STRUCTURES DE CONTRÔLES

STRUCTURES DE CONTRÔLES

Un ordinateur exécute un programme de manière séquentielle. Pour lui doter de l'intelligence relative afin d'être capable d'effectuer des choix ou des boucles sur une bloc d'instructions et de casser cette linéarité, il va falloir utiliser les structures de contrôle.

Parmi les structures de contrôle nous avons :

- LES STRUCTURES CONDITIONNELLES
- LES STRUCTURES ITERATIVES

LES STRUCTURES CONDITIONNELLES

INSTRUCTION CONDITIONNELLE

Syntaxe en Algorithme

```
Si(condition) alors
        {instructions}
Fin Si
```

Syntaxe en C/C++

```
if(condition) {
        /*instructions*/
}
```

Condition est une expression booléenne

Exemple:

```
…..

Si(jour <> 7) alors
   ecrire(''Je vais à l'école'');
Fin si
….
```

```
…..

if(jour != 7) {
   cout <<''Je vais à l'école''<<endl;
}
….
```

LES STRUCTURES CONDITIONNELLES

INSTRUCTION CONDITIONNELLE

Syntaxe en Algorithme

```
Si(condition) alors
        {instructions}
Sinon
        {instructions}
Fin Si
```

Syntaxe en C/C++

```
if(condition) {
        /*instructions*/
} else {
        /*instructions*/
}
```

Exemple:

```
…..

Si(jour <> 7 ET greve=Faux) alors
   ecrire(''Je vais à l'école'')
Sinon
   ecrire(''Il n'y a pas école'')
Fin si
….
```

```
…..

if(jour != 7 && greve==Faux) {
   cout <<''Je vais à l'école''<<endl;
} else {
  cout <<''Il n'y a pas école''<<endl;
}
….
```

LES STRUCTURES CONDITIONNELLES

INSTRUCTION CONDITIONNELLE - IF IMBRIQUÉES

Syntaxe en Algorithme

```
Si(condition 1) alors
        {instructions}
Sinon Si(condition 2) alors
        {instructions}

....

Sinon Si(condition n) alors
        {instructions}
Sinon
        {instructions}
Fin Si
```

Exemple:

```
…
Si(jour=1) alors
  Ecrire(''Lundi'')
Sinon Si(jour=2) alors
  Ecrire(''Mardi'')
Sinon Si(jour=3) alors
  Ecrire(''Mercredi'')
Sinon Si(jour=4) alors
  Ecrire(''Jeudi'')
Sinon Si(jour=5) alors
  Ecrire(''Vendredi'')
Sinon Si(jour=6) alors
  Ecrire('Samedi'')
Sinon
  Ecrire(''Dimanche'')
Fin Si
…
```

STRUCTURE A CHOIX MULTIPLE

Elle permet dans certain cas d'éviter une abondance d'instruction if imbriquées.

Syntaxe en Algorithme

```
Choix selon(expression)
        cas valeur1
          {instruction 1}
        interrompre
        cas valeur2
          {instruction 2}
        interrompre
                ...
        par défaut
          {suite_instruction}
Fin cas
```

Exemple:

```
...
  Choix selon(jour)
        cas 1
          Ecrire(''Lundi'')
         interrompre
        cas 2
          Ecrire(''Mardi'')
         interrompre
        cas 3
          Ecrire(''Mercredi'')
         interrompre
        cas 4
          Ecrire(''Jeudi'')
         interrompre
        cas 5
          Ecrire(''Vendredi'')
        interrompre
        cas 6
          Ecrire('Samedi'')
         interrompre
         par défaut
          Ecrire(''Dimanche'')
  Fin cas
...
```

STRUCTURE A CHOIX MULTIPLE

Syntaxe en C/C++

```
switch (expression)
{
        case valeur1:
          {instruction 1};
        break;
        case valeur2:
          {instruction 2};
        break;
                …
        default:
          {suite_instruction} ;
}
```

Exemple:

```
…
   switch(jour)
    {
            case 1:
              cout <<''Lundi''<<endl;
             break;
            case 2:
              cout <<''Mardi''<<endl;
             break;
            case 3:
              cout <<''Mercredi''<<endl;
             break;
            case 4:
              cout <<''Jeudi''<<endl;
             break;
            case 5:
              cout <<''Vendredi''<<;
            break;
            case 6:
              cout <<''Samedi''<<endl;
             break;
             default:
              cout <<''Dimanche''<<endl;
   }
…
```

EXERCICES D'APPLICATIONS

Application 6 :

Écrivez un programme qui calcule les solutions réelles d'une équation du second degré $ax^2+bx+c = 0$ en discutant la formule:

- Utilisez une variable d'aide d pour la valeur du discriminant b^2 - 4*a*c et décidez à l'aide de d, si l'équation a une, deux ou aucune solution réelle. Utilisez des variables du type entier pour a, b et c. Affichez les résultats et les messages nécessaires sur l'écran.

