Algorithmique & Programmation

Fahd Karami

fa.karami@uca.ma

Plan

I- **Les instructions d'entrées-sorties**

II- **Les structures Conditionnelles**

Les instructions d'entrées-sorties: lecture/Ecriture

- Les instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur
- La **lecture** permet d'entrer des donnés à partir du clavier
  - En pseudo-code, on note: **Lire (var);**

    la machine met la valeur entrée au clavier dans la zone mémoire nommée var
  - Remarque: Le programme s'arrête lorsqu'il rencontre une instruction Lire et ne se poursuit qu'après la frappe d'une valeur au clavier et de la touche Entrée

Les instructions d'entrées-sorties: lecture et écriture (2)

- **L'écriture** permet d'afficher des résultats à l'écran (ou de les écrire dans un fichier)

  - En pseudo-code, on note: **Ecrire (var) ;**

    la machine affiche le contenu de la zone mémoire var

  - Conseil: Avant de lire une variable, il est fortement conseillé d'écrire des messages à l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper

Exemple (Lecture / Ecriture)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le double de ce nombre ?

Exemple (Lecture/ Ecriture)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le double de ce nombre

```
Algorithme Calcul_double
 variables      A, B : entiers
 Début
        Ecrire("entrer la valeur de A: ");
        Lire(A);
        B ← 2*A;
        Ecrire("le double de :", A, "est :", B);
 Fin
```

Exercice (Lecture / Ecriture)

Ecrire un algorithme qui vous demande de saisir votre nom puis votre prénom et qui affiche ensuite votre nom complet?

Exercice: (Lecture / Ecriture)

Ecrire un algorithme qui vous demande de saisir votre nom puis votre prénom et qui affiche ensuite votre nom complet

```
Algorithme AffichageNomComplet
variables Nom, Prenom, Nom_Complet : chaîne de caractères
Début
        Ecrire("entrez  votre nom");
        Lire(Nom);
        Ecrire("entrez  votre prénom");
        Lire(Prenom);
        Nom_Complet ← Nom & Prenom;
        Ecrire("Votre nom complet est : ", Nom_Complet);
Fin
```

Méthode de construction d'un algorithme simple (1/4)

Exemple :

Écrire un algorithme qui consiste a calculer l'air S d'un cercle selon la formule S = Pi * R* R;

Rappel : Pi = 3.14159 et R le rayon du cercle

Méthode de construction d'un algorithme simple (2/4)

Méthodologie a suivre :

- constantes : Pi = 3.14159
- Variables : Rayon, Surface
- Types : Rayon, Surface : réel
- Expressions et affectation : Surface <- Pi * (Rayon)^2 ;
- Structures conditionnelles et les boucles : ------
- Opérations d'entrée-sortie : Lire (Rayon);
  Écrire (Surface);

Méthode de construction d'un algorithme simple (3/4)

Algorithme **Calcul_Aire**

```
Constantes
    Pi = 3,14159
Variables
    Rayon, Surface : réels
Début
    Lire (Rayon);
    Surface <- Pi * (Rayon*Rayon) ;
    Ecrire (Surface);
Fin
```

Méthode de construction d'un algorithme simple (3/4)

Programme Pascal	Programme C
Program Calcul_Aire; **CONST** Pi = 3.14159 **VAR** Rayon, Surface **: REAL;** **BEGIN** **READLN (Rayon);** Surface := Pi * SQR (Rayon); **WRITELN (**Surface**);** **END.**	**#include <stdio.h>** **#include <math.h>** **main ( ){** **float** Pi = 3.14159; **float** rayon, surface; **scanf** (« ° /° f », &rayon); **surface =** pi*pow (rayon,2); **printif (**« ° /° f\n »surface, **);** **}**

Algorithmique

II- Les structures Conditionnelles

Besoin a des concepts de ruptures de séquence

Algorithme

```
Calcul_Aire
Constantes
    Pi = 3,14159
Variables
    Rayon, Surface : réels
Début
    Lire (Rayon);
    Surface <- Pi * (Rayon)^2 ;
    Ecrire (Surface);
Fin
```

- Rare les algorithme qui peuvent se décrire uniquement par un enchaînement séquentiel d'opération élémentaire

- On a besoin a des concept de rupture de séquence comme les test et les boucles

Ex :

