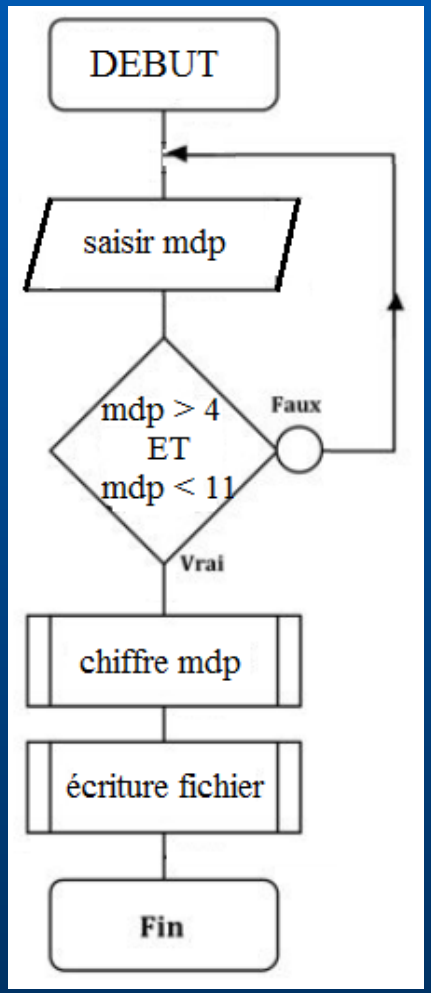
Introduction à l'algorithmique

Informatique et Création Numérique

Notions abordées

- Fonctions et procédures
- Fichier à accès aléatoire

assign(f, nomf) : assigner f au fichier nomf
reset(f) : ouvrir le fichier f en lecture
rewrite(f) : ouvrir le fichier f en écriture
read(f, donnee) : lecture de données
write(f, donnee) : écriture de données
close(f) : fermer le fichier



Introduction à l'algorithmique

Informatique et Création Numérique

Le mot de passe

```
program MotDePasse;
{ chiffrement et enregistrement d'un mot de passe }

const
  fichier = '.\passwd'; { nom du fichier des mdp }
  clef    = 2;          { clef de chiffrement }
var
  mot : string[20];

function caesar(mot : string; clef : integer) : string;
{ implémentation du chiffrement de caesar }
const
  modulo = 256; { caractères dans la table ASCII }
var
  i, num : integer;
  crypte : string[20];
begin
     crypte := '';

     for i := 1 to length(mot) do
     begin
       num := (ord(mot[i]) + clef) mod modulo;
       crypte := crypte + chr(num);
     end;

     caesar := crypte;  { retourne le mdp chiffré }
end;
```

```
procedure enregistre(mot : string; fichier : string);
{ enregistre le mdp chiffré dans un fichier }
var
   f : text;
begin
     assign(f, fichier);
     rewrite(f);         { ouverture fichier en écriture }
     write(f, mot);      { écriture données }
     close(f);           { fermeture fichier }
end;

begin
     { saisie mot depassse }
     repeat
       write('mot de passe : ');
       readln(mot);
     until (length(mot) > 4) and (length(mot) < 11);

     mot := caesar(mot, clef);    { chiffrement }
     enregistre(mot, fichier);    { enregistrement mdp }
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 4 : le tri à bulles

Mesurer la performance de l'algorithme du tri à bulles* sur une liste de 1000 éléments.

- Créer une liste de 1000 entiers [0 ; 500] générés aléatoirement.
- Trier la liste selon l'algorithme du tri à bulles.
- Enregistrer la liste triée dans un fichier.

* permutations successives d'éléments adjacents

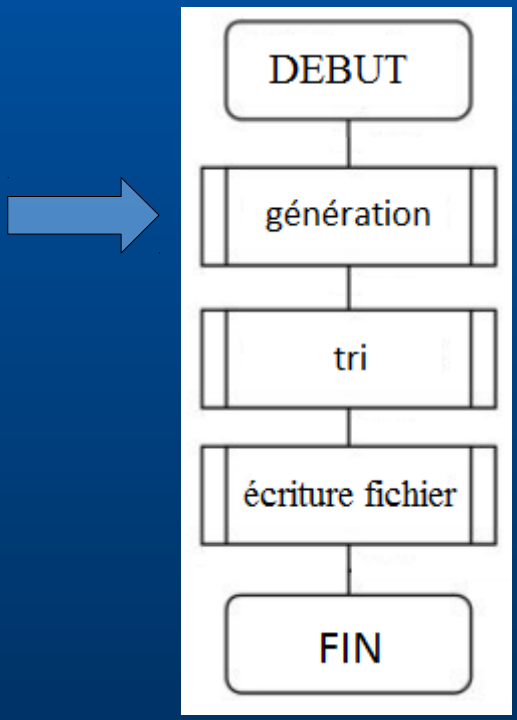
Introduction à l'algorithmique

Informatique et Création Numérique

Notions abordées

- Tableaux
- Passage par valeur et par référence

randomize : initialise le générateur
random(n) : génère un nombre [0..n-1]



Introduction à l'algorithmique

Informatique et Création Numérique

Le tri à bulles

