Programmation C++ (débutant)/Notion de variable

Le cours du chapitre 2 : la notion de variable

1. **Les variables**
   Une variable est un certain endroit en mémoire permettant de stocker une valeur. En C++, les variables sont typées : elles contiennent soit un entier, un réel, un caractère, … Le type va exprimer la nature des données contenues dans notre variable. Les variables portent un nom appelé identificateur.
2. **Déclaration des variables**
   Avant d'utiliser une variable, il faut la déclarer, c'est-à-dire fournir son nom et son type. La déclaration d'une variable est obligatoire. Le nom de la variable s'appelle l'identificateur. Le type de la variable indique la nature des données que va contenir cette variable : un entier, un réel, un caractère,…
   Syntaxe de la déclaration : type identificateur ;
   Exemple : int a;
   Cette déclaration déclare une variable a de type int.
3. **Intérêt de la déclaration**
   La déclaration des variables permet au programmeur d'indiquer la nature des données qui vont être stockées dans ces variables. Loin d'être un handicap, la déclaration permet d'éviter de nombreux bogues. Il s'agit d'un garde-fou qui évite au programmeur de nombreuses erreurs.
4. **Initialisation des variables**
   En C++, les variables ont une valeur quelconque après leur déclaration. Le programmeur doit donc initialiser les variables de son programme, sinon elles contiendront n'importe quoi. L'oubli d'initialisation d'une variable est un bogue très fréquent.
5. **Identificateurs valides**
   Un identificateur est constitué d'une suite des lettres, de chiffres et _ (underscore). Un identificateur ne peut pas commencer par un chiffre. Il ne peut pas contenir d'espaces, ni contenir le caractères - (tiret) qui serait interprété comme un moins. Il doit être explicite c'est-à-dire qu'il doit être en rapport avec ce que contient la variable. Si une variable contient le prix d'un kg de tomates, on va appeler notre identificateur prix_tomate par exemple.
6. **Le type int**
   Il s'agit d'un type de base prédéfini dans le langage C++. Il permet de manipuler des entiers positifs ou négatifs. En général sur 32 bits : les données sont représentées en complément à 2. On peut alors représenter tous les entiers de -2^31 à 2^31-1. Le nombre de bits et le système de représentation des données n'est pas déterminée en C++, ce qui pose de gros problèmes de portabilité des programmes. La manipulation des entiers est exacte sans erreur de calcul !
7. **L'affectation**
   L'affectation permet d'effectuer des calculs et de transférer le résultat dans une certaine variable.
   Syntaxe : identificateur= expression ;
   On commence par évaluer l'expression. On met le résultat dans la variable identificateur. L'écriture d'une valeur dans une variable remplace la valeur précédente qui est "écrasée". Il doit y avoir une correspondance des types.
8. **Exemple 1 : utilisation d'une variable entière**

```
#include <iostream>
using namespace std;

int main()
{
```

```
    int a;
    a = 80 + 20;
    cout << "La valeur de a est : " << a << endl;
    return 0;
}
```

- Dans cet exemple, nous déclarons une variable entière a grâce à la déclaration **int a;**.
- Nous utilisons ensuite l'affectation pour mettre dans a le résultat de l'expression 80+20, c'est-à-dire 100.
- Nous affichons alors la valeur de a grâce à cout.
- **Exécution de l'exemple 1**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  La valeur de a est 100

9. **Exemple 2 : incrémentation d'une variable**

```
#include <iostream>
using namespace std;

int main()
{
    int a;
    a = 80;
    a = a + 1;
    cout << "La valeur de a est : " << a << endl;
    return 0;
}
```

- L'expression `a = a + 1;` peut paraitre étrange : elle permet d'augmenter de 1 la valeur de a; a vaut 80 avant l'exécution de cette instruction. a vaut 81 après cette exécution.
- A la place de a=a+1; , on peut également écrire a++;.
- **Exécution de l'exemple 2**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  La valeur de a est 81

