Langage C : objectifs

- **Les objectifs du cours :**
  - **Objectif général :** Ecrire et exécuter des applications simples en langage C.
  - **Objectifs proposés :**
    - Compiler et exécuter un programme simple.
    - Utiliser la bibliothèque standard du langage.
    - Déclarer des constantes et des variables de types élémentaires, utiliser les opérations associées (déclaration, portée, affectation, opérateurs).
    - Utiliser les structures de contrôle du langage (if, switch, for, while).
    - Passer des adresses de variables (pointeurs) en paramètre de fonction.
    - Manipuler des tableaux.
    - Manipuler des chaînes de caractères, afficher et récupérer des informations sur le terminal.
    - Manipuler des types énumérés et des structures.
    - Ecrire une application à plusieurs modules, en écrivant des headers appropriés.
    - Lire et écrire des fichiers de données.
    - Effectuer des allocations dynamiques (tableaux, structures).

Séance 1 : Eléments de langage C

Langage C : séance 1

Construction d'un programme :

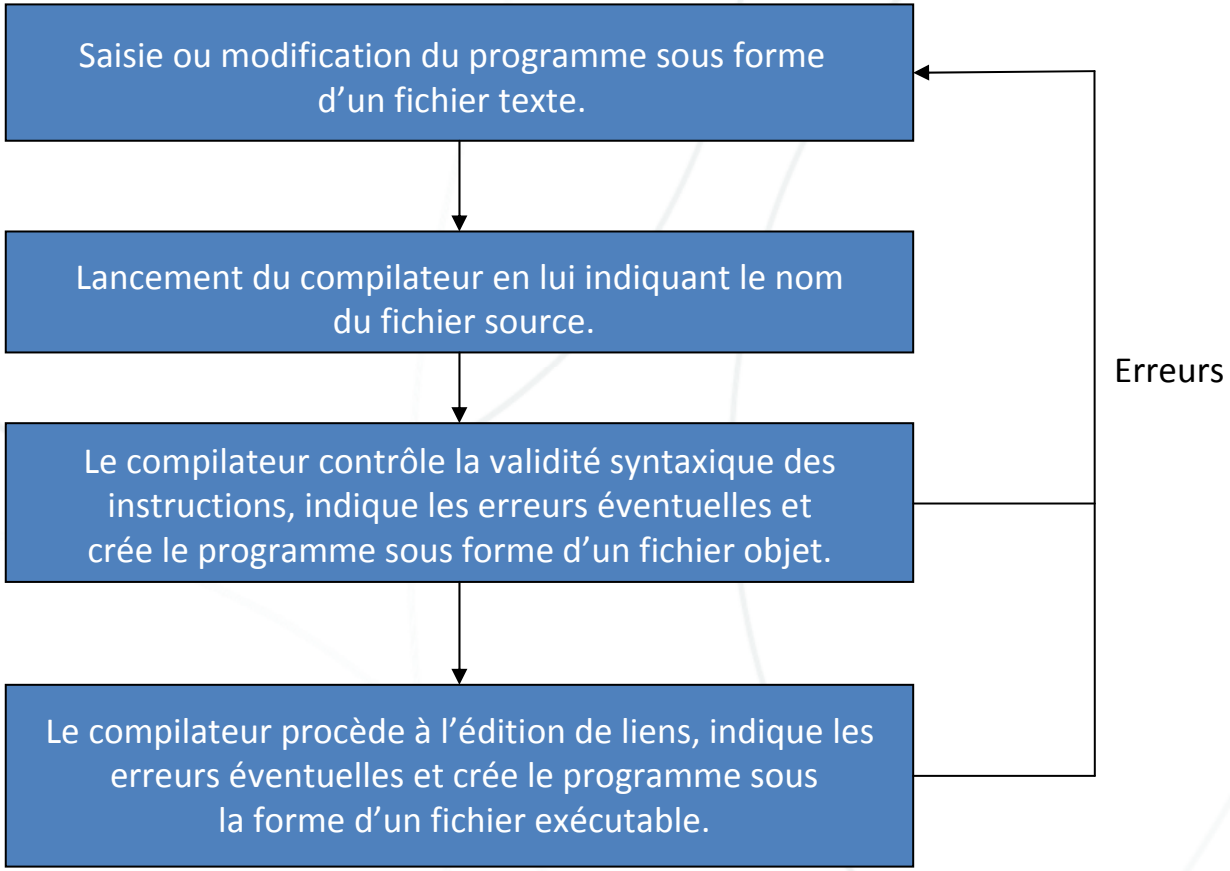
2 méthodes pour traduire un programme :

- L'interprétation. Les instructions du programme sont traduites une à une et exécutées (pas de fichier exécutable généré) : langage Php, Python.
- La compilation. Tout le programme source est traduit en instructions machines (fichier exécutable) : langage C, C++.

