Langage de programmation C++

Master 2 - Génie Industriel
Parcours - Industrie Numérique

Université de Picardie Jules Verne
INstitut Supérieur des Sciences Et Techniques (INSSET)
Saint-Quentin

Pr Mohamed Guessasma

Septembre 2025

Pourquoi ça ne marche pas ????

C’est fou comme c’est épuisant la programmation !!!

Pourquoi ça ne marche pas ????

C’est fou comme c’est épuisant la programmation !!!

Plan

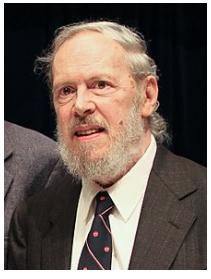
Historique

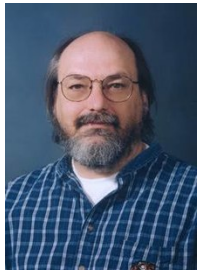
L'informatique nous passionne et nous voudrions apprendre à programmer ("à coder")? La programmation peut paraître compliquée au premier abord mais c'est un outil beaucoup plus accessible qu'on ne l'imagine ! Les programmes sont à la base de l'informatique. Ce sont eux qui nous permettent d'exécuter des instructions sur un ordinateur.

Historique

- Les années 1940: Plankalkül, ENIAC,...
- Les années 1950 et 1960: Fortran, Cobol, Basic,...
- Les années 1967 à 1978: C, ML, Pascal, SQL,...
- Les années 1980: C++, Perl,...
- Les années 1990 (l'ère Internet): Python, JavaScript, PHP,...

John-Backus
inventeur du Fortran





Dennis Ritchie et Ken Thompson
inventeurs du C et système Unix

Bjarne Stroustrup
inventeur du C++

Langages de programmation

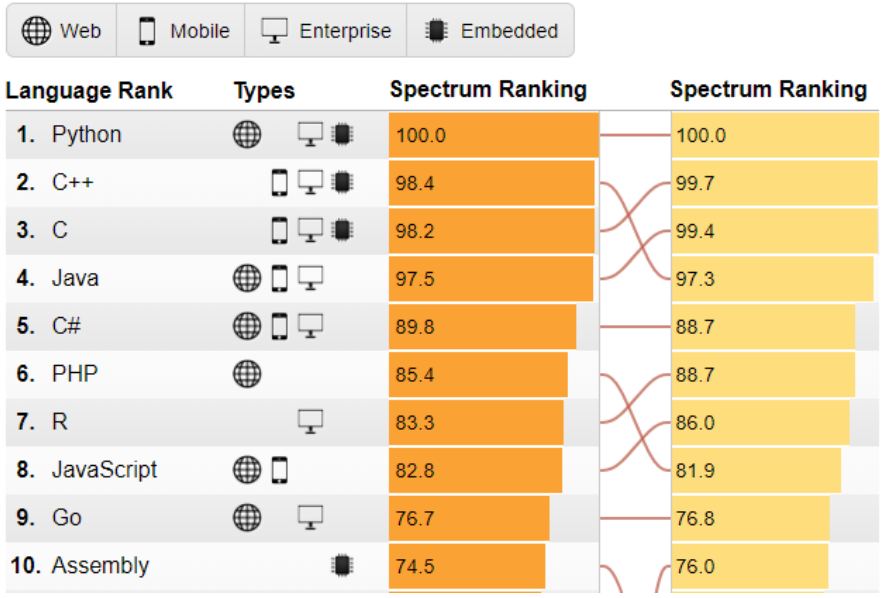
- Les lignes de code source (SLOC en anglais) qui composent un programme sont à la base de l'informatique. Ce sont ces lignes qui permettent d'exécuter des instructions sur un ordinateur. Ex. un jeu vidéo nécessite des dizaines de développeurs à plein temps !

Jeu vidéo Mario & Sonic

- Tout un chacun est en mesure de développer un programme informatique, faut-il encore avoir de bonnes connaissances en algorithmique et un sens de la logique bien aiguisé! Une multitude de programmes trouve un intérêt dans différents secteurs: technologique, médical, logistique, sécurité, bancaire, ressource humaine,...

