Université Paris-Nord — L1
Institut Galilée — Année 2008-2009
Licence 1 - 1er semestre

Rattrapage d'Éléments d'Informatique

Durée : 3 heures.

Documents autorisés : Aucun.

Recommandations : Un barème vous est donné à titre indicatif afin de vous permettre de gérer votre temps. La notation prendra en compte à la fois la syntaxe et la sémantique de vos programmes, c'est-à-dire qu'ils doivent compiler correctement. Une fois votre programme écrit, il est fortement recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.

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%d",&dividende,&diviseur);

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

```
  }

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

  return EXIT_SUCCESS;
}

/* definitions de fonctions utilisateurs */
```

1. Ajouter uniquement la déclaration de la fonction `quotient_par_soustraction` dans le programme principal, à la bonne place.
2. Est-ce que le programme compile ? Est-ce que l'édition de liens réussit ? **Les réponses doivent être succintement expliquées**.
3. Écrire la fonction `quotient_par_soustraction`.

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 : 0 4
le quotient de la division euclidienne est : 0
```

2 Boucles *for* et *while (7 points)*

Ces boucles ont exactement la même sémantique et on peut facilement réécrire l'une en l'autre. Par convention, on préfère utiliser la boucle *for* lorsque l'on connaît le nombre d'itérations à l'avance ; on utilise *while* dans le cas contraire, lorsque le nombre d'itérations n'est pas connu à l'avance. Par exemple, pour faire la somme des éléments d'un tableau on utilisera *for* et pour trouver un élément dans un tableau, ne sachant pas où il se trouve, on choisira *while*. L'intérêt de respecter cette convention est que la lecture d'un programme est facilité en indiquant à quoi sert la boucle.

Résoudre les problèmes suivants en utilisant soit *for*, soit *while*.

2.1 Nombre d'occurrences dans un tableau *(3,5 points)*

Écrire un programme qui fait initialiser un tableau d'entiers de taille `TAILLE` (constante symbolique à définir) par l'utilisateur, demande à l'utilisateur un entier, et affiche le nombre d'occurrences de l'entier dans le tableau. Des exemples de sorties sont les suivants :

```
Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
-2
Il y a 3 occurrences de -2 dans le tableau.

Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
3
```

```
Il y a 1 occurrences de 3 dans le tableau.

Saisissez 4 entiers : -2 -2 -2 3
Compte le nombre d'occurrences de quel entier ?
0
Il y a 0 occurrences de 0 dans le tableau.
```

2.2 Comptage du nombre d'éléments inférieurs et supérieurs à un seuil donné (*3,5 points*)

Écrire un programme principal qui demande à l'utilisateur d'entrer un entier positif ou nul `seuil`, d'entrer une suite d'entiers positifs ou nuls, puis affiche combien d'entiers de la suite sont strictement plus grand que `seuil` et combien sont strictement plus petit que `seuil`. La lecture d'un *entier négatif* marquera la fin de la suite. Un exemple de sortie est :

```
Entrez le seuil : 3
Entrez la suite en terminant par un nombre negatif : 1 2 3 4 5 0 -1
Il y a 2 nombres > seuil et 3 nombres < seuil.
```

N.B. : On ne peut pas stocker la série dans un tableau, car on ne connait pas la taille de la suite. Le calcul du nombre d'entiers plus grands et plus petits que le seuil se fait de façon incrémentale, après chaque lecture d'un nouvel entier.

3 Trace d'un programme avec fonctions (*4 points*)

Simulez l'exécution du programme, en réalisant sa **trace** : l'exécution du programme est représenté par un tableau **par appel de fonction**, à $n + 2$ colonnes : la première colonne étant le numéro de la ligne exécutée, les n colonnes suivantes, les colonnes des n variables de la fonction (n à déterminer) et la dernière colonne, la colonne servant à l'affichage à l'écran du programme. Seules les lignes modifiant l'état mémoire du programme sont à reporter dans le tableau. Notez bien que l'utilisateur doit saisir deux entiers. Vous considèrerez, comme indiqué en commentaires, qu'il saisit 4 puis 3.

```
1   #include <stdlib.h> /* EXIT_SUCCESS */
2   #include <stdio.h> /* printf, scanf */
3
4   /* declarations constantes et types utilisateurs */
5
6   /* declarations de fonctions utilisateurs */
7   int puissance(int base,int exposant);
8
9   int main()
10  {
11      int x;
12      int y;
13
14      printf("Entrez les valeurs de la base puis de l'exposant : ");
15      scanf("%d",&x); /* 4 est saisi par l'utilisateur */
```

```
16      scanf("%d",&y); /* 3 est saisi par l'utilisateur */
17
18      printf("%d exposant %d = %d\n",x,y,puissance(x,y));
19
20      return EXIT_SUCCESS;
21  }
22
23  /* definitions de fonctions utilisateurs */
24  int puissance(int base,int exposant)
25  {
26      int i; /* var. boucle */
27      int produit = 1; /* resultat */
28
29      for(i = 0;i < exposant;i = i + 1)
30      {
31          produit = produit * base;
32      }
33
34      return produit;
35  }
```

4 Écriture de fonctions (*5 × 1 point*)

1. Écrire la fonction `carre` qui prend en entrée un réel et qui renvoie le carré de ce réel.
2. Écrire la fonction `majeure` qui prend en entrée un entier représentant l'age en années d'une personne et affiche si cette personne est majeure ou mineure.
3. Écrire la fonction `demande_taille` qui demande à l'utilisateur sa taille en centimètres, et qui renvoie cette valeur.
4. Écrire la fonction qui prend en entrée deux entiers `n` et `m` et qui renvoie :

$$\sum_{i=0}^{i=n-1} \left(\sum_{j=0}^{j=n-1} i+j \right)$$

5. Écrire la fonction qui prend en entrée deux entiers `n` et `m` et qui renvoie :

$$\sum_{i=0}^{i=n-1} \left(\sum_{j=0}^{j=i-1} i+j \right)$$