



Pr. Youness KHOUREDIFI, PhD en Informatique
Professeur à la Faculté Polydisciplinaire – Khouribga –
Université Sultan Moulay Slimane – Béni Mellal –
Consultant IT : SQL 2016 Database Administration, Core
Infrastructure 2016, Azure Solutions Architect Expert,
Data Analyst Associate, Ingénieur DevOps.
y.khourdifi@usms.ma

TECHNIQUES DE PROGRAMMATION EN C



Chapitre

5

Les tableaux en langage C

Tableaux :

Activité : Programme des notes

- ☐ Créer un programme en C qui permet de stocker des notes afin de calculer la moyenne, le max et le min.

Tableaux :

Activité : Programme des notes

- ❑ Créer un programme en C qui permet de stocker des notes afin de calculer la moyenne, le max et le min.

```
#include <stdio.h>
int main () {
    float N1, N2, N3, N4;
    printf ( " Donner la note de l'étudiant num 1 : " );
    scanf ( "%f", &N1 );
    printf ( " Donner la note de l'étudiant num 2 : " );
    scanf ( "%f", &N2 );
    printf ( " Donner la note de l'étudiant num 3 : " );
    scanf ( "%f", &N3 );
    printf ( " Donner la note de l'étudiant num 4 : " );
    scanf ( "%f", &N4 );
    return 0;
}
```

Tableaux :

- ❑ Un **tableau** est une variable qui se compose d'un certain nombre de données de même type, rangées en mémoire les unes après les autres.
- ❑ Chaque donnée représente elle-même une variable.
- ❑ Le type d'un tableau peut être n'importe lequel :
 - **Type élémentaires** : char, short, int, long, float, double;
 - **Pointeur**
 - **Structure**
 - ...

Tableaux :

- ❑ Un **tableau** est une variable qui se compose d'un certain nombre de données de même type, rangées en mémoire les unes après les autres.
- ❑ Chaque donnée représente elle-même une variable.
- ❑ Le type d'un tableau peut être n'importe lequel :
 - **Type élémentaires** : char, short, int, long, float, double;
 - **Pointeur**
 - **Structure**
 - ...

Tableaux à une dimension

- ❑ Un tableau unidimensionnel est composé d'éléments qui ne sont pas eux-mêmes des tableaux.
- ❑ On pourrait le considérer comme ayant un nombre fini de colonnes, mais une seule ligne. Par exemple, le tableau suivant est constitué de N éléments de type int :

Tableaux :

- ❑ Un **tableau** est une variable qui se compose d'un certain nombre de données de même type, rangées en mémoire les unes après les autres.
- ❑ Chaque donnée représente elle-même une variable.
- ❑ Le type d'un tableau peut être n'importe lequel :
 - **Type élémentaires** : char, short, int, long, float, double;
 - **Pointeur**
 - **Structure**
 - ...



Tableaux à une dimension

- ❑ Un tableau unidimensionnel est composé d'éléments qui ne sont pas eux-mêmes des tableaux.
- ❑ On pourrait le considérer comme ayant un nombre fini de colonnes, mais une seule ligne. Par exemple, le tableau suivant est constitué de N éléments de type int :

Tableaux à une dimension

- ❑ Dans un tableau il faut préciser le **type** de données leurs **nombre**, ainsi que le **nom** sous lequel le programme pourra y accéder a ces éléments.
- ❑ La définition d'un tableau unidimensionnel admet la syntaxe suivante :

Type **Nom_Du_Tableau**[**Nombre d'éléments**] ;

- ❑ Cette déclaration signifie que le compilateur réserve **Nombre d'éléments** places en mémoire pour ranger les éléments du **Nom_Du_Tableau**.
- ❑ **Type** spécifie le type des éléments du tableau.
- ❑ Le **Nom_Du_Tableau** obéit aux règles régissant les noms de variables.
- ❑ **Nombre d'éléments** est une valeur constante entière, qui détermine le nombre d'éléments du tableau.

Remarque Nous avons ici à faire à une donnée statique, dont la taille n'est pas variable.

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ☐ Un élément du tableau est repéré par son **indice**.
- ☐ En langage C les tableaux commencent à l'indice **0**.
- ☐ L'indice maximum est donc **taille-1**.

