Module	Programmation orienté objet
Formateur	**Webster Sturgis MITHOU MBENGA**
	Ingénieur en génie logiciel
Téléphone	06579-7307
Mail	webspirit@hotmail.com

SEQUENCE 3 : CONCEPTS FONDAMENTAUX

Les variables & opérateurs arithmétiques

1. DÉFINITION D'UNE VARIABLE

En programmation informatique, une variable est un espace de stockage nommé associé à une valeur ou à une information. Elle représente un emplacement dans la mémoire de l'ordinateur où des données peuvent être stockées et manipulées par un programme. Le nom de la variable sert à identifier cet emplacement mémoire de manière symbolique.

1.1 Utilité d'une Variable

Les variables sont essentielles dans la programmation pour plusieurs raisons :

- ✓ **Stockage de Données :** Les variables permettent de stocker des données telles que des nombres, des caractères, des chaînes de caractères, des valeurs booléennes, etc. Ces données peuvent être manipulées et utilisées dans le programme.
- ✓ **Manipulation de l'Information :** Les variables sont utilisées pour effectuer des opérations et des calculs. Par exemple, additionner deux nombres, concaténer des chaînes de caractères, ou effectuer des opérations plus complexes en fonction des besoins du programme.
- ✓ **Maintien de l'État :** Les variables peuvent être utilisées pour maintenir l'état d'un programme. Par exemple, garder en mémoire le score d'un joueur dans un jeu vidéo ou le suivi d'un processus dans une application.
- ✓ **Passage d'Informations :** Les variables permettent de transmettre des informations entre différentes parties d'un programme. Les fonctions peuvent recevoir des variables en entrée, effectuer des opérations, et renvoyer des résultats sous forme de variables.
- ✓ **Adaptabilité :** En utilisant des variables, un programme devient plus adaptable et flexible. Les valeurs stockées dans les variables peuvent être modifiées au cours de l'exécution du programme, ce qui permet d'ajuster le comportement du programme en fonction des conditions rencontrées.

2. LES CONVENTIONS DE NOMMAGES

Les conventions de nommage sont des règles et des pratiques définies pour attribuer des noms cohérents et significatifs aux variables, fonctions, classes, et autres éléments du code source. Adopter des conventions de nommage uniformes facilite la lecture, la compréhension et la maintenance du code, surtout lorsque plusieurs développeurs travaillent sur un même projet. Voici quelques conventions de nommage couramment utilisées en programmation :

- ✓ **Camel Case** : Commencent par une minuscule et chaque mot suivant commence par une majuscule (ex. nomDeVariable, calculerSomme()).
- ✓ **Pascal Case (ou Upper Camel Case)** : Commencent par une majuscule, chaque mot suivant commence également par une majuscule (ex. NomDeClasse, StructureDeDonnees).
- ✓ **Snake Case :** Toutes en minuscules, les mots sont séparés par des underscores (ex. nom_de_variable, calculer_somme()).
- ✓ **Kebab Case :** Toutes en minuscules, les mots sont séparés par des tirets (ex. nom-de-variable, calculer-somme()).

Préfixes et Suffixes : Ajout de préfixes ou de suffixes pour indiquer le type ou la portée de l'élément. Par exemple, strNom pour une chaîne de caractères en C.

Noms Significatifs : Choisissez des noms qui reflètent clairement la fonction ou le but de l'élément. Évitez les noms génériques comme temp, data, etc.

Éviter les Abréviations Ambiguës : Évitez les abréviations qui pourraient prêter à confusion. Par exemple, utilisez compteur au lieu de ctr.

