université Paris Ouest Nanterre La Défense

TD 3 : les fonctions

Exercice 1. – Appels de fonction

Soit le programme suivant :

```
#include<stdio.h>
int f1(int a, int b) {
  return a*b;
}

int f2(int x, int y) {
  int res;
  res = x + f1(x,y);
  return res;
}

int f3(int u, int v, int w) {
  int res;
  res = f2(u,w) + f2(v,w);
  return res;
}

int main(int arv,char * arg[])  {
  prinft("%d \n",f3(3,2,4));
  return 0;
}
```

1. Qu'affiche le programme précédent. (PS : il faut répondre sans avoir implémenter le programme sur la machine !)
2. Modifiez les fonctions `f2` et `f3` pour qu'elles n'utilisent plus de variables locales (elles ne doivent pas non plus utiliser de variables globales !).

Exercice 2. – Afficher et retourner

Soit le programme suivant :

```
#include <stdio.h>

int main(int arv,char * arg[]){
  int a, b, c;
  a = 5;
  b = 3;
  c = min2Int(a,b);
  printf("le minimum de %d et de %d est %d \n",a,b,c);
  return 0;
}
```

1. Définissez la fonction `min2Int` appelée dans la fonction `main` précédente et qui **retourne** le minimum de 2 entiers.
2. En utilisant la fonction `min2Int` définie à la question précédente, écrivez une nouvelle fonction `min3Int` qui **affiche** le minimum de 3 entiers.
3. Écrivez la fonction `main` permettant de tester la fonction min3Int.
4. Peut-on utiliser la fonction `min3Int` pour afficher le minimum de 5 entiers ?

Exercice 3. – Portée des variables

Soit le programme suivant :

```
#include <stdio.h>

int a, b, c;

int f(int x) {
  return a*x;
}

int g() {
  int c;
  c=a;
  a=b;
  b=c;
  return c;
}

int main(int arv,char * arg[]) {
  int x,y;
  a=2;
  b=3;
  c=4;
  x=f(a);
  y=g();
  prinft("%d %d %d %d %d \n",x,y,a,b,c);
  return 0;
}
```

1. Quelles sont les variables globales ?
2. Quelles sont les variables locales à `f` ?
3. Quelles sont les variables locales à `g` ?
4. Quelles sont les variables locales à `main` ?
5. Quelles sont les paramètres formels de `f` ?
6. Quelles sont les paramètres formels de `g` ?
7. Quelles sont les paramètres effectifs de `f` ?
8. Que vaut `c` à la fin de l'appel à la fonction `g` ?
9. Que fait la fonction `g` ?
10. Qu'affiche le programme ?
11. Que retourne le programme ?

Exercice 4. – Réutiliser et imbriquer des fonctions

1. Donnez la signature et écrivez puis testez une fonction `divise` qui prend en argument deux entiers et retourne un entier. La fonction retourne la valeur *1* lorsque le premier argument est un diviseur du second argument. Elle retourne la valeur *0* sinon.
2. En utilisant la fonction `divise`, écrivez une fonction `ecrire_division` qui prend en argument deux entiers et écrit sur la sortie standard si le second divise le premier. Par exemple :

```
ecrire_division(4,2) -> 2 divise 4
ecrire_division(5,2) -> 2 ne divise pas 5
```

3. Toujours en utilisant la fonction `divise`, écrivez maintenant une fonction `nb_div` qui renvoie le nombre de diviseurs d'un entier `n` passé en paramètre.
4. Écrivez une fonction `premier` qui renvoie vrai ou faux selon que l'entier `n` passé en paramètre est premier ou non.
5. Considérons la fonction main suivante :

```
int main(int arv,char * arg[]) {
  int x = 4;

  printf("divise(3,6) = %d \n ",divise(3,6));
  printf("divise(3,7) = %d \n ",divise(3,7));
  if(premier(x))
    printf("%d est premier \n ",x);
  else
    printf("%d n'est premier \n ",x);

  return 0;
}
```

Donner la liste des appels de fonctions effectués par ce programme.

Exercice 5. – Types et appels de fonctions

L'objectif de cet exercice est de comprendre le passage de paramètre en C. Bien que ce soit un exercice d'approfondissement, il est fortement recommandé que tous les étudiants le fassent au moins chez eux, s'ils n'ont pas eu le temps de le voir en TD.

Soit le programme suivant :

```
#include <stdio.h>

int f1(int a, int b) {
  return a*b;
}

int f2(int a, int b) {
  if (b==1) {
    return a*a;
  } else {
    return -1;
  }
}
```

```
int f3(int a, char b) {
  if (b=='c') {
    return a*a;
  } else {
    return -1;
  }
}

int main(int arv,char * arg[]) {
  printf("%d \n",f1(3,2)+f2(2,1)-f3(4,'g'));
  printf("%d \n",f1(3,2)+f2(2,0)-f3(4,'c'));
  printf("%d \n",f1(2,3.0));
  printf("%d \n",f3(2,z));
  printf("%d \n",f3(2,"t"));

  return 0;
}
```

1. Qu'affiche ce programme ? Expliquez.
2. Que se passe-t-il si on ajoute les instructions suivantes dans la fonction `main` ? Expliquez.

```
printf("%d \n",f1(2,3.0));
printf("%d \n",f3(2,3));
printf("%d \n",f3(2,z));
printf("%d \n",f3(2,"t"));
```

3. Donnez des noms plus explicites aux fonctions `f1`, `f2` et `f3`.