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Affectation de la note 14 à l'étudiant numéro 1 :

`N[0] = 14;`

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Affectation de la note 14 à l'étudiant numéro 1 :

`N [0] = 14;`

T	14	17					11
	0	1	2			99

Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

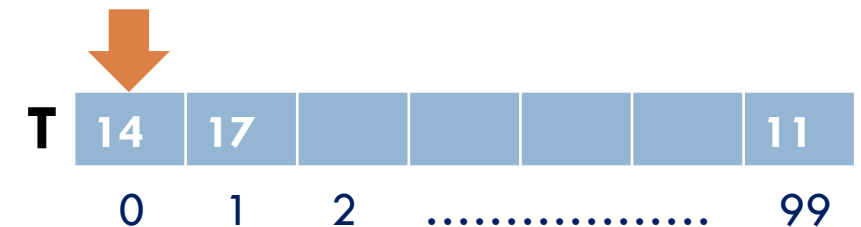
```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Affectation de la note 14 à l'étudiant numéro 1 :

`N [0] = 14;`



Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Exemple :

On déclare un tableau de type entier et de dimension 6.

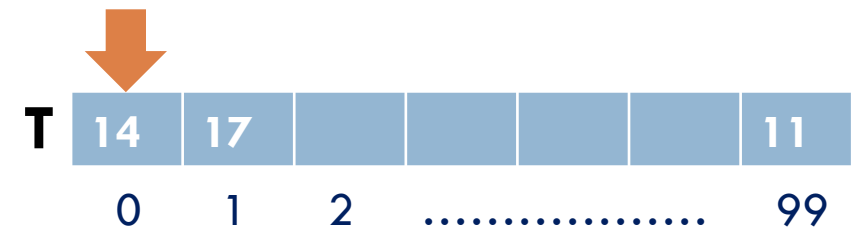
```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Utilisation de la lecture pour saisir la note de l'étudiant numéro 2 :

```
scanf("%f", &N[1]);
```



Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Example :

On déclare un tableau de type entier et de dimension 6.

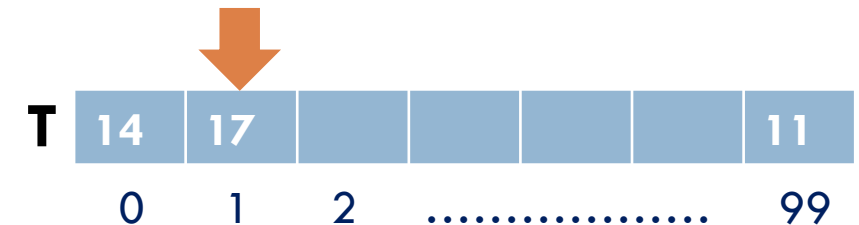
```
int T[6];
```

Pour accéder au éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Utilisation de la lecture pour saisir la note de l'étudiant numéro 2 :

```
scanf("%f", &N[1]);
```



Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Example :

On déclare un tableau de type entier et de dimension 6.

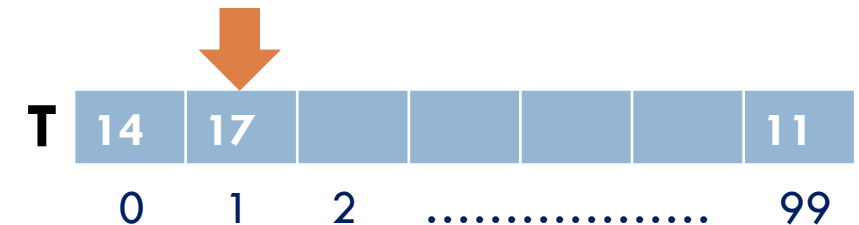
```
int T[6];
```

Pour accéder au éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab [indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Affichage de la note du dernier étudiant de la liste :

```
printf("%f", N[99]);
```



Accès aux éléments d'un tableau

Comment accède-t-on à un élément quelconque d'un tableau ?

- ❑ Un élément du tableau est repéré par son **indice**.
- ❑ En langage C les tableaux commencent à l'indice **0**.
- ❑ L'indice maximum est donc **taille-1**.