3. TYPES DE DONNEES DE BASE

Le langage C offre plusieurs types de données de base qui permettent aux programmeurs de représenter différentes informations. Voici quelques-uns des types de données fondamentaux :

a) **Entiers (int) :** Utilisé pour représenter des nombres entiers, positifs ou négatifs.
`int age = 25;`
b) **Décimaux (float et double)** : Utilisés pour représenter des nombres à virgule flottante. `float prix = 12.99;`
c) **Caractères (char)** : Utilisé pour représenter un seul caractère.
`char grade = 'A';`
d) **Booléens (bool)** : Utilisé pour représenter des valeurs booléennes (vrai/faux).
`bool estVrai = true;`

4. DECLARATION ET INITIALISATION VARIABLES

4.1 Déclaration de Variables

La déclaration de variables informe le compilateur sur le type de données que la variable va contenir. Elle se fait en spécifiant le type suivi du nom de la variable.

```
int nombre;
float prix;
char lettre;
```

4.2 Initialisation de Variables

L'initialisation consiste à attribuer une valeur initiale à une variable lors de sa déclaration. Cela peut se faire simultanément avec la déclaration ou ultérieurement dans le programme.

```
int age = 30;
float taille = 1.75;
char genre = 'M';
```

4.3 Les spécificateurs de format

Les spécificateurs de format sont utilisés dans le langage C avec les fonctions de formatage de chaînes telles que `printf`, `scanf`. Ils indiquent comment interpréter ou afficher les valeurs des variables dans une chaîne de format. Voici quelques-uns des spécificateurs de format les plus couramment utilisés :

%d	Spécifie un entier décimal.
%f	Spécifie un nombre à virgule flottante
%c	Spécifie un caractère
%s	Spécifie une chaîne de caractères

Ces spécificateurs de format permettent un contrôle précis sur la manière dont les données sont affichées ou interprétées lors de l'entrée/sortie dans un programme en langage C. Il est important de les utiliser correctement pour éviter des comportements indésirables ou des erreurs de programme.

4.4 Les caractères spéciaux

En programmation, les caractères spéciaux, également appelés caractères d'échappement, sont des séquences de caractères spécifiques utilisées pour représenter des caractères non imprimables ou pour effectuer des actions spéciales dans une chaîne de caractères. En langage C, ces caractères spéciaux sont précédés par le caractère d'échappement `\`. Voici quelques-uns des caractères spéciaux couramment utilisés :

\n	Saut de ligne
\t	Tabulation horizontale
\r	Retour chariot.
\b	Retour arrière (efface le caractère précédent).
\f`	Saut de page
\\	Caractère barre oblique inverse
\'	Apostrophe simple
\"	Guillemet double
\a	Bip sonore

4.5 Les fonctions d'entrée et sortie

En programmation, les fonctions d'entrée/sortie (E/S) sont des mécanismes permettant à un programme de communiquer avec l'environnement extérieur, généralement l'utilisateur ou d'autres parties du système. En langage C, les fonctions principales d'entrée/sortie sont définies dans la bibliothèque standard `stdio.h`. Voici quelques-unes des fonctions les plus couramment utilisées :

a) Fonctions d'Entrée (Input) :

- ✓ **scanf` :** Utilisée pour lire l'entrée formatée depuis le clavier ou d'autres sources.
  Exemple :

```
int age;
printf("Entrez votre âge : ");
scanf("%d", &age);
```

- ✓ **getchar` :** Lit un seul caractère depuis l'entrée standard (généralement le clavier).
  Exemple :

```
char caractere;
printf("Entrez un caractère : ");
caractere = getchar();
```

b) Fonctions de Sortie (Output):

- ✓ **printf`** : Utilisée pour afficher des données formatées à la console ou dans d'autres sorties.
  Exemple :

```
int nombre = 42;
printf("La réponse est : %d\n", nombre);
```

- ✓ **putchar** : Affiche un seul caractère à la console.
  Exemple :

```
char lettre = 'A';
putchar(lettre);
```

Ces fonctions offrent des moyens de base pour interagir avec l'utilisateur et afficher des résultats. Cependant, il est important de noter que certaines,

Exemples Combinés

```
#include <stdio.h>
int main() {
    // Déclaration de variables
    int nombre;
    float prix;
    char lettre;
    // Initialisation des variables
    nombre = 10;
    prix = 5.99;
    lettre = 'C';
    // Affichage des valeurs
    printf("Nombre: %d\n", nombre);
    printf("Prix: %.2f\n", prix);
    printf("Lettre: %c\n", lettre);
    return 0;
}
```

