ENTRÉES ET SORTIES

ENTRÉES ET SORTIES EN C++

BIBLIOTHÈQUES DE BASE

IOSTREAM

UTILITÉ

La bibliothèque **`iostream`** fournit des fonctionnalités d'**entrée** et de **sortie standard**.

Fonction	Description
cout	Sortie standard (console)
cin	Entrée standard (clavier)
cerr	Sortie d'erreur standard

Exemple :

```cpp
#include <iostream>

int main() {
    int age;

    std::cout << "Entrez votre âge: ";
    std::cin >> age;
    std::cout << "Vous avez " << age << " ans." << std::endl;

    return 0;
}
```

Pour utiliser `iostream`, il faut inclure la directive `#include <iostream>` en haut du programme.

SYNTAXE D'INCLUSION

```
#include <iostream>
```

UTILISATION DE `cout` ET `cin`

cout

SYNTAXE DE BASE

`cout` est un objet de flux de sortie standard. Il est utilisé pour envoyer des données vers la console.

```
std::cout << "Hello, World!";
```

OPÉRATEUR D'INSERTION <<

L'opérateur d'insertion << est utilisé pour insérer des données dans le **flux de sortie**.

```
std::cout << "Hello" << ", " << "World!";
```

`std::cout` sert à afficher le contenu à l'écran.

EXEMPLES D'UTILISATION

```
int a = 5;
std::cout << "The value of a is: " << a << std::endl;
```

Cet exemple illustre comment déclarer une variable et l'afficher à l'aide de `std::cout`. `std::endl` nous permet d'ajouter un retour à la ligne.

cin

SYNTAXE DE BASE

`cin` est un objet de **flux d'entrée standard**. Il est utilisé pour recevoir des données à partir de la console.

```
int x;
std::cin >> x;
```

L'opérateur `>>` est utilisé pour lire les données entrées dans la console.

OPÉRATEUR D'EXTRACTION >>

L'opérateur d'extraction `>>` est utilisé pour extraire des données du **flux d'entrée**.

```
std::cin >> a >> b;
```

EXEMPLES D'UTILISATION

```
int a, b;
std::cout << "Enter two integers: ";
std::cin >> a >> b;
std::cout << "Sum: " << (a + b) << std::endl;
```

Cet exemple montre comment lire deux entiers depuis l'entrée standard et afficher leur somme à l'écran.

MANIPULATEURS DE FLUX

endl

`endl` est un **manipulateur de flux** qui insère un **saut de ligne** et vide le tampon associé au flux.

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

int main() {
  cout << "Hello, ";
  cout << "World!" << endl;
}
```

Un autre moyen d'insérer un saut de ligne en C++ est d'utiliser l'opérateur " \n ".

UTILITÉ

Il permet de passer à la ligne suivante lors de l'affichage avec **cout**.

Exemple :

```cpp
#include <iostream>

int main() {
  std::cout << "Ligne 1" << std::endl;
  std::cout << "Ligne 2" << std::endl;
  return 0;
}
```

setw

`setw` est un **manipulateur de flux** qui définit la largeur minimale du champ à afficher.

```cpp
#include <iostream>
#include <iomanip>

int main() {
  std::cout << std::setw(10) << "Hello" << std::endl;
  std::cout << std::setw(10) << "World!" << std::endl;
  return 0;
}
```

`setw` est utilisé en association avec la bibliothèque `iomanip`, il faut l'inclure avec `#include <iomanip>`.

`setprecision`

`setprecision` est un manipulateur de flux qui définit le **nombre de chiffres de précision** à afficher pour les **nombres flottants**.

```cpp
#include <iostream>
#include <iomanip>

int main() {
    double nombrePi = 3.14159;
    std::cout << std::fixed << std::setprecision(2) << nombrePi << std::endl;
    return 0;
}
```

Dans cet exemple, l'affichage de `nombrePi` est limité à 2 chiffres après la virgule grâce à `setprecision`. Résultat : 3.14.

FLOTS FORMATÉS

IOS::FIXED

`ios::fixed` est un **indicateur de format** qui permet d'afficher les nombres à **virgule flottante** sous forme **décimale** (non scientifique).

```cpp
#include <iostream>
#include <iomanip>

int main() {
    double nombre = 12387654327456;

    std::cout << "Sans ios::fixed: " << nombre << std::endl;
    std::cout << std::fixed;
    std::cout << "Avec ios::fixed: " << nombre << std::endl;

    return 0;
}
```

UTILITÉ

- **Afficher les valeurs flottantes** avec un nombre fixe de décimales
- **Faciliter la lecture** des valeurs flottantes

```cpp
#include <iostream>
#include <iomanip>

int main() {
  double valeur = 3.14159;
  std::cout << std::fixed << std::setprecision(2) << valeur << std::endl;
  return 0;
}
```

Ceci est un exemple de code pour afficher les valeurs flottantes avec deux décimales. Vous pouvez modifier le nombre de décimales en changeant la valeur passée à std::setprecision().

USAGE AVEC `cout`

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

int main() {
    double num = 123.456789;
    cout << fixed << setprecision(2) << num; // Affiche 123.46
    return 0;
}
```

La fonction **setprecision()** permet de contrôler le nombre de décimales affichées lors de l'affichage de nombres à virgule flottante avec `cout`. **fixed** est utilisé pour indiquer que le format doit être fixe et non scientifique.

IOS::SCIENTIFIC

ios::scientific est un **indicateur de format** qui permet d'afficher les **nombres à virgule flottante** sous forme **scientifique** (notation exponentielle).

Exemple :

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

int main() {
  double num = 123;
  cout << "Sans ios::scientific : " << num << endl;

  cout << "Avec ios::scientific : " << scientific << num << endl;
  return 0;
}
```