10. **Exemple 3 : utilisation de plusieurs variables**

```
#include <iostream>
using namespace std;

int main()
{
    int a, bb = 9, c80;

    a = 80;
    a++;
    bb = bb + a;
    c80 = bb - 10;
    cout << "La valeur de a est : " << a << endl;
    cout << "La valeur de bb est : " << bb << endl;
    cout << "La valeur de c80 est : " << c80 << endl;
```

```
    return 0;
}
```

- Dans ce nouvel exemple, nous déclarons dans un premier temps 3 variables a,bb et c80.
- Nous effectuons différentes affectations sur ces variables.
- Nous affichons finalement le contenu final de ces variables.
- **Exécution de l'exemple 3**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  La valeur de a est 81
  La valeur de bb est 90
  La valeur de c80 est 80

11. **Différentes opérations**
    On peut additionner 2 entiers grâce à l'opérateur +.
    On peut soustraire 2 entiers grâce à l'opérateur -.
    On peut multiplier 2 entiers grâce à l'opérateur *.
    On peut diviser 2 entiers grâce à l'opérateur /. Il y a alors arrondi par troncature du résultat.
    On peut calculer le reste de la division de a par b grâce à l'opérateur %
12. **Exemple 4 : utilisation de différentes opérations**

```
#include <iostream>
using namespace std;

int main()
{
    int a = 10, b = 20, c, d, e, f;
    c = a + b;
    d = a * c;
    d = d - 80;
    e = d / 7;
    f = e % 4;

    cout << "La valeur de f est : " << f << endl;

    return 0;
}
```

    Lorsqu'on exécute notre programme, il s'affiche à l'écran :
    La valeur de f est 3

13. **cin**
    Il s'agit du flux d'entrée du programme. Par défaut le flux d'entrée d'un programme en C++ provient du clavier.
    En pratique, le cin permet d'envoyer le contenu de ce qui est saisit au clavier dans une variable.
    Exemple : cin>>i;
    Lorsque le programme exécute cette instruction, le programme s'arrête et attend que l'utilisateur tape au clavier une valeur entière. Lorsque l'utilisateur appuiera sur entrée , la valeur tapée ira dans la variable i.
14. **Exemple 5 : utilisation de cin**

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int a;
    cout << "Tapez la valeur de a : ";
    cin >> a;
    a = a + 10;
    cout << "La valeur de a est : " << a << endl;

    return 0;
}
```

- Dans cet exemple, nous déclarons une variable a.
- Nous saisissons ensuite au clavier la valeur de a.
- Nous effectuons un calcul sur cette variable.
- Nous affichons ensuite la valeur finale de a.
- Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez la valeur de a : **80**
  La valeur de a est 90.
  Remarque : dans cet exemple, l'utilisateur a choisi de taper la valeur 80 pour a.

15. **Le type double**

Le type double est un autre type prédéfini du langage C++. Il permet de stocker un réel. En général sur 64 bits, le format de représentation est souvent le format IEEE754. La taille et le système de représentation n'est pas imposé par le langage. Chaque opération peut être entachée d'une minuscule erreur de calcul. La propagation de cette erreur de calcul peut devenir dramatique !

16. **Exemple 6 : utilisation du type double**

```
#include <iostream>
using namespace std;

int main()
{
    double a, b, moy;

    cout << "Tapez une valeur réelle : ";
    cin >> a;
    cout << "Tapez une valeur réelle : ";
    cin >> b;

    moy = (a + b) / 2;

    cout << "la moyenne des 2 réels est : " << moy <<endl;

    return 0;
}
```

- Dans ce programme, on demande à l'utilisateur de taper successivement 2 valeurs réelles a et b.
- On calcule dans la variable moy la moyenne de a et de b.
- On affiche ensuite cette moyenne.

- **Exécution de l'exemple 6**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur réelle : **6.4**
  Tapez une valeur réelle : **3.2**
  La moyenne des 2 réels est : 4.8

17. **Compatibilité int-double**
    Les int et les double ne sont pas du tout représentés de la même manière. On peut sans problème copier un int dans un double :

```
int a;
double b;
b=a;
```

Pour mettre un double dans un int, il faut utiliser ce qu'on appelle un cast : on demande explicitement au compilateur de transformer le double en int et il y a alors troncature du double :

