Université M'Hamed BOUGARA - Boumerdes
Faculté des sciences – Département d'Informatique

Algorithmique et structure de données

Présentée par

Iza lyla

lylaiza@yahoo.fr

Contenu du module

- Introduction
- Historique
- Généralités et concepts de base
- Algorithmes séquentiels simple
- Les structures conditionnelles en algorithmique et en langage C
- Les boucles (en langage algorithmique et en C)
- Les tableaux et les chaînes de caractères
- Les types personnalisés

Les structures conditionnelles

Les structures de contrôle alternatives

Plan du cours 3: Les structures de contrôle (conditionnelles) alternatives

- Les structures de contrôle alternatives
- Les structures conditionnelles simple
- Les structures conditionnelles composées
- Les structures conditionnelles imbriquées
- Les structures conditionnelles de choix multiple
- Branchement

Les structures de contrôle (alternatives)

En programmation, on est souvent confronté à des situations ou on a besoin de choisir entre 2 ou plusieurs traitements selon la réalisation ou non d'une certaine condition (test). Un traitement conditionnel ou alternative est donc un traitement qui repose sur un test, ce dernier est exprimé à l'aide d'une ou plusieurs conditions.

Si « condition_verifié » alors action (ou traitement)

En c

if <condition> {<actions>}

1. ***La condition (ou expression logique ou prédicat)*** est un énoncé ou proposition qui peut être vrai ou faux, en d'autre terme une condition peut prendre une valeur logique soit « vrai » ou « faux ».

Exemple : La condition (10 < 15) est *vrai* et La condition (10 < 3) est *faux*

- ***Une condition peut être composée ,*** c'est-à-dire, qu'on peut avoir une ou plusieurs conditions à vérifier en même temps; pour ce la on utiles les opérateurs logiques : **ET , OU, NON, ….**

si « (condition1_verifié) ET (condition2_verifié)» alors action(ou traitement)

- ***Evaluation d'une expression logique*** le résultat de l'évaluation d'une expression logique est une valeur logique (booléenne) ayant la valeur vrai ou faux qu'on note V ou F:

L'évaluation de l'opérateur Non

C	Non C
V	F
F	V

L'évaluation de l'opérateur logique et

C1	C2	C1 et C2
V	V	V
V	F	F
F	V	F
F	F	F

L'évaluation de la négation des opérateurs de comparaison

OPERATEUR	NEGATION
=	<>
<>	=
<	>=
>	<=
<=	>
>=	<

L'évaluation de l'opérateur logique ou

C1	C2	C1 OU C2
V	V	V
V	F	V
F	V	V
F	F	F

l'évaluation de l'opérateur non sur les opérateurs logique : si A et B des variables logique (booléennes) alors :

- non (A et B) = (non A) ou (non B) .
- non (A ou B) = (non A) et (non B).
- non (non A) = A

Les structures de contrôle (alternatives)

On distingue plusieurs structures de traitement conditionnel à savoir :

- ***Les structures conditionnelles simple***
- ***La structures conditionnelles composées***
- ***La structures conditionnelles imbriquées***
- ***La structures conditionnelles de choix multiple***

Structures de contrôle simple (L'alternative simple)

la structure conditionnelle simple consiste à évaluer une condition ou une expression logique et d'effectuer le traitement relatif à la valeur de vérité trouvée.

```
Si <condition> alors <action (s)(traitement)> Finsi
```

En langage de programmation c

```
if  <condition> { actions (traitements)}
```

Principe : Si la condition est vérifiée, l'action (ou le groupe d'actions) est exécutée puis les actions qui suivent le ***finsi (fsi)***. Par contre si la condition n'est pas vérifiée, seules les actions qui suivent le ***finsi (fsi)*** seront exécutées (c. à d. rien à faire).

