

# Langage C

*Cours 1 - 2021*

*Introduction*



`#include<stdio.h>`

# L'ÉQUIPE PÉDAGOGIQUE

**RESPONSABLE : SYLVIE LEBRUN**

## INTERVENANTS PERMANENTS

- Julien VILLEMEJANE
- Xavier DELEN

## INTERVENANTS

- Anne NGUYEN
- Amal ZIDI
- Marc CHAMMAS
- Antoine TENART
- Maha BOUHADIDA

# OBJECTIFS

## PROGRAMMATION / LANGAGE C

- Introduction au langage C par l'exemple
- Langage de base
- Réalisation d'un mini-projet autour des images

## SYSTÈME À MICROPROCESSEUR

- Fonctionnement d'un système à microprocesseur
- Programmation bas niveau
- Gestion de la mémoire
- Vers des langages plus évolués

# DEROULEMENT

*Enseignement distanciel certain après la  
Toussaint. Avant, en fonction de la  
situation ...*

## SÉANCE 1

- Bases, prise en main de Code:Blocks

## SÉANCES 6-7

- Notions avancées
  - Pointeurs
  - Allocation dynamique
- Présentation mini-projet

## SÉANCES 2-3-4-5

- Notions de base
  - Tableaux 1D
  - Fonctions
  - Gestion de fichiers
  - Chaînes de caractères

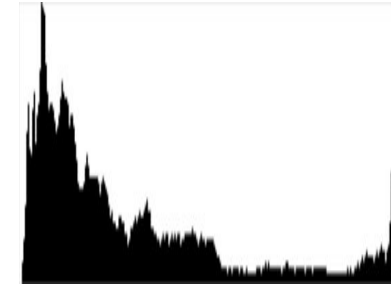
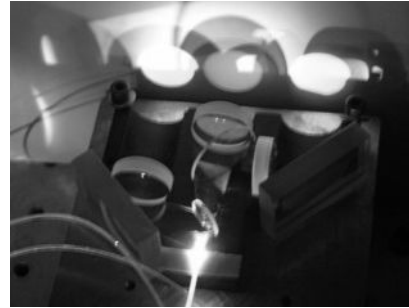
## SÉANCES 8-9-10-11

- Projet en groupe

## SÉANCE 12

- Présentation orale

# A PARTIR DE LA SÉANCE 8



## MINI-PROJET / TRAITEMENT D'IMAGES

- Travail en groupe

### OBJECTIF PRINCIPAL

### OBJECTIFS INTERMÉDIAIRES

- |  |   |
|--|---|
| <ul style="list-style-type: none"><li>- CALCULER L'HISTOGRAMME</li><li>- Ouvrir le fichier ASCII en lecture</li><li>- Afficher l'en-tête du fichier</li><li>- Ré-afficher l'image en console</li><li>- Créer un tableau <b>histogramme</b></li></ul> | <ul style="list-style-type: none"><li>- AFFICHER L'HISTOGRAMME</li><li>- Créer un fichier PGM</li><li>- Ecrire l'en-tête du fichier</li><li>- Créer une image DAMIER<ul style="list-style-type: none"><li>- N x N cases de K pixels chacune</li></ul></li><li>- Créer une image à partir d'un tableau 1D (type histogramme)</li></ul> |
|--|---|

# EVALUATION

## PROJET

- Présentation
- Evaluation en séance

## CRITÈRES

- Lisibilité du code : commentaires, indentation, variables...
- Explications claires
- Qualité du code : algorithme efficace
- Réutilisation possible du code : modularité

## EXAMEN écrit

Mercredi 12 janvier 2022

- QCM
- Problème

# GENERALITES

## CRÉÉ EN 1970

- Dennis Ritchie
- Brian Kernighan



**UNIX®**

## MULTIPLATEFORME

- Windows, Mac, Linux
- Compilateurs disponibles pour de nombreux processeurs et microcontrôleurs

## LANGAGE INTERMÉDIAIRE

- **Haut niveau** : compréhensible par l'être humain
- **Bas niveau** : suffisamment proche de la machine (fonction bas niveau)

# POUR QUOI FAIRE ?

## MANIPULATION DE DONNÉES

- Stockées dans des espaces mémoires
- Typées / Adressables par octet ( $1\text{o} = 8\text{ bits}$ )

## INTÉRÊTS

- Rapidité d'exécution / Code compilé
- Protection intellectuelle du code source

## UTILISATION

- Traitement de données
- Flux de données importants (vidéo...)
- Systèmes embarqués / grande réactivité

