Résumé sur les pointeurs

<table>
<tr><th></th><th>Algorithme</th><th>Langage C</th></tr>
<tr><td>Déclaration</td><td>nom_du_pointeur: * Type;
(Type est le type de l'objet pointé.)
Exemple :
p1:*caractère; //p1 est un pointeur pointant sur un objet de type caractère.
p2:*entier; //p2 est un pointeur pointant sur un objet de type entier.
p3:*réel; //p3 est un pointeur pointant sur un objet de type réel.</td><td>Type * nom_du_pointeur ;
(Type est le type de l'objet pointé.)
Exemple :
Char * p1; //p1 est un pointeur pointant sur un objet de type caractère.

int * p2 ; //p2 est un pointeur pointant sur un objet de type entier.

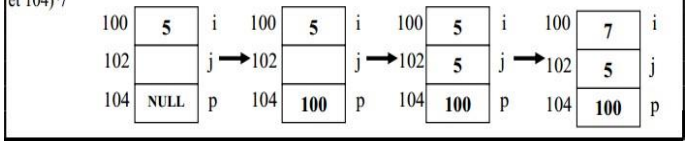
float * p3; // p3 est un pointeur pointant sur un objet de type réel.</td></tr>
<tr><td>Initialisation</td><td colspan="2">Après la déclaration et avant l'utilisation, un pointeur doit être initialisé (sinon, il peut pointer sur n'importe quelle région de la mémoire !):</td></tr>
<tr><td></td><td>1. l'affectation d'une valeur nulle:
p ← NULL ;

2. l'affectation de l'adresse d'une autre variable :
p ← &i ;

Algorithme Exemple0
Var
i : Entier
p : *Entier
Début
i←3
p←&i//p contient adresse de i
Ecrire("Le contenu de la case mémoire pointé par p est :", *p)
p←5 //changement de i à travers p
Ecrire("i =" , i , "*p =", *p)
Fin

3. l'allocation dynamique d'un nouvel espace-mémoire. L'allocation dynamique est l'opération qui consiste à réserver un espace-mémoire d'une taille définie.
allouer(p);

allouer(p): réserve un emplacement mémoire de la taille correspondant au type type, et met dans la variable p l'adresse de la zone mémoire réservée.</td><td>1. l'affectation d'une valeur nulle:
p = NULL;

2. l'affectation de l'adresse d'une autre variable:
p =&i ;
Exemple :
Opérateurs & et *
#include <stdio.h>
main( )
{
int i,j; /*i et j des entiers*/
int *p; /*p pointeur sur un entier*/
i=5; /*i reçoit 5*/
p=&i; /*p reçoit l'adresse de i*/
j=*p; /*j reçoit 5 : contenu de l'adresse p*/
p=j+2; / le contenu de l'adresse p devient 7 donc i aussi devient 7*/
}
/*Représentation mémoire(supposant que i, j et p se trouvent respectivement aux adresses 100, 102 et 104)*/
 

3. L'allocation dynamique en C se fait par l'intermédiaire (entre autre) de la fonction malloc(nombreOctets) de la libraire standard <u>stdlib.h</u> :

type* p = (type*) malloc (sizeof (type));
Exemple</td></tr>
</table>

		Allocation mémoire
		#include <stdio.h>
		#include <malloc.h>
		main( )
		{
		int *q; /*q pointeur sur des entiers*/
		q=(int*) malloc(2*sizeof(int)); /*réservation de 4 octets pour le stockage de deux entiers pointés par q*/
		*q=2; /*q pointe sur l'entier 2*/
		*(q+1)=5; /*On met 5 à l'adresse suivante, q pointe alors sur les entiers 2 et 5*/
		char *p; /*p pointeur sur char*/
		p=(char*) malloc(5*sizeof(char)); /*réservation de 5 octets pour le stockage de la chaîne de caractères pointée par p*/
		p="toto\0"; /*toto est la chaîne pointée par p*/
		}