Exemple: déterminer si un nombre et supérieure à un autre nombre

```
Algorithme comparaison
Var x, y : entier;
Debut
Lire (x, y);
Si (x>y) alors
Ecrire (‘la valeur de x est supérieur à la valeur de y’);
Finsi;
Fin.
```

Les structures conditionnelles composées

qui consiste à évaluer une expression qui n'est pas nécessairement à valeur booléenne (elle peut avoir plus de deux valeurs) et selon la valeur trouvée, effectue un traitement.

```
Si <condition> alors
<action(s)1 (traitement(s)1)>
Si non
<action(s)2 (traitement(s)2)>
Finsi
```

L'interprétation en c :

```
if  <condition> { <actions1>}
else
{<actions2>}
```

Les structures conditionnelles composées (principe d'utilisation)

Principe : ***s***i la condition est vérifiée, alors c'est l'action1 (ou groupe action1) qui sera exécutée, puis les actions qui suivent le ***finsi*** . Si cette condition n'est pas vérifiée, alors il faut exécuter l'action2 (ou le groupe d'actions2), puis l'exécution des actions qui suivent le ***finsi***.

Remarque : si les actions (ou traitements) contiennent plus d'une instruction alors on doit les délimitée par ***debut*** et ***fin.***

Exemple 3 : déterminer le minimum de deux nombre entiers

```
Algorithme min
var x, y, min : entier;
debut
    ecrire (‘donnez les valeurs de x et y’);
    lire (x,y),
        si (x < y) alors
        min ← x ;
        si non
        min ← y ;
        finsi
    ecrire (‘le minimum entre x et y est :’, min);
fin.
```

La traduction de l'exemple 3 en c :

```
#include <stdio.h>
Void main()
{
 int x,y,min ;
 printf ("donner les valeurs de x et y \n" );
 scanf ("%d,%d" &x,&y);
 if (x < y) { min =x;}
 else
 {min = y}
 printf (" le minimum entre x et y est : %d \n ",min);

}
```

Les structures conditionnelles imbriquées

structures alternatives sont dites imbriqués lorsque l'une contient l'autre.

Exemple : déterminer le minimum de trois nombres :

```
Algorithme min_trois_nb;
      var x,y,y : entier ;
      debut
          ecrire ('introduire les valeurs de x, y et z ');
          lire (x,y,z);
            si ((x<y) et (x<z)) alors
                  min ← x;
            si non
               si ((x>y) et (x>y)) alors
                     si (y<z) alors
                        min ← y;
                     si non   min ← z;
                si non
                  si ((x<y) et (x>z)) alors
                            min ← z;
                si non
                   si ((x>y) et (x<z)) alors
                       min ← y ;
                             finsi; finsi; finsi;finsi;finsi;
Fin.
```

Traduction en c e l'exemple précèdent :

```
#include <stdio.h>
int main() {
int x, y, z;
    printf("introduire les trois variables \n");
    scanf (" %d ", &x);
    scanf (" %d ", &y);
    scanf (" %d ", &z);

    printf ("les valeur de x y et z sont %d %d %d", x, y, z);
    int min;
```

```
if ((x<y) & (x<z)) {
                    min = x;}
                    else
                    {
                         if ((x>y) & (x>z))
                         { if (y<z)
                         { min = y ;
                         }else min = z;

                         }else
                         if ((x<y)&(x>z)) {
                              min = z;
                         } else
                         if ((x>y)&(x<z))
                         min = y;

                         {
                         }
                                        }
          printf ("le min est : %d,\n" ,min);
Return 0;}
```

Méthode 2 pour le calcul de min des trois nombres :

```
#include <stdio.h>
int main() {
int x, y, z;
      printf("introduire les trois variables \n");
      scanf (" %d ", &x);
      scanf (" %d ", &y);
      scanf (" %d ", &z);
      printf ("les valeur de x y et z sont %d %d %d \n", x, y, z);
      int min;
if (x<y)
     { if (x<z)
      { min = x;}
     else{
     if (y<z)
     {min =  y;
       }else min = z;} }
     else  {
     if (y<z) min = y;
     else min =z;}
  printf ("le min est %d ", min)   ;
  return 0;     }
```

Les structures conditionnelles de choix multiple

Syntaxe en algorithmique

```
Selon <expresion>;
Debut
Cas valeur1 : <séquence d'instruction1>;
Cas valeur2 : < séquence d'instruction2>;
…
Cas valeurN : < séquence d'instructionN>;
defaut              : <séquence par defaut>;
```

Les structures conditionnelles de choix multiple

Syntaxe en C :

```
Switch <variable de choix>
{
Case choix1 : <sequence 1>;[break];
…
Case choixN : <sequence N>;[break];
[default] : <sequence par defaux>;
}
```

Les structures conditionnelles de choix multiple selon (switch en c)

cette structure de contrôle présente plusieurs cas d'exécution du traitement. permet d'effectuer une suite de tests d'égalité consécutifs pour une valeur donnée.

