Licence Informatique | **UE INF101**
Licence Mathématiques | **Initiation à la programmation**
Université Bordeaux 1 | **Travaux Dirigés**
2006-2007 | **feuille 7**

1 Travaux Dirigés

PORTÉE ET MASQUAGE

Exercice 1. La compilation du programme `portee1.c`

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

int
main(int argc, char * argv[])
{
  {
    int a = 3;
    printf("%d\n",a);
  }
  int b = a;
  printf("%d\n",b);
  return EXIT_SUCCESS;
}
```

donne le résultat suivant :

```
gcc -std=c99 -Wall -Werror  portee1.c -oportee1
portee1.c: In function 'main':
portee1.c:11: erreur: 'a' undeclared (first use in this function)
portee1.c:11: erreur: (Chaque identificateur non declaré est rapporté une seule fois
portee1.c:11: erreur: pour chaque fonction dans laquelle il apparaît.)

Compilation exited abnormally with code 1 at Tue Feb 21 18:36:52
```

Expliquer et corriger l'erreur.

Exercice 2. L'exécution du programme `masquage1.c`

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

int main(int argc, char * argv[])
{
  int x = 3;
  {
    int x = 17;
    printf("%d\n",x);
  }
  printf("%d\n",x);
  return EXIT_SUCCESS;
}
```

donne le résultat ci-dessous. Expliquer.

```
xxxx@lucifer:~/InitProg/src/td07$ ./masquage1
17
3
xxxx@lucifer:~/InitProg/src/td07$
```

Exercice 3. Qu'affiche le programme `masquage2.c` suivant ? Expliquer.

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

int a = 1, b = 2, d = 3;

void f(void) {
  int a = 5, c = 6;
  a++;
  printf("f():␣\ta␣=␣%d␣\tb␣=␣%d␣\tc␣=␣%d␣\td␣=␣%d\n", a, b, c, d);
  {
    int a = 7;
    c++;
    printf("1er␣bloc␣f():␣\ta␣=␣%d␣\tb␣=␣%d␣\tc␣=␣%d␣\td␣=␣%d\n", a, b, c, d);
    {
      int a = 8;
      c++;
      printf("2eme␣bloc␣f():␣\ta␣=␣%d␣\tb␣=␣%d␣\tc␣=␣%d␣\td␣=␣%d\n", a, b, c, d);
    }
    printf("1er␣bloc␣f():␣\ta␣=␣%d␣\tb␣=␣%d␣\tc␣=␣%d␣\td␣=␣%d\n", a, b, c, d);
  }
}

void g (int d) {
  int a = d, c = 9;
  printf("g():␣\ta␣=␣%d␣\tb␣=␣%d␣\tc␣=␣%d␣\td␣=␣%d\n", a, b, c, d);
}

int main(int argc, char *argv[]) {
  int d = 4;
  printf("main():␣\ta␣=␣%d␣\tb␣=␣%d␣\td␣=␣%d\n", a, b, d);
  f();
  printf("main():␣\ta␣=␣%d␣\tb␣=␣%d␣\td␣=␣%d\n", a, b, d);
  g(d);
  printf("main():␣\ta␣=␣%d␣\tb␣=␣%d␣\td␣=␣%d\n", a, b, d);
  return EXIT_SUCCESS;
}
```

VARIABLE STATIQUE

Exercice 4. Qu'affiche le programme `static1.c` suivant ? Expliquer.

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

int suivant (void) {
  static int s = 0;
  return s++;
}

int main(int argc, char *argv[]){
  for(int i=-5; i < 0; i++)
    printf("l'appel␣suivant()␣retourne:\t%d\n", suivant());
}
```

Exercice 5. On considère les fontions suivantes :

```
int
g(int v)
{
  return 2 * v + 3;
}

int
suite_g(int n)
{
  int v = 0;
  for(int i = 1; i < n; i++)
    v = g(v);
  return v;
}

int
serie_g(int k)
{
  int s = 0;
  for(int i = 1; i < k; i++)
    s += suite_g(i);
  return s;
}
```

1. Quel est le résultat de l'appel `suite_g(4)` ?
2. Exprimer en fonction de `k` le nombre d'appels à la fonction `g` nécessaires au calcul `serie_g(k)`.
3. On désire améliorer la complexité du calcul précédent en modifiant seulement la fonction `suite_g`. Pour réaliser cette optimisation, utiliser *des* variables statiques pour mémoriser le dernier résultat calculé.

Exercice 6. Qu'affiche le programme suivant ?

```
#include <stdio.h>

int a = 1;

void f(void) {
  a++;
}

void g(void) {
  int a = 10;
  a++;
}

void affiche (void) {
  printf("%d\n", a);
}

void main(void) {
  affiche();
  f();
  affiche();
  f();
  affiche();
  g();
  affiche();
}
```

Exercice Complémentaire 1. Analyser chacune des fonctions suivantes (fonctionnement, valeur de retour en fonction des paramètres, effets de bord) : On suppose que l'appel des fonctions est effectué avec un paramètre qui vaut 3.

```
int f1(int i) {
    return i+1;
}

int f2(int i) {
    return ++i;
}

int f3(int i) {
    return i++;
}
```

```
int f4(int i) {
    printf("%d\n",i++);return i;
}

int f5(int i) {
    printf("%d\n",i==0);return i;
}

int f6(int i) {
    printf("%d\n",i=0);return i;
}
```

2 Travaux Pratiques

Télécharger et décompacter l'archive contenant les fichiers tp.

Exercice 7. Compiler et exécuter (après avoir éventuellement apporté les corrections nécessaires) les fichiers correspondants aux exercices de la partie td. Vérifier que vous obtenez bien les résultats attendus.

Exercice 8. Le fichier `suite.c` contient les fonctions de l'exercice sur les suites.

1. Compiler et exécuter (on ne modifiera pas pour l'instant la fonction `suite_g`).
2. On veut maintenant compter le nombre d'appels à la fonction g et l'afficher avant la terminaison du programme. Modifier le programme en conséquence.
3. Améliorer la complexité en modifiant la fonction `suite_g`.