Il existe des méthodes « dites intermédiaires » : compilation (génération d'un « pseudo-code » ou bytecode, fichier « *.class ») puis interprétation➔langage Java.

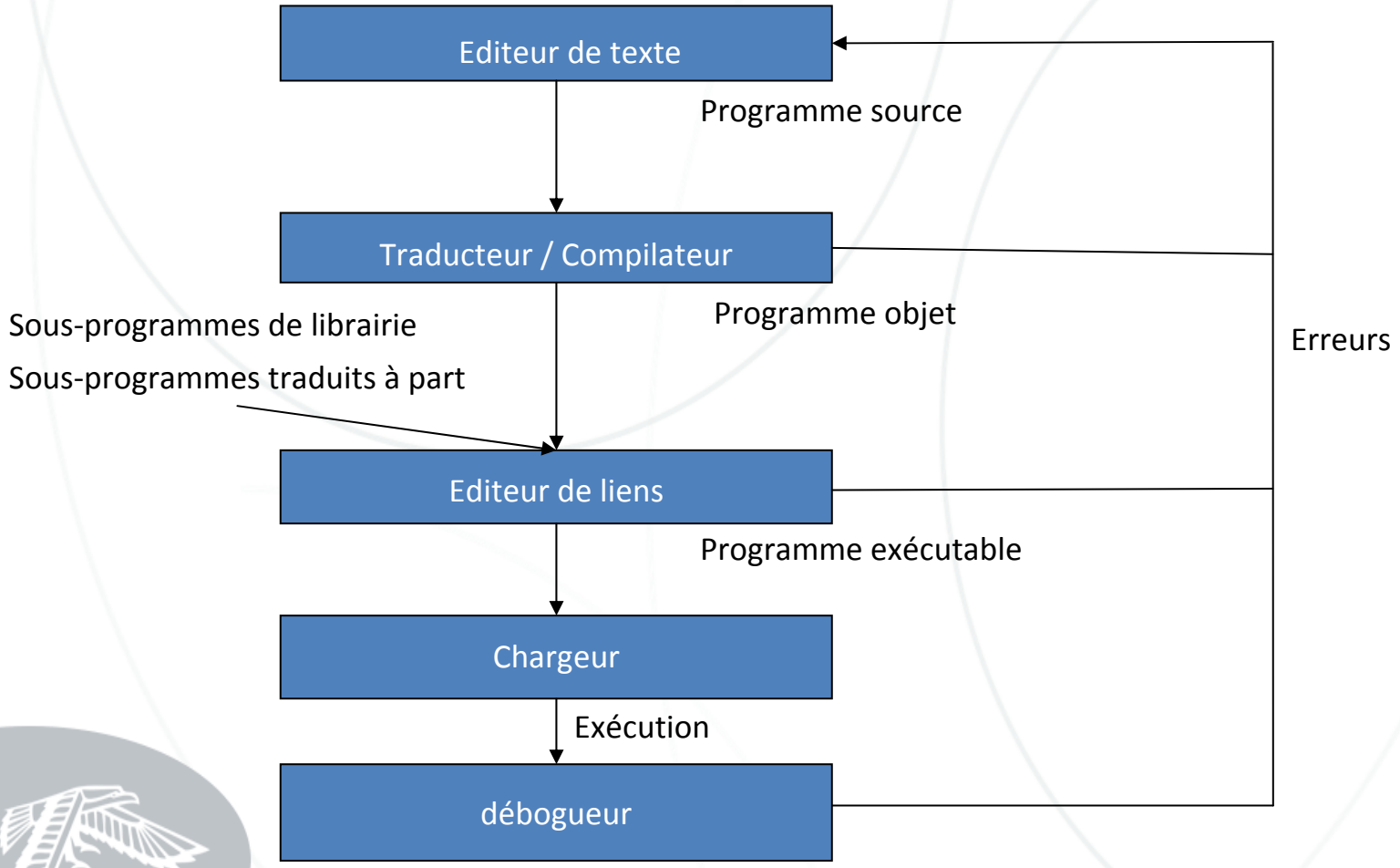
Langage C : séance 1

La programmation : du source à l'exécutable



Langage C : séance 1

Environnement de programmation minimum :



Langage C : séance 1

Définitions :

Compilateur :

programme qui traduit le code source écrit dans un langage de haut niveau (facilement compréhensible par l'humain) vers un langage d'assemblage ou langage machine, généralement représenté sous forme binaire (code objet ou bytecode).