Principe :

1. A la base cette structure de contrôle présente plusieurs cas d'exécution du traitement, mais un seul sera exécuté. En premier lieu, la variable de choix est évaluée
   - Si sa valeur est égale à un cas i, la séquence i est exécutée.
   - Si la valeur de la variable de cas n'est pas égale à aucun cas, alors la séquence par défaut est exécutée.

   L'action ***defaut*** (***en c default***) est optionnelle (facultative), s'exécute dans le cas où la variable de choix ne prend aucune des valeurs.
2. En c, dans le cas où on veut exécuter une seul séquence, on met l'instruction break à la fin de chaque séquence.
3. La valeur de choix doit être de type ***entier positif*** ou de type ***caractère.***

- **Exemple:**

```
Algorithm choix_mult;
Var j : entier;
debut
Lire (j)
Selon i
debut
cas 1:ecrire (‘ c’est samedi ’);// on peut trouver choix <valeur> au lieu
                //de cas
Cas 2: ecrire (‘c’est dimanche’);
Cas 3: ecrire (‘c’est lundi’);
Cas 4: ecrire (‘c’est mardi’);
Cas 5: ecrire (‘c’est mercredi’);
Cas 6: ecrire (‘c’est jeudi’);
Cas 7: ecrire (‘c’est vendredi’);
defaut : ecrire (‘autre chose’);
fin
Fin;
```

Interprétation de l’exemple précèdent en c :

```
#include <stdio.h>
int main(){

    int j ;
    scanf("%d", &j);
    switch (j)
    {
        case 1 : printf("c'est samedi \n"); break ;
        case 2 : printf("c'est dimanche \n");break;
        case 3 : printf("c'est lundi \n");break ;
        case 4 : printf("c'est mardi \n");break ;
        case 5 : printf("c'est mercredi \n");break;
        case 6 : printf("c'est jeudi \n"); break ;
        case 7 : printf("c'est vendredi \n"); break ;
        default: printf("autre chose \n'");
            }
    printf("la valeur de j est %d \n", j);
return 0;
```

Branchement

- L’instruction de branchement permet de sauter à un endroit précis du programme. Cette endroit est repéré par une étiquette.
- Pour pouvoir sauter à une instruction, if faux utiliser une ***étiquette*** pour désigner cette instruction. **En c** l’instruction ***goto*** permet d’effectuer un saut jusqu’à l’instruction étiquette correspondante.
- **Il y a deux type de branchement :**
  - **Le branchement conditionnel :** c’est un saut soumis à une condition. Dans ce cas, cette instruction fait partie du bloc sinon
  - **Le branchement inconditionnel :** c’est un saut sans condition et qui ne fait pas partit du bloc du si ou du sinon.

Exemple d'une instruction de branchement conditionnelle

en Algorithmique

```
Algorithme branchement
Var a : entier;
    debut
    lire (a);
    si (a<0) alors
        aller a etiq;
    finsi;
    ecrire ('positif ou nul');
Etiq : ecrire ('negatif');
Fin.
```

En c :

```
#include <stdio.h>
int main (){
Int a;
Print(« donner la valeur de a \n »);
scanf(« %d »,&a);
If (a< 0) goto etiq ;
printf(positif ou nul); break;
Etiq : printf(« negatif »);
}
```

Exemple d'instruction de branchement non conditionnelle

En algorithmique

```
Algorithme branchnocond;
Var x : entier;
debut
lire (x);
aller a etiq;
ecrire (x+1);
etiq : ecrire (x-5);
N.B. etiq est une valeur entière
naturelle.
```

En langage de programmation c

```
#include <stdio.h>
int main () {
int x;
scanf(« %d »,x);
goto etiq;
printf ( x+1 );
etiq : printf (x-5);
```

- **N’hésitez pas à poser ou envoyer vos questions**

- **RDV au cours suivant**