

# TP1 C++

## OBJECTIFS

- Premiers pas en C++

## PRE-REQUIS

- Aucun

## DUREE

2-3 heures

## RESSOURCES

- QtCreator

## Premiers pas : le classique « Hello World »

### 1 CREATION D'UN PROJET QTCREATOR

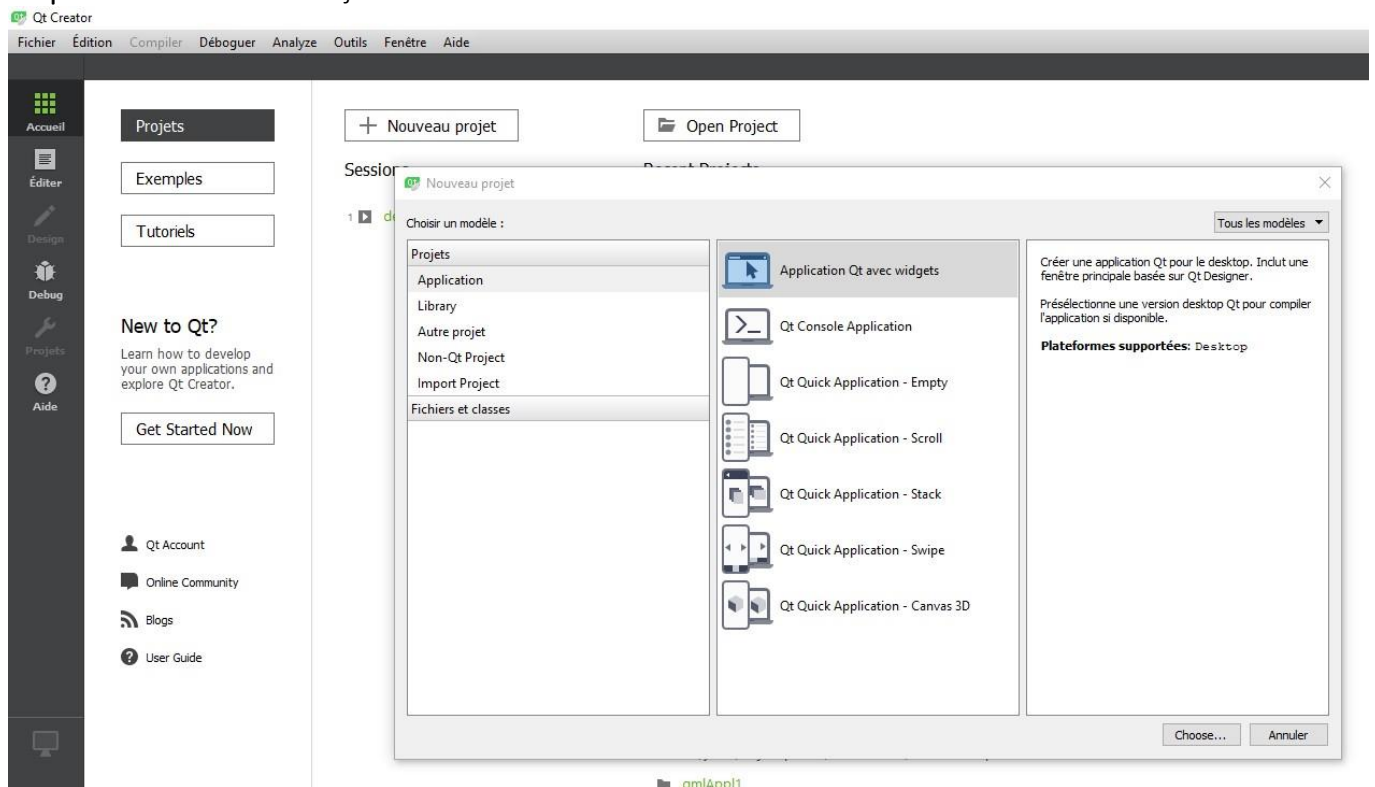
Le code source sera écrit à l'aide d'un éditeur de texte, **notepad++** par exemple (dans ce cas il faudra procéder à la compilation séparément) ou un IDE (Interface de développement) comme **QtCreator** (dans ce cas les étapes de compilation seront automatisées).