Example :

On déclare un tableau de type entier et de dimension 6.

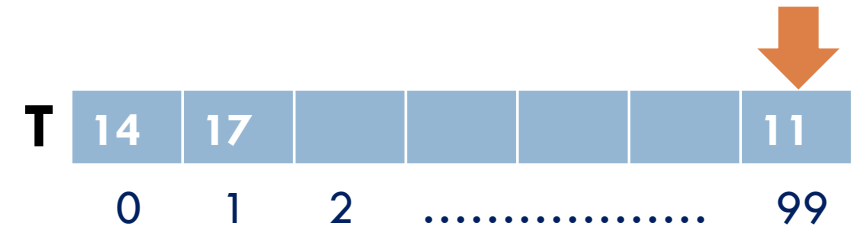
```
int T[6];
```

Pour accéder aux éléments de T on procède comme suit:

- Syntaxe d'affectation : `nom_tab[indice] = valeur;`
- Syntaxe lecture : `scanf("%...", &nom_tab[indice]);`
- Syntaxe écriture : `printf("%...", nom_tab[indice]);`

Affichage de la note du dernier étudiant de la liste :

```
printf("%f", N[99]);
```



T

Exemple 1 : Programme qui lit et qui affiche un tableau

```
1  #include <stdio.h>
2  int main()
3  ▼ {   int T[6], i; //Lecture des éléments du tableau T
4      for (i=0; i<6; i++)
5      ▼ {
6          printf("La saisie de l'element numero %d : ", i);
7          scanf("%d", &T[i]);
8      }
9      printf("Vous avez donner comme tableau \n");
10     for(i=0; i<6; i++)
11     printf("%d  ,\n",T[i]);
12 }
```

```
La saisie de l'element numero 0 : 2
La saisie de l'element numero 1 : 12
La saisie de l'element numero 2 : 22
La saisie de l'element numero 3 : 34
La saisie de l'element numero 4 : 65
La saisie de l'element numero 5 : 7
Vous avez donner comme tableau
2 ,
12 ,
22 ,
34 ,
65 ,
7 ,
Process returned 0 (0x0)   execution time : 14.874 s
Press any key to continue.
```

Exemple 2 : Programme qui lit et qui affiche un tableau

```
1  #include <stdio.h>
2  int main()
3  {   int i, taille;
4  do {
5      printf("Donnez la taille du tableau : ");
6      scanf("%d", &taille);
7  }
8  while (taille <=0);
9  float tab[taille];
10 for (i=0; i<taille; i++)
11 {
12     printf("La saisie de l'element numero %d : ", i);
13     scanf("%f", &tab[i]);
14 }
15 printf("Tab = \n");
16 for(i=0; i<taille ; i++)
17     printf("%f \n",tab[i]);
18 }
```

```
Donnez la taille du tableau : 4
La saisie de l'element numero 0 : 12
La saisie de l'element numero 1 : 32
La saisie de l'element numero 2 : 11
La saisie de l'element numero 3 : 8
Tab =
12.000000
32.000000
11.000000
8.000000
```

```
Process returned 0 (0x0)   execution time : 18.239 s
Press any key to continue.
```

Exercices :

Exercice 1 :

- ☐ Ecrire un programme qui effectue le produit scalaire de deux vecteurs de même taille (3 éléments) représentés par des tableaux à une dimension.

Exercices :

Exercice 1 :

- ❑ Ecrire un programme qui effectue le produit scalaire de deux vecteurs de même taille (3 éléments) représentés par des tableaux à une dimension.

```
1  #include <stdio.h>
2  int main()
3  {
4      float U[3], V[3];
5      int i;
6      float P;
7      printf("Veuillez saisir les valeurs des deux vecteurs : \n");
8      for (i = 0; i < 3; i++)
9      {
10         printf("U[%d] = ", i);
11         scanf("%f", &U[i]);
12         printf("V[%d] = ", i);
13         scanf("%f", &V[i]);
14     }
15     P = 0;
16     for (i = 0; i < 3; i++)
17     {
18         P = P + U[i] * V[i];
19     }
20     printf("Le produit scalaire des deux vecteurs est : %.2f", P );
21 }
```


Exercices :

Exercice 2 :

- ☐ Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers qu'on stocke dans un tableau T.
- ☐ Ensuite, le programme détermine et affiche le minimum des éléments du tableau T.

