ESIEE IN4A11

Programmation

Algorithmique
Exercices simples

Jean-Claude GEORGES

Année scolaire 2007-2008

1 Exemple d'écriture d'une fonction

Écrire une fonction qui calcule la somme des n premiers nombres
- à l'aide d'une boucle `for` ;
- en utilisant la récursivité ;
- en utilisant une formule.

Exemple de solution :

```
/* int somme_nombres_iteratif(int n)
 * calcule la somme des n premiers nombres
 * Principe : accumulation dans une variable int des n premiers nombres
 * arguments : n entier positif
 * retour : somme des n premiers entiers si n>0
0 si n est négatif ou nul
 * JCG écrit le 22/03/2001 modifié le 1/09/2006
 */
int somme_nombres_iteratif(int n) {
  int i, s=0;
  for (i=1;i<=n;++i) s+=i;
  return s;
}
/*********************************************************************/
/* int somme_nombres_recursif(int n)
 * calcule la somme des n premiers nombres
 * Principe : S(n)=0 si n ==0, S(n)=n+S(n-1) sinon
 * arguments : n entier positif
 * retour : somme des n premiers entiers si n>0
0 si n est nul
indéterminé si n est négatif
 * JCG écrit le 22/03/2001 modifié le 1/09/2006
*/
int somme_nombres_recursif(int n) {
  if (n==0)
    return 0;
  else
    return n + somme_nombres_recursif(n-1);
}
/*********************************************************************/
/* int somme_nombres_formule(int n)
 * calcule la somme des n premiers nombres
 * Principe : S(n)=(n*(n+1))/2
 * arguments : n entier positif
 * retour : somme des n premiers entiers si n>0
0 si n est nul
somme des |n|-1 premiers entiers si n est négatif
 * JCG écrit le 22/03/2001 modifié le 1/09/2006
*/
int somme_nombres_formule(int n) {
  return (n*(n+1))/2;
}
/*********************************************************************/
#include <stdio.h>
int main(void) {
  int n; scanf("%d",&n);
  printf("somme_nombres_iteratif(%d) = %d \n",n,somme_nombres_iteratif(n));
  printf("somme_nombres_recursif(%d) = %d \n",n,somme_nombres_recursif(n));
  printf("somme_nombres_formule (%d) = %d \n",n,somme_nombres_formule(n));
  return 0;
}
```

2 Factorielle (*)

Écrire une fonction qui retourne la factorielle d'un entier positif ou nul ($5! = 5 \times 4 \times 3 \times 2 \times 1$). Que se passe-t-il en cas de dépassement de capacité ?

Prototype : `int factorielle(int n) ;`

3 PGCD (*)

Écrire une fonction qui retourne le plus grand commun diviseur (*pgcd*) de deux nombres entiers positifs par l'algorithme d'Euclide :

$$\text{pgcd}(a,b) = \begin{cases} a & \text{si } b = 0 \\ \text{pgcd}(b, a \bmod b) & \text{sinon} \end{cases}$$

Prototype : `int pgcd(int a, int b) ;`

4 Conversion Fahrenheit - Centigrade (*)

Écrivez la fonction qui convertit les degrés Fahrenheit en degrés centigrades $\left[\theta_C = \frac{5}{9}(\theta_F - 32)\right]$

Prototype : `double F_vers_C(double F) ;`

5 Volume d'une sphère (*)

Écrivez la fonction qui calcule le volume d'une sphère étant donné son rayon $\left(V = \frac{4}{3}\pi R^3\right)$

Prototype : `double volume_sphere(double rayon) ;`

6 Conversion kilomètres-miles (*)

Écrivez la fonction qui convertit les kilomètres en miles (1 mile = 1,609 km)

Prototype : `double km_vers_mile(double km) ;`

7 Plus petit diviseur premier (**)

Écrire la fonction qui retourne le plus petit diviseur premier d'un nombre entier supérieur à 1.

Prototype : `int plus_petit_diviseur_premier (int n) ;`

8 Racine carrée entière (**)

Écrire une fonction qui calcule la racine carrée entière d'un nombre entier positif par soustractions successives des nombres impairs.

