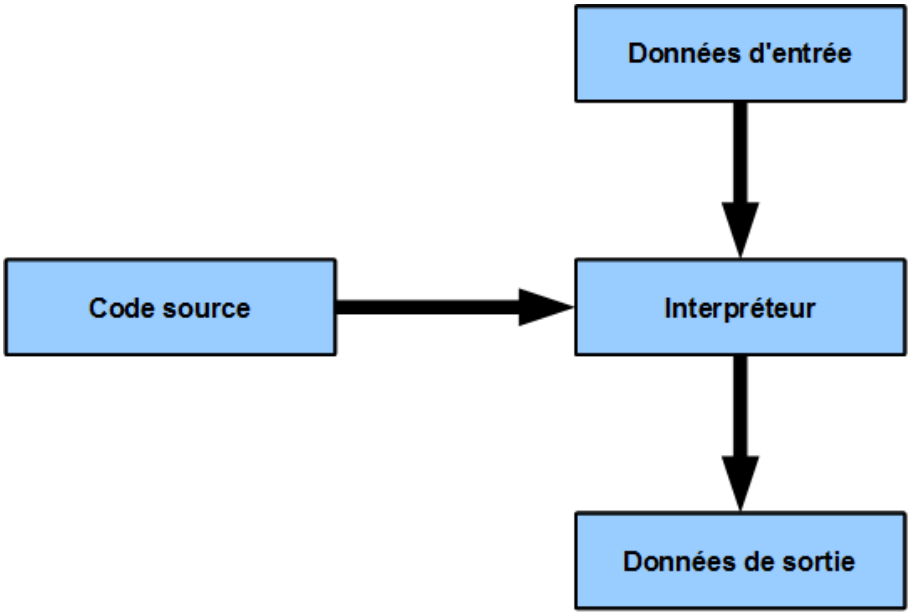
1. Langages compilés et langages interprétés.

On peut distinguer deux grands types de langages : les langages interprétés et les langages compilés. En voici quelques exemples de chaque catégorie :

- Langages interprétés : Java, Python, Ruby ;
- Langages compilés : C, C++, Pascal et OCaml.

Langages interprétés

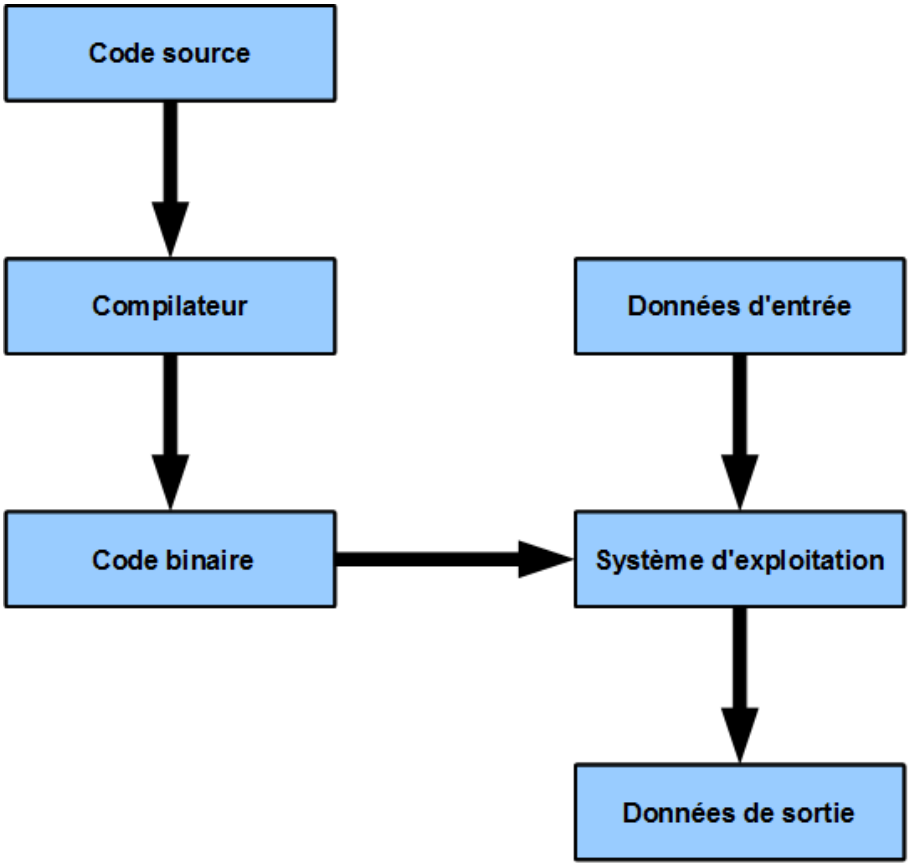
Dans ces langages, le code source (celui que vous écrivez) est interprété, par un logiciel qu'on appelle interpréteur. Celui-ci va utiliser le code source et les données d'entrée pour calculer les données de sortie :



L'interprétation du code source est un processus « pas à pas » : l'interpréteur va exécuter les lignes du code une par une, en décidant à chaque étape ce qu'il va faire ensuite.

