

TD3: Mémoire statique

Module ArcSys

Objectifs pédagogiques :

- Comprendre la durée de vie des variables (E1, E2, E6);
- Comprendre la visibilité des variables et fonctions (E1);
- Comprendre la différence entre un tableau et un pointeur (E3);
- Manipulation de pointeurs (E2, E3, E4, E5, E7).

★ Exercice 1: Visibilité et durée de vie.

Supposons les deux fichiers suivants :

```

1 #include <stdio.h>
2
3 void func3(void);
4 int pi = 3;
5 static char c = 'c';
6
7 static void func1(int a) {
8     printf("Life be%came interesting in %d\n", c, a);
9 }
10 void func2(void) {
11     static int year = 1972;
12     int g = 0;
13
14     func1(year - g);
15     func3();
16 }
17 int main(void) {
18     func2();
19     return 0;
20 }
    
```

fichier1.c

```

1 #include <stdio.h>
2
3 extern int pi;
4 void func3(void) {
5     int i;
6     for (i = 0; i < pi; i++) {
7         printf("%d", pi);
8     }
9     printf("\n");
10 }
    
```

fichier2.c

Définissez chaque fonction et variable en terme de visibilité (globale, fichier, limité à une fonction ou à un bloc) et durée de vie dans le tableau suivant :

	Visibilité	Durée de vie
printf		
main		
func1		
func2		
func3		
pi		
c		
year		
a		
g		
i		

★ Exercice 2: Erreurs classiques.

- ▷ Question 1: Quel est le problème 1 ci-dessous ? Comment le corriger, sans changer la ligne 8 ?
- ▷ Question 2: Quel est le problème avec le programme 2 ci-dessus ? Comment le corriger ?
- ▷ Question 3: Quel est le problème avec le programme 3A à gauche ? Pourquoi cela fonctionne avec 3B ?
- ▷ Question 4: Expliquez le problème du programme 4 (demandez-vous où est stockée la chaîne "Good").

```

1  Programme 1
2  #include <stdio.h>
3  static int* func(int n) {
4      n = n + 10;
5      return &n;
6  }
7  int main(void) {
8      int* a;
9
10     a = func(19);
11     printf("%d\n", *a);
12     return 0;
13 }

```

```

1  Programme 2
2  #include <stdio.h>
3  int main(void) {
4      char* nom;
5
6      printf("Quel est votre nom ?");
7      scanf("%s", nom);
8
9      return 0;
10 }

```

```

1  Programme 3A
2  #include <stdio.h>
3  int* func(int n) {
4      int array[2] = { 14, 15 };
5      if (n == 0) {
6          return array;
7      } else {
8          return NULL;
9      }
10 }
11 int main(int argc, char* argv[]) {
12     int *t = func(0);
13     printf("t[0]=%d; t[1]=%d\n", t[0], t[1]);
14 }

```

```

1  Programme 3B
2  #include <stdio.h>
3  int* func(int n) {
4      static int array[2] = { 14, 15 };
5      if (n == 0) {
6          return array;
7      } else {
8          return NULL;
9      }
10 }
11 int main(int argc, char* argv[]) {
12     int *t = func(0);
13     printf("t[0]=%d; t[1]=%d\n", t[0], t[1]);
14 }

```

```

1  Programme 4
2  void edit(char* s) {
3      s[0] = 'B';
4      s[1] = 'a';
5      s[2] = 'd';
6      s[3] = '!';
7  }
8  int main(void) {
9      edit("Good");
10
11     return 0;

```

★ Exercice 3: Tableau vs. pointeur

Supposons le programme suivant :

```

1  #include <stdio.h>
2  #include <string.h>
3
4  int main(void) {
5      char hello[] = "hello world!";
6      char* s1 = "hello world!";
7
8      printf("a = %zd\n", sizeof(hello));
9      printf("b = %zd\n", sizeof(s1));
10     printf("c = %zd\n", strlen(hello));
11     printf("d = %zd\n", strlen(s1));
12
13     return 0;
14 }

```

Affichage, si l'on utilise le modèle mémoire 32 bits :

```

a = 13
b = 4
c = 12
d = 12

```

- ▷ Question 1: Qu'est-ce que hello ?
- ▷ Question 2: Qu'est-ce que s1 ?
- ▷ Question 3: Expliquez le résultat de l'affichage.
- ▷ Question 4: Où sont stockées les 2 chaînes de caractères ? Quelle est leur durée de vie ?
- ▷ Question 5: Dessinez le schéma mémoire de ce programme.