Editeur de liens :

programme qui permet de créer des fichiers exécutables ou des bibliothèques dynamiques ou statiques, à partir de fichiers objets.
Lien statique : le fichier objet et la bibliothèque sont liés dans le même fichier exécutable.
Lien dynamique : le fichier objet est lié avec la bibliothèque, mais pas dans le même fichier exécutable (liens établis lors du lancement de l'exécutable).

Langage C : séance 1

Chaque programme chargé en mémoire dispose de plusieurs segments :

- le segment de code, contenant les instructions du programme en langage machine.
- le segment de données, contenant les données déclarées comme **globales** dans le programme.
- le segment de pile, permettant d'exécuter des programmes contenant des sous-programmes, et stockant le contexte d'exécution du sous-programme, ses **paramètres** ainsi que ses **variables locales**.
- le tas, ou mémoire dynamique, pour stocker des données. La taille de cette zone varie au cours de l'exécution du programme.

Langage C : séance 1

Les différents blocs mémoire utilisés par un programme :

Program	• Code machine • Données statiques globales •Liens avec bibliothèques
Heap	• Tas (➔ malloc)
Free memory	• Mémoire libre (utilisée pour la croissance du tas et de la pile)
links	• Liens dynamiques : Pour charger dynamiquement des bibliothèques.
Stack	• Pile : Variables locales de chaque fonction, adresse de retour, …

Langage C : séance 1

Caractéristiques générales du langage C :

- Un programme écrit en langage C se stocke dans un fichier texte ayant l'extension « .c ».
- Ce fichier doit posséder la fonction principale d'entrée : la fonction « main »
- Pour utiliser des fonctions d'entrées/sorties du langage C, on doit inclure un fichier standard avec une directive au préprocesseur.
- Exemple de programme en langage C :

```
#include <stdio.h>
int main()
{
    printf("bonjour");
    return 0;
}
```

bloc d'instructions

Instruction « printf » du fichier « stdio.h »

Instruction « return » du langage c

Langage C : séance 1

Caractéristiques générales du langage C :

- Distinction entre minuscules et majuscules, les mots réservés du langage en minuscules.
- Notion de déclaration de variables : toute variable est typée et déclarée avant son usage, en début de bloc.
- Notion d'affectation symbolisée par le signe =.
- Les commentaires :

  // jusqu'à la fin de ligne

  ou

  /* sur plusieurs

  lignes

  ***/**

Rm : contrairement au langage Python, l'indentation en langage C n'est pas significative pour le compilateur.

Langage C : séance 1

Les types prédéfinis :

- **void** : aucune valeur.
  utilisé en retour de fonction, pour indiquer aucune valeur retournée.
  ex : **void** f1(); *// donc pas d'instruction **return** obligatoire dans f1*
- **char** : un caractère (1 octet).
- **short , int, long** : pour des entiers.
- **float, double :** pour des réels.
- **enum :** pour définir un type énumératif.

exemple :

typedef enum {lundi,mardi,mercredi, jeudi , vendredi ,samedi,dimanche} Semaine;
Semaine unJour = lundi;

- **pas de type booléen** : 0(zéro) c'est faux,
  toute valeur différente de 0 , c'est vrai.
- **typedef :** Permet de définir un nouveau type a partir d'un type existant.

Syntaxe : **typedef** typeExistant nouveauType;
Exemple : typedef int Euro;

Langage C : séance 1

Les valeurs constantes :

- Constante entière :

  En base dix : 12

  En base octale : 012 *// précédé de zéro*

  En base héxa : 0X12 *// précédé de zéro et x (ou X)*

- Constante réelle :

  plusieurs notations possibles : 3. , .4 , 2e4 , 2.3e4

- Constante caractère :

  entouré de quote : 'a'

  pour les caractères non imprimables : '\001' (valeur ASCII du caractère)

  et pour les caractères d'échappement : '\n', '\t', …

- Constante chaînes de caractères :

  entouré de guillemets : "bonjour"

  Le compilateur rajoute un caractère '\0' en fin de chaîne (marqueur).

Langage C : séance 1

Déclaration et initialisation des variables :

- Variable entière :

  int i; *// variable déclarée mais non initialisée*

  long j=0; *// variable déclarée et initialisée*

- Variable réelle :

  float varf = .4 ;

  double vard = 0 ;

- Variable caractère :

  char car = ‘a’ ;

- Variable chaînes de caractères :

  char chaine[20] ; *// taille précisée obligatoirement si non initialisée*

  char unechaine[]= "bonjour"; *// taille non obligatoire ici*

Langage C : séance 1

Déclaration de variables constantes :

Il existe deux méthodes :

- Première méthode :

Déclaration d'une variable, dont la valeur sera constante pour tout le programme:

const float PI=3.14159 ;

On utilise le mot réservé « const » et on écrit par convention les constantes en majuscules.

