|  |  |  |
| --- | --- | --- |
|  | **INSTITUT SUPERIEUR DES ETUDES TECHNOLOGIQUES DE NABEUL** | |
|  | Travaux Pratiques N° 3 : LES STRUCTURES REPETITIVES | |
| **Classe : TI 1** | | **Enseignante : Sana REFAI** |

## 1. Objectifs

Le but de ce TP est d'apprendre à utiliser les différentes structures conditionnelles dans un programme C.

## 2. La structure répétitive While

Il s'agit de l'instruction:

**Tant que (expression vraie) Faire**

**< BLOC D'INSTRUCTIONS >**

**Fin TantQue**



Syntaxe:

**while (<expression>)**

**<bloc d'instructions>**

* **<expression>** peut désigner :
* Une variable numérique.
* Une expression fournissant un résultat numérique.
* **<bloc d’instructions>** :
* Une suite d’instructions comprise entre {}.
* Une seule instruction.
* Tant que l ’**<expression>** donne une valeur différente de 0 le **<bloc d’instructions>** est

exécuté.

* Si l ’**<expression>** donne 0, l’exécution continue avec l’instruction qui suit le bloc.
* Le **<bloc d’instructions>** est exécuté 0 ou n fois.

**/\* Afficher 10 fois ” Bonjour ”  \*/**

Exemple:

**int cpt = 0;**

**while (cpt < 10)**

**{**

**printf (” Bonjour\n”);**

**cpt ++;**

**}**

**Autre possibilité :**

Exemple:

**/\* Afficher 10 fois ” Bonjour ”  \*/**

**int cpt = 1;**

**while (cpt <= 10)**

**{**

**printf (” Bonjour\n”);**

**cpt ++;**

**}**

## 3. LA STRUCTURE REPETITIVE DO WHILE

Il s'agit de l'instruction:

**Repeter**

**< BLOC D'INSTRUCTIONS >**

**TantQue (expression vraie)**



Syntaxe:

**do**

**<bloc d'instructions>**

**while (<expression>);**

* Le **<bloc d ’instructions>** est exécuté **au moins une fois** et aussi longtemps que l ’**<expression>** fournit une valeur différente de 0.
* La structure **do while** est comme la structure **while** avec la différence :
* **while** évalue la condition **avant** d’exécuter le bloc d’instructions.
* **do while** évalue la condition **après** avoir exécuté le bloc d’instructions.
* En pratique, la structure **do - while** n'est pas si fréquente que **while**; mais dans certains cas, elle fournit une solution plus élégante.

**#include <stdio.h>**

Exemple:

**#include <conio.h>**

**#include <math.h>**

**/\* Calcul de la racine carrée \*/**

**int main ( )**

**{**

**float Nbre ;**

**do**

**{**

**printf (”Entrez un entier positif : ”);**

**scanf (”%d”, &Nbre);**

**} while (Nbre < 0);**

**printf (”La racine carrée de %.2f est %.2f\n ”,Nbre, sqrt(Nbre));**

**getch();**

**}**

## 4. La structure répétitive for

Il s'agit de l'instruction:

**Pour i =1 à n Faire**

**< BLOC D'INSTRUCTIONS >**

**FinPour**



Syntaxe:

**for (<expr1>; <expr2>; <expr3>)**

**<bloc d’instructions>**

* L’instruction **for** est équivalent à :

**<expr1>**

**while (<expr2>){**

**<bloc d’instructions> <expr3>;**

**}**

* **<expr1>** est évaluée une fois avant le passage de la boucle (pour initialiser les données de la boucle).
* **<expr2>** est évaluée avant chaque passage de la boucle (pour décider si la boucle est répétée ou non).
* **<expr3>** est évaluée à la fin de chaque passage de la boucle (pour réinitialiser les données de la boucle)

***Remarque******:***

* **<expr1>** et **<expr3>** peuvent contenir plusieurs initialisations ou réinitialisations séparées par des virgules.
* Les 3 expressions **<expr1>**, **<expr2>** et **<expr3>** ne portent pas forcément sur la même variable.
* Les 3 expressions peuvent être omises, mais pas les ‘**;’**.

**...**

**int n, tot;**

**for (n=1, tot = 0; (n < 11); n++)**

**tot += n;**

**printf (“La somme des nombres de 1 à 10 est %d\n“, tot);**

**...**

## 

## 5. CHOIX DE LA STRUCTURE REPETITIVE

Choisir la structure qui reflète le mieux l’idée du programme :

* Si le bloc d’instructions ne doit pas être exécuté si la condition est fausse, alors utilisez **while** ou **for.**
* Si le bloc d’instructions doit être exécuté au moins une fois, alors utilisez **do while**
* Si le nombre d’exécutions du bloc d’instructions dépend d’une ou plusieurs variables qui sont modifiées à la fin de chaque répétition, alors utilisez **for.**
* Si le bloc d’instructions doit être exécuté aussi longtemps qu’une condition extérieure est vraie, alors utilisez **while.**

## 6. Les Exercices

### 6.1 Exercice 1

Ecrire un programme C qui permet de saisir un entier positif (en utilisant « while ») puis (en utilisant « do …while »)

### 6.2 Exercice 2

Ecrire un programme C qui permet de répéter l’affichage de la question « Voulez vous Continuez ? » jusqu’à ce que la réponse soit la lettre o ou O pour oui et la lettre n ou N pour non, il s’arrêtera. ( en utilisant « while ») puis (en utilisant « do …while »)

### 6.3 Exercice 3

Ecrire un programme C qui permet de calculer le factoriel d’un entier n saisi au clavier.

### 6.4 Exercice 4

Ecrire un programme C qui permet d’afficher les nombres pairs compris entre 0 et 100

### 6.5 Exercice 5

Ecrire un programme C qui permet d’afficher les nombres premiers inferieurs à 100

### 6.6 Exercice 6

1. Afficher le triangle isocèle suivant formé de plusieurs lignes en utilisant les instructions de contrôle. Exemple d’exécution:

Donner le nombre de lignes : 6

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

2. Modifier le programme pour qu’il affiche ce triangle dans le sens inverse :

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

### 6.7 Exercice 7

Écrire un programme C qui permet de déterminer si un entier naturel donné par l'utilisateur est un nombre parfait. On rappelle qu'un nombre est parfait s'il est égal à la somme de ses diviseurs stricts (sauf lui-même). Ainsi par exemple, l'entier 6 est parfait car 6 = 1 + 2 + 3.

### 6.8 Exercice 8

Écrire un programme C qui lit deux entiers a et b puis calcule et afficha leur PGCD( Plus grand Commun Diviseur) en utilisant la méthode suivante :

-si a= b ; PGCD(a,b) = a

-si a > b ; PGCD(a,b) = PGCD(a-b,b)

-si b > a ; PGCD(a,b) = PGCD(a,b-a)

Exemple :

PGCD(18,45)=PGCD(18,27)=PGCD(18,9)=PGCD(9,9)=9