Le fichier source doit être sauvegardé avec obligatoirement l'extension **.cpp** pour C++

Lancer l'IDE sous Windows : **C:\Qt\Qt5.8.0\Tools\QtCreator\bin\qtcreator.exe**

Attention : votre version n'est peut-être pas la 5.8.0 ! Pour ce qu'on a à faire, n'importe quelle version au-delà de la version 4 suffit amplement !

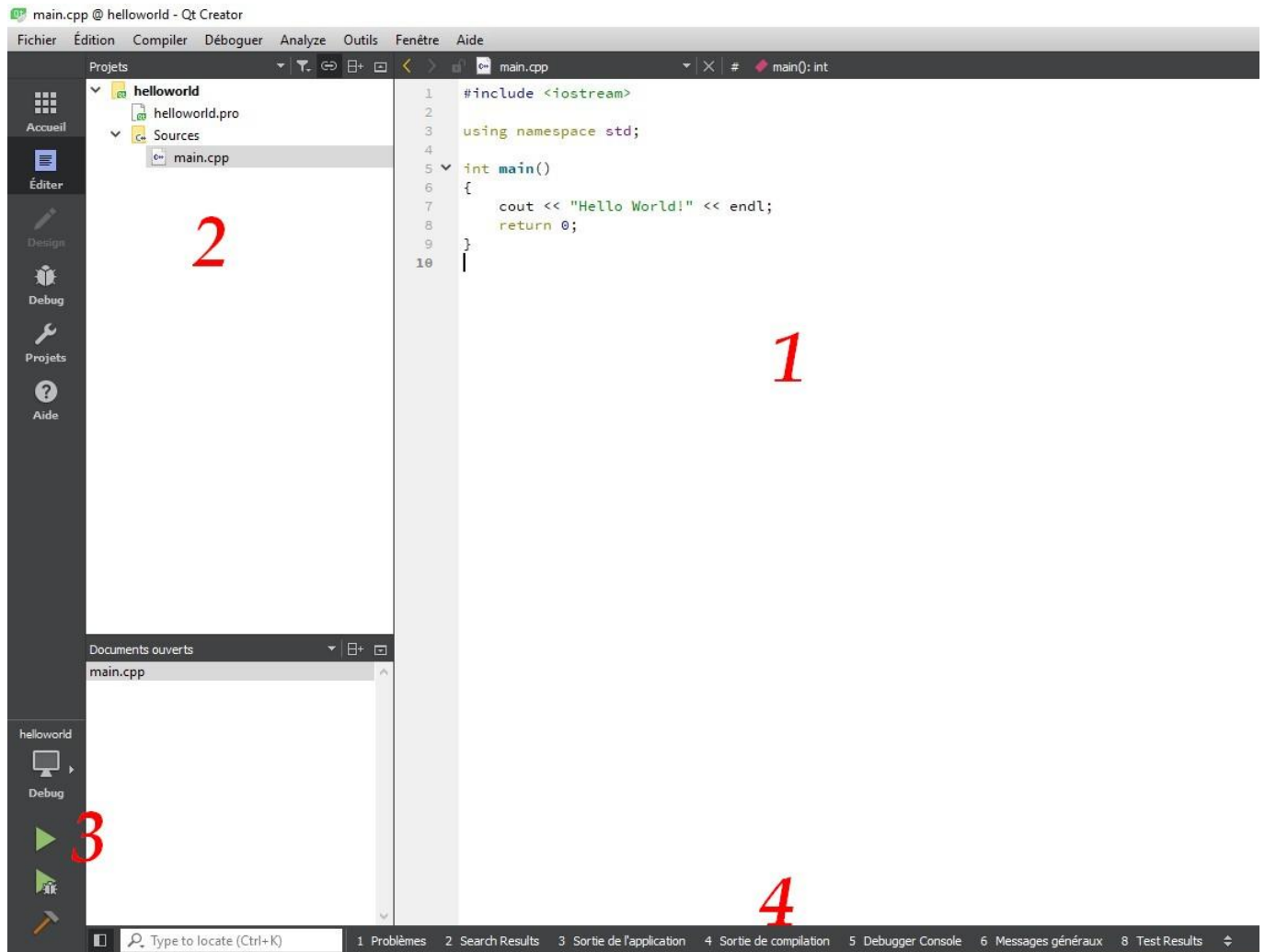
Cliquer sur « Nouveau Projet »