Exercices :

Exercice 2 :

- ☐ Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers qu'on stocke dans un tableau T.
- ☐ Ensuite, le programme détermine et affiche le minimum des éléments du tableau T.

```
2  int main()
3  {
4      int T[10];
5      int i, min;
6      printf("Veuillez saisir les elements du tableau : \n");
7      for (i = 0; i < 10; i++)
8      {
9          printf("T[%d] = ", i );
10         scanf("%d",&T[i]);
11     }
12     min = T[0];
13     for (i = 0; i < 10; i++)
14     {
15         if (min>T[i])
16         {
17             min = T[i];
18         }
19     }
20     printf("Le minimum des elements du tableau est : %d", min);
21 }
```

Exercices :

Exercice 3 :

- ☐ Ecrire un programme qui demande à l'utilisateur d'entrer des éléments dans un tableau, puis le programme place les éléments pairs et impairs dans deux tableaux séparés.

Exercices :

Exercice 3 :

- ❑ Ecrire un programme qui demande à l'utilisateur d'entrer des éléments dans un tableau, puis le programme place les éléments pairs et impairs dans deux tableaux séparés.

```
1  #include<stdio.h>
2  int main()
3  {
4      int T[100], P[100], I[100];
5      int i, Taille, Pcmp, Icmp;
6
7      printf("Veuillez saisir la taille du tableau : ");
8      scanf("%d", &Taille);
9      printf("Veuillez saisir les elements du tableau : \n");
10     for (i=0; i<Taille;i++) {
11         printf("T[%d] = ", i+1);
12         scanf("%d", &T[i]); }
13     Pcmp=0;
14     Icmp=0;
15     for (i=0; i<Taille;i++) {
16         if (T[i]%2==0) {
17             P[Pcmp]=T[i];
18             Pcmp++; }
19         else {
20             I[Icmp]=T[i];
21             Icmp++; }
22     }
23     printf("\nLes elements pairs du tableau sont ");
24     for(i=0; i<Pcmp;i++)
25         printf("%d ", P[i]);
26
27     printf("\nLes elements impairs du tableau sont ");
28     for(i=0; i<Icmp;i++)
29         printf("%d ", I[i]);
30     return 0;
31 }
```

Les pointeurs en langage C

Introduction :

- ❑ Toutes les variables qu'on utilise dans nos programmes sont stockées quelque part dans la mémoire centrale.
- ❑ La mémoire peut être assimilée à un tableau où chaque case est identifiée par une "adresse".
- ❑ Pour retrouver une variable, il suffit donc, de connaître l'adresse de la case (l'emplacement mémoire où elle est stockée)
- ❑ C'est le compilateur qui fait le lien entre l'identificateur (nom) d'une variable et son adresse dans la mémoire.

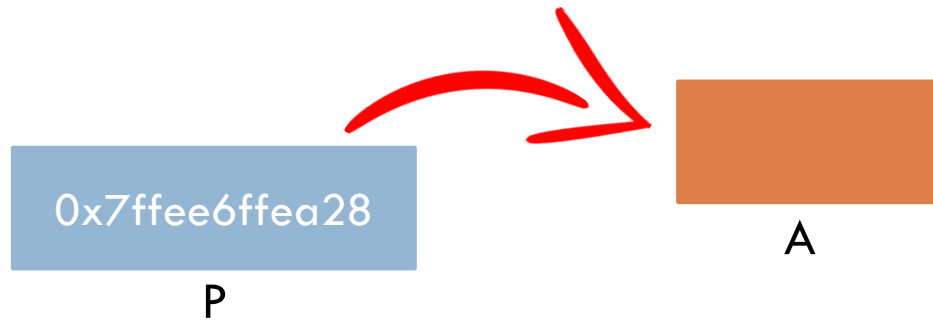


Il peut être plus intéressant d'écrire une variable non plus par son identificateur mais directement par son adresse.

Les pointeurs en langage C

Définition d'un pointeur :

- ❑ Un pointeur est une variable spéciale qui contient l'adresse d'une autre variable.
- ❑ Chaque pointeur est limité à un type de données.
- ❑ Si un pointeur P contient l'adresse d'une variable A, on dit que 'P pointe sur A'.



