Université Paris-Nord — L1
Institut Galilée — Année 2011-2012
Licence 1 - 1er semestre

Travaux dirigés 12

1 Écriture et appel de fonctions

1.1 Un programme de géométrie

Nous souhaitons écrire un programme de géométrie permettant le calcul d'aires de formes simples : l'utilisateur spécifie les dimensions des figures et le programme calcule les aires de ces figures. Afin de tester les fonctions, un premier programme est écrit :

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* Declaration des constantes et types utilisateurs */
#define PI 3.1415926

/* Declaration des fonctions utilisateurs */
/* calcule l'aire d'un rectangle */
double aire_rectangle(double longueur,double largeur);
/* calcule l'aire d'un cercle */
double aire_cercle(double rayon);
/* calcule l'aire d'un triangle */
double aire_triangle(double base,double hauteur);

int main()
{
    double longueur;
    double largeur;
    double cote;
    double rayon;
    double base;
    double hauteur;

    printf("Entrez la longueur et largeur du rectangle : ");
    scanf("%lg",&longueur);
    scanf("%lg",&largeur);
    printf("L'aire du rectangle est %g.\n",aire_rectangle(longueur,largeur));

    printf("Entrez le rayon du cercle : ");
    scanf("%lg",&rayon);
    printf("L'aire du cercle est %g.\n",aire_cercle(rayon));

    printf("Entrez la base et la hauteur du triangle : ");
    scanf("%lg",&base);
    scanf("%lg",&hauteur);
    printf("L'aire du triangle est %g.\n",aire_triangle(base,hauteur));

    return EXIT_SUCCESS;
}

/* Definitions des fonctions utilisateurs */
```

- Est-ce que le programme compile ? Est-ce que l'édition de liens réussit ? **Les réponses doivent être succintement expliquées**.
- Définir les fonctions manquantes.
- Afin de s'assurer de la correction du programme, faire sa trace.

1.2 Calcul de la constante e

- Écrire la fonction `factorielle` qui prend en entrée un entier positif ou nul n et qui renvoie sa factorielle $n!$, avec la convention $0! = 1$.
- Écrire la fonction `e_approchee` qui prend en entrée un entier n et qui renvoie $\sum_{i=0}^{i=n} \frac{1}{n!}$. Cette serie converge vers la constante e, la constante de Néper, la base des logarithmes naturels. On utilisera la fonction `factorielle`
- Écrire un complet programme simple qui permet de tester `e_approchee`.

2 Révision

2.1 Calcul du quotient d'une division entière par soustractions successives (*4 points*)

Nous voulons écrire une fonction `quotient_par_soustraction` qui prend deux entiers strictement positifs a et b en paramètre, et retourne le quotient de la division entière de a par b. Le nombre a est appelé le dividende et b le diviseur. Cette fonction doit calculer le quotient par soustractions successives **sans utiliser l'opérateur / du C (division entière)**. Le programme principal est déjà écrit pour pouvoir tester la fonction :

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

/* declarations constantes et types utilisateurs */

/* declarations de fonctions utilisateurs */

int main()
{
  int dividende;
  int diviseur;

  printf("Entrez le dividende et le diviseur : ");
  scanf("%d",&dividende);
  scanf("%d",&diviseur);

  while(dividende <= 0 || diviseur <= 0) /* pas les deux strictement positifs */
  {
      printf("Erreur. Entrez le dividende et le diviseur : ");
      scanf("%d",&dividende);
      scanf("%d",&diviseur);
  }

  printf("le quotient de la division euclidienne est : %d\n",
         quotient_par_soustraction(dividende,diviseur));

  return EXIT_SUCCESS;
}

/* definitions de fonctions utilisateurs */
```

1. Est-ce que le programme compile ?
2. Compléter le programme pour que l'édition de liens réussise et produise un exécutable.

Deux exemples d'exécution du programme sont :

```
Entrez le dividende et le diviseur : 23 4
le quotient de la division euclidienne est : 5
```

```
Entrez le dividende et le diviseur : 1 4
le quotient de la division euclidienne est : 0
```

2.2 Exercices sur des tableaux, sans fonctions (*5 points*)

Pour les deux exercices suivants, tout le traitement sera effectué dans le `main`, sans faire appel à des fonctions utilisateurs.

2.2.1 Somme de deux vecteurs (*1,5 points*)

Rappel : la somme de deux vecteurs du plan se fait terme à terme comme ceci :

$$(x, y) + (x', y') = (x + x', y + y').$$

Nous voulons faire un programme qui réalise la somme de deux vecteurs de dimension N. Un vecteur sera représenté par un tableau.

Écrire un programme qui, étant donnés deux tableaux de réels, u et v, de tailles N, calcule dans un tableau w de taille N la somme terme à terme de u et de v, puis affiche le contenu de w. Définir `N` comme une constante symbolique valant 2. Les tableaux u et v seront initialisés à des valeurs de votre choix.

Tout le traitement sera effectué dans le `main`*, sans faire appel à des fonctions utilisateurs.*

2.2.2 Unicité des éléments d'un tableau (*3,5 points*)

Nous disposons d'un tableau t de N caractères et nous souhaitons savoir si chaque caractère apparaissant dans le tableau n'y apparaît qu'une seule fois, autrement dit on veut savoir si chaque caractère est unique.

1. Écrire un programme qui, étant donné un tableau initalisé t, teste si le premier élément du tableau est unique et affiche `Vrai` si c'est le cas, `Faux` sinon.
2. Écrire un programme qui étant donné un tableau initialisé t, teste si tous les éléments sont uniques et affiche `Vrai` si c'est le cas, `Faux` sinon.

Tout le traitement sera effectué dans le `main`*, sans faire appel à des fonctions utilisateurs.*