★ **Exercice 4: Écrire du code avec des pointeurs.**

Implémentez la fonction `replace` (qui remplace toute la chaîne après le caractère `n` par la chaîne passée en argument) et `swap` (qui échange deux pointeurs entre eux) pour compléter programme suivant :

```

1 int main(void) {
2     char hello[] = "hello world!";
3     char a[] = "abc";
4     char* s1 = hello;
5     char* s2 = a;
6
7     replace(s1, 6, "dennis");
8
9     printf("%s\n", s1);
10    printf("%p %p\n", s1, s2);
11
12    swap(...);
13
14    printf("%s\n", s1);
15    printf("%p %p\n", s1, s2);
16
17    return 0;
18 }
```

Affichage attendu :

```

hello dennis
0x7ffc0f9e2230 0x7ffc0f9e2220
abc
0x7ffc0f9e2220 0x7ffc0f9e2230
```

★ **Exercice 5: Déchiffrer des expressions compliquées et des programmes avec des pointeurs.**

▷ **Question 1:** Complétez le tableau suivant (les expressions s'exécutent les unes après les autres).

	a	b	c
	2	4	6
a++;			
b = a++;			
b = ++a;			
a += 5 + a;			
c = a = b;			
c = (a == b);			
c = (++a == --b);			
a *= b++ - c;			
b == b ? ++ a : ++ b;			

(ce sont les valeurs initiales)

▷ **Question 2:** Décomposez chaque opération pour déterminer l'affichage si on exécute le programme suivant :

```

1 #include <stdio.h>
2
3 static void abcd(char *b, char *a) {
4     int g = 0;
5     int *r = &g;
6     char **v = &a;
7
8     *(v[g]) = b[0];
9     g = 2;
10    (*r)++;
11    v = &b;
12    *(v[0] + 1) = a[g];
13
14    printf("%s %s\n", a, b);
15 }
16
17 int main(void) {
18     char dennis[] = "dennis";
19     char brian[] = "brian";
20     abcd(dennis, brian);
21     return 0;
22 }
```

▷ Question 3: (optionnelle)

Dessinez un tableau où une ligne représente une opération et une colonne représente les variables. Dans chaque case décrivez la modification effectuée par l'opération.

```
1 int a[] = {10, 20, 30, 40, 50};
2 int main(void) {
3     int i, *pi, *pk, *b[2], **p1;
4     pi = &a[0];
5     pk = &a[1];
6     p1 = &pk;
7     (*p1)--;
8     **p1 = 0;
9     b[0] = &a[4];
10    b[1] = b[0];
11    b[0]--;
12    b[0]--;
13    *(b[0]) = 3;
14    pi++;
15    *pi = 4;
16    a[a[2]] = 1;
17    for (i = 0; i < 5; i++)
18        printf(" a[%d] = %d ", i, a[i]);
19    printf("\n");
20
21    return 0;
22 }
```

▷ Question 4: Regarde l'océan (optionnelle)

Même chose avec le programme suivant :

```
1 #include <stdio.h>
2
3 int main() {
4     char mot[] = "VACANCE";
5     char* ptr, ** ptr2;
6
7     mot[1] = '0';
8     ptr = mot + 2;
9     *ptr = mot[0] + 3;
10    ptr++;
11    ptr2 = &ptr;
12    **ptr2 = *(mot + 3);
13    *(++ptr2) = 'G';
14    *(ptr + 1) =>(*ptr2 + 2);
15    *(ptr + 2) = 'S';
16    printf( "Nouveau mot %s \n", mot);
17    *(*ptr2++) = mot[7];
18    printf( "Nouveau mot %s \n", mot);
19 }
```