

LU1IN002 : Éléments de programmation 2

Cours 3

Tableaux et pointeurs

Cours : Jean-Lou Desbarbieux, François Bouchet,
Mathilde Carpentier
et toute l'équipe de l'UE
LU1IN002 Sorbonne Université 2022/2023

Sections : ScFO 21 - CMI Méca - SHI - SPH 2

Structures linéaires

En C : les tableaux

- ▶ suites de valeurs contiguës en mémoire
- ▶ les valeurs sont toutes du même type

Deux types de tableaux en C :

- ▶ les tableaux “statiques”
 - ▶ de taille fixe (exemple `float tab[12];`)
 - ▶ localisés en mémoire dans la pile
- ▶ les tableaux “dynamiques” \Rightarrow au cours suivant
 - ▶ de taille variable, mais la mémoire est gérée par le programmeur avec les fonctions `malloc` et `free` principalement
 - ▶ localisés en mémoire dans le tas

Les tableaux **statiques** en C

```
1  int main() {  
2      int tab[12]; //déclaration  
3      tab[0]=2;    //affectation  
4      tab[5]=4;  
5      tab[9]=tab[5]+tab[0];  
6      tab[5]=tab[5]+1;  
7      printf("tab[5]:%d\n", tab[5]);  
8      return 0;  
9  }
```

⚠ **Attention** : pas d'indice négatif !

Les tableaux **statiques** en C

⚠ **Attention** : pour afficher un tableau entier, il faut afficher chacune des valeurs (avec une boucle)

```
1  int main() {  
2      float tab[5]={1.1,10.9,5.0,1.5,5.6};  
3      //déclaration et initialisation  
4      int i;  
5      for(i=0;i<5;i++){  
6          printf("tab[%d]:%f\n", i, tab[i]);  
7      }  
8      return 0;  
9  }
```

Les tableaux **statiques** en C

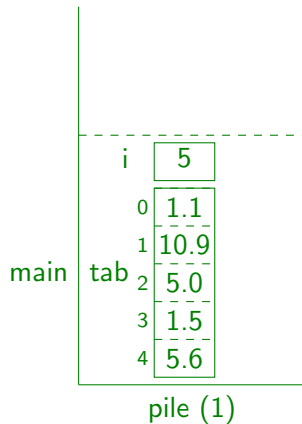
Le nombre de cases du tableau est souvent défini avec une directive **#define**

```
1 #define N 5 //instruction pour le préprocesseur
2 int main() {
3     float tab[N]={1.1,10.9,5.0,1.5,5.6};
4     //déclaration et initialisation
5     int i;
6     for(i=0;i<N;i++){
7         printf("tab[%d]:%f\n", i, tab[i]);
8     }
9     return 0;
10 }
```

⚠ **Attention** : Le **N** défini ici n'est PAS une variable. On ne peut pas modifier sa valeur pendant l'exécution du programme.

Example

```
1 #define N 5
2 int main() {
3     float tab[N]=
4         {1.1,10.9,5.0,1.5,5.6};
5     int i;
6     for(i=0;i<N;i++){
7         printf("tab[%d]:%f\n", i,
8             tab[i]);
9     } // (1)
10     return 0;
11 }
```



Les tableaux **statiques** en C

Déclarations des tableaux :

```
1
2 #define NB 14
3
4 int main() {
5     int tab[NB]; //déclaration
6
7     float t2[4]={1.2, 9, 5.6}; //déclaration et
        affectation
8
9     tab[0]=2; //affectation
10
11     //Attention: tab={1,2,3} interdit en dehors
        des déclarations
12     return 0;
13 }
```

Les tableaux **statiques** en C

Exemple :