Choisir « Non-Qt Project » et « Plain C++ Application »

Donner un nom à votre projet : par exemple « helloworld », puis **cliquer** sur suivant/suivant/terminer

Vous devriez obtenir :



1 : Zone d'édition du code source

2 : Arborescence du projet => fichiers et répertoires du projet *helloworld*

3 : Boutons d'exécution normale et en pas à pas

4 : Onglets donnant des renseignements sur le fonctionnement du programme : compilation sans erreurs, exécution du programme etc..

## 2 ECRITURE DU CODE SOURCE

Normalement QtCreator a fourni un fichier *main.cpp* avec un contenu déjà exploitable.

Voici le source :

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Remarque : il peut y avoir autant de sauts de ligne que l'on veut, ça ne change en rien le programme.

### 3 COMPILATION ET EXECUTION

Cliquer sur le bouton « Execute » (triangle vert). Si le programme n'a jamais été produit (production d'un exécutable) alors le compilateur va commencer la compilation. Ouvrir l'onglet « Sortie de compilation » et observer les différentes étapes de la production d'un exécutable :

The screenshot shows the Qt Creator IDE. The main window displays a C++ source file named `main.cpp` with the following code:

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello World!" << endl;
8     return 0;
9 }
10

```

Overlaid on the code editor is a small console window titled `D:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe`. It displays the output of the program:

```

Hello World!
Appuyez sur <ENTRÉE> pour fermer cette fenêtre...

```

At the bottom of the IDE, the 'Sortie de compilation' (Compilation Output) window is open, showing the following log:

```

17:08:51: Début : "D:\Qt\5.9.1\mingw53_32\bin\qmake.exe" "G:\lycee\Projets perso\helloworld\helloworld.pro" -spec win32-g++ "CONFIG+=debug" "CONFIG+=qml_debug"
Info: creating stash file G:\lycee\Projets perso\build-helloworld-Desktop_Qt_5_9_1_MinGW_32bit-Debug\qmake.stash
17:08:53: Le processus "D:\Qt\5.9.1\mingw53_32\bin\qmake.exe" s'est terminé normalement.
17:08:53: Début : "D:\Qt\Tools\mingw530_32\bin\mingw32-make.exe" qmake_all
mingw32-make: Nothing to be done for 'qmake_all'.
17:08:57: Le processus "D:\Qt\Tools\mingw530_32\bin\mingw32-make.exe" s'est terminé normalement.
17:08:57: Début : "D:\Qt\Tools\mingw530_32\bin\mingw32-make.exe"
D:/Qt/Tools/mingw530_32/bin/mingw32-make -f Makefile.Debug
mingw32-make[1]: Entering directory 'G:\lycee\Projets perso\build-helloworld-Desktop_Qt_5_9_1_MinGW_32bit-Debug'
g++ -c -fno-keep-inline-dllexport -pipe -g -std=gnu++11 -Wextra -Wall -W -fexceptions -mthreads -DUNICODE -DQT_QML_DEBUG -I..\helloworld -I. -ID:\Qt\5.9.1\mingw53_32\mkspecs\win32-g++ -o debug\main.o ../helloworld/main.cpp
g++ -WL,-subsystem,console -mthreads -o debug\helloworld.exe debug\main.o
mingw32-make[1]: Leaving directory 'G:\lycee\Projets perso\build-helloworld-Desktop_Qt_5_9_1_MinGW_32bit-Debug'
17:09:06: Le processus "D:\Qt\Tools\mingw530_32\bin\mingw32-make.exe" s'est terminé normalement.
17:09:06: Temps écoulé : 00:18.

```

