C - Les types de variables

Dans la programmation C, les variables permettent de stocker et manipuler des données. Dans cet article, nous allons explorer les principaux types de variables en C avec des exemples pratiques.

5TTR 6TTR Découverte

Types de variables en C

C est un langage à **typage statique** (*statically typed*): les types des variables sont **contrôlés au moment de la compilation**.

Une fois la variable déclarée avec un type, elle ne peut plus changer de type par la suite.

Voici les principaux types de variables utilisés dans la programmation en C:

Les entiers (`int` , `unsigned int`)

Les entiers permettent de stocker des nombres entiers, positifs ou négatifs.

Les **int** stockent des valeurs positives et/ou négatives (ils utilisent un bit de signe).

Les **unsigned int** stockent des valeurs positives uniquement.

Les flottants (`float` , `double`)

Les flottants sont utilisés pour stocker des nombres décimaux. Ils sont utiles pour les calculs de précision.

- **float** : précision jusqu'à 6-7 chiffres après la virgule.
- **double** : souvent identique à `float` (pas de différence).

Exemple : Calcul de la distance parcourue par un robot.

```
1 | float distanceParcourue = 12.34; // En mètres
2 | double angleRotation = 45.5; // En degrés
```

Les caractères (`char`)

Les caractères permettent de stocker une lettre ou un symbole ASCII. Ils sont souvent utilisés pour gérer des messages ou des commandes.

Exemple : Communication avec un robot via Bluetooth.

```
char commandeRecue = 'A'; // Commande reçue : avancer
```

Les chaînes de caractères (`char[]` ou `String`)

Les chaînes de caractères sont utilisées pour manipuler des mots ou des messages.

- **char[]** : tableau de caractères, plus optimisé pour les microcontrôleurs.
- **String** : classe de manipulation de chaînes, plus facile à utiliser mais consomme plus de mémoire.

Exemple : Affichage d'un message sur un écran LCD.

```
char messageLCD[] = "Robot prêt"; // Message statique
```

Les booléens (`bool`)

Les booléens stockent des valeurs `true` (vrai) ou `false` (faux). Ils sont utiles pour gérer des états.

Exemple : Vérification de la détection d'un obstacle.

```
bool obstacleDetecte = false; // Initialisation
```

Bonnes pratiques pour utiliser les variables

1. **Noms explicites** : Donnez des noms significatifs à vos variables pour comprendre leur rôle.
    - Mauvais exemple : `int a;`
    - Bon exemple : `int vitesseMoteur;`
2. **Initialisation** : Assurez-vous que vos variables sont initialisées avant leur utilisation pour éviter des comportements imprévisibles.
3. **Limitation de la portée** : Déclarez vos variables au niveau approprié (locale ou globale) pour éviter les conflits et réduire la consommation de mémoire.
4. **Évitez les types coûteux** : Préférez des types simples (comme `char[]`) plutôt que `String` si la mémoire est limitée.

Variables globales et locales

Variables globales

Déclarées en dehors de toute fonction, les variables globales sont accessibles dans tout le programme.

Exemple : Stocker la vitesse du robot.

```
int vitesseGlobale = 100; // Accès depuis n’importe quelle fonction
```

Variables locales

Déclarées à l'intérieur d'une fonction, elles ne sont accessibles que dans cette fonction.

Exemple : Utiliser une variable temporaire pour un calcul.

```
void avancer() {
    int vitesseLocale = 120; // Variable spécifique à cette fonction
    analogWrite(3, vitesseLocale);
}
```

Conclusion

Les variables sont au cœur de vos projets de programmation. Bien choisir leur type et leur portée, les initialiser correctement, et les utiliser de manière structurée est essentiel pour garantir la fiabilité et la clarté de votre code. Avec ces bases et les exemples fournis, vous êtes prêt à gérer efficacement les données de vos programmes.