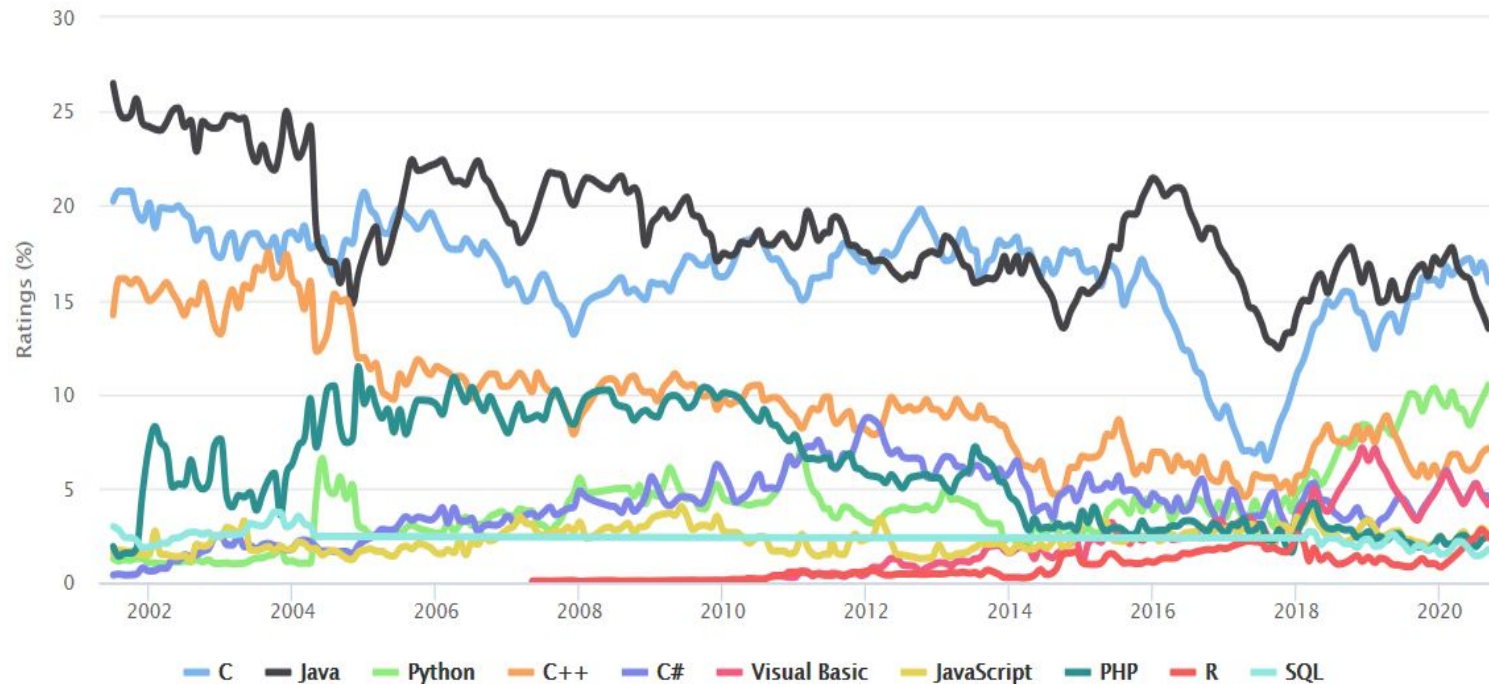




# UN LANGAGE RÉPANDU

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# DÉVELOPPEMENT LOGICIEL

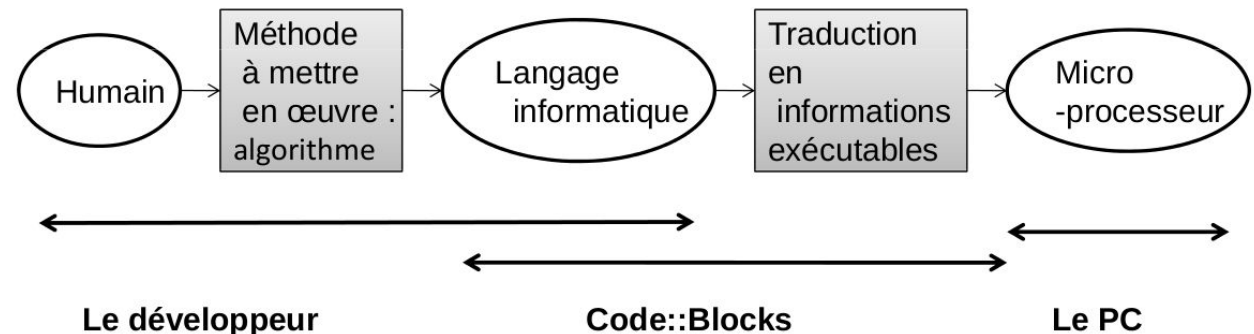
## EN 2 PHASES

### RÉSOLUTION

- Compréhension du problème
- Ébauche de solution
- Liste des variables nécessaires
- Solution détaillée / Algorithme
- Essais pas à pas avec des données

### MISE EN OEUVRE

- Traduction dans un langage
- Ecriture du programme
- Correction syntaxique
- Essais avec des données



# LANGAGE COMPILÉ / PROCEDURAL

## CODE SOURCE

- Syntaxe particulière / Typage de données
- Modularité / Réutilisation grâce à des fonctions/procédures / Bibliothèques existantes

## COMPILATION

- Nécessite un compilateur par type de calculateur
- Difficilement portable / C Standardisé
- Protection intellectuelle du code source

## EXÉCUTION

- Rapidité d'exécution

# Comparaison des langages informatiques / Suite de Fibonacci

Calcul des **30 premiers termes de la suite de Fibonacci**  
Itération de **1M** de fois le calcul – sans affichage

## Python

```
fibonacci_numbers = [0, 1]

for i in range(2,MAX):
    fibonacci_numbers.
        append(fibonacci_numbers[i-1] +
fibonacci_numbers[i-2])
```

## MATLAB

```
Fibonacci = zeros(1,MAX);

Fibonacci(1) = 0;
Fibonacci(2) = 1;

for i = 3:MAX
    Fibonacci(i) = Fibonacci(i-1) +
Fibonacci(i-2);
end
```

## C

```
int fibonacci[MAX];
short compteur;

fibonacci[0] = 0;
fibonacci[1] = 1;
for(compteur = 2; compteur < MAX; compteur+
+)
{
    fibonacci[compteur] = fibonacci[compteur - 2]
+ fibonacci[compteur - 1];
}
```

Exécuté sur Intel Core i5-4258U / 6Go RAM DDR3 1600 / Xubuntu 16.04  
Exécuté avec Python 3.5 / MATLAB 2016b / Compilé avec GCC 5.4.0

## Résultats

Moyenne de 10 exécutions

**7,57 s**  
0,24 s

**0,27 s**  
0,01 s

**0,13 s**  
0,03 s

# DURANT LES SÉANCES...