Les pointeurs en langage C

Déclaration et initialisation d'un pointeur en C

Déclaration

- ❑ Un pointeur est une variable dont la valeur est égale à l'adresse d'une autre variable. En C, on déclare un pointeur par l'instruction :

type *nom_du_pointeur;

Où


- ❑ **type** est le type de la variable pointée,
- ❑ l'identificateur **nom_du_pointeur** est le nom de la variable pointeur
- ❑ ***** est l'opérateur qui indiquera au compilateur que c'est un pointeur.

Exemple : **int *p;** On dira que :

- ❑ **p** est un pointeur sur une variable du type **int**, ou bien p peut contenir l'adresse d'une variable du type int
- ❑ ***p** est de type **int**, c'est l'emplacement mémoire pointé par p.

Les pointeurs en langage C

Remarques :

- ☐ A la déclaration d'un pointeur p, il ne pointe a priori sur aucune variable précise : p est un pointeur non initialisé.
-  Toute utilisation de p devrait être précédée par une initialisation.
- ☐ La valeur d'un pointeur est toujours un entier (codé sur 32bits ou 64bits). Le type d'un pointeur dépend du type de la variable vers laquelle il pointe. Cette distinction est indispensable à l'interprétation de la valeur d'un pointeur. En effet :
 - ☐ Pour un pointeur sur une variable de type char, la valeur donne l'adresse de l'octet où cette variable est stockée.
 - ☐ pour un pointeur sur une variable de type short, la valeur donne l'adresse du premier des 2 octets où la variable est stockée
 - ☐ Pour un pointeur sur une variable de type float, la valeur donne l'adresse du premier des 4 octets où la variable est stockée.

Les pointeurs en langage C

Quelque types de variables en C :

Type	Taille	Signé	Non signé
char	1 octet	-128 à 127	0 à 2^8-1
short int	2 octets	-2^{15} à $2^{15}-1$	0 à $2^{16}-1$
int	4 octets	-2^{31} à $2^{31}-1$	0 à $2^{32}-1$
long int	4/8 octets (processeur 32/64 bits)	-2^{63} à $2^{63}-1$	0 à $2^{64}-1$
int *	4/8 octets (processeur 32/64 bits)	-2^{63} à $2^{63}-1$	0 à $2^{64}-1$
float	4 octets	-2^{31} à $2^{31}-1$	0 à $2^{32}-1$
double	8 octets	-2^{63} à $2^{63}-1$	0 à $2^{64}-1$
long double	12/16 octets (processeur 32/64 bits)	-2^{95} à $2^{95}-1$ -2^{127} à $2^{127}-1$	0 à $2^{96}-1$ 0 à $2^{128}-1$

Les pointeurs en langage C

Initialisation :

- ❑ Pour initialiser un pointeur, le langage C fournit l'opérateur unaire `&`.
- ❑ Ainsi pour récupérer l'adresse d'une variable A et la mettre dans le pointeur P (P pointe vers A) :

`P = &A`

Exemple 1 :

*`int A, B, *P; /* supposons que ces variables occupent la mémoire à partir de l'adresse 01A0 */`*