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i;  
4      for(i=0; i<5;i++){  
5          tab[i]=i*i;  
6      }  
7      for(i=0; i<5;i++){  
8          printf(" tab[%d]=%d \n", i, tab[i]);  
9      }  
10     return 0;  
11 }
```

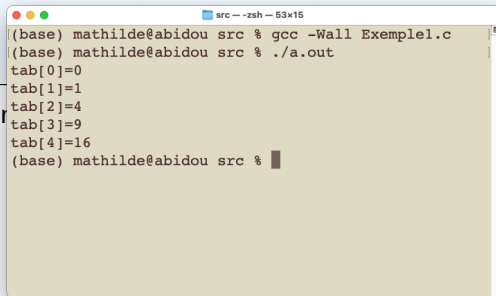
Qu'affiche ce programme ?

Les tableaux **statiques** en C

Exemple :

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i;  
4      for(i=0; i<5;i++){  
5          tab[i]=i*i;  
6      }  
7      for(i=0; i<5;i++){  
8          printf("tab[%d]=%d \n", i, tab[i]);  
9      }  
10     return 0;  
11 }
```

Qu'affiche ce programme



```
(base) mathilde@abidou src % gcc -Wall Exemple1.c  
(base) mathilde@abidou src % ./a.out  
tab[0]=0  
tab[1]=1  
tab[2]=4  
tab[3]=9  
tab[4]=16  
(base) mathilde@abidou src %
```

Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Que se passe t-il avec ce code ?

```
1  int main() {
2      int tab[10]; //déclaration
3      int i;
4      for (i=0; i<5;i++){
5          tab[i]=i*i;
6      }
7      // (...)
8      for (i=0; i<10;i++){
9          printf("tab[%d]=%d \n", i, tab[i]
10             );
11      }
11     return 0;
12 }
```

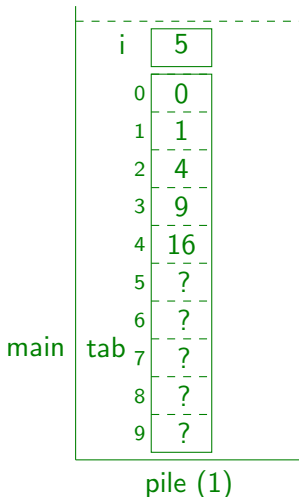
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Que se passe t-il avec ce code ?

```
1  int main() {  
2    int tab[10]; //déclaration  
3    int i;  
4    for (i=0; i<5;i++){  
5        tab[i]=i*i;  
6    }  
7    // (...)  
8    for (i=0; i<10;i++){  
9        printf("tab[%d]=%d \n", i, tab[i]  
10    }  
11    return 0;  
12 }
```



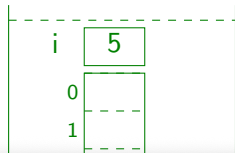
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Que se passe t-il avec ce code ?

```
1  int main() {
2      int tab[10]; //déclaration
3      int i;
4      for (i=0; i<5;i++){
5          tab[i]=i*i;
6      }
7      // (...)
8      for (i=0; i<10;i++){
9          printf("tab[%d]=%d \n", i, tab[i]
10             );
11      }
12      return 0;
13 }
```



```
src -- zsh -- 53x15
(base) mathilde@abidou src % gcc -Wall Ex
(base) mathilde@abidou src % ./a.out
tab[0]=0
tab[1]=1
tab[2]=4
tab[3]=9
tab[4]=16
tab[5]=0
tab[6]=0
tab[7]=0
tab[8]=-357559656
m tab[9]=32766
(base) mathilde@abidou src %
```

pile (1)