```
int a;
double b;
a=(int)b;
```

18. **Exemple 7 : mettre un int dans un double**

```
#include <iostream>
using namespace std;

int main()
{
    int a;
    double b;
    cout << "Tapez une valeur entière : ";
    cin >> a;

    b = a;

    cout << "La valeur de b vaut : " << b <<endl;

    return 0;
}
```

- Dans ce programme, on déclare une variable entière (de type int) a et une variable réelle (de type double) b.
- On demande à l'utilisateur de saisir la valeur de a.
- On met la valeur de a dans la variable b : cette opération s'effectue sans cast.
- On affiche ensuite la valeur de b.
- **Exécution de l'exemple 7**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur entière : **18**
  La valeur de b vaut : 18

19. **Exemple 8 : mettre un double dans un int**

```
#include <iostream>
using namespace std;
```

```
int main()
{
    double a;
    int b;

    cout << "Tapez une valeur réelle : ";
    cin >> a;

    b = (int)a;

    cout << "La valeur de b vaut : " << b <<endl;

    return 0;
}
```

- Dans ce programme, on déclare une variable réelle (de type double) a et une variable entière (de type int) b.
- On demande à l'utilisateur de saisir la valeur de a.
- On met la valeur de a dans la variable b : cette opération s'effectue avec un cast. Il y aura troncature de la valeur de a.
- On affiche ensuite la valeur de b.
- **Exécution du l'exemple 8**
  Lorsqu'on exécute notre programme, il s'affiche à l'écran :
  Tapez une valeur réelle : **6.78**
  La valeur de b vaut : 6

20. **Les commentaires**

Il est recommandé d'inclure dans tout programme des commentaires permettant de rendre le programme plus facilement compréhensible. Un programme doit être compréhensible par un autre programmeur : dans 6 mois, vous aurez oublié comment marche votre programme ou un autre programmeur peut être amené à le modifier.Le compilateur ne tient pas compte de tout ce qui est en commentaire !
Les commentaires se présentent sous 2 formes:

- Commentaires sur plusieurs lignes commençant par /* et finissant par */.
  /*kkkkkkk JJJJJJJJJJJJJ*/
- Commentaires sur une seule ligne commençant par //
  // kkkkkkkkkkkkkkkkkk

21. **Exemple 9 : les commentaires**

```
/****************************************
  MON PROGRAMME, AUTEUR : MOI
****************************************/

#include <iostream>
using namespace std;

int main()
{
    double a; int b;
```

```
    // Saisie de la variable a
    cout << "Tapez une valeur réelle : ";
    cin >> a;

    // On met a dans un entier
    b = (int)a;

    /* Affichage */
    cout << "La valeur de b vaut : " << b <<endl;

    return 0;
}
```

Exercices du chapitre 2

EXERCICE 1

Écrire un programme qui demande à l'utilisateur de taper la largeur et la longueur d'un champ et qui en affiche le périmètre et la surface.

EXERCICE 2

Écrire un programme qui demande à l'utilisateur de taper 5 entiers et qui affiche leur moyenne. Le programme ne devra utiliser que 2 variables.

EXERCICE 3

Écrire un programme qui demande à l'utilisateur de saisir 2 entiers A et B, qui échange le contenu des variables A et B puis qui affiche A et B.

EXERCICE 4

Écrire un programme qui demande à l'utilisateur de taper le prix HT d'un kilo de tomates, le nombre de kilos de tomates achetés, le taux de TVA (Exemple 5.5, 19.6,...). Le programme affiche alors le prix TTC des marchandises.

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de saisir les coordonnées de deux points du plan A et B et qui affiche la distance entre A et B.
Indication 1 : on pourra utiliser le théorème de Pythagore.
Indication 2 : dans le fichier include cmath, il y a une fonction sqrt qui calcule la racine carrée.
Exemple d'utilisation : x = sqrt(y)
x et y doivent être des double.
</pre>

Fonction racine carré : sqrt(nombre)
Fonction carré : pow(nombre, 2)
Les résultats renvoyés par des deux fonctions précédentes sont de type **'double**'.