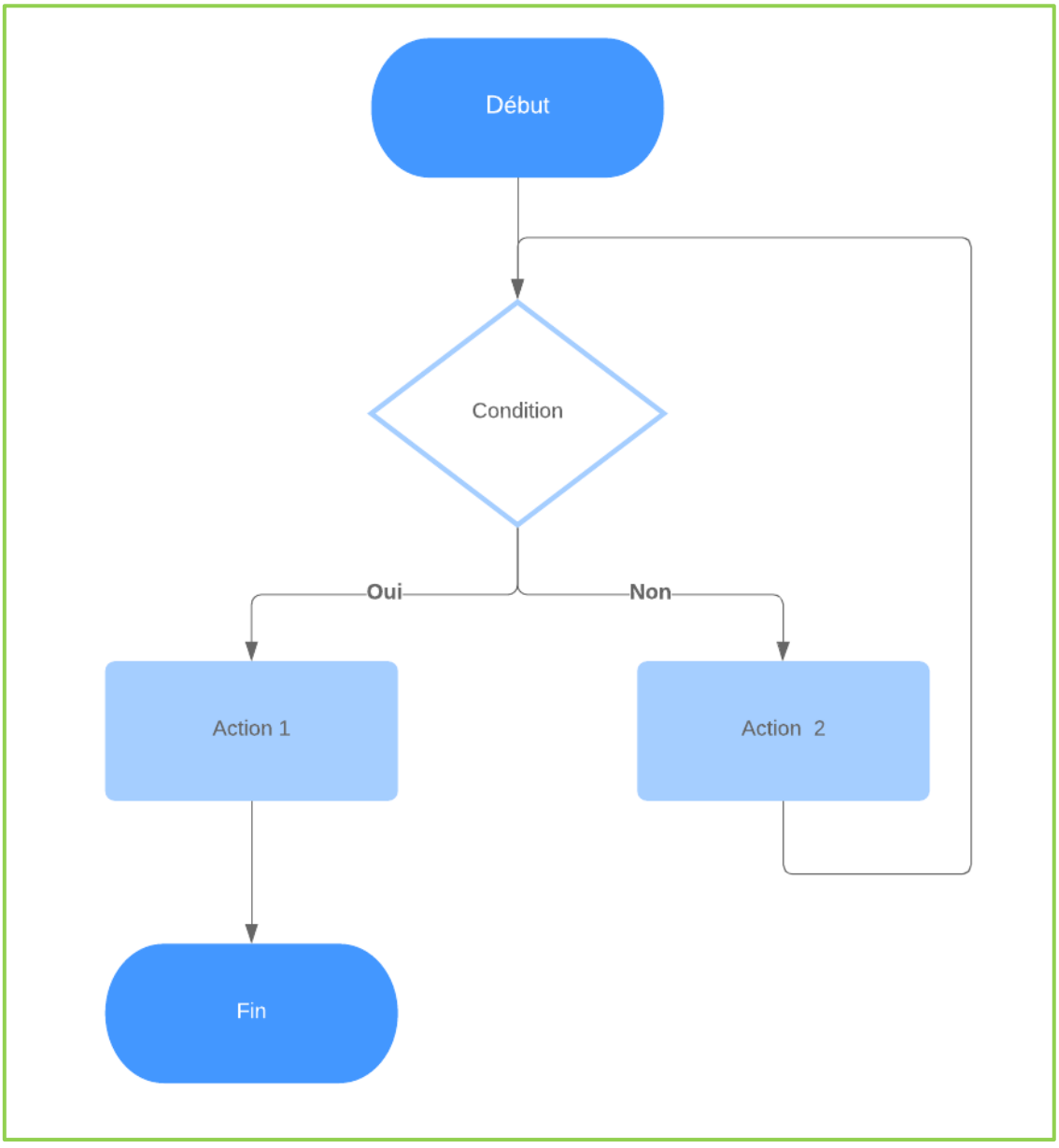
CM – Programmation C / C++

Chapitre 3 : Algorigramme



I- Principe d'un algorigramme :

