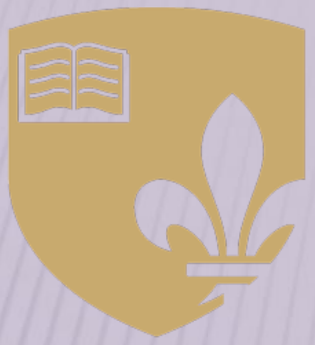

COLLEGE DE PARIS SUPÉRIEUR

DAKARTECH

PROGRAMMATIONS C

Dr Pape Abdoulaye **BARRO**
Enseignant – Chercheur
Spécialiste en Télémétrie & Systèmes Intelligents

POINTEURS ET TABLEAUX

PROGRAMMATIONS C
POINTEURS

Un **pointeur** est une variable spéciale qui peut contenir l'adresse d'une autre variable.

- Si un pointeur P contient l'adresse d'une variable N, on dit que 'P pointe sur N'.
- Les **pointeurs** et les **noms de variables** ont presque le même rôle (à exception prés):
  - Ils donnent accès à un espace mémoire.
  - Un pointeur peut 'pointer' sur différentes adresses tant disque le nom d'une variable reste toujours lié à la même adresse.

PROGRAMMATIONS C

POINTEURS>DÉCLARATION, AFFECTATION, MANIPULATION

- **Déclaration**
  - Syntaxe: <type> *<nom>
  - **Exemple**: int *p;
- **Affectation**
  - Syntaxe: <pointeur> = <&(variable)>
  - **Exemple**:
    - int n=10;
    - int *p = NULL;
    - p=&n;
- **Manipulation**
  - printf("entrer une valeur");
  - scanf("%d", p); // écrire dans la case mémoire pointée par p
  - printf("La valeur est : %d", *p);
  - printf("L'adresse est :%d ", p);

PROGRAMMATIONS C
POINTEURS>ALLOCATION DYNAMIQUE

- Pour demander manuellement une case mémoire, on utilise l'opérateur malloc qui signifie « **M**emory **ALLOC**ation ».
- malloc est une fonction ne retournant aucune valeur (à revoir):
  - **Exemple**:
    - int* p= NULL;
    - p = **malloc**(sizeof(int));
- On peut libérer la ressource après usage via l'opérateur ***free***
- ***Free*** *également est une fonction ne revoyant aucune valeur.*
  - **Exemple**: *free(*p);

PROGRAMMATIONS C
POINTEURS

- **Exemple 1:**

Ecrivez un programme déclarant une variable i de type int et une variable p de type pointeur sur int. Affichez les dix premiers nombres entiers en n'incrémentant que *p et en n'affichant que i

- **Solution:**

```
# include <stdio.h>

int main(){
        // déclaration
        int i = 0;
        int *p;
        p=&i;
        // Affichage
        printf("les dix premiers nombres entiers: \n");
        while(i<10){
                printf("\n%10d", *p);
                i++;
        }
        return 0;
}
```

PROGRAMMATIONS C
POINTEURS

- **Exemple 2:**

Même chose que dans l'exemple 1 mais cette fois ci, en n'incrémentant que *p et en n'affichant que i.

- **Solution:**

```
# include <stdio.h>

int main(){
        // déclaration
        int i;
        int *p;
        p=&i;
        *p = 0;
        // Affichage
        printf("les dix premiers nombres entiers: \n");
        while(*p<10){
                printf("\n%10d", i);
                *p=*p+1;
        }
        return 0;
}
```

PROGRAMMATIONS C

POINTEURS>ALLOCATION DYNAMIQUE D'UN TABLEAU

L'allocation dynamique d'un tableau est un mécanisme très utile. Elle permet de demander à créer un tableau ayant exactement la taille nécessaire (pas plus, ni moins).

