Demander à l'utilisateur d'entrer des données et les stocker dans des variables

`cin` est un objet de la classe `istream` en C++ qui est défini dans l'en-tête `iostream`. Il est utilisé pour lire des données à partir du périphérique d'entrée standard, c'est-à-dire le clavier. Il est associé au flux d'entrée standard C `stdin`. Le "c" dans `cin` signifie "caractère" et "in" signifie "entrée". Par conséquent, `cin` signifie "entrée de caractères".

L'objet `cin` est utilisé avec l'opérateur d'extraction `>>` pour lire un flux de caractères. Par exemple, pour lire une variable entière et une chaîne de caractères, vous pouvez utiliser le code suivant:

```
int var1;
std::cin >> var1;
std::string var2;
std::cin >> var2;
```

Cela lira une valeur entière à partir du clavier et la stockera dans la variable `var1`, puis lira une chaîne de caractères à partir du clavier et la stockera dans la variable `var2`. Vous pouvez également utiliser plusieurs opérateurs d'extraction `>>` dans une seule instruction pour lire plusieurs valeurs à la fois.

En résumé, `cin` est un objet en C++ qui permet de lire des données à partir du clavier en utilisant l'opérateur d'extraction `>>`. Voici un exemple simple de code C++ qui utilise `cin` pour saisir des données pour le prénom et la taille:

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

int main() {
    string prenom;
    double taille;

    cout << "Entrez votre prenom: ";
    cin >> prenom;

    cout << "Entrez votre taille (en metres): ";
    cin >> taille;

    cout << "Bonjour, " << prenom << "! Vous mesurez " << taille << " metres." << endl;

    return 0;
}
```

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

int main() {
    string prenom;
    double taille;

    cout << "Entrez votre prenom: ";
    cin >> prenom;

    cout << "Entrez votre taille (en metres): ";
    cin >> taille;

    cout << "Bonjour, " << prenom << "! Vous mesurez " << taille << " metres." << endl;

    return 0;
}
```

Ce code utilise l'objet `cin` pour saisir des données à partir de l'entrée standard (généralement le clavier). Il déclare deux variables, `prenom` de type `string` et `taille` de type `double`, pour stocker les données saisies par l'utilisateur. Le programme affiche des messages à l'écran pour demander à l'utilisateur d'entrer son prénom et sa taille, puis utilise `cin` pour lire les données saisies. Enfin, il utilise `cout` pour afficher un message de bienvenue à l'utilisateur en utilisant les données saisies.

Il y a une règle à apprendre: Quand on utilise des chevrons et `getline()` , on doit impérativement placer l'instruction `cin.ignore()` après la ligne `cin>>a` .

Modifier les valeurs des variables

En C++, l'affectation de variables se fait en utilisant l'opérateur d'affectation. Lorsque vous utilisez cet opérateur, la valeur à droite de l'opérateur est affectée à la variable à gauche de l'opérateur. Par exemple, pour affecter la valeur `5` à une variable `x`, vous pouvez utiliser l'instruction suivante :

```cpp
int x; // Déclare une variable de type entier
x = 5; // Affecte la valeur 5 à la variable x
```

Vous pouvez également affecter une valeur à une variable lors de sa déclaration en utilisant la syntaxe suivante :

```cpp
int x = 5; // Déclare une variable de type entier et lui affecte la valeur 5
```

Une fois qu'une variable a été déclarée et initialisée, vous pouvez modifier sa valeur en utilisant l'opérateur d'affectation. Par exemple, pour incrémenter la valeur de `x` de `1`, vous pouvez utiliser l'instruction suivante :

```
x = x + 1; // Incrémente la valeur de x de 1
```

Vous pouvez également utiliser des opérateurs d'affectation composés pour effectuer des opérations arithmétiques et logiques sur les variables. Par exemple, pour ajouter `5` à la valeur de `x`, vous pouvez utiliser l'opérateur d'affectation composé `+=` comme suit :

```
x += 5; // Ajoute 5 à la valeur de x
```

Calculer avec des variables

Calcul IMC

```
#include <iostream> // Inclut la bibliothèque d'entrée/sortie
#include <string> // Inclut la bibliothèque de chaînes de caractères
using namespace std; // Utilise l'espace de noms standard

int main() { // Fonction principale
    string prenom; // Déclare une variable de type chaîne pour stocker le prénom
    double taille, poids, imc; // Déclare des variables de type double pour stocker la taille, le poids et l'IMC

    cout << "Entrez votre prenom: "; // Demande à l'utilisateur d'entrer son prénom
    cin >> prenom; // Lit l'entrée de l'utilisateur et la stocke dans la variable prenom

    cout << "Entrez votre taille (en metres): "; // Demande à l'utilisateur d'entrer sa taille
    cin >> taille; // Lit l'entrée de l'utilisateur et la stocke dans la variable taille

    cout << "Entrez votre poids (en kilogrammes): "; // Demande à l'utilisateur d'entrer son poids
    cin >> poids; // Lit l'entrée de l'utilisateur et la stocke dans la variable poids

    imc = poids / (taille * taille); // Calcule l'IMC en divisant le poids par la taille au carré

    cout << "Bonjour, " << prenom << "! Vous mesurez " << taille << " metres et pesez " << poids << " kilogrammes." << endl;
    cout << "Votre IMC est de " << imc << "." << endl; // Affiche l'IMC calculé

    return 0; // Termine le programme avec succès
}
```