```

g++ -c -fno-keep-inline-dllexport -pipe -g -std=gnu++11 -Wextra -Wall -W -fexceptions -mthreads -DUNICODE
-DQT_QML_DEBUG -I..\helloworld -I. -ID:\Qt\5.9.1\mingw53_32\mkspecs\win32-g++ -o debug\main.o
..\helloworld\main.cpp

```

La commande `g++ -c ..... main.cpp` effectue la compilation du programme source .cpp en fichier objet (main.o)

```
g++ -WL,-subsystem,console -mthreads -o debug\helloworld.exe debug\main.o
```

La commande `g++ -o ...exe ....o` effectue la production d'un exécutable (.exe) à partir des fichiers .o. On appelle cette étape « l'édition de liens ».

L'exécution à proprement parlé s'effectue dans une fenêtre « DOS »

Si vous « re-cliquer » sur le bouton « triangle vert », QtCreator se contente d'exécuter votre programme. En effet, puisqu'aucune modification n'a eu lieu dans le code source, il n'est pas nécessaire de recompiler !

Ajouter une ligne simplement en tapant sur « Entrée » au bout d'une ligne de code. « re-cliquer » sur le bouton « triangle vert » et vous constatez que la compilation recommence du début.

Conclusion : le compilateur compare la date et l'heure de la dernière compilation et du (des) fichier (s) source. Si ceux-ci sont postérieurs à la « compil » alors il en déduit qu'ils ont été modifiés mais pas recompilés.

## La première variable

### 4 MODIFICATION DU CODE SOURCE : AFFECTATION ET AFFICHAGE

Saisir le code suivant (remplacer le code précédent par celui-ci)

```
#include <iostream>

using namespace std;

int main()
{
    int compteur(0);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```

La ligne `int compteur(0);` ;

=> crée un objet de type *int* avec le nom *compteur* et **initialise** celui-ci avec la valeur 0

La ligne `compteur = 10;` ;

=> **affecte** la valeur 10 à la variable *compteur*

Les lignes *cout* effectuent des affichages dans la fenêtre console « DOS »

### 5 TYPES SCALAIRES

Remplacer la ligne `compteur = 10;` par `compteur = 10.2;`

Recompiler et exécuter le programme. Vous constatez que l'affichage n'a pas changé. Dans la variable *compteur* y a-t-il 10 ou 10.2 ?

Le C++ est un langage fortement typé cela signifie que la variable (objet) est définie en tant que variable entière par conséquent même si vous lui affectez une valeur décimale (réelle) comme 10.2, le compilateur la transforme en une valeur entière (il effectue une troncature vers l'entier inférieur le plus proche)

Pour faire en sorte qu'on puisse mettre la valeur 10.2 dans *compteur*, il faut indiquer au compilateur que *compteur* est un réel.

Remplacer la ligne `int compteur(0);` par la ligne `float compteur(0);` ;

Affecter à *compteur* la valeur 10.2 et constater désormais que l'affichage est correct. *compteur* contient bien la valeur 10.2

Remplacer *float* par *double* et constater que le résultat est similaire.

*Pour résumer :*

On a des types (classes) entiers (*int*), réels (*float*, *double*).

La différence entre *float* et *double* se situe au niveau de la précision. *float* est un réel simple précision, *double* est un réel double précision (plus de chiffres après la virgule)

#### Remarque :

En C++, tous les types sont en fait des classes. Les variables sont des objets.

