Chapitre 2 : Les structures de contrôles

1. Introduction
2. Les S.C. Conditionnelles

1. Les S.C. Répétitives (boucles)

En programmation, l’ordre des instructions est primordiale, le processeur les exécutent dans l’ordre dans lesquels elles apparaissent : on dit que l’exécution est séquentielle.

Parfois, il est nécessaire que le processeur n’exécute pas toutes les instructions ou encore il recommence plusieurs fois les mêmes instructions. Pour cela il faudra casser la séquence. C’est le rôle des structures de contrôles.

Il existe deux grands types de structure de contrôles :

* Les S.C. conditionnelle (if)
* Les S.C. répétitives (boucles)

II) Les S.C. Conditionnelles

* Les S.C.C. permettent d’exécuter des instructions en fonction de certaines conditions.
* Une condition (ou expressions conditionnelles ou expression booléenne) est évaluée (jugée vraie ou fausse). Si elle est vraie, un traitement est réalisé sinon, un autre traitement sera exécuté.
* Il existe 2 types de S.C.C. stricte

1. S.C.C. Stricte

Syntaxe :

If(<condition>){ 🡸 début du bloc

//Traitement 🡸bloc d’instruction

} 🡸fin du bloc

Exemple :

If(nb>0){

Cout << « Le nombre est positif » ;

}

Explication : Si le nombre est positif alors afficher le message « Le nombre est positif »

1. S.C.C. alternative

Syntaxe :

If(<condition){

//Traitement 1

}else{

//Traitement2

}

Exemple :

If(nb>0){

cout << « Le nombre est positif » ;

}else{

cout<< « Le nombre est négatif  ou nul» ;

}

III)Les S.C. Répétitives (boucles)

* Les SCR (boucle) permettent de répéter un traité autant de fois qu’il est nécessaire, soit :

- un nombre défini de fois

- tant qu’une condition est vraie

* Il existe 3 grands types de boucles :

1. La boucle « while » tant que

* Elle permet de répéter un traitement tant qu’une condition est vraie.

Syntaxe :

while(<condition){

//Traitement à répéter

}

/ ! \ Si d’emblée, la condition n’est pas vraie, le traitement ne sera jamais exécuté.

Exemple :

Ecrire un programme qui calcule le cube d’un nombre saisi par l’utilisateur du programme s’arrête une fois que l’utilisateur choisi 0 comme valeur.

#include ….

…….

Int main(…){

//variables

Int nb, resultat ;

//saisie

Cout<< »Entrez un nombre (0 pour quitter !) : » ;

Cin>>nb ;

While(nb != 0){

Resultat=nb+nb ;

Cout<< « Le cube de » <<nb<< « est : »<<resultat<<endl ;

//Nouvelle saisie

Cout<< « Entrez un nouveau nombre (0 pour quitter) : » ;

Cin<<nb ;

}

Cout<< « Terminé ! »<<endl ;

}

2. La boucle for (pour)

🡺 Permet de répéter un traité un nombre de fois connu d’avance.

Syntaxe :

for(i=<valeur\_initiale> ; i<=<valeur\_finale> ; i++){

}

For(i=<valeur\_finale> ;i>=<val-initiale> ;i--){

}

Exemple :

for(i=1 ; i<=10 ; i++) 🡸 réalisé 10 fois

Type d’écriture :

i++ ; => i=i+1

i+=< val> ; => i=i+<val> ;

++i;

--i;

i--;

i/;

i\*;

i-;

Exemple en programmation :

For( 1; 2; 3){ 🡸 1 : instruction(s) a éxecuter avant le démarrage de la boucle. => 1 fois !

2 : Condition à vérifier avant chaque passage de la boucle => N fois

3 : Instruction(s) à éxecuter après chaque passage de la boucle => N fois

//Traitement à répéter

}

Différence avec la boucle while :

1

while(2){

//Traitement a répéter

3

}

Exemple :

For(i=1,j=20 ;i<9&&j>4 ;j=j-4,i=j+2){

}

For(i=1; i<=10;i++){

A l’écran => i=1

i=2

i=3

i=10

Cout<<”i=”<<i<<endl;

}

/ !\ La boucle for se base sur une variable (expl : i) jouant le rôle d’un compteur. Cette varialbe doit être obligatoirement déclarée en type int

3. La boucle do…while

🡺idem que « while «

🡺les différences sont :

* La vérification de la condition de répétition se fait après chaque passage
* Le traits a répéter est éxécuté au moins une fois

Syntaxe :

Do{

//Traitement a répéter

}while(<condition>) ;

IV Expressions conditionnelles

* Ou expr booléennes ou logique est une expression dont la valeur est soit vrai ou faux

1. Condition simple

* Est une simple comparaison de 2 expressions de même type.

Syntaxe : <expr1><symbole><expr2>

Les symbole de comparaison sont <,>,==,<=,<=, !=

1. Condition complexe

C’est une combinaison de plusieurs conditions simple ou variables booléenne reliées entre elles par des opérateurs logiques

Et => && ou and

Ou =>|| ou or | <=(altG+G)

Non => !

If((a<b)&&(c<3)){

}