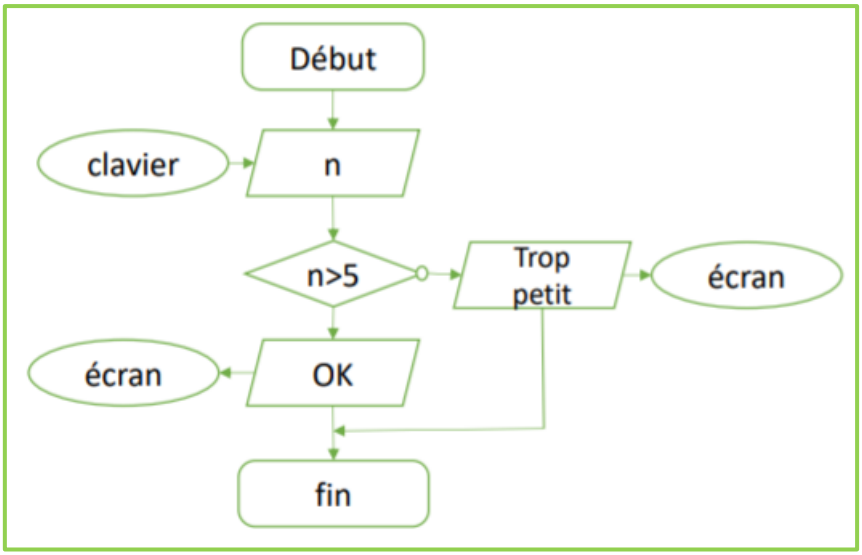
1) *Définition d'un algorigramme :*

Un algorigramme est un schéma permettant de détailler les actions que réalisera un programme. Il se parcourt du haut vers le bas et de gauche à droite. Il est composé de formes géométriques qui représentent des actions, reliées par des flèches qui symbolisent le déroulé du programme (les différentes flèches ne doivent pas se croiser pour une meilleure lisibilité).

Exemple d'un programme et de son algorigramme :

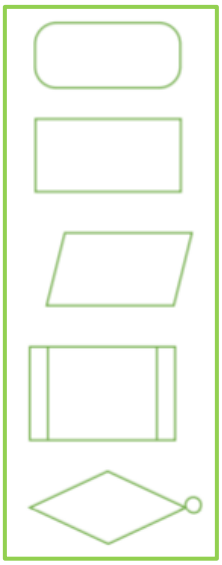
```
#include <iostream>

using namespace std;


int main()
{
    int n;
    cin >> n;

    if(n>5)
        cout << "Ok" << endl;
    else cout << "Trop petit" << endl;

    return 0;
}
```

Le programme permet d'entrer une valeur entière dans une variable et renvoie un message positif si elle est inférieure à 5 ou renvoie un message négatif si elle est supérieure ou égale à 5. L'algorigramme qui lui correspond est le suivant :



II- Principaux symboles :

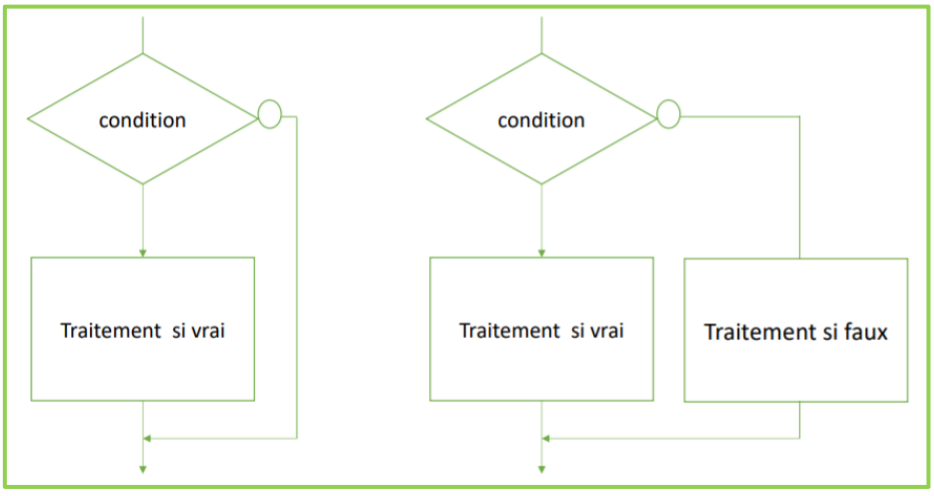
Un algorigramme utilise cinq symboles différents :



- Début et fin du programme (avec « Début » ou « Fin » écrit).
- Symbole général d'une action (écrite comme dans le code).
- Entrée / Sortie (précisé par une ellipse indiquant le flux à côté).
- Sous-programme.
- Test logique. Le rond indique la sortie `false`.