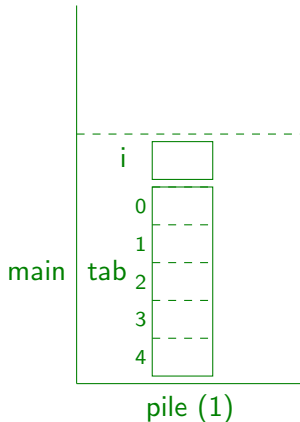
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i; //(1)  
4      for(i=0; i<10;i++){  
5          tab[i]=i*i; //(2)  
6      }  
7      for(i=0; i<10;i++){  
8          printf("tab[%d]=%d \n", i, tab[i  
9          ] );  
10     }  
11     return 0;  
12 }
```



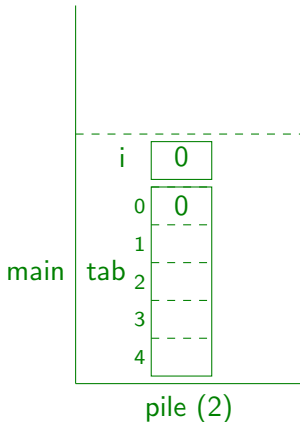
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i; //(1)  
4      for(i=0; i<10;i++){  
5          tab[i]=i*i; //(2)  
6      }  
7      for(i=0; i<10;i++){  
8          printf("tab[%d]=%d \n", i, tab[i]  
9      )};  
10     return 0;  
11 }
```



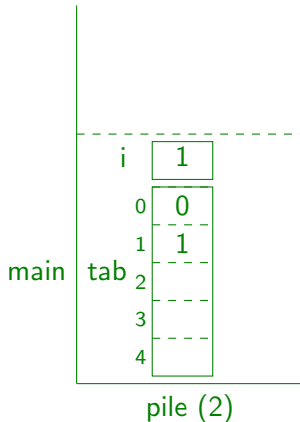
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i; //(1)  
4      for(i=0; i<10;i++){  
5          tab[i]=i*i; //(2)  
6      }  
7      for(i=0; i<10;i++){  
8          printf("tab[%d]=%d \n", i, tab[i]  
9      )};  
10     return 0;  
11 }
```



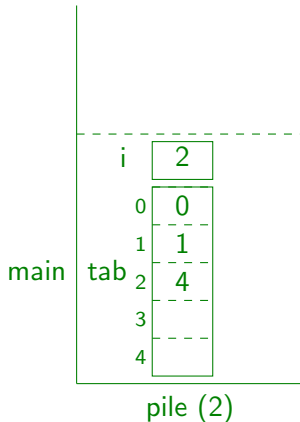
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2    int tab[5]; //déclaration  
3    int i; //(1)  
4    for(i=0; i<10;i++){  
5        tab[i]=i*i; //(2)  
6    }  
7    for(i=0; i<10;i++){  
8        printf("tab[%d]=%d \n", i, tab[i  
9        ] );  
10   }  
11   return 0;  
}
```



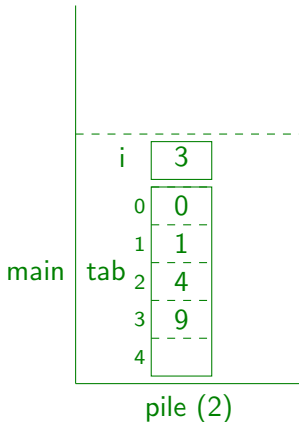
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {
2      int tab[5]; //déclaration
3      int i; //(1)
4      for(i=0; i<10;i++){
5          tab[i]=i*i; //(2)
6      }
7      for(i=0; i<10;i++){
8          printf("tab[%d]=%d \n", i, tab[i]
9      );
10     }
11     return 0;
12 }
```



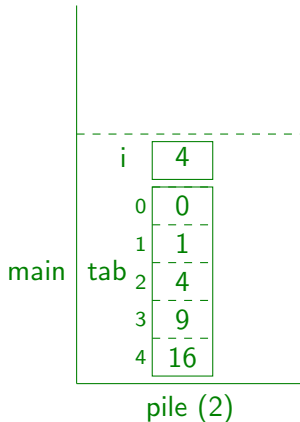
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i; //(1)  
4      for(i=0; i<10;i++){  
5          tab[i]=i*i; //(2)  
6      }  
7      for(i=0; i<10;i++){  
8          printf("tab[%d]=%d \n", i, tab[i  
9      ] );  
10     }  
11     return 0;  
12 }
```



Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {  
2      int tab[5]; //déclaration  
3      int i; //(1)  
4      for(i=0; i<10;i++){  
5          tab[i]=i*i; //(2)  
6      }  
7      for(i=0; i<10;i++){  
8          printf("tab[%d]=%d \n", i, tab[i  
9          ]);  
10     }  
11     return 0;  
12 }
```

		i	5
main	tab	0	0
		1	1
		2	4
		3	9
		4	16
			25

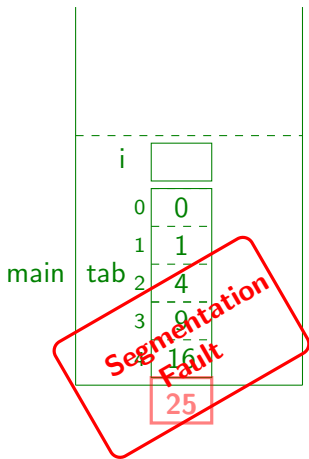
Les tableaux **statiques** en C

<https://app.wooclap.com/HPEDPE>



Et celui-ci ?

```
1  int main() {
2      int tab[5]; //déclaration
3      int i; //(1)
4      for(i=0; i<10;i++){
5          tab[i]=i*i; //(2)
6      }
7      for(i=0; i<10;i++){
8          printf("tab[%d]=%d \n", i, tab[i]
9              );
10     }
11     return 0;
12 }
```



Les tableaux statiques en C

<https://app.wooclap.com/HPEDPE>

Et celui-ci ?

```
1  int main() {
2      int tab[5]; //déclaration
3      int i; //(1)
4      for (i=0; i<10;i++){
5          tab[i]=i*i; //(2)
6      }
7      for (i=0; i<10;i++){
8          printf("tab[%d]=%d \n", i, tab[i]
9              );
10     }
11     return 0;
12 }
```

main

i	
0	0
1	1
2	4
3	9
4	16

⚠ **Attention** : Il n'y a aucun contrôle du compilateur des bornes du tableau : on peut tout à fait essayer d'afficher une case qui n'existe pas, ce qui **peut générer une erreur à l'exécution** (typiquement un *segmentation fault*).



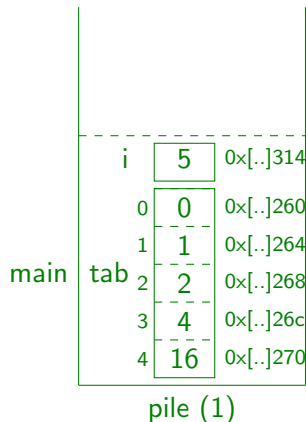
Tableau, mémoire et adresse

Un tableau `tab` de taille n en C est une zone mémoire **dont on a l'adresse du premier octet** dans laquelle on pourra mettre n données ayant toute la même taille. `tab` contient l'adresse du premier octet du tableau.

Équivalence pointeurs et adresses

tab est en fait l'adresse de la première case du tableau.

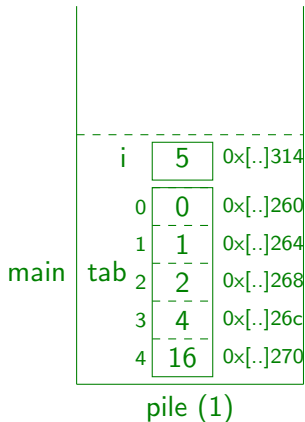
```
4  #define T 5
5  int main() {
6      int tab[T];
7      int i;
8      for(i=0; i<T; i++){
9          tab[i]=i*i;
10     }
11     printf(" tab:%p &tab[0]:%p\n",
12           tab, &tab[0]);
13     printf("  tab[1]:%d &tab[1]:%p\n",
14           tab[1], &tab[1]);
15     printf("  tab[2]:%d &tab[2]:%p\n",
16           tab[2], &tab[2]); //
17                               (1)
18     return 0;
19 }
```



Équivalence pointeurs et adresses

tab est en fait l'adresse de la première case du tableau.

