**CM 1**

Langage compilé (code source transformé en code machine pour être interprété)

Langage de haut niveau du point de vue du système (en comparaison avec assembleur) mais bas niveau comparé à des langages objets. Très utilisé, proche de la machine. On peut directement manipuler la mémoire à la main en C.

**Tous les os sont codés en C**

**Avantages** :

* Plus de contrôle sur l'exécution du programme (proche système, haut niveau)
* Plus de possibilité d’optimisation de perf
* Permet de bosser sur l’archi (car bas niveau) type drivers etc…

**Inconvénients** :

* Nécessité de gérer la mémoire
* Prend plus de lignes de code pour accomplir une tâche
* Plus de possibilités de faire des erreurs

-> Langage puissant mais dangereux, pas utile à moins de devoir faire des traitements seulement réalisables en C

**Différences entre C et Java :**

* Pas d'exceptions
* Pas de classe, de package, d’interfaces
* Pas de garbage collector

Syntaxe du C <https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/7669321-ecrivez-votre-premier-programme>

Pour compiler un programme en C, il faut 3 outils

* **Pré-processeur :** s’occupe de réécrire le code par rapport aux macros (#)
* **Compilateur :** transforme le code en code assembleur (lisiblepar la machine)
* **Linker :** gère les librairies et les dépendances

**CM 2**

Comment définir un type ? :

* Domaine de valeurs
* Type d’opérateurs

→ **Void** : pas de domaine de valeurs, pas d’opérateurs applicables

→ **Int** : domaine de valeurs : les entiers, opérateurs sur les nombres

→ **short** : deux fois moins de bits que le int

→ **long** : deux fois plus de bits que le int

→ **float** : nombre à virgule

→ **double** : deux fois plus de bits que le float

→ **Char** :

3 types d'éléments qu’on peut regrouper en C :

* **Arrays (tableaux) :** listes ordonnées d’éléments -> initialisation : int T[5]. Pas d’opérateurs globaux type == ou + sur eux
* **Structures :** assimilable aux objets Java mais sans les méthodes. Les paramètres et valeurs retournées sont copiés (pas d’effet de bord). On ne peut pas utiliser de == sur les structures.
* **Énumérations :** permettent de grouper des valeurs connexes. Par exemple, une valeur couleur peut prendre SEULEMENT les valeurs Bleu, Rouge, Blanc et Jaune. On aura des erreurs de compilation si on a oublié une valeur dans un switch. On peut utiliser tous les opérateurs logiques et arithmétiques. On peut par exemple faire white+1 et on aura l’indice d’après dans l’enum. (possible aussi en Java mais plus chiant et compliqué à utiliser)

Mémoire en C :

-> **Première règle :** les éléments successifs sont stockés dans l’ordre dans la mémoire. De son côté le compilateur sera libre d’ajouter de l’espace pour respecter les contraintes d’alignement.

Alias :

-> on utilise typedef pour déclarer les variables de type (définir des alias).

**Syntaxe :** typedef struct point **point\_t (alias pour tout le reste dans la ligne)**

-> Inclusion du header : on liste les prototypes de fonctions qu’on va vouloir utiliser. Ces lignes commencent par # et sont appelés headers.

-> #include <file> (portée : système) | #include “file” (portée : projet)