QROC RAN Informatique – Novembre 2014

Répondre sur la feuille impérativement. Polycopié de cours seul autorisé.

NOM :
Prénom :
Filière :

Compilation

Chaque ligne du tableau ci-dessous contient une portion de code qui doit être considérée comme indépendante des autres.

- Indiquer (par une croix) si la portion de code est correcte ou provoque une erreur de compilation ou provoque une erreur du *linker* ou peut provoquer un effet indésirable à l'exécution

Code	Correct	Pb Compilation	Pb Linker	Pb possible à l'exécution

`#include <stdio.h>`
`#include <math.h>`
`#define MAX 360`
`double t[MAX];`
`int i,j;`
`int main() {`
`  for (i = 0; i <= MAX; i++)`
`  t[i]= sin(i*2*M_PI/10);`
`  for (i = 0; i <= MAX; i++)`
`  printf("%lg\n", t[i]);`
`  return 0;`
`}`
				X (déborde ment de tableau)

`#include <stdio.h>`
`int main() {`
`  double x = 7,y;`
`  y = x + 2;`
`  printf("x=%d y=%d\n", x, y);`
`  return 0;`
`}`
				X (spécifica teur de format incorrect)

`#include <stdio.h>`
`int f321(int m);`
`int main() {`
`  int n = 7;`
`  printf("%d",f321(n*n + 5));`
`  return 0;`
`}`
`int f321(int m){`
`  return m*m;`
`}`
	X			

`#include <stdio.h>`
`int main() {`
`  scanf("%d",&n);`
`  while(n!=1)`
`  if (n%2==0) n/=2;`
`  else n=3*n+1;`
`printf("fin\n");`
`return 0;`
`}`
		X
		(n non déclaré)		

Fonctions-procédures

Le programme ci-dessous représente une version procédurale, légèrement différente de celle vue en cours, de l'algorithme de résolution d'une équation du premier degré.

- Ecrire la procédure *reso1* qui est compatible avec l'appel présenté ci-dessous.

```
#include <stdio.h>
// écrire la procédure reso1 ci-dessous


void reso1(int a, int b, double* psol, int *pcode) {
    if (a==0)
        if (b==0) *pcode=2 ;
        else *pcode=0 ;
    else {
        *pcode=1 ;
        *psol=-b/(double)a ;
    }
}



int a,b,code;
double x;
int main() {
    printf("a : ");scanf("%d",&a);
    printf("b : ");scanf("%d",&b);
    reso1(a,b,&x,&code);
    switch(code) {
        case 0:printf("0 sol.");break;
        case 1:printf("1 sol.= %lg",x);break;
        case 2:printf("inf. sol.");break;
    }
    return 0;
}
```

Fonction

La suite de Fibonacci est définie par énumération ainsi :

n	0	1	2	3	4	5	6	n

fib(n)	0	1	1	2	3	5	8	fib(n-1)+ fib(n-2)

La définition mathématique récurrente de cette fonction est la suivante :

$fib(0)=0$
$fib(1)=1$
$n\in\mathbb{N};\forall n>1\ fib(n)=fib(n-1)+fib(n-2)$

- Rappeler la fonction récursive, directement déduite de cette définition mathématique, de la fonction de Fibonacci.

```
int fib(int n) {
    if (n==0 || n==1) return n ;
    else return fib(n-1)+fib(n-2) ;
}
```

Il existe une généralisation de cette suite de Fibonacci pour des valeurs de n négatives, notée ***fibGen*** ci-dessous.