```
4 #define T 5
5 int main() {
6     int tab[T];
7     int i;
8     for (i=0; i<T; i++){
9         tab[i]=i*i;
10    }
11    printf(" tab:%p &tab[0]:%p\n",
12          tab, &tab[0]);
13    printf(" tab[1]:%d &tab[1]:%p\n",
14          tab[1], &tab[1]);
15    printf(" tab[2]:%d &tab[2]:%p\n",
16          tab[2], &tab[2]); //
17    (1)
18    return 0;
19 }
```



```
src -- -zsh -- 63x15
(base) mathilde@abidou src % gcc -Wall prog_tableau_adresse.c
(base) mathilde@abidou src % ./a.out
tab:0x7ffee1864260 &tab[0]:0x7ffee1864260
tab[1]:1 &tab[1]:0x7ffee1864264
tab[2]:4 &tab[2]:0x7ffee1864268
(base) mathilde@abidou src %
```


Tableaux et fonctions

Il est possible de passer un tableau en argument d'une fonction.
Exemple : une fonction retournant la somme des éléments du tableau :

- ▶ **float** somme_tab(**float** tab[], **int** len) ;
- ▶ **float** somme_tab(**float** *tab, **int** len) ;

Ces deux prototypes sont équivalents mais le premier est plus explicite et est donc conseillé quand on manipule des tableaux.

De Python à C : schéma d'accumulation

Avec les listes en Python

En Python :

```
1 def somme_liste(L):  
2     """ list[Number] -> Number  
3     Retourne la somme des éléments de la liste L.  
4     """  
5     # s : Number  
6     s = 0 # la somme cumulée des éléments  
7     # n : Number (élément courant)  
8     for n in L:  
9         s=s+n  
10    return s
```

:- (il n'y a pas de boucle **for** ... **in** en C

De Python à C : schéma d'accumulation

Avec les listes en Python

En Python :

```
1 def somme_liste(L):
2     """ list[Number] -> Number
3         Retourne la somme des éléments de la liste L.
4         """
5     # s : Number
6     s = 0 # la somme cumulée des éléments
7     # n : Number (élément courant)
8     for i in range(len(L)):
9         s=s+L[i]
10    return s
```

:-) Il y a en C une boucle ressemblant au `for i in range(...)` du Python.

Tableau, taille et boucle

En C :

```
1 float somme_tab(float tab[], int len) {  
2     /* donne la somme des éléments du tableau tab  
3     Hypothèse: tab est de longueur len  
4     */  
5     float s = 0;  
6     int i  
7     for (i=0; i < len; i++) {  
8         s = s + tab[i];  
9     }  
10    return s;  
11 }
```

⚠ Attention :

- ▶ déclaration sans indication de taille (**float** tab[])
- ▶ pas de fonction **len**(tab) \Rightarrow le paramètre **len**

Tableau, taille et boucle

En C :

```
1 #define N 4
2 float somme_tab(float tab[], int len) {
3     /* donne la somme des éléments du tableau tab
4        Hypothèse: tab est de longueur len */
5     float s = 0;
6     int i;
7     for (i=0; i < len; i++) {
8         s = s + tab[i];
9     }
10    return s;
11 }
12 int main(){
13     float t[N]={1,4,2,6};
14     printf("somme %f\n", somme_tab(t, N));
15     return 0;
16 }
```

Tableau, taille et boucle

Et avec un while?

En Python :

```
1 def somme_liste(L):
2     """ list[Number] -> Number
3         Retourne la somme des éléments de la liste L.
4         """
5     # s : Number
6     s = 0 # la somme cumulée des éléments
7     # i : Number (élément courant)
8     i=0
9     while(i<len(L)):
10         s=s+L[i]
11         i=i+1
12     return s
```

Tableau, taille et boucle

En C :

