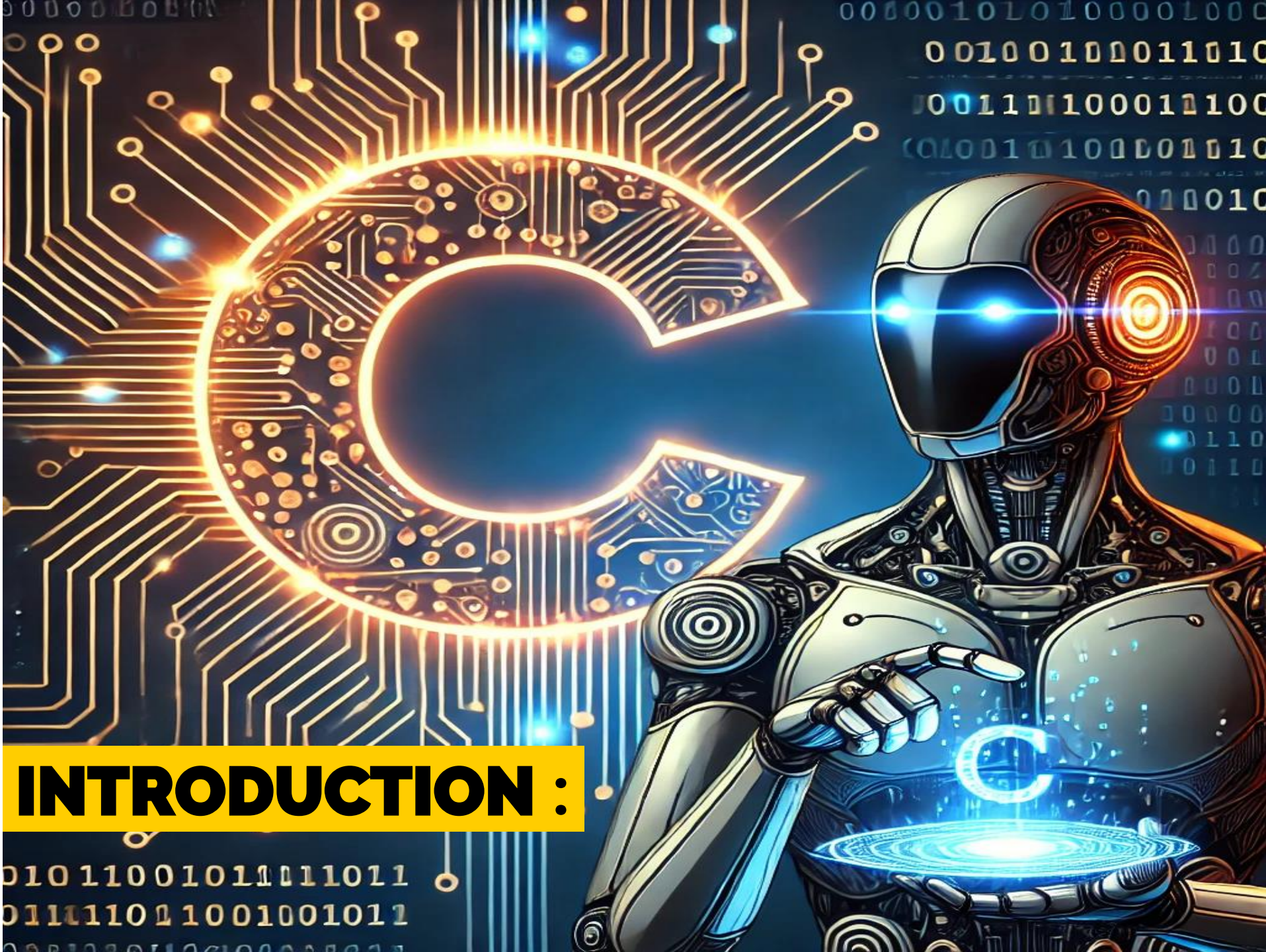
SAGIM

Langage C >> Data Structure

Chapitre 4:
Data Structure en C

LES STRCTURES EN LANGAGE C

Janvier 17, 2025


INTRODUCTION :

les structures

- les structures sont un type de données composé d'un ensemble de variables de DIFFÉRENTS types, regroupées sous un même nom.
- Elles permettent de créer des types de données personnalisés adaptés à des besoins spécifiques.