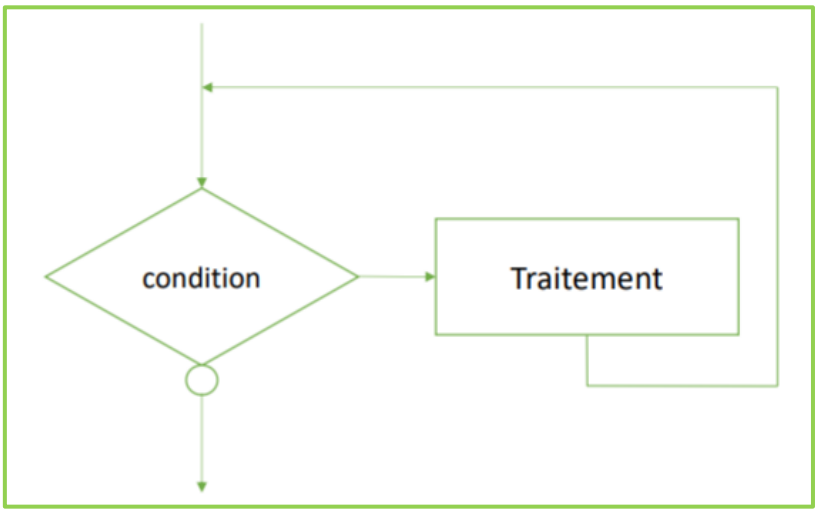
III- Principales séquences d'algorigramme :

1) Séquence « if » :



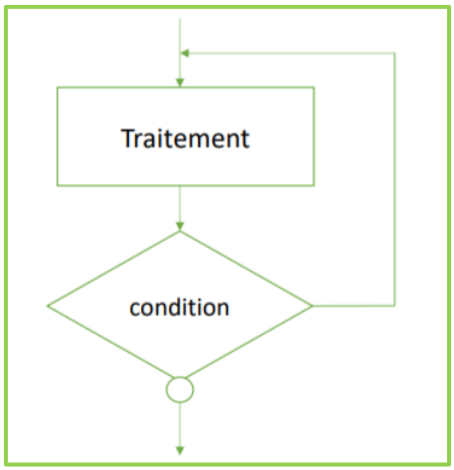
L'algorigramme de la séquence `if` est légèrement différent si il possède un `else` (schéma de droite) ou non (schéma de gauche). Elle démarre par un test logique qui contient la condition à respecter. Si il renvoie `true`, on va à l'action à réaliser. Si il renvoie `false`, l'action qui suit n'est pas réalisée sauf si il y a un `else`, ce qui mène à une autre action à réaliser.

2) Séquence « while » :



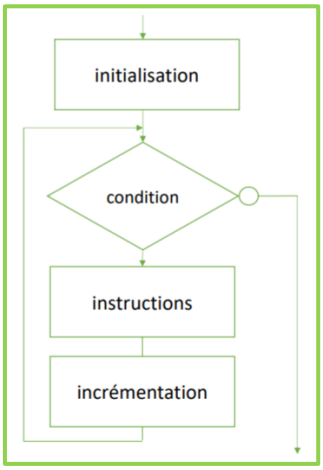
L'algorigramme de la séquence **`while`** commence par un test logique qui contient la condition à respecter de la boucle. Si la condition est respectée (renvoie **`true`**), on va à l'action à réaliser puis on retourne à la condition. Si la condition n'est pas respectée (renvoie **`false`**), on sort de la boucle.

3) Séquence « do while » :



L'algorigramme de la séquence **`do while`** possède les mêmes symboles que la séquence **`while`** excepté que l'on commence par l'action à réaliser de la boucle suivie du test logique contenant la condition. Le fonctionnement devient alors identique à la boucle **`while`**.

