

Exercices en salle 8 Arithmétique des pointeurs et mémoire dynamique

1 Exercices rapides d'appropriation

- 1. Comment lire un float entré au clavier depuis le console?
- 2. À quoi sert la fonction malloc et comment l'utiliser?
- 3. Qu'est-ce-que le pointeur null?
- 4. Si j'ai un pointeur int *tab, quel est l'équivalent de int x = tab[5]?
- 5. Si j'ai un pointeur int *tab, quel est l'équivalent de tab[5] = 12?
- 6. Quelle est la différence entre *(p++) et (*p)++?
- 7. À quoi servent les fonctions calloc et realloc et comment les utiliser?

2 Exercices sur les pointeurs

1. Pour chacun des ensembles d'instructions suivants, identifiez ce qui se trouve en mémoire.

```
char c = 'A';
                               // un char occupe 1 octet en mémoire
    char *pc = &c;
                               // un char* occupe 8 octets en mémoire
    printf ("%c\n", *pc);
   pc: 1001
    c: 1000
(b)
    float f = 12.5;
                               // un float occupe 4 octets en mémoire
    float *pf = &f;
                               // un float* occupe 8 octets en mémoire
    float \circ = 2 * *pf;
    printf ("%f\n", o);
    o: 1012
   pf: 1004
    f: 1000
```

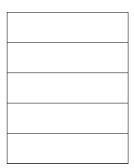


	// un long occupe 8 octets en mémoire // un long* occupe 8 octets en mémoire		
<pre>printf ("%ld\n", *o);</pre>			
o : 1016			
pl : 1008			
l : 1000			
(d)			
int a = 10; double d = 0.75; *(&d) = a * d;	// un int occupe 4 octets en mémoire // un double occupe 8 octets en mémoire		
<pre>printf ("%f\n", d);</pre>			
d : 1004			
a : 1000			

2. Soit le programme suivant. Identifiez ce qui se trouve en mémoire. Quelle information se trouve sur la pile et quelle information se trouve sur le tas.

```
int a = 10;
double d = 0.75;
double *result = (double*) malloc (sizeof (double));

*result = a * d;
printf ("%f\n", *result);
free (result);
```





3. Soit le programme suivant. Identifiez ce qui se trouve en mémoire et trouvez ce qu'il affiche lors de son exécution.

```
int N = 3;
int *tab = (int*) malloc (N * sizeof (double));
int i;
for (i = 0; i < N; i++)
{
    tab[i] = 2 * (i + 1);
}
int *last = tab + (N - 1);
printf ("%d\n", *last);
free (tab);</pre>
```



4. Soit le programme suivant. Identifiez ce qui se trouve en mémoire et trouvez ce qu'il affiche lors de son exécution.

```
int a = 15;
int *pa = &a;
int **ppa = &pa;

*(*ppa) -= 5;

printf ("%d\n", a);
```





5. Soit le programme suivant. Dessinez l'état en mémoire juste avant l'appel de la fonction newTab (ligne 17), à la fin de l'exécution de la fonction newTab (ligne 12) et enfin à la fin de la fonction main (ligne 21).

```
int* newTab (int N)
2
       int *tab = (int*) malloc (N * sizeof (int));
3
       int i;
4
       for (i = 0; i < N; i++)
5
6
7
          tab[i] = 0;
8
9
       return tab;
10
11
   int main()
12
13
14
       int *tab = newTab (4);
       printf ("Le 2ème élément vaut d\n", tab[1]);
15
16
       free (tab);
17
       return 0;
18
19
```

	1			
	I			
	1			
	I			
	I			
	I			
	I			
1	I	1		
	I			
	l .			
1	I	I		
	I			
	I			
	I			
	1		1	
	I			
1	I	I		
1	I	I		
	I			
	I			
1	I	I		
	1		1	
	I			
1	I	I		
	I			
1	I	I		
	I			
	I			
	1		1	
	I			
1	I	I		
	I			
	l .	I	1	



6. Soit la variable int **tab et la situation suivante en mémoire :

tab : 1000	2000
2000	5000
	4000
4000	8
	3
5000	16
	7
	-2

Quelles sont les valeurs des expressions suivantes :

- (a) tab
- (b) &tab
- (c) *tab
- (d) tab[0]
- (e) tab + 2
- (f) *(tab + 2)
- (g) tab[1][2]
- (h) &(tab[1])
- (i) *(tab + 1) + 2
- (j) *(*(tab + 1) + 2)