`A = 10;`

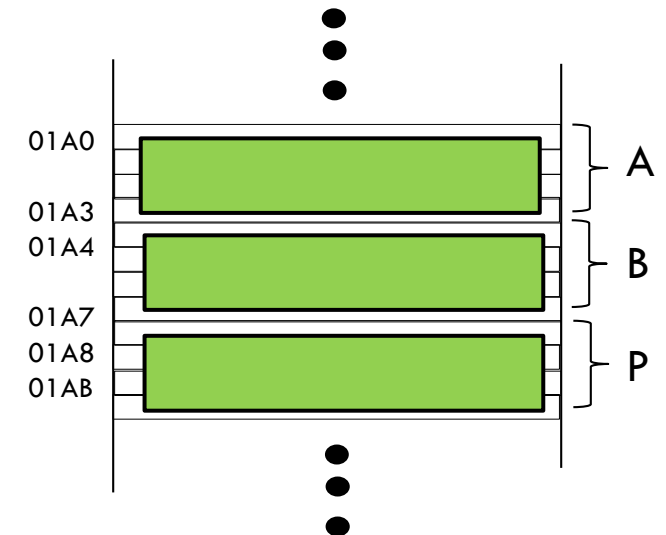
`B = 50;`

`P = &A ; // se lit mettre dans P l'adresse de A`

*`B = *P; /* mettre dans B le contenu de la variable pointé par *P */`*

*`*P = 20; /* mettre la valeur 20 dans la variable pointé par p*/`*

`P = &B; // P pointe sur B`



Les pointeurs en langage C

Exemple 1 : Solution

```
int A, B, *P;
```

```
A = 10;
```

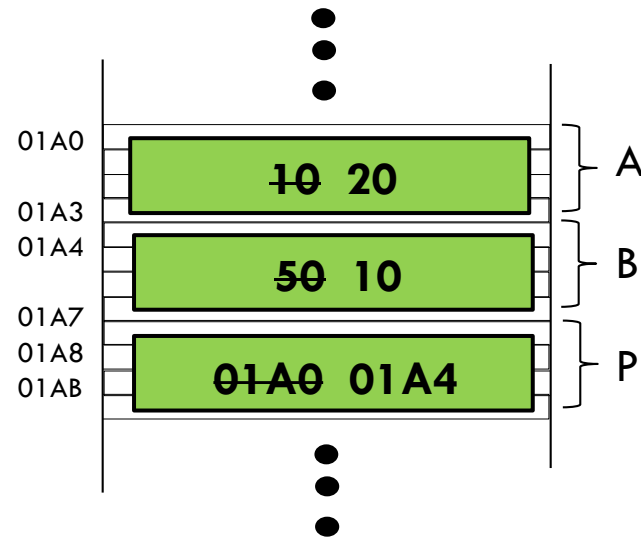
```
B = 50;
```

```
P = &A;
```

```
B = *P;
```

```
*P = 20;
```

```
P = &B;
```



Les pointeurs en langage C

Arithmétique des pointeurs :

On peut appliquer sur un pointeur quelques opérations arithmétiques :

- ☐ Ajouter un entier à un pointeur.
- ☐ Soustraire un entier d'un pointeur.
- ☐ Soustraire un pointeur d'un autre pointeur (de même
- ☐ type).

Les pointeurs en langage C

- ❑ Soit i et j deux entiers et p un pointeur sur un élément de type T ,
- ❑ L'expression $p' = p + i$ ($p' = p - i$) désigne un pointeur p' sur un élément de type T ,
- ❑ La valeur de p' est égale à la valeur de p incrémenté (décrémenté) de : $i * \text{sizeof}(T)$

Exemple:

```
main () {  
    int i = 5;  
    int *p1 = NULL, *p2 = NULL;  
    p1 = &i + 2;  
    p2 = p1 - 2;  
    int j = p1 - p2;  
}
```

Lvalue	Adresse	Valeur
i	6422000	5
p1	6422004	?
p2	6422008	?
j	6422016	?

Avec : $\text{sizeof}(\text{int}) == 4$ Octet

Les pointeurs en langage C

Solution :

```
main () {  
    int i = 5;  
    int *p1 = NULL, *p2 = NULL;  
    p1 = &i + 2;  
    p2 = p1 - 2;  
    int j = p1 - p2;  
}
```

Avec : sizeof (int) == 4 Octet

Lvalue	Adresse	Valeur
i	6422000	5
p1	6422004	6422008
p2	6422008	6422000
j	6422016	2

Les pointeurs en langage C

Opérations élémentaires sur les pointeurs :

- ❑ L'opérateur & : 'adresse de' : permet d'obtenir l'adresse d'une variable.
- ❑ L'opérateur * : 'contenu de' : permet d'accéder au contenu d'une adresse.
- ❑ Si un pointeur P pointe sur une variable X, alors *P peut être utilisé partout où on peut écrire X.