- Si on veut créer un tableau de **n** élément de type int (par exemple), on fera appel à malloc.
  - **Exemple**:
    - int *t = NULL;
    - t = (int *)malloc(n*sizeof(int));

Conversion de type

Calcul du nbr d'octets

- Nous avons les équivalences suivantes:
  - ***t = … // 1ere case ou bien t[0]**
  - ***(t+1)=…//2eme case ou bien t[1]**
  - ***(t+2)=…//3eme case ou bien t[2]**
  - **…**
  - ***(t+i)=…//ieme case ou bien t[i]**

PROGRAMMATIONS C

POINTEURS>ALLOCATION DYNAMIQUE D'UN TABLEAU

- **Autres fonctions:**
  - calloc: identique à malloc mais avec initialisation des cases réservées à 0.
  - realloc : permet d'agrandir une zone mémoire déjà réservée
    - **Exemple**:
      - t = (int *) calloc (taille, sizeof(int));
      - taille = taille+10;
      - t =(int *) realloc(t, taille*sizeof(int));

    - **Remarque**: l'utilisation des fonctions telles que **malloc**, **calloc** et **realloc** nécessite l'inclusion d'une bibliothèque nommée **stdlib.h**.

PROGRAMMATIONS C

POINTEURS>ALLOCATION DYNAMIQUE D'UN TABLEAU

- **Exemple3:**

Écrire un programme qui lit un entier n au clavier, alloue un tableau de n entiers initialisés à 0, remplir le tableau par des valeurs saisies au claviers et affiche le tableau.

- **Solution:**

```
#include <stdio.h>
#include <stdlib.h>

int main(){
        // déclaration
        int n, *tab=NULL;
        printf("Veuillez donner la taille du tableau : ");
        scanf("%d", &n);

        while(n<0){
                printf("la taille du tableau doit etre positive: ");
                scanf("%d", &n);
        }

        tab = (int*)calloc(n, sizeof(int));

        // remplissage du tableau
        printf("Veuillez donner les éléments du tableau: ");
        for(int i=0; i<n; i++){
                scanf("%d", &tab[i]);
        }
        // suite …
```

```
        // Affichage du tableau
        for(int i=0; i<n; i++){
                if(i==0){
                    printf("tab=%d", tab[i]);
                }else if(i!=n-1){
                    printf(",%d", tab[i]);
                }else{
                    printf(",%d}", tab[i]);
                }
        }

        return 0;
}
```

PROGRAMMATIONS C
CAS PRATIQUES N°6

- **Application 26:**

Ecrivez un programme déclarant une variable i de type int et une variable p de type pointeur sur int. Affichez les dix premiers nombres entiers en :

  - n'incrémentant que *p
  - n'affichant que i

- **Application 27:**

Écrire un programme qui lit un entier n au clavier, alloue un tableau de n entiers initialisés à 0, remplir le tableau par des valeurs saisies au claviers et affiche le tableau.

- **Application 28:**

Ecrire un programme qui place dans un tableau T les N premiers nombres impairs, puis qui affiche le tableau. Vous accèderez à l'élément d'indice i de t avec l'expression *(t + i).

- **Application 29:**

Ecrivez un programme qui demande à l'utilisateur de saisir un nombre n et qui crée une matrice T de dimensions n*n avec un tableau de n tableaux de chacun n éléments. Nous noterons tij=0 j-ème élément du i-ème tableau. Vous initialiserez T de la sorte : pour tous i, j, tij=1 si i=j (les éléments de la diagonale) et tij=0 si i≠j (les autres éléments). Puis vous afficherez T.

- **Application 30:**

Écrire un programme allouant dynamiquement un emplacement pour un tableau d'entiers, dont la taille est fournie par l'utilisateur. Utiliser ce tableau pour y placer des nombres entiers lus également au clavier. Créer ensuite dynamiquement un nouveau tableau destiné à recevoir les carrés des nombres contenus dans le premier. Supprimer le premier tableau, afficher les valeurs du second et supprimer le tout.

- **Application 31:**

Écrire un programme qui demande à l'utilisateur de lui fournir un nombre entier entre 1 et 7 et qui affiche le nom du jour de la semaine ayant le numéro indiqué (lundi pour 1, mardi pour 2, … dimanche pour 7).

Affaires à suivre