La Libération mémoire	C'est l'opération qui consiste à **libérer l'espace-mémoire alloué**. **Liberer (p);** **Liberer (p):** libère la place de la zone mémoire dont l'adresse est dans **p** (et la rend disponible pour l'allocation d'autres variables), elle laisse la valeur du pointeur en l'état (n'efface pas l'adresse qui est dans la variable pointeur).	En C, la libération mémoire se fait par l'intermédiaire de la fonction **free** de la libraire standard **stdlib.h** : **void free(p) ;** Tout espace-mémoire alloué dynamiquement via **malloc** (ou équivalent) doit obligatoirement être désalloué par **free**.
Arithmétique des pointeurs	La valeur d'un pointeur étant un entier, on peut lui appliquer un certain nombre d'opérateurs arithmétiques classiques. Les seules opérations arithmétiques valides sur les pointeurs sont : • **l'addition d'un entier à un pointeur**. Le résultat est un pointeur de même type que le pointeur de départ ; • **la soustraction d'un entier à un pointeur**. Le résultat est un pointeur de même type que le pointeur de départ ; • **la différence de deux pointeurs pointant tous deux vers des objets de même type**. Le résultat est un entier.	

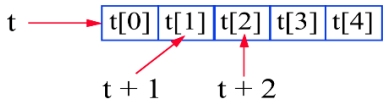
	Exemple:
	
	Si K: **entier**; p:* ***type***;
	L'expression "**p+k**" désigne un pointeur sur un objet de type ***type*** dont la valeur est égale **à la valeur de p incrémentée de k*sizeof(*type*)**.
	// **sizeof(type)** c'est **taille(type)**.
	
	Il en va de même pour la **soustraction** d'un **entier** à un **pointeur** et pour les opérateurs **d'incrémentation** et de **décrémentation** ++ et- -
		Exemple sur l'addition
		incrémentation
		#include <stdio.h>
		main( )
		{
		int i; /*i entier, supposons qu'il se trouve à l'adresse 100*/
		int *p; /*p pointeur sur un entier*/
		p=&i; /*p reçoit 100 : adresse de i*/
		p++; /*p s'incrémente de 2 (devient 102)*/
		char c; /*c char, supposons qu'il se trouve à l'adresse 200*/
		char *q; /*q pointeur sur char*/
		q=&c; /*q reçoit 200 : adresse de c*/
		q++; /*q s'incrémente de 1 (devient 201)*/
		}

Pointeurs et tableaux

Tout tableau unidimensionnel peut être remplacé par un pointeur vers son premier élément.

Équivalences:

t équivalente à t +0 équivalente à &t[0]
t+i équivalente à &t[i]
*t équivalente à t[0]
*(t+i) équivalente à t[i]

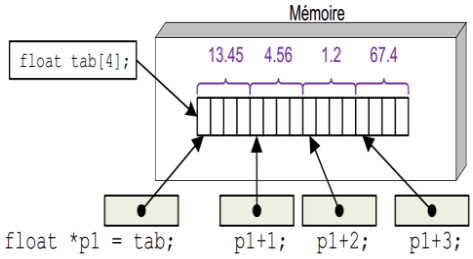


Les boucles suivantes sont équivalentes

```
Pour i ← 1 jusqu'à n faire
      Lire t [i] ;
      i← i +1 ;
   Fin pour ;
```

```
Pour i ← 1 jusqu'à n faire
      Lire *(t +i) ;
      i← i +1 ;
   Fin pour ;
```

```
Pour P1 ← t jusqu'à t+n faire
      Lire *p1 ;
      p1← p1+1 ;
   Fin pour ;
```



Pointeurs et enregistrements	Lorsqu'on utilise un **pointeur sur une structure**. Ses champs seront référencés par le nom du pointeur suivi d'une flèche puis le nom du champ en question.	Accès aux données d'une structure
		`#include <stdio.h>`
		`typedef struct /*définit le type structure date*/`
		`{int jour;`
		`int mois;`
		`int annee;`
		`}date;`
		`main( )`
		`{`
		`date d1;` `/*déclare une variable d1 du type défini date*/`
		`date *d2;` `/*déclare une variable d2 pointeur sur le type défini date*/`
		`/*affecte à d1 la date 01/03/2000*/`
		`d1.jour=1;`
		`d1.mois=3;`
		`d1.annee=2000;`
		`/*affecte à d2 la date 01/03/2000*/`
		`d2->jour=1;`
		`d2->mois=3;`
		`d2->annee=2000;`
		`}`