```
program TriBulle;
{ implémentation du tri à bulles }

const
  fichier = '.\liste.txt';      { nom du fichier de la liste triée }
  taille  = 1000;               { taille de la liste }
  max     = 500;                { valeur max  }
type
  liste = array [1..taille] of integer;
var
  l : liste;

function aleas(taille : integer; max : integer) : liste;
{création d'une liste de nombres aléatoires }
var
  i : integer;
  l : liste;
begin
     randomize;
     for i := 1 to taille do
         l[i] := random(max+1);
     aleas := l;
end;

function echange(var x, y : integer) : boolean;
{ echange 2 éléments dans une liste }
var
        z : integer;
begin
        z := x;
        x := y;
        y := z;

        echange := true;
end;
```

```
procedure triBulle(var l : liste);
{ implémentation du tri à bulles }
var
   i, j : integer;
   permut : boolean;
begin
     j := 0;

    repeat
        permut := false;

        for i := 1 to taille - j do
        begin
            if l[i] > l[i+1]
            then
                permut := echange(l[i], l[i+1]);
        end;
        j := j + 1;
    until permut = false;
end;
```

```
procedure enregistre(l : liste; fichier : string);
{ enregistre la liste dans un fichier }
var
   f : file of liste;
begin
     assign(f, fichier);
     rewrite(f);        { ouverture fichier en écriture }
     write(f, l);       { écriture données }
     close(f);          { fermeture fichier }
end;
```

```
begin
    l := aleas(taille, max);    { création d'une liste d'entiers aléatoires }
    triBulle(l);                { tri de la liste }
    enregistre(l, fichier);     { enregistrement de la liste triée }
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 5 : occurrence des lettres

Déterminer l'occurrence des lettres de l'alphabet dans un texte*.

- Utiliser un fichier comportant au moins 1000 caractères.
- Afficher les occurrences en % par ordre décroissant.

Notion de tableaux multidimensionnels
Downto dans boucle for

* utilisé en cryptographie pour le déchiffrement par méthode statistique

Introduction à l'algorithmique

Informatique et Création Numérique

Occurrence des lettres

```
program occurence;
{ recherche occurences de lettres dans un texte }

const
  fichier = '.\occurence.txt';  { nom du fichier texte }
  taille  = 26;
type
  liste = array [1..2] of array [1..taille] of integer;
var
  l : liste;
  f : text;
  lettre : char;
  index, compteur : integer;

procedure initialise(var l : liste; valeur : integer);
{ initialise un tableau }
var
  i : integer;
begin
     for i:= 1 to taille do
     begin
          l[1][i] := ord('a') + i - 1;
          l[2][i] := valeur;
     end
end;

procedure echange(var x, y : integer);  ...

procedure triBulle(var l : liste);  ...

procedure affichage(var l : liste; total : integer);
var
   i : integer;
begin
     for i := taille downto 1 do
         writeln(chr(l[1][i]), ' : ', l[2][i] / total * 100 :0:2, '%');
end;

begin
     compteur := 0;
     initialise(l, 0);

     assign(f, fichier);
     reset(f);           { ouverture fichier en écriture }

     while not eof(f) do
     begin
       read(f, lettre);
       index := ord(LowerCase(lettre)) - ord('a') + 1;
       if (index > 0) and (index < taille)
       then
           begin
                l[2][index] := l[2][index] + 1;
                compteur := compteur + 1;
           end;
     end;

     { affichage résultat}
     triBulle(l);
     affichage(l, compteur);

     close(f);           { fermeture fichier }
end.
```