```
1  #define N 4
2  float somme_tab(double tab[], int len) {
3      /* donne la somme des éléments du tableau tab
4       Hypothèse: tab est de longueur len*/
5      float s = 0;
6      int i=0;
7      while(i<len){ //for (i=0; i < len; i++) {
8          s = s + tab[i];
9          i=i+1;
10     }
11     return s;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     printf("somme %f\n", somme_tab(t, N));
16     return 0;
17 }
```

Passage par adresse

⚠ **Attention** : Les tableaux sont passés par **adresse** et leur contenu peut être **modifié** par une fonction.

Passage par adresse

⚠ **Attention** : Les tableaux sont passés par **adresse** et leur contenu peut être **modifié** par une fonction.

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf("tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Apres l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf("tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

Passage par adresse

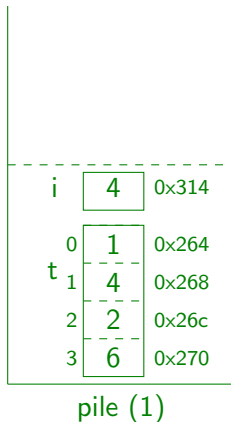
```
src -- -zsh -- 71x11
(base) mathilde@abidou src % gcc -Wall mettre_au_carre.c
(base) mathilde@abidou src % ./a.out
tab[0]:1.000000 tab[1]:4.000000 tab[2]:2.000000 tab[3]:6.000000
Après l'appel
tab[0]:1.000000 tab[1]:16.000000 tab[2]:4.000000 tab[3]:36.000000
(base) mathilde@abidou src %
```

```
4 #def
5 void
6 /
7 int i,
8 for (i=0; i < len; i++) {
9     tab[i] = tab[i] * tab[i]; //(2)
10 }
11 return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf("tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf("tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

Passage par adresse

```
4  #define N 4
5  void mettre_au_carre(float tab[], int len) {
6      /* Met au carré tous les éléments de tab*/
7      int i;
8      for (i=0; i < len; i++) {
9          tab[i] = tab[i] * tab[i]; //(2)
10     }
11     return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf("tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf("tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

main

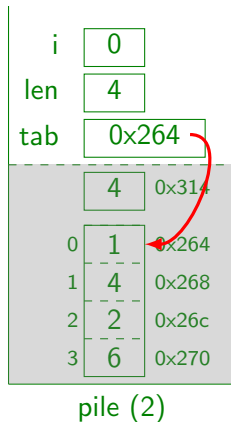


Passage par adresse

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf(" tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf(" tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

mettre_au_carre

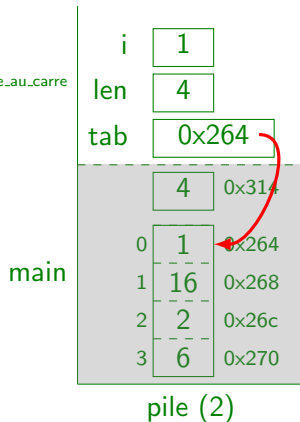
main



Passage par adresse

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf(" tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf(" tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

mettre_au_carre

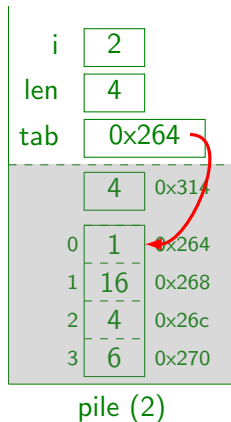


Passage par adresse

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf(" tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf(" tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

mettre_au_carre

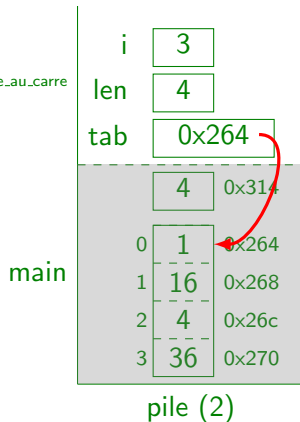
main



Passage par adresse

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf(" tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf(" tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

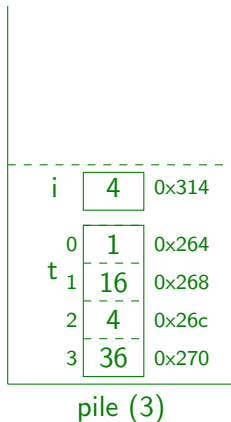
mettre_au_carre



Passage par adresse