Application 7 :

Écrivez un programme qui permet de calculer la superficie d'un cercle, d'un rectangle ou d'un triangle. L'utilisateur saisira "C", "R" ou "T" selon la superficie de la figure qu'il souhaite calculer, ensuite il saisira les dimensions.

Selon le choix de l'utilisateur, le programme doit pouvoir lui demander de saisir les dimensions appropriées.

Afficher ensuite à l'écran selon son choix la superficie demandée

STRUCTURES ITERATIVES

Une itération consiste en la répétition d'un blocs d'instructions jusqu'à ce qu'une certaine condition soit vérifiée.

Il en existe 2 sortes:

- Le nombre d'itérations est connu d'avance
- Le nombre d'itération dépend du résultat précédemment obtenue.

Supposons qu'on veut afficher tous les nombres entiers comprises entre 9 et 999. il va falloir faire:

```
Ecrire(''9'')
Ecrire(''10'')
...
Ecrire(''999'')
```

Une tache répétitive fastidieuse. D'où la nécessité de trouver une solution alternative.

STRUCTURES ITERATIVES
ITERATION POUR

Syntaxe en Algorithme

```
Pour i allant de MIN à MAX par pas de PAS faire
        {instructions}
Fin pour
```

Syntaxe en C/C++

```
for (initialisation ; condition ; incrémentation){
        /*instructions*/
 }
```

BOUCLE AVEC COMPTEUR

Exemple:

```
…..
Pour i allant de 9 à 999 faire
   ecrire(''i='', i)
Fin pour
….
```

```
…..
int compteur;
for (compteur = 9; compteur <1000 ; compteur++)
{
        cout << compteur << endl;
}
….
```

STRUCTURES ITERATIVES

ITERATION FAIRE…TANT QUE

Syntaxe en Algorithme

```
Faire
        {instructions}
Tant que (condition_de_reprise)
```

```
Répéter
        {instructions}
Jusqu'à(condition_de_sortie)
```

Syntaxe en C/C++

```
do{
        /*instructions*/
}while(condition_de_reprise);
```

le contenu de la boucle sera toujours lu au moins une fois.

Exemple:

```
…..
Faire
   Ecrire(''veuillez entrer un entier ?'')
   Lire(nombre)
Tant que (nombre<0)
….
Répéter
   Ecrire(''veuillez entrer un entier ?'')
   Lire(nombre)
Jusqu'à (nombre>0)
….
```

```
….
int nombre(0);
do
{
   cout << ''veuillez entrer un entier ?'' << endl;
   cin >> nombre;
} while (nombre< 0);
…
```

STRUCTURES ITERATIVES

ITERATION TANT QUE … FAIRE

Syntaxe en Algorithme

```
Tant que (condition) faire
        {instructions}
Fin tant que
```

Syntaxe en C/C++

```
while(condition ){
        /*instructions*/
 }
```

La condition d'entrée doit être définie au préalable sinon, en implémentant votre algorithme, vous risquez d'avoir des comportements étranges.

Exemple:

```
…..
Ecrire(''Entrez un entier naturel'')
Lire(n)
result ← 0;
i←1;
Tant que(i<=n) faire
        result ← result + i;
        i ← i+1;
Fin tant que
Ecrire(''Somme ='', result)
….
```

```
…..
int result(0), i(1), n;
cout << ''Entrez un entier naturel ?'' << endl;
cin >> n;
while(i<=n)
{
   result = result + i;
   i = i+1;
}
cout << ''Somme ='' << result << endl;
….
```

EXERCICES D'APPLICATIONS

Application 8 :

Écrivez un programme qui calcule les solutions réelles d'une équation du second degré $ax^2+bx+c = 0$ en discutant la formule:

- Utilisez une variable d'aide d pour la valeur du discriminant $b^2 - 4*a*c$ et décidez à l'aide de d, si l'équation a une, deux ou aucune solution réelle. Utilisez des variables du type entier pour a, b et c. On suppose que les valeurs saisies sont non nulles. Affichez les résultats et les messages nécessaires sur l'écran

Application 9:

Ecrire un programme qui permet de faire les opérations suivantes :

- Ecrire un programme qui affiche la somme des n premiers entiers naturels. La valeur de n est saisie au clavier lors de l'exécution.
- Ecrire un programme qui affiche la somme des entiers compris entre les entiers d et f. Les valeurs de d et f sont saisies au clavier lors de l'exécution.
- Ecrire un programme qui affiche la somme des valeurs absolues des entiers compris entre les entiers relatifs d et f. Les valeurs de d et f sont saisies au clavier lors de l'exécution.