Partie 2 : présentation du langage C

- Un bref historique
- Bibliographie
- Pourquoi les normes
- Pourquoi le C
- Du python au C

Bref historique

- 1967 Langage BPCL (Richards, Bell Labs)

  Basic Combined Programming Language
- 1970 Langage B (Thompson, BL)

  Pas de types, manipulation de mots machine -> programmes non portables
  Structures de contrôle, pointeurs, récursivité.
- 1972 1er compilateur C (Kernighan, Ritchie, BL)
- 1978 1ère spécification publique du C
- 1989 Norme « ANSI » -> C « ISO » ou « ANSI »
- 1999 Norme ISO/IEC 9899 : C99.

Bibliographie

- A. Braquelaire. Méthodologie de la programmation en langage C. Norme C99, API POSIX. Dunod, Paris, 4ème éd. 2005.
- C. Delannoy, Programmer en langage C – Cours et exercices corrigés, Eyrolles, 2006
- B. Kernighan et D. Ritchie, Le langage C, norme ANSI, Masson Paris 1990
- I. Horton, Beginning C, Apress, 2006
- J.-M. Léry, Le langage C, Pearson Education, 2005

Pourquoi des normes

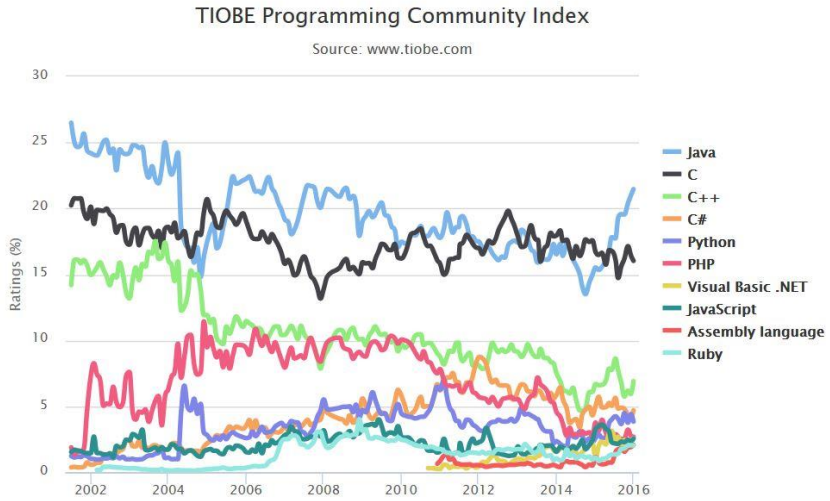
- Plusieurs vendeurs ont développé leur propre version du langage C.
  -> Ils ont proposé des extensions incompatibles entre elles.

- Norme : fournit une signification précise et unique des programmes.

- Un programme qui respecte la norme C99 est portable : on peut le déployer sur n'importe quelle architecture actuelle.

- Les programmes ne respectant pas la norme C99 seront (ici) considérés comme incorrects.

Pourquoi C


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
Java
C
C++
C#
Python
PHP
Visual Basic .NET
JavaScript
Assembly language
Ruby

Pourquoi C

Toujours autant utilisé

- Portable : Un programme C conforme aux normes est utilisable sur une grande variété de machines.
- Puissant : Nombreux domaines d'application OS (Unix), réseau, BD, graphique...
- Efficace : Permet de développer des programmes rapides.
- Haut niveau : Détails hardware cachés au programmeur.
- Souple : Très permissif, accès à la mémoire.

-> facile à aborder, difficile à bien maîtriser.

Du python au C

- C est un langage compilé
- Typage statique (chaque variable, constante et expression a un type défini à la compilation)

Du python au C

```
f: x->x²+x+1
```

```
def f(x):
    return x*x+x+1
```

Entier, réel, etc ?

L'instruction est-elle finie ?

Entier, réel,etc?

Le code de la fonction est-il fini ?

Le programmeur doit être explicite C ne déduit rien

Du python au C

f: x->x²+x+1

```
int f (int x)
{
 return x*x+x+1;
}
```

Du python au C

```
def pair(n):
  p=0
  i=1
  if n%2==0:
    print(''Pair'')
    return p
  else :
    print(''Impair'')
    return i
```

Entier, réel, etc ?

Entier, réel, etc ?

Où s'arrête le bloc du if ?

Où s'arrête le bloc du else?

Où s'arrête chaque instruction ?

Où s'arrête le code de la fonction ?

Du python au C

```
int pair(int n){
  int p=0;
  int i=1;
  if (n%2==0) {
    printf(''Pair'');
    return p; }
  else {
    printf(''Impair'');
    return i;}
}
```