```
4 #define N 4
5 void mettre_au_carre(float tab[], int len) {
6     /* Met au carré tous les éléments de tab*/
7     int i;
8     for (i=0; i < len; i++) {
9         tab[i] = tab[i] * tab[i]; //(2)
10    }
11    return;
12 }
13 int main(){
14     float t[N]={1,4,2,6};
15     int i;
16     for (i=0; i < N; i++) {
17         printf(" tab[%d]:%f ", i, t[i]);
18     } //(1)
19     mettre_au_carre(t, N);
20     printf("\n Après l'appel \n ");
21     for (i=0; i < N; i++) {
22         printf(" tab[%d]:%f ", i, t[i]);
23     } //(3)
24     printf("\n");
25     return 0;
26 }
```

main



Comment initialiser un tableau d'entiers avec des valeurs saisies au clavier ?

Il faut utiliser la fonction **scanf** de la bibliothèque `stdio.h` :

```
1  int main() {
2      char c;
3      int i;
4
5      printf(" Saisir un caractère: ");
6      scanf("%c", &c); //Il faut donner l'adresse de c
7      printf(" Caractère lu: %c\n", c);
8
9      printf(" Saisir un entier: ");
10     scanf("%d", &i);
11     printf(" entier lu: %d\n", i);
12
13     return 0;
14 }
```

=> Dessiner la mémoire au tableau

Fonctions et pointeurs : scanf

La fonction scanf :

```
1 void init_tab(int tab[], int taille)
2 {
3     int i = 0;
4     for(i=0;i<taille;i++) {
5         printf("tab[%d] = ? : ", i);
6         scanf("%d", &tab[i]); //adresse de la ieme case, s'écrit
                                aussi tab+i
7     }
8     printf("\n");
9 }
10
11 int main() {
12     int tab[5];
13     init_tab(tab, 5);
14     return 0;
15 }
```

=> Dessiner la mémoire au tableau

Calcul d'adresse

Dans un tableau, les indices indiquent un décalage par rapport à l'adresse de base du tableau.

En C : on peut calculer un décalage d'adresse.

Arithmétique des pointeurs : addition *pointeur + entier*

Soit les variables suivantes :

```
1  int ns[5];  
2  int *a=ns;
```

Adresses :

&ns[0]	≡	a+0	≡	a
&ns[1]	≡	a+1		
...				
&ns[41]	≡	a+41		

Valeurs contenues :

ns[0]	≡	*(a+0)	≡	*a
ns[1]	≡	*(a+1)		
...				
ns[41]	≡	*(a+41)		

Fonctions et pointeurs : scanf

La fonction scanf :

```
1 void init_tab(int tab[], int taille)
2 {
3     int i = 0;
4     for(i=0;i<taille;i++) {
5         printf("tab[%d] = ? : ", i);
6         scanf("%d", tab+i); //adresse de la ieme case
7     }
8     printf("\n");
9 }
10
11 int main() {
12     int tab[5];
13     init_tab(tab, 5);
14     return 0;
15 }
```

Fonctions et pointeurs

<https://app.wooclap.com/HPEDPE>



Qu'affiche ce programme ?

```
1 void AfficheTab(int t[], int n){
2     int i;
3     printf("t=%p t:%p\n", t, t+1);
4     for(i=0;i<n;i++){
5         printf("t[%d]:%d\n", i, t[i]);
6     }
7 }
8
9 int main() {
10     int t1[4]={1,2,3,4};
11     printf("t1=%p t1+1:%p\n", t1, t1+1);
12     AfficheTab(t1,4);
13     AfficheTab(t1+1,3);
14     return 0;
15 }
```

C'est tout pour aujourd'hui !