Principe : Si $\sum_{i=1}^{p}(2i-1) \leq n < \sum_{i=1}^{p+1}(2i-1)$, alors $p \leq \sqrt{n} < p+1$

Exemple : racine de 43

$43 - 1 = 42, 42 - 3 = 39, 39 - 5 = 34, 34 - 7 = 27, 27 - 9 = 18, 18 - 11 = 7$

6 soustractions en tout, donc la racine entière de 43 est 6

Prototype : `int racine_entiere(int n) ;`

9 Racine carrée (**)

Écrire une fonction qui calcule la racine carrée d'un nombre réel positif par l'algorithme de Newton.
Principe : La suite définie par

$$\left| \begin{array}{l} u_0 = 1 \\ u_{n+1} = \dfrac{u_n + \dfrac{a}{u_n}}{2} \end{array} \right.$$

converge vers $\sqrt{a}$
Prototype : `double racine_carre(double x) ;`

10 Suite bizarre (**)

Écrire la fonction qui calcule le $n^{\text{ème}}$ terme de la suite initialisée par

$$\left| \begin{array}{l} u_0 = \dfrac{1}{3} \\ u_{n+1} = 4u_n - 1 \end{array} \right.$$

Testez pour $n = 100$. Quel est le résultat théorique ? Expliquez l'écart.
Prototype : `double suite_bizarre(int n) ;`

11 Suite de Fibonacci (**)

Écrire une fonction qui retourne le nème terme d'une suite de Fibonacci initialisée par a et b.

$$\left| \begin{array}{lcll} u_0 & = & a & \\ u_1 & = & b & \\ u_n & = & u_{n-1} + u_{n-2} & \text{pour } n \geq 2 \end{array} \right.$$

Prototype : `int fibonacci(int n, int a, int b) ;`

12 Puissance entière itérative (**)

Écrire une fonction qui calcule a^n avec a réel et n entier positif. Utiliser un algorithme itératif.
Prototype : `double puissance(double a, int n) ;`

13 Puissance entière récursive (**)

Écrire une fonction qui calcule a^n avec a réel et n entier positif. Utiliser un algorithme récursif selon le principe :

$$a^n = \left| \begin{array}{ll} 1 & \text{si } n = 0 \\ (a^{n/2})^2 & \text{si } n \text{ est pair} \\ a \cdot a^{n-1} & \text{si } n \text{ est impair} \end{array} \right.$$

Comparer les temps d'exécution avec la fonction précédente pour $n = 100, 1000, 10000$
Prototype : `double puissance_rapide(double a, int n) ;`

14 Développements limités (**)

Écrire les fonctions qui calculent les développements limités à l'ordre n de $\sin x$, $\cos x$, e^x et $\arctan x$.
Rappel :

$$\left|\begin{array}{lcl} \sin x & = & x - \dfrac{x^3}{3!} + \dfrac{x^5}{5!} - \dfrac{x^7}{7!} + \ldots \\ \cos x & = & 1 - \dfrac{x^2}{2!} + \dfrac{x^4}{4!} - \dfrac{x^6}{6!} + \ldots \\ \mathrm{e}^x & = & 1 + x + \dfrac{x^2}{2!} + \dfrac{x^3}{3!} + \dfrac{x^4}{4!} + \ldots \\ \arctan x & = & x - \dfrac{x^3}{3} + \dfrac{x^5}{5} - \dfrac{x^7}{7} + \ldots \end{array}\right.$$

Prototypes :

```
double dev_lim_sin(double x, int n) ;
double dev_lim_cos(double x, int n) ;
double dev_lim_exp(double x, int n) ;
double dev_lim_arn(double x, int n) ;
```

15 Notes (**)

Un professeur note les résultats d'un test portant sur 50 questions en utilisant la table suivante :

bonnes réponses	0-10	11-20	21-30	31-40	41-50
note	E	D	C	B	A

Écrire la fonction qui retourne la note, étant donné un nombre bonnes réponses.
Prototype : `char note (int bonnes_reponses)`

16 Salaire (**)

Écrivez la fonction ayant en paramètres le nombre d'heures effectuées par un salarié et son salaire horaire, qui retourne sa paye hebdomadaire. Les heures sup. (au-delà de 35 heures) sont payées à 150%.
Prototype : `double salaire_hebdomadaire(int nb_heures, double salaire_horaire)`

17 Table de Pythagore (**)

Écrivez une fonction qui affiche la table de Pythagore de la multiplication.

```
    1   2   3   4   5   6   7   8   9  10
 1  1   2   3   4   5   6   7   8   9  10
 2  2   4   6   8  10  12  14  16  18  20
 3  3   6   9  12  15  18  21  24  27  30
 4  4   8  12  16  20  24  28  32  36  40
 5  5  10  15  20  25  30  35  40  45  50
 6  6  12  18  24  30  36  42  48  54  60
 7  7  14  21  28  35  42  49  56  63  70
 8  8  16  24  32  40  48  56  64  72  80
 9  9  18  27  36  45  54  63  72  81  90
10 10  20  30  40  50  60  70  80  90 100
```