Popularité des langages de programmation

- Il existe plusieurs langages de programmation: C, Java, C++, Phyton sont les langages les plus utilisés par les développeurs :

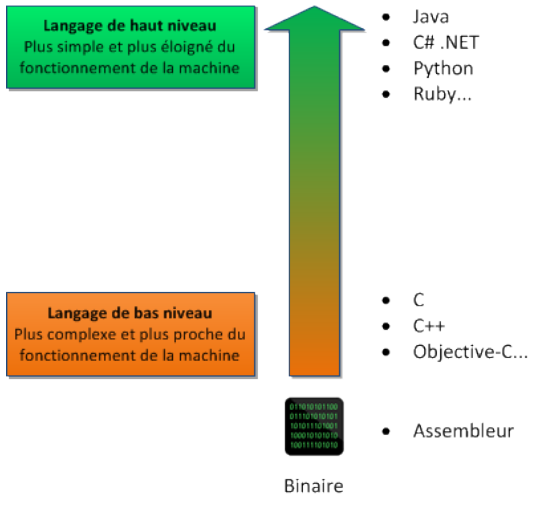


Source IEEE

- Faut-il choisir un langage selon le seul critère de popularité ? Bien qu'il existe des langages très performants mais peu utilisés, le choix d'un langage utilisé par un grand nombre de de personnes est toutefois à privilégier (aide, conseil, forum,...).

Catégories de langage

Au delà du critère de popularité, un langage de programmation est choisi par / à son niveau. ∃ des langages de haut niveau et de bas niveau.



Catégories de langages

Un langage de haut niveau est assez éloigné du binaire, il permet de développer de façon plus souple et rapide. Alors qu'un langage de bas niveau demande plus d'efforts et donne plus de contrôle.

Plan

C++

L'ancêtre du **C++** est le langage ALGOL (ALGorithmic Oriented Language). Insatisfait des possibilités offertes par le C, Bjarne Stroustrup a créé en 1983 le **C++** en y ajoutant les possibilités qui, selon lui, manquaient. Le symbole ++ est l'opérateur d'incrémentation, augmentation de la valeur d'une variable de 1.

Pourquoi le **C++** plutôt qu'un autre langage? Quelles sont les forces et les faiblesses du **C++**?

- ✔ Très répandu et très rapide
- ✔ Portable (Windows, Mac OS et Linux)
- ✔ Nombreuses bibliothèques
- ✔ Multi-paradigmes (programmation procédurale ou POO)
- ✘ Langage complexe

Intérêt de la programmation

- À l'instar d'autres langages de programmation, le C++ ne réalise pas uniquement un ensemble d'opérations basiques et une série de déclarations, il nous permet aussi de réaliser nos propres opérations et déclarations.
- On veut dire par opérations et déclarations, respectivement fonctions et procédures auxquelles fait appel un programme donné.
- Les fonctions et procédures (subprograms : sous-programmes) utilisent des paramètres dont le progarmme se sert pour effectuer les tâches qu'il contient.

Subprograms : Définition d'une fonction

- Fonction avec paramètres et retour de résultat (entier, réel, chaîne de caractères, tableau...)

```
1  #include<stdio.h>
2  #include<iostream>
3  // A comment
4  int main(void)
5  {
6  printf("Hello World\n");
7  return 0;
8  }
```

Function

```
1  int times(int x, int y) {
2  int p = 0;
3  // While loop
4  if (y%2 == 0) {
5   y = y/2;
6   x = x*2;
7           }
8  else {
9   p = p + x;
10  y = y - 1;
11          }
12 }
13 return p;
14 }
```

Compilation sous linux

La compilation permet de traduire un programme écrit dans un langage source (langage lisible) en un programme écrit dans un langage cible, exécutable par une machine.

- La compilation d'un programme écrit en **C++** nécessite le compilateur g++ ()
- Compilation: g++ loop.cpp
- Exécutable: a.out
- Lancement de l'exécutable: ./a.out loop.cpp
- Makefile

Aperçu C++ : Hello World !

Pour se donner une idée simple du langage **C++**, le programme suivant permet d'afficher l'expression « Hello World! » à l'écran:

```
1  #include <iostream>
2  //En-tete standard pour les in out
3  using namespace std;
4  // Espace de "nommage" standard
5
6  int main()
7  {
8  std::cout <<"Hello World!"<< endl;
9  std::cout <<"I'm a C++ ! "<< endl;
10 std::cout <<"Your favorite" << endl ;
11 std::cout <<"programming language"<< endl;
12 return 0;
13 }
```

Aperçu C++ : Boucle (Lopp)

```
1  #include <iostream>
2  using namespace std ;
3  int main() {
4  // Variable Declaration
5  int a;
6  //Get Input Value
7  cout << "Enter the Number :";
8  cin >> a;
9  // for Loop Block
10 for (int counter = 1; counter <= a; counter++)
11 cout << "Execute " << counter << " time" $<<$ endl;
12 return 0;
13 }
```

