Ecole Nationale Supérieure d'Arts et métiers.
Mathématiques
Alg-C

Programmation langage C

Introduction à l'algorithmique

A.BELCAID
anasss.belcaid@gmail.com

Salle 1, TD 2 / 22 février 2016

Definition

Les structures de contrôle permettent à un programme d'éviter une exécution purement séquentielle (*chaîne linéaire d'instructions*), en contrôlant l'**enchaînement** logique des instructions.

Definition

Les structures de contrôle permettent à un programme d'éviter une exécution purement séquentielle (*chaîne linéaire d'instructions*), en contrôlant l'**enchaînement** logique des instructions.

Exemple

La **résolution d'une équation du second degré** dans $\mathbb{R}$

Le **programme** qui doit résoudre ce problème **devra donc adapter son comportement** en fonction des valeurs prises par certaines variables (notamment le discriminant de l'équation).

Définition

La structure de contrôle if...then permet d'exécuter des instructions en **fonction** de la valeur d'une **condition** de type booléen

Sa syntaxe est la suivante :

syntaxe

```
1 if Condition then
2 |   Instructions.
3 end
```

Algorithme 1 : Strictement positif

```
1 if q >0 then
2 |   Ecrire(q,'DH')
3 end
```

Algorithme 2 : Appartenance à un intervalle

```
1 if (x ≥ a) et (x ≤ b) then
2 |   Ecrire(x,' ∈ [a, b]')
3 end
```

If..then ..else

- Il arrive assez souvent qu'en fonction de la valeur d'une **condition**, le programme doit exécuter des instructions si elle est vraie et **d'autres** instructions si elle est fausse.

If..then ..else

- Il arrive assez souvent qu'en fonction de la valeur d'une **condition**, le programme doit exécuter des instructions si elle est vraie et **d'autres** instructions si elle est fausse.
- Plutôt que de **tester** une condition puis son contraire, il est possible d'utiliser la structure If..then ..else.

- Il arrive assez souvent qu'en fonction de la valeur d'une **condition**, le programme doit exécuter des instructions si elle est vraie et **d'autres** instructions si elle est fausse.
- Plutôt que de **tester** une condition puis son contraire, il est possible d'utiliser la structure If..then ..else.

Syntaxe

```
1 if condition then
2 |   Instructions 1;
3 else
4 |   Instructions 2;
5 end
```

If..then ..else

- Il arrive assez souvent qu'en fonction de la valeur d'une **condition**, le programme doit exécuter des instructions si elle est vraie et **d'autres** instructions si elle est fausse.
- Plutôt que de **tester** une condition puis son contraire, il est possible d'utiliser la structure If..then ..else.

Syntaxe

```
1 if condition then
2 |   Instructions 1;
3 else
4 |   Instructions 2;
5 end
```

Remarque

Les instructions peuvent contenir eux mêmes des branchement conditionnelles.

Exemple

Algorithme 3 : Parité

```
1 Var a : Entier

2 if (a%2 = 0) then
3 |   Ecrire(a, ' est pair');
4 else
5 |   Ecrire(a , 'est impair');
6 end
```

Exercice

Ecrire un **algorithme** qui demande la saisie de **deux** réels , et affiche le signe de leur produit. **sans le calculer**

Exercice

Ecrire un **algorithme** qui demande la saisie de **deux** réels , et affiche le signe de leur produit. **sans le calculer**

```
1 Var a, b : Réel

2 Ecrire('Donner deux réels : ')
3 lire(a)
4 lire(b)
5 if ((a > 0 et b > 0) ou (a < 0 et b < 0)) then
6 |   Ecrire('Le produit est positif')
7 else
8 |   Ecrire('Le produit est négatif')
9 end
```

Exercice

Développer un **algorithme** qui demande la saisie de 3 entiers, puis informe l'utilisateur s'ils sont **rangés** dans un ordre monotone.

Exercice

Développer un **algorithme** qui demande la saisie de 3 entiers, puis informe l'utilisateur s'ils sont **rangés** dans un ordre monotone.