ou mieux

```
+---+---+---+---+---+---+---+---+---+---+---+
|  x|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10|
+---+---+---+---+---+---+---+---+---+---+---+
|  1|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10|
+---+---+---+---+---+---+---+---+---+---+---+
|  2|  2|  4|  6|  8| 10| 12| 14| 16| 18| 20|
+---+---+---+---+---+---+---+---+---+---+---+
|  3|  3|  6|  9| 12| 15| 18| 21| 24| 27| 30|
+---+---+---+---+---+---+---+---+---+---+---+
|  4|  4|  8| 12| 16| 20| 24| 28| 32| 36| 40|
+---+---+---+---+---+---+---+---+---+---+---+
|  5|  5| 10| 15| 20| 25| 30| 35| 40| 45| 50|
+---+---+---+---+---+---+---+---+---+---+---+
|  6|  6| 12| 18| 24| 30| 36| 42| 48| 54| 60|
+---+---+---+---+---+---+---+---+---+---+---+
|  7|  7| 14| 21| 28| 35| 42| 49| 56| 63| 70|
+---+---+---+---+---+---+---+---+---+---+---+
|  8|  8| 16| 24| 32| 40| 48| 56| 64| 72| 80|
+---+---+---+---+---+---+---+---+---+---+---+
|  9|  9| 18| 27| 36| 45| 54| 63| 72| 81| 90|
+---+---+---+---+---+---+---+---+---+---+---+
| 10| 10| 20| 30| 40| 50| 60| 70| 80| 90|100|
+---+---+---+---+---+---+---+---+---+---+---+
```

Prototype : `void affiche_pythagore(int n) ;`

18 Nombre de bits à un (***)

Écrire une fonction qui retourne le nombre de bits à 1 d'un entier quelconque.
Prototype : `int nb_bits_a_1(int n) ;`

19 Codage machine d'un entier (***)

Écrire la fonction qui affiche le codage d'un entier quelconque.
Prototype : `void affiche_codage(int n) ;`

20 Calcul de π (***)

Écrire une fonction qui calcule une valeur approchée de π par le calcul du développement limité à l'ordre n de $\arctan 1 (= \frac{\pi}{4})$, multiplié par 4. Jusqu'à quelle valeur de n faut-il pousser le calcul pour avoir trois chiffres significatifs ?

Écrire une fonction qui calcule une valeur approchée de π par le calcul des développements limités à l'ordre n correspondant à la formule de Machin $\left(\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}\right)$. Jusqu'à quelle valeur de n faut-il pousser le calcul pour avoir trois chiffres significatifs ?

21 Curiosité (********)

Commentez et étudiez ce programme. Discutez de son intérêt.

```
/* calcul de Pi à 240 décimales */
#include <stdio.h>
long a=10000, b, c=840, d, e, f[841], g;
int main () {
  for (;b-c;)
    f[b++]=a/5;
  for (;d=0,g=c*2;c-=14,printf("%.4d",e+d/a),e=d%a)
    for (b=c;d+=f[b]*a,f[b]=d%--g, d/=g--,--b;d*=b);
  return 0;
}
```

22 Représentation mémoire (**)

Soit le morceau de programme suivant :

```
int a;
int *pa;
double x;
int **p;
a=8;
pa=&a;
x=3.14159;
p=&pa;
**p=281;
```

En supposant que la mémoire soit structurée en octets, que les entiers soient codés sur 2 octets et les pointeurs sur 4 octets et que la zone de mémoire automatique soit située en début d'exécution à l'adresse 2006 (décimal), représentez la mémoire après l'exécution de ce début de programme.

23 Échanges (**)

- Soit la fonction suivante :
  ```
  void echange1 (int x, int y) {
    int z;
    z=x;x=y;y=z;
  }
  ```
  Pourquoi ne fonctionne-t-elle pas lorsqu'on l'appelle avec par exemple
  `a=2;b=3;echange1(a,b);`
  Représentez la mémoire lors de l'exécution de ce morceau de programme.
- Soit la fonction suivante :
  ```
  void echange2 (int *x, int *y) {
    int *z;
    *z=*x;*x=*y;*y=*z;
  }
  ```
  Pourquoi peut-elle ne pas fonctionner lorsqu'on l'appelle avec par exemple
  `a=2;b=3;echange2(&a,&b);`
  Représentez la mémoire lors de l'exécution de ce morceau de programme.
- Soit la fonction suivante :
  ```
  void echange3 (int *x, int *y) {
    int z;
    z=*x;*x=*y;*y=z;
  }
  ```

et l'appel suivant :
`a=2;b=3;echange2(&a,&b);`
Représentez la mémoire lors de l'exécution de ce morceau de programme.

– Soit la fonction suivante

```
void echange(type *x, type *y) {
  *y=*x + *y;
  *x=*y - *x;
  *y=*y - *x;
}
```

où *type* peut être `double` ou `int`. Faites tourner cette fonction à la main avec par exemple
`a=2;b=3;echange(&a,&b);`
Fonctionne-t-elle dans tous les cas ? Donnez un exemple avec des `double` et un exemple avec des `int` où elle est incorrecte.