Langages compilés

Dans ces langages, le code source est tout d'abord transformé, par un logiciel qu'on appelle compilateur, en un code binaire qu'un humain ne peut pas lire mais qui est très facile à exploiter pour un ordinateur. C'est alors directement le système d'exploitation qui va utiliser le code binaire et les données d'entrée pour calculer les données de sortie :



Comparatif

	Avantages	Inconvénients
Compilé	Prêt à l'emploi Plus performant Le code source reste privé	Non multiplateforme Prise en main plus complexe Étape supplémentaire (compilation)
Interprété	Multiplateforme Prise en main plus facile Plus facile à tester et à débugger	Nécessite un interpréteur Plus lent Le code source est publique

Les langages interprétés ont pendant longtemps été mal perçus par les développeurs. Mais l'amélioration constante des processeurs diminue le problème de leur extrême lenteur, la facilité d'évolution d'un programme interprété a changé la vision des développeurs pressés, et la rapidité de mise en place de petits programmes les ont rendus indispensables au quotidien, pour des automatisations de tâches plus ou moins complexes.

2. Le langage C++

Une partie importante du travail de l'informaticien est de réaliser des programmes. De nombreux langages de programmation sont utilisés en entreprise, un des plus courant est le langage C et son évolution le C++. C'est pour cette raison que nous allons découvrir ce langage qui est le socle de construction de nombreux langages comme le java, le PHP et d'autre tout aussi utiles.

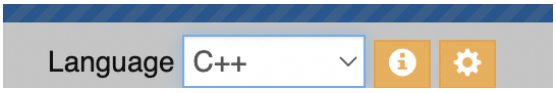
Le langage C a été conçu en 1972 aux laboratoires Bell Labs. C'est un langage structuré et modulaire, dans la philosophie générale de la famille Algol. Mais c'est aussi un langage proche du système, qui a notamment permis l'écriture et le portage du système Unix. Par conséquent, la programmation orientée système s'effectue de manière particulièrement aisée en C, et on peut en particulier accéder directement aux fonctionnalités du noyau Unix.

Le C++ peut être considéré comme un successeur du C. Tout en gardant les points forts de ce langage, il corrige certains points faibles et permet la programmation objet.

Premier programme C++

Pour notre découverte du C++, nous allons utiliser le site www.onlinegdb.com qui à l'avantage de compiler un code C++ simple puis de l'exécuter dans la foulée. Il aura l'avantage de simplifier son utilisation.

Tout d'abord connectons nous sur le site, et précisons que nous allons programmer en C++ dans l'option an haut à gauche du site :



Comme vous le voyez dans la capture suivante, le site propose le traditionnel « Hello World » avec toute la structure classique d'un programme C++

```
/*******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <iostream>

using namespace std;

int main()
{
    cout<<"Hello World";

    return 0;
}
```

Import de la bibliothèque d'entrées-sorties. (console)

Espace de noms pour la portée des variables, constantes et fonctions.

En C++, la fonction `main()` représente le programme principal. Elle retourne toujours 0 en cas de fin normale du programme ou 1 en cas d'erreur.

L'affichage en console se fait par la commande `cout <<` . On affiche la phrase entre les guillemets.

Toutes les instructions se terminent par un point-virgule.

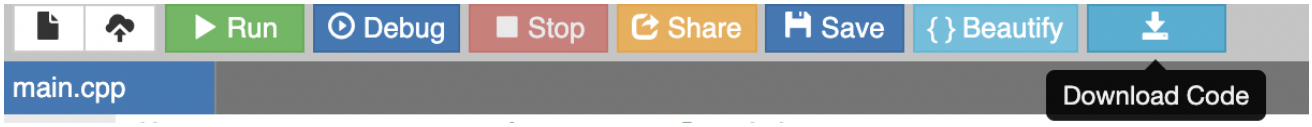
La saisie des variables

En C++, les variables doivent être déclarées et typées :