Essayez d'entrer Jean René comme prénom

Il faut apporter une modification au code…

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

int main() {
    string prenom;
    double taille, poids, imc;

    cout << "Entrez votre prenom: ";
    getline(cin, prenom);

    cout << "Entrez votre taille (en metres): ";
    cin >> taille;

    cout << "Entrez votre poids (en kilogrammes): ";
    cin >> poids;

    imc = poids / (taille * taille);

    cout << "Bonjour, " << prenom << "! Vous mesurez " << taille << " metres et pesez " << poids << " kilogrammes." << endl;
    cout << "Votre IMC est de " << imc << "." << endl;

    return 0;
}
```

Dans ce code, nous utilisons la fonction `getline` pour lire une ligne entière de l'entrée standard (`cin`) et la stocker dans la variable `prenom`. Cette fonction lit tous les caractères jusqu'à ce qu'elle rencontre un caractère de fin de ligne (`\n`), ce qui permet d'accepter les espaces dans le prénom saisi par l'utilisateur.

Liste des opérateurs en C++

♦ **Opérateurs arithmétiques** : `+` (addition), `-` (soustraction), `*` (multiplication), `/` (division), `%` (modulus)

♦ **Opérateurs d'affectation** : `=` (affectation simple), `+=` (addition et affectation), `-=` (soustraction et affectation), `*=` (multiplication et affectation), `/=` (division et affectation), `%=` (modulus et affectation)

♦ **Opérateurs de comparaison** : `==` (égal à), `!=` (différent de), `<` (inférieur à), `>` (supérieur à), `<=` (inférieur ou égal à), `>=` (supérieur ou égal à)

♦ **Opérateurs logiques** : `&&` (et logique), `||` (ou logique), `!` (non logique)

♦**Opérateurs d'incrémentation/décrémentation** : `++` (incrémentation), `--` (décrémentation)

♦ **Opérateurs de manipulation de bits** : `&` (et bit à bit), `|` (ou bit à bit), `^` (ou exclusif bit à bit), `~` (non bit à bit), `<<` (décalage à gauche), `>>` (décalage à droite)

Les constantes

En C++, une constante est une valeur qui ne peut pas être modifiée pendant l'exécution du programme. Vous pouvez déclarer une constante en utilisant le mot-clé `const` suivi du type de la constante et de son nom. Par exemple, pour déclarer une constante de type entier nommée `PI` et lui affecter la valeur `3.14`, vous pouvez utiliser l'instruction suivante :

```
const double PI = 3.14;
```

Une fois qu'une constante a été déclarée et initialisée, vous ne pouvez pas modifier sa valeur. Si vous essayez de le faire, le compilateur générera une erreur. Par exemple, l'instruction suivante générera une erreur de compilation :

```
PI = 3.14159; // Erreur: impossible de modifier la valeur d'une constante
```

Les constantes sont utiles lorsque vous avez des valeurs qui ne changent pas tout au long de l'exécution du programme, comme des valeurs mathématiques (par exemple, PI), des taux de conversion (par exemple, le nombre de pieds dans un mètre), etc.

Quand utiliser les constantes ?

Elles sont utilisées en C++ pour stocker des valeurs qui ne doivent pas être modifiées pendant l'exécution du programme. Mais également elles sont utilisées pour stocker des valeurs qui sont utilisées à plusieurs endroits dans le code, valeurs qui ne changent pas, comme des constantes mathématiques (par exemple, PI), des taux de conversion (par exemple, le nombre de pieds dans un mètre), des tailles de tableaux fixes, etc.

L'utilisation de constantes peut :

- rendre votre code plus lisible et plus facile à maintenir (vous pouvez donner un nom significatif à une valeur qui est utilisée à plusieurs endroits dans votre code.)
- aider à prévenir les erreurs de programmation, car le compilateur vous avertira si vous essayez de modifier la valeur d'une constante par inadvertance.

Voici un exemple d'utilisation d'une constante en C++ :

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

const double PI = 3.14159; // Déclaration d'une constante

int main() {
    double rayon = 2.5; // Déclaration d'une variable
    double aire = PI * rayon * rayon; // Utilisation de la constante PI pour calculer l'aire d'un cercle

    cout << "L'aire du cercle est " << aire << endl; // Affichage du résultat

    return 0;
}
```

Dans cet exemple, nous déclarons une constante `PI` pour stocker la valeur de la constante mathématique PI. Nous utilisons ensuite cette constante pour calculer l'aire d'un cercle en utilisant la formule `aire = PI * rayon * rayon`. L'utilisation de la constante `PI` rend notre code plus lisible et plus facile à comprendre.

ATTENTION : Les constantes sont écrites en **MAJUSCULE**.

Exercices

Ecrivez un programme pour convertir une valeur entrée en mille marin et afficher le résultat en km/h dans la Console

Dans cet exercice, vous demandez à l'utilisateur d'entrer une vitesse en milles marins, puis vous utilisez une constante avec le ratio pour convertir cette vitesse en kilomètres par heure.

Enfin, vous affichez le résultat à l'écran.