Algorithme 5 : If imbriquée

```
1  Var a, b, c : Réel
2  Var test : Booléen
3  Ecrire('Donner trois réels : ')
4  lire(a) ;lire(b) ;lire(c)
5  if a > b then
6  |   if b > c then
7  |   |   test ← true
8  |   else
9  |   |   test ← false
10 |   end
11 else
12 |   if b < c then
13 |   |   test ← true
14 |   else
15 |   |   test ← false
16 |   end
17 end
18 if test then
19 |   Ecrire('Ordre monotone')
20 else
21 |   Ecrire('Pas d ordre')
22 end
```

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.
- Il existe **trois** type de boucle :

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.
- Il existe **trois** type de boucle :

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.
- Il existe **trois** type de boucle :
  - **For**

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.
- Il existe **trois** type de boucle :
  - **For**
  - **While**

- Certains **algorithmes** nécessitent de répéter des **instructions** afin d'obtenir un résultat.
- Cette **répétition** est réalisée en utilisant une structure de contrôle de type **itératif** nommée boucle.
- Il existe **trois** type de boucle :
  - **For**
  - **While**
  - **Repeat... until**

- La boucle For est utilisée pour **répéter** un ensemble d'instructions.
- le nombre de répétition *n* est souvent **connu à l'avance**.
- L'usage principal de la boucle **For** est la gestion de l'évolution d'un compteur(type entier).

- La boucle For est utilisée pour **répéter** un ensemble d'instructions.
- le nombre de répétition n est souvent **connu à l'avance**.
- L'usage principal de la boucle **For** est la gestion de l'évolution d'un compteur(type entier).

Syntaxe

```
1 for i= val1 à val2 do
2 |   Instructions
3 end
```

Algorithme 6 : Répétition

```
1 Var a : Entier

2 for a =1 à 10 do
3 |   Ecrire(a)
4 end
```

Algorithme 7 : ???

```
1 Var a,i,S : Entier

2 Ecrire('donner un entier : ')
3 lire(a)
4 S ← 0
5 for i=1 à 100 do
6 |   S ← S + a
7 |   Ecrire(S)
8 end
```

Exercice

Soit $(u_n)_{n\in\mathbb{N}}$ une suite **donnée**. Ecrire un algorithme qui demande la saisie d'un entier n, puis affiche la somme partielle S_n de la série $\sum u_n$.

$$S_n = \sum_{i=0}^{n} u_i$$

Exercice

Soit $(u_n)_{n\in\mathbb{N}}$ une suite **donnée**. Ecrire un algorithme qui demande la saisie d'un entier n, puis affiche la somme partielle S_n de la série $\sum u_n$.

$$S_n = \sum_{i=0}^{n} u_i$$

Algorithme 9 : Somme partielle d'une série

```
1 Var i, n : Entier
2 Var U, S , Réel

3 Ecrire( 'Donner n = ') ;lire(n)
4 S ← 0
5 for i =0 à n do
6 |   U ← u_i
7 |   S ← U
8 end
9 Ecrire('Sn= ',S)
```

Factoriel

Ecrire un algorithme qui demande la saisie d'un **entier** n,puis affiche son factoriel.

Factoriel

Ecrire un algorithme qui demande la saisie d'un **entier** n,puis affiche son factoriel.

Maximum

Ecrire un algorithme qui demande **successivement** 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le **plus grand** parmi ces 20 nombres.

Factoriel

Ecrire un algorithme qui demande la saisie d'un **entier** n,puis affiche son factoriel.

Maximum

Ecrire un algorithme qui demande **successivement** 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le **plus grand** parmi ces 20 nombres.

Algorithme 12 : Maximum de 20 valeurs

```
1 Var a,i,max : Entier

2 Ecrire('Donner une valeur : ');
  lire(max)
3 for i = 1 à 19 do
4 |   Ecrire('Donner une valeur : ');
  |   lire(a)
5 |   if a > max then
6 |   |   max ← a
7 |   end
8 end
9 Ecrire('max est :',max)
```

- Dans certains **problèmes**, on ne connaît pas à l'avance le nombre de **répétition** du procédé itératif , mais plutôt une condition logique pour **terminer** les itérations.