Exemple : `int X=1, Y, *P`

Après l'instruction, `P = &X`; On a :

<code>Y = X + 1</code>	équivalente à	<code>Y = *P + 1</code>
<code>X += 2</code>	équivalente à	<code>*P += 2</code>
<code>++X</code>	équivalente à	<code>++ *P</code>
<code>X++</code>	équivalente à	<code>(*P)++</code>

Les pointeurs en langage C

Opérations élémentaires sur les pointeurs :

- ❑ Le seul entier qui puisse être affecté à un pointeur d'un type quelconque P est la constante entière 0 désignée par le symbole **NULL** défini dans **<stddef.h>**
- ❑ On dit alors que **le pointeur P ne pointe « nulle part »**.

Exemple :

```
#include <stddef.h>
...
int *p, x, *q;
short y = 10, *pt=&y;
p = NULL; // Correct
p = 0; // Correct
x = 0;
p = x // Incorrect ! bien que x vaille 0
q = &x;
p = q; // Corrct : p et q pointe sur des variables de même type
p = pt; // Incorrect : p et pt pointe sur des variable de type différent
```


Les pointeurs en langage C

Les pointeurs et les tableaux :

En C, il existe une relation très étroite entre tableaux et pointeurs. Ainsi, chaque opération avec des indices de tableaux peut aussi être exprimée à l'aide de pointeurs.

Adressage et accès aux composantes d'un tableau à une dimension :

En déclarant un tableau T de type int (int T[N]) et un pointeur P sur des variables entières (int *P),
l'instruction P = T crée une liaison entre le pointeur P et le tableau T en mettant dans P l'adresse du premier élément de T (de même P = &T[0]).

```
int T[n];
```

```
int *p;
```

Créer une liaison entre le tableau T et le pointeur p:

`p = T;`



`p = &T[0];`

Les pointeurs en langage C

A partir du moment où $\mathbf{P} = \mathbf{T}$, la manipulation du tableau \mathbf{T} peut se faire par le biais du pointeur \mathbf{P} .

En effet :

- | | |
|-----------------------|-----------------------------|
| ➤ p pointe sur T[0] | *p désigne T[0] et p[0] |
| ➤ p+1 pointe sur T[1] | *(p+1) désigne T[1] et p[1] |
| ➤ ... | ... |
| ➤ p+i pointe sur T[i] | *(p+i) désigne T[i] et p[i] |

où $i \in [0, N-1]$

Les pointeurs en langage C

Exemple :

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    int T[5] = {1, 2, 3, 4, 5};
    int i, *p;
    p = T;
    for (i=0; i<5; i++){
        printf ("*(p+%d)=%d \t", i, *(p+i));
        printf ("p[%d]= %d \t", i, p[i]);
        printf ("T[%d]=%d \n", i, T[i]);
    }
    exit(0); // ou exit(EXIT_SUCCESS) sortir du prog avec succès
}
```

Donner le résultat d'exécution de ce programme:



La fonction *exit* avec les paramètres suivants: 1 ou *EXIT_FAILURE* pour signaler un problème.

Les pointeurs en langage C

```
*(p+0)=1      p[0]= 1      T[0]=1
*(p+1)=2      p[1]= 2      T[1]=2
*(p+2)=3      p[2]= 3      T[2]=3
*(p+3)=4      p[3]= 4      T[3]=4
*(p+4)=5      p[4]= 5      T[4]=5

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
```

❑ Nous retenons que si p pointe sur T alors:

`*(p+i)` **`= p[i]`** **`= T[i]`** // valeurs

`p+i` **`= &p[i]`** **`= &T[i]`** // adresses

Les pointeurs en langage C

Exercice :

- ❑ Ecrire 2 procédures pour lire et afficher les éléments d'un tableau de réelles à l'aide d'un **pointeur** et de l'indice **i**:

```
void lecture_pti (float t[], int n);  
void affichage_pti (float t[], int n);
```

Les pointeurs en langage C

Lecture et affichage d'un tableau avec un pointeur et l'indice i :

```
void lecture (float t[], int n)
{ int i ;
  for(i=0 ; i<n ; i++)
  {
    printf("donner une valeur");
    scanf("%f", &t[i]) ;
  }
}

void affichage (float t[], int n)
{ int i ;
  for(i=0 ; i<n ; i++)
  printf("%f", t[i]);
}
```



Les pointeurs en langage C

Lecture et affichage d'un tableau avec un pointeur et l'indice i :