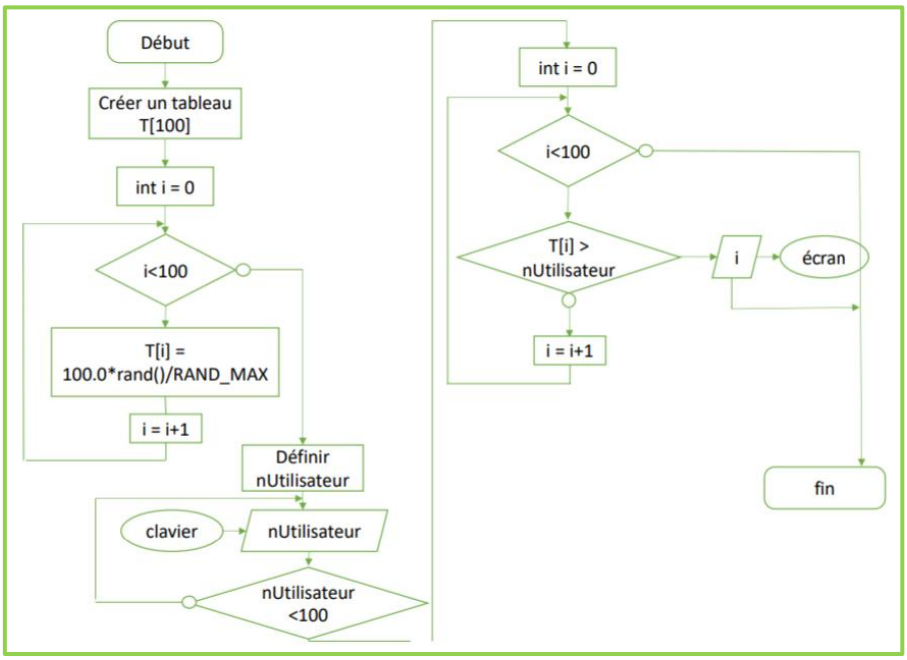
4) Séquence « for » :



L'algorithme de la séquence **`for`** commence par l'initialisation de la boucle avec une variable. On suit avec un test logique qui contient la condition de la boucle. Si la condition est respectée (renvoie **`true`**), on passer sur les différentes actions qui composent les instructions de la boucle, puis on suit par l'expression de la boucle (ici, c'est une incrémentation car on initialise souvent avec une variable entière). On retourne après à la condition. Si la condition n'est pas respectée (renvoie **`false`**), on sort de la boucle.

Exemple – Création de l'algorigramme d'un programme :

On souhaite créer un programme qui permet de construire un tableau de 100 entiers dont chaque case est initialisée avec un nombre entier aléatoire entre 0 et 100. On demande à l'utilisateur d'entrer un nombre entier tel que ce nombre soit inférieur à 100, puis on cherche le premier nombre supérieur à celui-ci dans le tableau et on affiche sa position à l'intérieur. Un algorigramme de ce programme est le suivant :



On débute l'algorigramme en créant le tableau de 100 entiers, puis on initialise une variable entière à 0 pour une boucle **for** qui, si la variable est inférieure à 100, on affecte à la case du tableau une valeur aléatoire entre 0 et 100. On incrémente ensuite la variable entière qui initialise la boucle et on continue tant que la condition est respectée. Dès que la boucle se finie, on déclare une nouvelle variable entière à laquelle l'utilisateur affecte une valeur par le clavier, puis on met la condition de la boucle **do while** qui continue si la valeur entrée est supérieure à 100. Si la condition n'est pas respectée, on initialise une nouvelle variable pour une deuxième boucle **for** qui mène à la condition de la boucle qui est identique à la première. Dès que la valeur de la case du tableau est supérieure à la valeur entrée par l'utilisateur (test logique), on affiche la position de ce nombre dans le tableau et on arrête le programme, sinon on recommence au début de la boucle.

Poser l'algorigramme permet de mieux se rendre compte de comment le programme va fonctionner et comment il devra être écrit. Le code du programme peut ressembler à ceci :

```
#include <iostream>
#include <cstdlib>

using namespace std;


int main()
{
    double T[100];

    for(int i=0;i<100;i++)
        T[i]=100.0*rand()/RAND_MAX;

    double nUtilisateur;
    do {
        cout << "Entrer un nombre < 100" << endl;
        cin >> nUtilisateur;
    } while(nUtilisateur>100);

    for(int i=0;i<100;i++)
        if(T[i]>nUtilisateur)
        {
            cout << "Le premier nb > " << nUtilisateur << "est a la position " << i << endl;
            break;
        }

    return 0;
}
```