- ⇒ La classe est le modèle des objets que l'on va créer
- ⇒ Les objets sont des instances de ces classes

Ex : La classe « Etre humain », l'objet « Einstein »

#### Bonne pratique

**Prendre** l'habitude de toujours initialiser vos variables. Même si celles-ci seront modifiées dans le code juste après !

## 6 OPERATIONS ARITHMETIQUES

Saisir le code suivant :

```
#include <iostream>

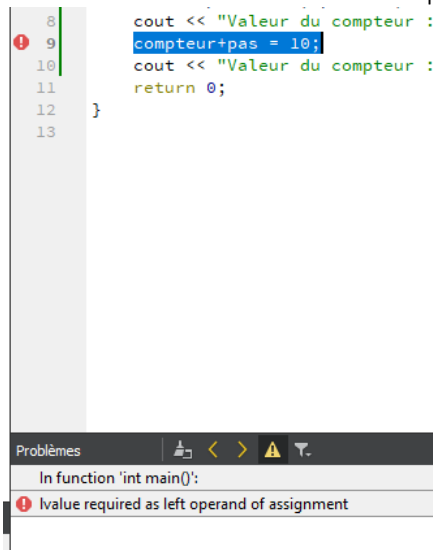
using namespace std;

int main()
{
    int compteur(0), pas(1);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10+pas;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```

Compiler et exécuter ce code. **Constater** que *compteur* contient bien la valeur 11. Donc le compilateur a effectué d'abord l'addition de 10 avec *pas* puis range le résultat dans *compteur*.

**Essayer** de remplacer la ligne `compteur = 10+pas;` par `compteur+pas = 10;`

Vous obtenez une erreur de compilation :



```
8      cout << "Valeur du compteur :
9      compteur+pas = 10;
10     cout << "Valeur du compteur :
11     return 0;
12     }
13
```

Problèmes

In function 'int main()':

lvalue required as left operand of assignment

*lvalue* signifie « la valeur de gauche » (left value)

A GAUCHE DU SIGNE D'AFFECTATION, ON NE PEUT AVOIR QU'UNE SEULE VARIABLE !!! puisqu'il s'agit d'une position mémoire (adresse mémoire)

*Remarque :*

Le nombre d'espaces dans le code n'a aucune importance ; il faut simplement que chaque ligne d'instruction soit terminée par le « ; »

## LES AUTRES OPERATIONS ARITHMETIQUES

Saisir le code suivant :

```
#include <iostream>

using namespace std;

int main()
{
    int compteur(0), pas(3);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10+pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10-pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10*pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10/pas;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```

**Compiler et exécuter** le code. Vous remarquerez que le résultat de la dernière opération correspond à une division euclidienne (division entière).  $10/3 = 3$  C'est tout à fait logique puisque *compteur* est un entier, *pas* est un entier et *10* est un entier.

**Modifier** le programme précédent :

```
#include <iostream>

using namespace std;

int main()
{
    float compteur(0);
    int pas(3);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10+pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10-pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10*pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10/pas;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```

*compteur* est un réel alors que *pas* est toujours un entier.

**Remarquez** que le résultat ne change pas ! ça n'est pas parce qu'on change le type de *compteur* en réel que la division devient une division « réelle ».

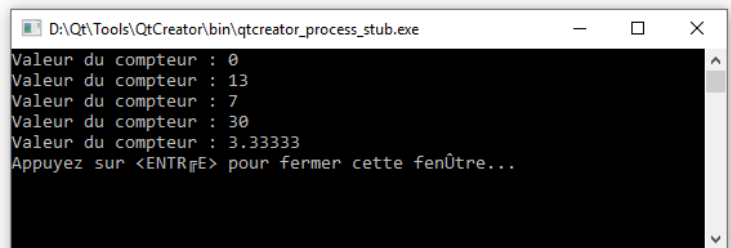
Pour que le résultat change, il faut faire en sorte que l'un des opérandes de la division soit réel. Vous avez le choix :

- soit vous faites : `10./pas` Sous entendu 10. est vu comme 10.0 donc réel