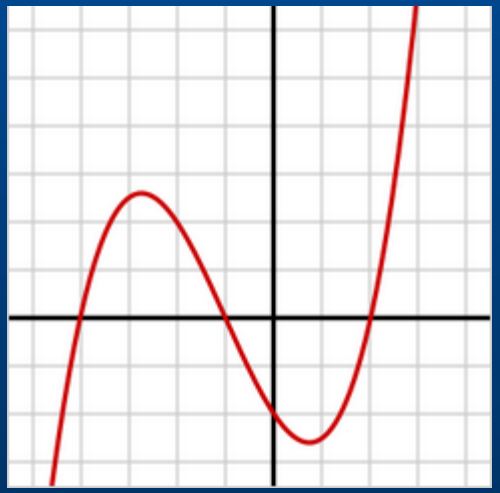
Introduction à l'algorithmique

Informatique et Création Numérique

Exemple 6 : pour terminer...

Écrire un programme pour rechercher les racines d'un polynôme :

- Quelconque de degré 2
- D'équation : $y = 0{,}25.x^3 + 0{,}75.x^2 - 1{,}5.x - 1{,}9$



Introduction à l'algorithmique

Informatique et Création Numérique

Polynôme degré 2

Racine carrée : sqrt()
Valeur absolue : abs()

```
program Degre2;
{ recherche des racines d'un polynome de degre 2 }

var
  a, b, c : real;               { déclarations des variables }
  delta, x1, x2 : real;

begin
     { préparation du traitement }
     writeln('Entrer les coefficients ax^2 + bx +c :');

     repeat
           write('a : '); readln(a);  { Lecture }
     until ( a <> 0 );
     write('b : '); readln(b);  { Lecture }
     write('c : '); readln(c);  { Lecture }

    // traitement
    delta := (b*b) - 4 * (a*c);
    x1 := (-b + sqrt(abs(delta))) / (2*a);
    x2 := (-b - sqrt(abs(delta))) / (2*a);

    // affichage
    writeln('delta: ', delta:2:0);

    if ( delta <> 0)
    then
        begin
          if ( delta < 0 )
          then
              begin
                x1 := -b / (2*a);
                x2 := sqrt(abs(delta)) / (2*a);
                write('racines complexes : ', x1:0:2, ' +j', x2:0:2, ' et ',x1:0:2, ' -j', x2:0:2);
              end
          else
              write('racines reelles : ', x1:0:2, ' et ',x2:0:2);
        end
    else
        write('racine double : ', x1:0:2);
end.
```

Introduction à l'algorithmique

Informatique et Création Numérique

Polynôme degré 3

```
program Degre3;
{ recherche des racines d'un polynome de degre 3 par dichotomie }

var
  a, b : real;          { déclarations des variables }
  delta, y1, y2, x : real;


function power(a : real; n : integer) : real;
{ fonction puissance }
begin
     if n = 0
     then
         power := 1.0
     else
         if n > 0
         then
             power := a * power(a, n - 1)
         else
             power := 1/a * power(a, n + 1);
end;

function polynome(x : real) : real;
{ la fonction polynome }
begin
     polynome := 0.25 * power(x, 3) - 0.75 * power(x, 2) - 1.5 * x - 1.9;
end;
```

```
begin
     writeln('Entrer un interval [a,b] :');

     repeat
       write('a : '); readln(a);  { Lecture }
       write('b : '); readln(b);  { Lecture }

       y1 := polynome(a);
       y2 := polynome(b);
       writeln('[a,b] = ', y1:0:2, ' ', y2:0:2);
     until (a < b) and ((y1*y2) < 0.0);

    // traitement
    delta := 0.01;
    while (abs(y1) > delta) and (abs(y2) > delta) do
    begin
        x := a;
        a := a + (b - a) / 2;

        y1 := polynome(a);
        y2 := polynome(b);

        if (y1 * y2) > 0
        then
            b := x;
    end;

    // affichage
    if y1 < y2
    then
        writeln('la racine est ', a:0:2)
    else
        writeln('la racine est ', b:0:2);
end.
```

puissance : power()
→ écrire une fonction récursive