FICHE RESUME SUR LES POINTEURS EN C++

Donnée	Adresse
i	&i
*p	p

Un pointeur est une variable qui contient l'adresse mémoire d'une autre variable

```
int *p ;
```

- Déclare un pointeur nommé p
- *p est un entier
- p est une variable contenant l'adresse mémoire d'un entier

```
int i=5 ;
int *p=&i ;
```

idem que **int *p ; p=&i ;**

- Déclare un entier i
- Déclare un pointeur nommé p
- *p est un entier
- p est une variable contenant l'adresse mémoire de i

Données	Adresses
i=5=*p	&i=p

```
double tab[5]={2.4,6.1,5.3,9.4} ;
double *p=tab ;
```

idem que **double *p ; p=tab;**

- Le nom d'un tableau est identique à l'adresse de sa 1ère case

Données	Adresses
tab[0]	&tab[0]=tab=p
tab[1]	&tab[1]=p+1
tab[2]	&tab[2]=p+2
tab[3]	&tab[3]=p+3
tab[4]	&tab[4]=p+4

```
void saisie(float *t , int i)
{
  cout << "Saisir tab["<<i<<"] : " ;
  cin >> t[i] ;
}
```

- Quand on veut créer une fonction avec un tableau en paramètre, on passe en fait l'adresse de la première case de ce tableau en paramètre, c'est-à-dire le nom du tableau.

Données	Adresses
t[0]	t=**0x0018FE64**
t[1]	t+1=0x0018FE68

Exemple possible d'adresse. On constate aussi que les adresses se décalent de 4 en 4 car les données sont des float qui occupent 4 octets

```
int main()
{
  float tab[5];
  saisie(tab , 2) ; // saisie de la case n°2
}
```

Données	Adresses
tab[0]	&tab[0]=tab=**0x0018FE64**
tab[1]	&tab[1]=0x0018FE68
tab[2]	&tab[2]= 0x0018FE6C
tab[3]	
tab[4]	

```
short *p ;
float *q ;
int n ;

q=new float ;

cout <<"Saisir la taille du tableau
à créer : " ;
cin >> n ;

p = new short[n] ;
…

delete q ;
delete[] p ;
```

- L'opérateur **new** permet de créer des tableaux dynamiques (dont la taille n'est pas connue avant le lancement du programme).
- L'opérateur **delete** permet de libérer la mémoire allouée par **new**. Pensez à écrire autant de **delete** que de **new**

Données	Adresses
***q**	q

Si on saisit n=3

Données	Adresses
p[0]	p
p[1]	p+1
p[2]	p+2