- ✓ un algorithme qui résout une équation de deuxième degré
- ✓ un algorithme qui calcule une série numérique

Les structures conditionnelles et les boucles

- Les tests simples : permet de réaliser un choix parmi deux possibilités (Ex :Booléenne : vrais ou faux)
- Les instructions conditionnelles : c'est un concept de tests multiples, permet de comparer un objet à une série de valeurs, et exécuter si la condition est vérifier (Ex : recherche des nombres premier dans une ensemble)
- Les itérations : consiste a exécuté un bloc d'instructions un certain nombre de fois (Ex : calcul d'une suite numérique)
- Les boucles conditionnelles : consiste a exécuté un bloc d'instructions un certain nombre de fois si la condition est vérifier (Ex : On veut afficher le 100 premiers nombres :. Tant que i est plus petit que 100, afficher la valeur de i).

Tests: instructions conditionnelles (1)

- Les instructions conditionnelles servent à n'exécuter une instruction ou une séquence d'instructions que si une condition est vérifiée
- On utilisera la forme suivante:

Si condition **alors**

instruction ou suite d'instructions1

Sinon

instruction ou suite d'instructions2

Finsi

  - La condition ne peut être que vraie ou fausse
  - Si la condition est vraie, se sont les instructions1 qui seront exécutées
  - Si la condition est fausse, se sont les instructions2 qui seront exécutées
  - La condition peut être une condition simple ou une condition composée de plusieurs conditions

Tests: instructions conditionnelles (2)

- La partie Sinon n'est pas obligatoire, quand elle n'existe pas et que la condition est fausse, aucun traitement n'est réalisé
  - On utilisera dans ce cas la forme simplifiée suivante:

Si condition **alors**

instruction ou suite d'instructions I

Finsi

Exemple (Si…Alors…Sinon)

Écrire un algorithme qui consiste a afficher la valeur absolue d'un nombre réel?

Exemple (Si…Alors…Sinon)

```
Algorithme AffichageValeurAbsolue (version1)
Variable x : réel
Début
        Ecrire (" Entrez un réel : ");
        Lire (x);
        Si ( x < 0) alors
                Ecrire ("la valeur absolue de ", x, "est:",-x);
        Sinon
                Ecrire ("la valeur absolue de ", x, "est:",x);
        Finsi
Fin
```

Exemple (Si…Alors)

Écrire un algorithme qui consiste a afficher la valeur absolue d'un nombre réel?

Cette fois vous ne devrez pas utilisez l'instruction(Sinon).

Exemple (Si…Alors)

```
Algorithme AffichageValeurAbsolue (version2)
Variable x,y : réel
Début
        Ecrire (" Entrez un réel : ");
        Lire (x);
        y← x;
        Si x < 0 alors
                y ← -x;
        Finsi
        Ecrire ("la valeur absolue de ", x, "est:",y);
Fin
```

Exercice (tests)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur,puis qui teste et affiche s'il est divisible par 3?

Exercice (tests)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 3

```
Algorithme Divsible_par3
Variable n : entier
Début
        Ecrire (" Entrez un entier : ");
        Lire (n);
        Si (n%3=0) alors
                Ecrire (n," est divisible par 3");
        Sinon
                Ecrire (n," n'est pas divisible par 3");
        Finsi
Fin
```

Conditions composées

- Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques:

  ET, OU, OU exclusif (XOR) et NON

- Exemples :
  - x compris entre 2 et 6 : (x > 2) ET (x < 6)
  - n divisible par 3 ou par 2 : (n%3=0) OU (n%2=0)
  - deux valeurs et deux seulement sont identiques parmi a, b et c :

    (a=b) XOR (a=c) XOR (b=c)

- L'évaluation d'une condition composée se fait selon des règles présentées généralement dans ce qu'on appelle tables de vérité