les structures

- Voici un exemple de définition d'une structure en C:

```
struct Point {
        int x;
        int y;
            };
```

les structures

- Cette structure, nommée Point, contient deux membres entiers : **x** et **y** représentant les coordonnées d'un point dans un plan cartésien.
- Pour utiliser une structure, vous devez déclarer une variable de ce type:

```
struct Point p1;
```

les structures

- Cette déclaration crée une variable nommée p1 de type Point. Vous pouvez ensuite accéder aux membres de la structure à l'aide de l'opérateur :

```
p1.x = 10;
p1.y = 20;
```

Principales utilisations :

- **Représentation de données complexes:**

Les structures permettent de regrouper des données logiquement liées, comme les coordonnées d'un point, les informations d'un employé (nom, prénom, salaire, etc.), les caractéristiques d'un produit, etc.

- **Création de types de données personnalisés:**

En définissant des structures, vous pouvez créer vos propres types de données adaptés à vos besoins spécifiques, ce qui rend votre code plus lisible et plus modulaire.

Principales utilisations :

- **Passage de paramètres à des fonctions:**

Vous pouvez passer des structures entières en argument à des fonctions, ce qui permet de transmettre plusieurs valeurs liées entre elles.

- **Retour de valeurs de fonctions:**

Certaines fonctions peuvent retourner des structures comme valeur de retour.

Exemple d'utilisation :

```
#include <stdio.h>
struct Point {
    int x;
    int y;
};
struct Point addPoints(struct Point p1, struct Point p2) {
    struct Point result;
    result.x = p1.x + p2.x;
    result.y = p1.y + p2.y;
    return result;
}
```

Exemple d'utilisation :

```
int main() {
    struct Point p1 = {10, 20};
    struct Point p2 = {5, 15};
    struct Point p3 = addPoints(p1, p2);
    printf("p3.x = %d, p3.y = %d\n", p3.x, p3.y);
    return 0;
}
```

Exemple d'utilisation :

- La fonction **addPoints** prend deux structures **Point** en entrée et retourne une nouvelle structure **Point** qui représente la somme des coordonnées des deux points d'entrée.
- Les structures sont un concept fondamental en C et sont largement utilisées dans la programmation pour organiser et manipuler des données **complexes**.

SAGIM

Programmation
Procedurale en c

Exemple d’utilization d’une structure

GESTION DES LIVRES AVEC LES STRUCTURES .

Exemple pratique

Gestion Livres (1)

```c
#include <stdio.h>
#include <string.h>
struct Livre {
    char titre[100];
    char auteur[50];
    int anneePublication;
    int nbPages;
};
```

Gestion Livres (2)

```
void afficherLivre(struct Livre livre) {
  printf("Titre : %s\n", livre.titre);
  printf("Auteur : %s\n", livre.auteur);
  printf("Année de publication : %d\n", livre.anneePublication);
  printf("Nombre de pages : %d\n\n", livre.nbPages);
}
```

Gestion Livres (3)

```
int main() {
struct Livre livre1 = { "Le prince", "Michiaveli", 1654, 256};
struct Livre livre2 = { "Le prince charmant",    "Dostoisvki", 1949, 328};
    afficherLivre(livre1);
    afficherLivre(livre2);
    return 0;
}
```

Nouveauté :

- **&s->id :** opérateur qui accède au membre **id** de la structure pointée par **S**, et l'opérateur **&** obtient l'adresse mémoire de ce membre.

  L'opérateur **->** est utilisé pour accéder aux membres d'une structure via un **pointeur**.

- **Le spécificateur "%[^\n]" :** Dans le contexte de la fonction **scanf** est utilisé pour lire une chaîne de caractères jusqu'à ce qu'un caractère de saut de ligne (\n) soit rencontré.