```
void lecture (float t[], int n)
{ int i ;
  for(i=0 ; i<n ; i++)
  {
    printf("donner une valeur");
    scanf("%f", &t[i]) ;
  }
}

void affichage (float t[], int n)
{ int i ;
  for(i=0 ; i<n ; i++)
    printf("%f", t[i]);
}
```



```
void lecture_pti ( float t[], int n )
{ int i;
  float *pt ;
  pt = &t[0]; // ou pt=t
  for(i=0 ; i<n ; i++)
  {
    printf("donner une valeur");
    scanf("%f", pt+i) ; // ou &pt[i]
  }
}

void affichage_pti (float t[], int n)
{ int i;
  float *pt ;
  pt = &t[0]; // ou pt=t
  for(i=0 ; i<n ; i++)
    printf("%f", *(pt+i)); // ou pt[i]
}
```

Les pointeurs en langage C

Exercice :

- ❑ Réécrire les 2 procédures précédentes à l'aide d'un **pointeur** et sans utiliser l'indice **i**:

```
void lecture_pt (float t[], int n);  
void affichage_pt (float t[], int n);
```


Les pointeurs en langage C

Lecture et affichage d'un tableau avec un pointeur et sans utiliser i :

```
void lecture_pti ( float t[], int n )
{ int i;
  float *pt ;
  pt = &t[0]; // ou pt=t
  for(i=0 ; i<n ; i++)
  {
    printf("donner une valeur");
    scanf("%f", pt+i) ; // ou &pt[i]
  }
}

void affichage_pti (float t[], int n)
{ int i;
  float *pt ;
  pt = &t[0]; // ou pt=t
  for(i=0 ; i<n ; i++)
    printf("%f", *(pt+i)); // ou pt[i]
}
```



```
void lecture_pt(float t[], int n)
{
  float *pt ;
  for( pt=t; pt<t+n ; pt++)
  {
    printf("donner une valeur");
    scanf("%f", pt) ;
  }
}

void affichage_pt(float t[], int n)
{
  float *pt ;
  for( pt=t; pt<t+n; pt++)
    printf("%f", *pt);
}
```

Les pointeurs en langage C

Sources d'erreurs :

Un grand nombre d'erreurs lors de l'utilisation de C provient de la confusion entre soit contenu et adresse, soit pointeur et variable.

Résumons :

`int A ;` //déclare une variable simple de type `int`

`A` désigne le contenu de `A`

`&A` désigne l'adresse de `A`

`int B[10] ;` déclare un tableau de 10 éléments de type `int`

`B` \Leftrightarrow `&B[0]` désigne l'adresse de la première composante de `B`.

`*(B+i)` \Leftrightarrow `B[i]` désigne le contenu de la composante `i` du tableau

`B+i` \Leftrightarrow `&B[i]` désigne l'adresse de la composante `i` du tableau

Les pointeurs en langage C

Exercice :

Soit P un pointeur qui 'pointe' sur un tableau A : `int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};`

`int *P;`

`P = A;`

Quelles valeurs ou adresses fournissent ces expressions sachant que **`&A[0]=2193480`** :

a) P

b) `*P+2`

c) `*(P+2)`

d) `&A[7]-P`

e) `P+1`

f) `&A[4]-3`

g) `A+3`

h) `*(P+*(P+8)-A[7])`

i) `(*P)++`

j) `*P++`

Les pointeurs en langage C

Solution :

Soit P un pointeur qui 'pointe' sur un tableau A : `int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};`

`int *P;`

`P = A;`

Quelles valeurs ou adresses fournissent ces expressions sachant que **&A[0]=2193480** :

a) `P` = **2193480**

b) `*P+2` = **14**

c) `*(P+2)` = **34**

d) `&A[7]-P` = **L'indice 7**

e) `P+1` = **L'adresse de A[1]**

f) `&A[4]-3` = **L'adresse de A[1]**

g) `A+3` = **L'adresse de A[3]**

h) `*(P+*(P+8)-A[7])` = **23**

i) `(*P)++` = **13**

j) `*P++` = **23**



La soustraction de deux pointeurs qui pointent dans le même tableau est équivalente à la soustraction des indices correspondants.