- Dans certains **problèmes**, on ne connaît pas à l'avance le nombre de **répétition** du procédé itératif , mais plutôt une condition logique pour **terminer** les itérations.
- Dans ce cas, on utilise la structure de **contrôle** While.

- Dans certains **problèmes**, on ne connaît pas à l'avance le nombre de **répétition** du procédé itératif , mais plutôt une condition logique pour **terminer** les itérations.
- Dans ce cas, on utilise la structure de **contrôle** While.

While

```
while (Condition) do
    Instructions
end
```

- Dans certains **problèmes**, on ne connaît pas à l'avance le nombre de **répétition** du procédé itératif , mais plutôt une condition logique pour **terminer** les itérations.
- Dans ce cas, on utilise la structure de **contrôle** While.

While

```
while (Condition) do
    Instructions
end
```

- Dans certains **problèmes**, on ne connaît pas à l'avance le nombre de **répétition** du procédé itératif , mais plutôt une condition logique pour **terminer** les itérations.
- Dans ce cas, on utilise la structure de **contrôle** While.

While

```
1 while (Condition) do
2 |   Instructions
3 end
```

Remarque

les **instructions** de la boucle **While** ne seront jamais exécutées,si la **Condition** est fausse au **départ**.

Algorithme 13 : Countdown

```
1 while (n > 0) do
2 |   Ecrire(n)
3 |   n ← n − 1
4 end
```

Algorithme 14 : $\max\{k \mid \sum_{i=0}^{k} i < A\}$

```
1 i ← 0
2 S ← 0
3 while (S < A) do
4 |   i ← i + 1
5 |   S ← S + i
6 end
7 Ecrire(i − 1)
```

Exercice 1

Ecrire un algorithme qui demande la **saisie** d'un entier A, puis affiche le plus grand entier i, tel que

$$i! \leq A$$

Exercice 1

Ecrire un algorithme qui demande la **saisie** d'un entier A, puis affiche le plus grand entier i, tel que

$$i! \leq A$$

Exercice 2

Développer un algorithme qui demande une suite d' entiers **positifs** se terminant avec -1. puis calcul :

1. leur **min**.
2. leur **somme**.
3. leur **moyenne**

Exercice 1

Ecrire un algorithme qui demande la **saisie** d'un entier A, puis affiche le plus grand entier i, tel que

$$i! \leq A$$

Exercice 2

Développer un algorithme qui demande une suite d' entiers **positifs** se terminant avec -1. puis calcul :

1. leur **min**.
2. leur **somme**.
3. leur **moyenne**

Algorithme 17 : Statistiques

```
1  Var a,count : Entier
2  Var min,S :Entier
3  Var moy : Réel
4  S ← 0 ; min ← 0 ; count ← 0
5  Ecrire('Donner une valeur : ') ;lire(a)
6  while (a ≠ −1) do
7  |   count ← count + 1
8  |   S ← a
9  |   if a > max then
10 |   |   max ← a
11 |   end
12 end
13 if count > 0 then
14 |   moy ← S / count
15 end
16 Ecrire('somme= ',S)
17 Ecrire('max= ',max)
18 Ecrire('moyenne= ',moy)
```

- Parfois on a besoin d'exécuter les instructions d'une boucle au moins un fois avant de passer au **test**.
- pour cela on devrait utiliser la boucle repeat ... until.

- Parfois on a besoin d'exécuter les instructions d'une boucle au moins un fois avant de passer au **test**.
- pour cela on devrait utiliser la boucle repeat ... until.

Syntaxe

```
1 repeat
2 |   Instructions
3 until (Condition);
```

exemple

```
1 repeat
2     ecrire("Donner un nombre positif :");
3     lire(A)
4 until (A > 0);
```

exemple

```
1 repeat
2     ecrire("Donner un nombre positif :");
3     lire(A)
4 until (A > 0);
```

exemple

```
1 repeat
2     ecrire("Est ce que vous voulez continuer (O=Oui,N=Non) :");
3     lire(c)
4 until ((c='O') OU (c='N'));
```