Tables de vérité

C1	C2	**C1 ET C2**
VRAI	VRAI	
VRAI	FAUX	
FAUX	VRAI	
FAUX	FAUX	

C1	C2	**C1 OU C2**
VRAI	VRAI	
VRAI	FAUX	
FAUX	VRAI	
FAUX	FAUX	

C1	C2	**C1 XOR C2**
VRAI	VRAI	
VRAI	FAUX	
FAUX	VRAI	
FAUX	FAUX	

C1	**NON C1**
VRAI	
FAUX	

Tables de vérité

C1	C2	**C1 ET C2**
VRAI	VRAI	**VRAI**
VRAI	FAUX	**FAUX**
FAUX	VRAI	**FAUX**
FAUX	FAUX	**FAUX**

C1	C2	**C1 OU C2**
VRAI	VRAI	**VRAI**
VRAI	FAUX	**VRAI**
FAUX	VRAI	**VRAI**
FAUX	FAUX	**FAUX**

C1	C2	**C1 XOR C2**
VRAI	VRAI	**FAUX**
VRAI	FAUX	**VRAI**
FAUX	VRAI	**VRAI**
FAUX	FAUX	**FAUX**

C1	**NON C1**
VRAI	**FAUX**
FAUX	**VRAI**

Tests imbriqués

- Les tests peuvent avoir un degré quelconque d'imbrications

```
Si condition1 alors
                Si condition2 alors
                        instructionsA
               Sinon
                        instructionsB
                Finsi
 Sinon
                Si condition3 alors
                        instructionsC
               Finsi
 Finsi
```

Tests imbriqués: exemple (version 1)

```
Algorithme NegPos

Variables n : entier
Début
        Ecrire ("entrez un nombre : ");
        Lire (n);
        Si (n < 0) alors
                Ecrire ("Ce nombre est négatif");
        Sinon
                Si  (n = 0)  alors
                        Ecrire ("Ce nombre est nul");
                Sinon
                        Ecrire ("Ce nombre est positif");
                Finsi
        Finsi
Fin
```

Tests imbriqués: exemple (version 2)

```
Algorithme NegPos2
Variables n : entier
    Début
        Ecrire ("entrez un nombre : ");
        Lire (n);
        Si (n < 0) alors
                Ecrire ("Ce nombre est négatif");
        Finsi
        Si (n = 0)  alors
                Ecrire ("Ce nombre est nul");
        Finsi
        Si (n > 0) alors
                Ecrire ("Ce nombre est positif");
        Finsi
    Fin
```

Remarque : dans la version 2 on fait trois tests systématiquement alors que dans la version 1, si le nombre est négatif on ne fait qu'un seul test .

Conseil : utiliser les tests imbriqués pour limiter le nombre de tests et placer d'abord les conditions les plus probables (minimiser la complexité)

Tests imbriqués: exercice

Le prix de photocopies dans une reprographie varie selon le nombre demandé: 0,5 DH la copie pour un nombre de copies inférieur à 10, 0,4DH pour un nombre compris entre 10 et 20 et 0,3DH au-delà.

Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, qui calcule et affiche le prix à payer ?

Tests imbriqués: corrigé de l'exercice

Algorithme cop

```
Variables copies : entier
                  prix : réel
  Début
        Ecrire ("Nombre de photocopies : ");
        Lire (copies);
        Si (copies < 10)  alors
            prix ← copies*0.5;
        Sinon
                    Si  (copies < 20) alors
                        prix ← copies*0.4;
                    Sinon
                        prix ← copies*0.3;
                    Finsi
        Finsi
        Ecrire ("Le prix à payer est : ", prix);
  Fin
```

Tests imbriqués: Exercice 2

Écrire l'algorithme du traitement qui calcule le discriminant DELTA d'trinome du second degré AX2 + BX + C et qui, en fonction de son signe, calcule la ou les racines réelles du trinome ou afiche, si besoin est qu'il n'ya pas de racine réelle.

Les trois coefficients A, B et C seront saisis au clavier avant traitement.

Tests imbriqués: corrigé de l'exercice 2

```
Algorithme Eq

Variables     A, B, C, Delta, sol1, sol2 : réels
   Début
        Lire (A, B, C);
        Delta ← B*B – 4*A*C;

        Si (Delta < 0) alors
        Ecrire ( « le trinome n'a pas de racine réelle »);
        Sinon
                Si  (Delta > 0)  alors
                        sol1 ← (-B + racine(Delta)) / 2*A;
                        sol2 ← (-B – racine(Delta)) / 2*A;

                        Ecrire (« le trinome possède deux racines réelles : », sol1, sol2);
                Sinon
                        sol1 ← -B /(2*A);
                        Ecrire (« le trinome possède une racine réelle : », sol1);
                Finsi
        Finsi

    Fin
```