Dans ce cas, le compilateur réserve de la place en mémoire (ici 4 octets), pour la variable PI, mais dont on ne peut changer la valeur.

- Deuxième méthode :

Définition d'un symbole à l'aide de la directive de compilation **#define**

```
#define  PI  3.14159
float perimetre, rayon= 6;
perimetre =  2* rayon * PI;
```

Langage C : séance 1

Les Entrées/Sorties :

- Affichage écran :

  - On utilise la fonction « printf » de la librairie standard du langage C.

  ➔ Demander à utiliser la librairie standard des E/S :

  ```
  #include <stdio.h>    // à mettre en début de fichier
  ```

  - Paramètres de la fonction « printf » :

  ```
  printf( "bonjour" );
  ```
  ➔ bonjour

  Ou

  ```
  int i=8;
  printf("valeur de i=%d", i);
  ```
  ➔ valeur de i=8

  - Un format à spécifier pour chaque type de variable :

  ```
  %d          : entier
  %f  ou %lf  : float ou double
  %c          : char
  %s          : chaînes de caractères
  %p          : adresse (pointeur)
  ```

Langage C : séance 1

Les Entrées/Sorties : le cas particulier des chaînes de caractères

- Affichage écran : la fonction « printf »

  - On utilise la fonction « printf » de la librairie standard du langage C (stdio.h).

    printf("bonjour");

  - Si on souhaite stocker la chaîne dans une variable, on utilise un tableau de caractères :

    char chaine[]= "bonjour"; // *chaine ne pourra contenir que 8 caractères max ('\0' inclus).*

    Ou char chaine[10]= "bonjour"; *// taille de chaine : 10 max*

    Ou char chaine[10]; *// taille à préciser obligatoirement dans ce cas*

    strcpy(chaine, "bonjour"); *// fonction (de "string.h") qui recopie "bonjour" dans chaine*

    Puis pour l'affichage : printf(" %s ", chaine);

    Pour les sauts de ligne : on utilise le caractère spécial '\n'.

    printf(" %s \n ", chaine);

Langage C : séance 1

Les Entrées/Sorties : le cas particulier des chaînes de caractères

- Affichage écran : la fonction « puts »

  - On utilise la fonction « puts » de la librairie standard du langage C (stdio.h).

    puts(“bonjour”); *// saut de ligne automatique : équivalent à printf(“bonjour\n”);*

  - Pour afficher le contenu d’une chaîne de caractères :

    char chaine[]= “bonjour”;
    puts(chaine);

  Le paramètre de “puts” doit être **obligatoirement** une chaîne de caractères !

- Affichage d’un caractère : la fonction « putchar »

  Si la variable c est un **char** :

  putchar(c); *// équivalent à printf("%c\n",c);*

Langage C : séance 1

Quelques opérateurs de base :

• **Une expression** ➔ suite d'opérateurs et d'opérandes retournant toujours une valeur.

exemple : x=y
x+f(x)

• Une expression terminé par un point virgule (;) ➔ **une instruction**.

• **L'opérateur d'affectation : =**

exemple : x=y; *// x prend la valeur de y*

affectation multiple :
i=j=k=l;
équivalent à : k=l; j=k; i=j;

Langage C : séance 1

Quelques opérateurs de base :

- **Les opérateurs arithmétiques :**
  - unaires : **+ , -**
  - binaires :
    - **+** (addition).
    - **-** (soustraction).
    - **/** (division) si les deux opérandes entières ➔ résultat entier (1/2 ➔ 0) sinon résultat réel (1.0/2 ➔ 0.5)
    - * (multiplication).
    - % (modulo) reste de la division entière (3%2 ➔ 1).
- **Les opérateurs booléens :**
  - **!** Inversion booléenne
  - **&&** ET logique
  - **| |** OU logique
- **Les opérateurs relationnels :**
  - == égalité
  - != différent de
  - < ou <=
  - > ou >=

Langage C : séance 1

L'instruction « si … alors … sinon… »:

Syntaxe :

```
if ( expression_condition  )
   {
         instruction1;
   }
else
   {
         instruction2;
   }
```

Parenthèses obligatoires

Le bloc est facultatif pour une seule instruction,
Mais conseillé pour une meilleure lisibilité !

La clause « else » est facultative