Aperçu C++ : Nested if...else (si...sinon imbriqués)

```
1  #include <iostream>
2  using namespace std ;
3  int main() {
4  int number;
5  // Get Input Value
6  cout << "Enter an integer: ";
7  cin >> number;
8  if ( number > 0)
9  {cout << "Positive integer: " << number << endl;}
10 else if (number < 0)
11 {cout << "Negative integer: " << number << endl;}
12 else {cout << "You entered 0" << endl;}
13 cout << "This line is always printed." << endl;
14 return 0;}
```

Aperçu C++ : While (tant que)

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4  int number, i = 1, factorial = 1;
5  cout << "Enter a positive integer: ";
6  cin >> number;
7  while ( i <= number ) {
8          factorial = factorial * i;
9                  ++i;}
10 // Get Output Value
11 cout << "Factorial of " << number << " = " << factoria
12 return 0;
13 }
```

Plan

Compilation de programmes source

Compilation d'un programme source en **C++** :

- en ligne de commande, ex: g++(+ version number − 4.8) main.cpp (création de l'exécutable a.out)
- ou avec un makefile dont l'utilité est de créer un exécutable nommable (ex: toto, run, execute,...) à partir d'un code source. Utile surtout quand il y a plusieurs programmes sources à compiler (main, function, bibliothèque,...)

Exemple de makefile

```
CC = g++(+ version number  − 4.8)
SRC = $(wildcard */*.cpp)
OBJS = $(SRC:.cpp=.o)
AOUT = run
all : $(AOUT)
run : $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $^
%.o : %.c
        $(CC) $(CFLAGS) -o $@ -c $<
clean :
        @rm */*.o
cleaner : clean
        @rm $(AOUT)
```

Plan

décimal: 13 ≡ binaire: 1101

Création d'un programme en C++ qui permet de convertir un nombre décimal en nombre binaire et vis-versa.

1. Programme avec main (principal)
2. Programme avec main et function
3. Programme avec main, function et header

$[n \times n]^{-1}$

Inversion d'une matrice carrée [n×n] à l'aide de la librairie Eigen.
Objets utilisés:

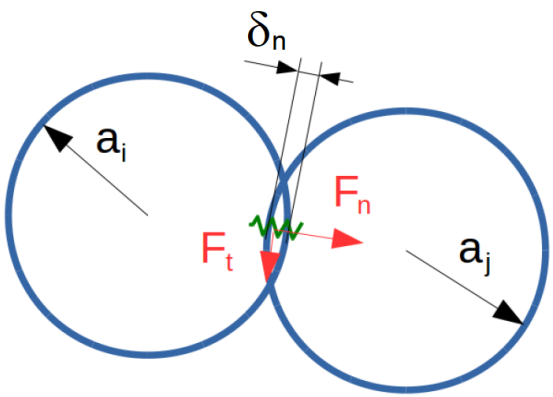
1. Matrix2f: matrice 2×2
2. A.determinant(): calcul du déterminant
3. A.inverse(): calcul de l'inverse
4. A × A.inverse(): produit de la matrice avec son inverse
5. ...

Plan

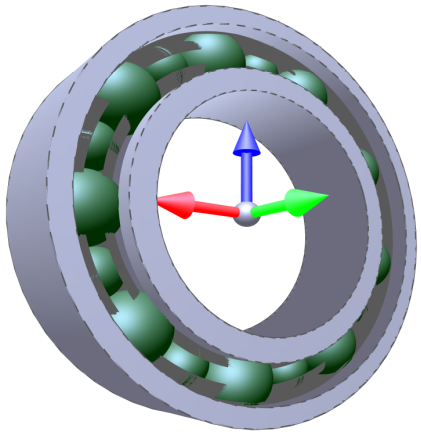
Méthode des Éléments Discrets - MED

C'est quoi la MED?

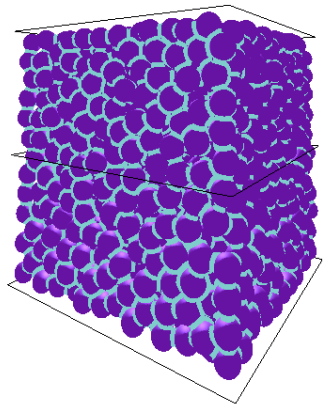
Modélisation du comportement des milieux discrets, divisés, multi-contacts (ex. sable, neige, produits pharmaceutiques, grains de blé, gravier,...) constitué d'un grand nombre de particules ($\times 10^3 \leq Nb_{particules} \leq \times 10^6$) interagissant entre elles.



Modèle de contact



Roulement à billes



Forces capillaires

Méthode des Éléments Discrets - MED

Principe de la MED

1. Modèle de raideur au contact (ressort + amortisseur)
2. Loi de frottement (Coulomb)
3. Détection des contacts
4. Résolution de la 2^e loi de Newton pour chaque particule
5. Actualisation des positions des particules

Merci de votre attention