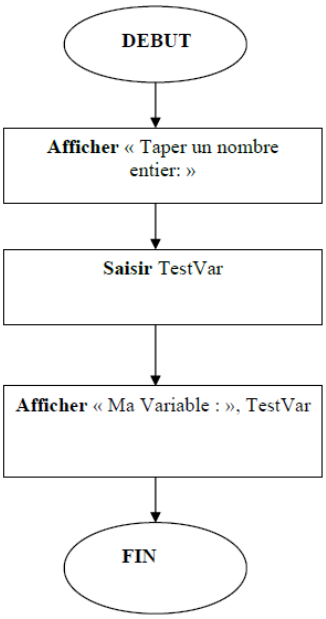
- `int` pour un entier (un nombre sans virgule)
- `char` pour un caractère (toutes les touches du clavier)
- `float` pour un réel (un nombre avec une virgule)
- `bool` pour un nombre binaire (valeurs possibles : true ou false).

La saisie de ces variables se fait par l'instruction `cin >> NomDeLaVariable`.

Récupérez votre programme avec le bouton « Download Code » dans le menu suivant :



Et testez le code ci-dessous.




```c
/* La saisie des variables */

#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    int TestVar;

    cout << "Saisissez un nombre entier : ";
    cin >> TestVar;
    cout << "Ma variable : " << TestVar << endl;

    return EXIT_SUCCESS;
}
```

Import de la bibliothèque qui permet une gestion dynamique de la mémoire et fournit des fonctions de calcul, conversion, ainsi que des constantes utiles.

L'instruction `<<endl` permet d'ajouter une fin de ligne (end of line).

Exercice 1

En utilisant le même principe, faites le test pour un réel, un caractère et un booléen

Structures conditionnelles

Récupérez votre programme et testez le code suivant:

```cpp
/* Les structures de controle */

#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    int TestVar;

    cout << "Saisissez un nombre entier : ";
    cin >> TestVar;

    if (TestVar < 10)
    {
        cout << "Ma variable est inférieure à 10 " << endl;
    }
    else
    {
        cout << "Ma variable est supérieure à 10 " << endl;
    }

    return EXIT_SUCCESS;
}
```

Exercice 2

En utilisant le même principe, déterminez si le nombre est pair ou impair.

La boucle FOR

Indiquez pour les 2 exemples de boucles for ci-dessous les résultats obtenus :

```cpp
/* La boucle FOR (1)*/

#include <iostream>
using namespace std;

int main()
{
    int i;
    for(i=0;i<10;i=i+1)
        cout<<"BONJOUR"<<endl;
    return 0;
}
```

```cpp
/* La boucle FOR (2)*/

#include <iostream>
using namespace std;

int main()
{
    int i;
    for (i=0; i<10; i=i+1)
        cout<<"La valeur de i est : "<<i<<endl;
    cout<<"La valeur finale de i est : "<<i<<endl;
    return 0;
}
```

Résultats obtenus :

La boucle WHILE

Indiquez pour les 2 exemples de boucles while ci-dessous les résultats obtenus :

```cpp
/* La boucle WHILE (1)*/

#include <iostream>
using namespace std;

int main()
{
    int i=0;
    while(i<10)
    {
        cout<<"La valeur de i vaut : "<<i<<endl;
        i++;
    }
    cout<<"La valeur finale de i vaut : "<<i<<endl;
    return 0;
}
```

```cpp
/* La boucle WHILE (2)*/

#include <iostream>
using namespace std;

int main()
{
    int i=-1;

    while (i<0 || i>20)
    {
        cout <<"Tapez une valeur entre 0 et 20 bornes incluses : ";
        cin >> i;
    }
    cout << "Vous avez saisi : " << i << endl;
    return 0;
}
```

Résultats obtenus :

Exercice 3 :

Le programme python suivant permet de générer un nombre aléatoire entre 1 et 100 puis demande à l'utilisateur de saisir un nombre et le guide avec « Plus petit » ou « Plus grand ». Le joueur à 5 tentatives pour trouver le nombre sinon il a perdu.

Vous trouverez le fichier devi.py dans le répertoire CLASSE.

A faire :

Inspirez vous de ce code pour réaliser le même programme en C++.

Aide :

Pour générer un nombre aléatoire en C++, vous aurez besoin des lignes suivantes :

```
1  /* Devi nombre */
2
3  #include <cstdlib>
4  #include <ctime>
5  #include <iostream>
6
7
8  using namespace std;
9
10 int main()
11 {
12
13     std::srand(std::time(nullptr));
14     int nba = (rand() % 100) + 1;
```

Import de la bibliothèque time

Initialisation du random puis génération d'un nombre aléatoire entre 1 et 100.