5. COMMENTER LE CODE

Commenter le code est une pratique essentielle en programmation. Les commentaires aident à expliquer le but, le fonctionnement et la logique du code source. Ils facilitent la compréhension du code par d'autres développeurs (ou même par vous-même à un stade ultérieur), améliorent la maintenabilité du code et contribuent à la collaboration au sein d'une équipe de développement. Voici quelques points sur l'importance de commenter le code et sur la manière de le faire efficacement :

5.1 Importance de Commenter le Code :

- ✓ Compréhension du code : Les commentaires permettent aux autres développeurs (ou à vous-même) de comprendre rapidement le but et le fonctionnement du code.
- ✓ Facilitation de la maintenance : Lorsque des mises à jour ou des corrections doivent être apportées, les commentaires aident à localiser rapidement les parties du code qui doivent être modifiées.
- ✓ Collaboration : En travaillant en équipe, les commentaires aident à expliquer les choix de conception, les décisions stratégiques et les intentions du code, facilitant la collaboration et la communication.
- ✓ Formation : Les commentaires sont utiles pour expliquer le code à des débutants ou à des nouveaux membres de l'équipe, facilitant ainsi le processus de formation.

- ✓ Réduction des Erreurs : Des commentaires clairs et précis peuvent réduire le risque d'interprétation erronée du code, minimisant ainsi les erreurs potentielles lors de la modification ou de l'extension du code.

5.2 Commenter le Code de Manière Efficace :

- ✓ Commentaires descriptifs : Les commentaires doivent décrire le pourquoi du code, pas simplement ce qu'il fait. Expliquez la logique, l'intention, les choix de conception, etc.
- ✓ Mises à Jour des Commentaires : Assurez-vous de mettre à jour les commentaires lorsque le code est modifié. Des commentaires obsolètes peuvent être plus nuisibles que pas de commentaires du tout.
- ✓ Évitez les Commentaires Évidents : Inutile de commenter chaque ligne de code si elle est évidente. Les commentaires doivent ajouter de la valeur en expliquant des concepts complexes ou des décisions non évidentes.
- ✓ Conventions de Commentaires : Adoptez des conventions de commentaires cohérentes au sein de votre équipe ou de votre projet. Cela peut inclure l'utilisation de blocs de commentaires pour décrire les fonctions, les classes, etc.
- ✓ Utilisation de Commentaires Temporaires : Si vous avez besoin d'ajouter rapidement un commentaire pour expliquer quelque chose temporaire, assurez-vous de revenir plus tard pour le mettre à jour ou le supprimer.

6. LES OPERATEURS ARITHMETIQUES

En langage C, les opérateurs arithmétiques sont utilisés pour effectuer des opérations mathématiques sur des variables numériques. Voici les principaux opérateurs arithmétiques en C :

```
+ (Addition) : Ajoute deux valeurs.
int somme = a + b;
```

```
- (Soustraction) : Soustrait la valeur de droite de la valeur de gauche
int difference = a - b;
```

```
* (Multiplication) : Multiplie deux valeurs
int produit = a * b;
```

```
/ (Division) : Divise la valeur de gauche par la valeur de droite. Notez
que si les deux opérandes sont des entiers, la division sera une division
entière.
float quotient = (float)a/b;   // Pour obtenir un résultat en virgule
flottante
```

```
% (Modulo) : Donne le reste de la division entière de la valeur de gauche
par la valeur de droite.
int reste = a % b;
```

7. CONCLUSION

La compréhension des types de données de base en C et la manière de déclarer et initialiser des variables sont des compétences cruciales pour tout programmeur. Ces concepts fondamentaux servent de base à la construction de programmes plus complexes. En continuant d'explorer le langage C, vous découvrirez des structures de données plus avancées et des fonctionnalités puissantes pour développer des applications robustes.