Travaux Dirigés n°4

Visibilité des variables et appels à fonctions

Exercice 1

On considère le programe suivant :

```
#include <stdio.h>
int chose(int a);
int machin();

int chose(int a){
  return a+17+machin();
}
int machin(){
  int a=-1;
  return a;
}
int main(){
  int a=1;
  a=chose(a);
  printf("%d\n", a);
}
```

Que se passse-t-il si l'on enlève les déclarations des fonctions `chose` et `machin` ? Qu'affiche le programme ?

Exercice 2

On considère le programe suivant :

```
#include <stdio.h>
int a = 27;

int chose(int a);
int machin();
int chose(int a){
  return a+17+machin();
}
int machin(){
  return a;
}
int main(){
  int a=1;
  a=chose(a);
```

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

Qu'affiche le programme ?

Exercice 3

Définissez l'affichage produit par l'exécution du programme. Expliquez.

```
#include <stdio.h>

int nb = 3;
//============================================================

void mal_ecrit_1 (int nb)
{
  printf ("nb mal_ecrit_1 = %d\n",nb++);
  {
    int nb = 14;
    printf ("nb mal_ecrit_1 = %d\n",nb);
  }
  printf ("nb mal_ecrit_1 = %d\n",nb);
}
//============================================================

int mal_ecrit_2 (int x, int y)
{
  printf ("nb mal_ecrit_2 = %d\n",x + nb);
  printf ("nb mal_ecrit_2 = %d\n",y + nb);
  return (nb *= 0);
}
//============================================================

int main(void)
{
    int x = 3;
    int y = 6;
    printf ("nb main = %d\n",nb);
    mal_ecrit_1(nb);
    printf ("nb main = %d\n",nb);
    printf("resultat de mal_ecrit_2 = %d\n",mal_ecrit_2(y,x));
    printf ("nb = %d\n",nb);

    return 0;
}
```

Les Pointeurs

Exercice 4

Analyser le programme suivant et donner la suite des affichages produits (on choisira arbitrairement les adresses avec des entiers supérieurs à 1000) :

```
#include <stdio.h>
```

```
int main (void) {
  int a, b;
  int *p, *q;
  a = 3; b = 4;
  printf("a = %d, adresse de a = %p,", a, &a);
  printf("b = %d; adresse de b = %p\n",b, &b);
  p = &a; q = &b;
  printf("p = %p, valeur pointee par p = %d,", p, *p);
  printf("q = %p, valeur pointee par q = %d\n",q, *q);
  *p += 1; *q += 1;
  printf("a = %d, b = %d\n", a, b);
  p = q;
  printf("p = %p, valeur pointee par p = %d,", p, *p);
  printf("q = %p, valeur pointee par q = %d\n",q, *q);
  *p += 10;
  printf("a = %d, b = %d\n", a, b);
  return 0;
}
```

Exercice 5

Dans cet exercice, on se propose d'écrire pas-à-pas un programme. Tout doit être fait dans la fonction principale.

- Déclarer un entier **i** et un pointeur sur un entier **p**.
- Affecter la valeur 5 à **i**.
- Afficher la valeur de **i**, et celle de l'adresse de **i**.
- Faire pointer **p** à l'adresse de **i**.
- Que contient alors l'expression ***p** ?
- Déclarer un nouveau pointeur sur entier **q** et le faire pointer à la même adresse que **p**.
- Afficher les expressions du programme contenant la valeur 5, ainsi que leur adresse respective.
- Augmenter de 2 façons différentes la valeur de **i** de 2 sans passer par l'intermédiaire de **i**.

Exercice 6

Si i et j sont des entiers et p et q, des pointeurs sur des entiers, donner le résultat des instructions suivantes :

```
1) p = &i ;  2) p = &*&i ;  3) i = *&*&i ;
4) *p = &j;  5) i = (int)p ;  6) q = &p ;
```

Exercice 7

Trouver l'erreur contenue dans les extraits de programmes suivants :

```
1) int *p ;
   *p = 5 ;

2) int x = 5 ;
   int *p = &x ;
   p = 9 ;
```

Exercice 8

On rappelle que si on définit un tableau `t` de 5 entiers `t` désigne l'adresse du premier élément du tableau et `t+1` est l'adresse du deuxième élément du tableau, etc.

Soit l'extrait code suivant :

```
int t[]={4, 3, 2, 1};
int *p1, **p2, *pt[2];
p1 = &t[2];
p2 = &p1;
pt[0] = &t[3];
(*p2)++;
**p2 = 5;
pt[1] = t + 1;
(*p1)++;
t[t[t[2]]]++;
*pt[1]=*(pt[0]-1)+1;
```

Que vaut le tableau t à la fin de ces instructions ?

Exercice 9

On considère le programme `C` suivant :

```
int main ( )
{
int a = 1, b = 2, c = 3;
int *p1, *p2;
p1 = &a;
p2 = &c;
*p1 = ( *p2 )++;
p1 = p2;
p2 = &b;
*p1 -= *p2;
++( *p2 );
*p1 *= *p2;
a = ( ++( *p2 ) ) * *p1;
p1 = &a;
*p2 = *p1 /= *p2 ;

return (0);
}
```

Remplissez le tableau suivant :

	a	b	c	p1	p2
Init					
`p1 = &a;`					
`p2 = &c;`					
`*p1 = ( *p2 )++;`					
`p1 = p2;`					
`p2 = &b;`					
`*p1 -= *p2;`					
`++( *p2 );`					
`*p1 *= *p2;`					
`a = ( ++( *p2 ) ) * *p1;`					
`p1 = &a;`					
`*p2 = *p1 /= *p2;`					