```
#include <iostream>

using namespace std;

int main()
{
    float compteur(0);
    int pas(3);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10+pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10-pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10*pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10/pas;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```



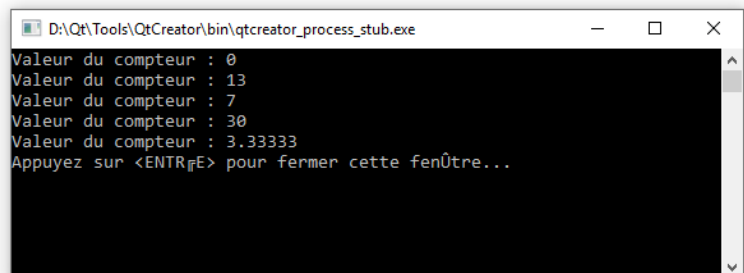
```
D:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Valeur du compteur : 0
Valeur du compteur : 13
Valeur du compteur : 7
Valeur du compteur : 30
Valeur du compteur : 3.33333
Appuyez sur <ENTRÉE> pour fermer cette fenêtre...
```

- soit vous modifiez le type de *pas* en le « passant » en réel

```
#include <iostream>

using namespace std;

int main()
{
    float compteur(0);
    float pas(3);
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10+pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10-pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10*pas;
    cout << "Valeur du compteur : " << compteur << endl;
    compteur = 10/pas;
    cout << "Valeur du compteur : " << compteur << endl;
    return 0;
}
```



```
D:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Valeur du compteur : 0
Valeur du compteur : 13
Valeur du compteur : 7
Valeur du compteur : 30
Valeur du compteur : 3.33333
Appuyez sur <ENTRÉE> pour fermer cette fenêtre...
```

## LA PREMIERE ITERATION

Saisir le code suivant :

```
#include <iostream>

using namespace std;

int main()
{
    int compteur(0);
    while (compteur <= 3)
    {
        cout << "Valeur du compteur : " << compteur << endl;
        compteur++;
    }

    cout << "Valeur finale du compteur : " << compteur << endl;
    return 0;
}
```

*Bonne pratique*

**Prendre** l'habitude d'indenter (tabulations) le code se situant à l'intérieur d'un bloc (ici par exemple le bloc *while*)

**Chercher** sur internet le rôle de *while*

Je vous conseille de consulter plutôt ce site : <http://www.cplusplus.com/reference/>

## SYNTHESE

Proposer une synthèse de ce TP en montrant ce que vous avez compris et retenu.

Pour vous aider, je vous propose une fiche de synthèse que vous pourrez utiliser (à rendre à la fin du TP).



# Fiche de synthèse TP1 – C++

**Nom :**

**Prénom :**

Désignation	Détails	Compléments
QtCreator	Environnement de développement logiciel (EDI)	Le compilateur C++ est installé dans l'environnement QtCreator
C++	Langage	Nécessite un compilateur C++. Ici, il s'agit de g++
int	classe (type)	Modèle de variables (objets)
compteur	Objet (variable)	
lvalue		
Affectation		
Initialisation		
while		
Division entière		

## ***A retenir***

Le signe '=' de l'affectation n'est pas une égalité au sens mathématique du terme, il s'agit  
 .....  
 .....

Lorsqu'on effectue une opération sur des variables, il faut connaître leur  
 .....  
 while est une ..... Elle sert à .....

## ***Questions qu'on peut se poser***

Quelle est la différence entre variable et type ?

Quelle est la différence entre affectation et initialisation ?

A quoi sert un compilateur ?

A quoi sert une boucle et peut-on s'en passer ?

Quel est le concept mathématique s'approchant le plus d'un programme informatique ?

Peut-on représenter une égalité mathématique dans un programme ?