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 :

fichier1.c

```c
#include <stdio.h>

void func3(void);
int pi = 3;
static char c = 'c';

static void func1(int a) {
    printf("Life be%came interesting in %d\n", c, a);
}
void func2(void) {
    static int year = 1972;
    int g = 0;

    func1(year - g);
    func3();
}
int main(void) {
    func2();
    return 0;
}
```

fichier2.c

```c
#include <stdio.h>

extern int pi;
void func3(void) {
    int i;
    for (i = 0; i < pi; i++) {
        printf("%d", pi);
    }
    printf("\n");
}
```

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"`).

Programme 1

```
1  #include <stdio.h>
2
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 }
```

Programme 2

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

Programme 3A

```
1  #include <stdio.h>
2
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 }
```

Programme 3B

```
1  #include <stdio.h>
2
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 }
```

Programme 4

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

★ 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 $n$ième caractère 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.

```
int a[] = {10, 20, 30, 40, 50};
int main(void) {
    int i;
    int* pi;
    int* pk;
    int* b[2];
    int** pl;

    pi = &a[0];
    pk = &a[1];
    pl = &pk;
    (*pl)--;
    **pl = 0;
    b[0] = &a[4];
    b[1] = b[0];
    b[0]--;
    b[0]--;
    *(b[0]) = 3;
    pi++;
    *pi = 4;
    a[a[2]] = 1;
    for (i = 0; i < 5; i++)
        printf(" a[%d] = %d ", i, a[i]);
    printf("\n");

    return 0;
}
```

▷ **Question 4: Regarde l'océan (optionnelle)**

Même chose avec le programme suivant :

```
#include <stdio.h>

int main() {
    char mot[] = "VACANCE";
    char* ptr;
    char** ptr2;

    mot[1] = 'O';
    ptr = mot + 2;
    *ptr = mot[0] + 3;
    ptr++;
    ptr2 = &ptr;
    **ptr2 = *(mot + 3);
    *(++*ptr2) = 'G';
    *(ptr + 1) = *(*ptr2 + 2);
    *(ptr + 2) = 'S';
    printf( "Nouveau mot %s \n", mot);
    *(*ptr2++) = mot[7];
    printf( "Nouveau mot %s \n", mot);
}
```