- En remarquant que
  $\forall n\in\mathbb{Z}\ fib(n)=fib(n-1)+fib(n-2)\Rightarrow fib(n-2)=fib(n)-fib(n-1)$
  compléter les 6 valeurs manquantes ci-dessous correspondant aux valeurs négatives de n

n	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	**n**
fibGen(n)	-8	5	-3	2	-1	1	0	1	1	2	3	5	8	**fibGen(n-1)+ fibGen(n-2)**

- Compléter le code ci-dessous pour prendre en compte les valeurs de n négatives

```
#include <stdio.h>
int fib(int n) {
   // supposée correctement fait dans la question précédente
}
int fibGen(int n) {
     if (n<0) {
          return fibGen(n+2)-fibGen(n+1) ;
    }
     else return fib(n) ;
}

int main() {
    int n;
    for(n=-10;n<=10;n++) printf("fibGen(%d)=%d\n",n,fibGen(n));
    return 0;
}
```

Chaînes de caractères

On considère le programme de détection d'anagramme suivant

```
#include <stdio.h>
#include <string.h>
int isAna(char *mot1, char *mot2) {
     char alpha1[26],alpha2[26];
     int i;
     int ana=1;
     for(i=0;i<26;i++) alpha1[i]=alpha2[i]=0;
     for(i=0;i<strlen(mot1);i++) alpha1[mot1[i]-'a']++;
     for(i=0;i<strlen(mot2);i++) alpha2[mot2[i]-'a']++;
     for(i=0;i<26;i++)
          if (alpha1[i]!=alpha2[i]) ana=0;
     return ana;
}

int main() {
```

```
     char mot1[50],mot2[50];
    printf("mot1 ? : ");scanf("%s",mot1);
    printf("mot2 ? : ");scanf("%s",mot2);
    if (isAna(mot1,mot2))
       printf ("%s est l'anagramme de %s\n",mot1,mot2);
    else
       printf("%s et %s ne sont pas des anagrammes\n",mot1,mot2);
    return 0;
}
```

Voici deux exemples d'exécution de ce programme :

mot1 ? : parisien

mot2 ? : aspirine

parisien est l'anagramme de aspirine

mot1 ? : salut

mot2 ? : radar

salut et radar ne sont pas des anagrammes

- Quel est le principe de la fonction *isAna* ? En particulier quel est le rôle de chaque tableau *alpha1* et *alpha2* ?

alpha1 et *alpha2* sont 2 tableaux qui contiennent le nombre d'occurrence de chaque lettre de l'alphabet contenue dans, respectivement, *mot1* et *mot2*. Leur dimension est donc 26, ce qui correspond au nombre de lettres dans l'alphabet.

Chaque élément de ces tableaux est associé à une lettre de l'alphabet, en commençant par la lettre *a sur l'indice zéro.* Le contenu de chaque élément du tableau correspond aux nombre d'occurrences de la lettre correspondante dans le mot donné.

On détermine que 2 mots sont anagrammes l'un de l'autre si la répartition des occurrences dans chaque tableau *alpha1* et *alpha2* est identique.

Le code **`for(i=0;i<26;i++) alpha1[i]=alpha2[i]=0; initialise chacun des tableaux`**

Le code

```
for(i=0;i<strlen(mot1);i++) alpha1[mot1[i]-'a']++;
```

```
for(i=0;i<strlen(mot2);i++) alpha2[mot2[i]-'a']++;
```

lit chaque mot lettre à lettre et incrément la case correspondante de chaque tableau alpha. L'indice *mot1[i]-'a'* permet d'associer naturellement la lettre trouvée dans le mot avec sa position dans le tableau *alpha*.

Le code

```
for(i=0;i<26;i++)
     if (alpha1[i]!=alpha2[i]) ana=0;

return ana;
```

et retourne le résultat booléen dans *ana*

On suppose qu'on dispose d'une fonction, appelée *sort*, qui retourne à partir d'une chaîne de caractères sa version ordonnée.

Par exemple :

`printf("%s", sort("salut"))` produit l'affichage suivant : **`alstu`**

- En supposant que vous disposez de cette fonction *sort,* écrire votre propre version de *isAna*

```
int isAna(char *mot1, char *mot2) {
    // vous disposez de sort et des fonctions de string.h
    return strcmp(sort(mot1),sort(mot2))==0 ;
}
```