Faculté d'ingénierie et de technologie – Génie informatique

Algorithmique et programmation

Damien Berthet & Vincent Labatut

TP 04 | instructions de contrôle

Présentation

Le but de ce TP est de manipuler les structures de contrôle vues en cours, et en particulier les boucles.

1 Utilisation de la bibliothèque mathématique

Vous aurez besoin d'utiliser la fonction mathématique `math.h`, qui contient certaines fonctions nécessaires lors de quelques calculs. Comme pour les autres bibliothèques, il est nécessaire de mentionner explicitement son utilisation dans votre programme grâce à la directive `#include` :

```
#include <math.h>
```

Sous Windows, c'est suffisant. Sous Linux, il faut en plus paramétrer votre projet de la façon suivante :

- Dans *Project Explorer* (à gauche), faites un clic-droit sur votre projet et sélectionnez *Properties*.
- Allez dans *C/C++ Build* puis *Settings*.
- Dans l'onglet *Tool Settings*, allez dans *MinGW C Linker* puis *Libraries*.
- À droite, dans *Libraries (-l)*, cliquez sur le bouton permettant d'ajouter une bibliothèque, et entrez simplement `m`, puis validez avec *OK*.
- Cliquez sur *OK* pour fermer la fenêtre.

2 Test simple avec alternative

Exercice 1

Écrivez un programme qui demande à l'utilisateur de saisir trois entiers a, b et c, puis qui affiche le plus grand de ces trois entiers. Respectez exactement la forme de l'exemple donné ci-dessous.

exemple :

```
Entrez les valeurs de a, b et c (sous la forme a:b:c) : 8:12:-32
Le plus grand des entiers est 12.
```

Exercice 2

Écrivez un programme qui demande à l'utilisateur d'entrer trois réels a, b et c puis qui affiche les solutions de l'équation $ax^2 + bx + c = 0$ en fonction des valeurs de a, b et c. Vous devez utiliser la fonction `sqrt()` définie dans la bibliothèque `math.h`, qui calcule la racine carrée d'un réel.

exemples :

- Deux solutions :

```
Entrez les valeurs des coefficients a, b et c du trinome (a doit etre non-nul).
a = 2.1
b = 1.2
c = 0.1
```

```
L'equation  2.10x^2 +  1.20x +  0.10 = 0 admet deux solutions :
   x1 = -0.470142  et  x2 = -0.101287.
```

- Une seule solution :

```
Entrez les valeurs des coefficients a, b et c du trinome (a doit etre non-nul).
a = 4
b = 4
c = 1
L'equation 4.00x^2 + 4.00x + 1.00 = 0 admet exactement une solution :
   x = -1.000000
```

- Pas de solution :

```
Entrez les valeurs des coefficients a, b et c du trinome (a doit etre non-nul).
a = 0.1
b = -1.2
c = 5.2
L'equation  0.10x^2 + -1.20x +  5.20 = 0 n'admet pas de solution.
```

3 Boucles simples

Exercice 3

Écrivez un programme qui :

- Saisit un entier ;
- Renverse l'entier ;
- Affiche le résultat.

exemple :

```
Entrez l'entier a renverser : 1234
Resultat : 4321
```

Exercice 4

Donnez le *tableau de situation* pour la valeur utilisée dans l'exemple précédent.

4 Boucles imbriquées

Exercice 5

Écrivez un programme qui affiche (en respectant l'alignement indiqué dans l'exemple ci-dessous) la somme des p premiers entiers, pour p allant de 2 à $N = 10$.

exemple :

```
1 + 2 = 3
1 + 2 + 3 = 6
...
...
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
```

Exercice 6

Faites une copie du programme précédent, et modifiez-la de manière à demander à l'utilisateur de saisir la valeur de n :

- Si l'utilisateur saisit une valeur strictement positive, le programme effectue la même tâche qu'à l'exercice précédent, puis redemande une valeur n à l'utilisateur pour recommencer le traitement.
- Si l'utilisateur saisit une valeur négative ou nulle, le programme s'arrête.

exemple :

```
Entrez n ou une valeur<=0 pour terminer : 2
1 + 2 = 3
Entrez n ou une valeur<=0 pour terminer : 4
1 + 2 = 3
1 + 2 + 3 = 6
1 + 2 + 3 + 4 = 10
Entrez n ou une valeur<=0 pour terminer : -1
```

```
Le programme se termine.
```

5 Suite de Fibonacci

On considère la suite de Fibonacci (u_n), $n \in \mathbb{N}$ définie par :

$$\begin{cases} u_0 = 0; u_1 = 1 \\ \forall n \geq 2 : u_n = u_{n-1} + u_{n-2} \end{cases}$$

Exercice 7

En utilisant une boucle, écrivez un programme qui calcule *itérativement* le $n^{ème}$ terme de la suite de Fibonacci. La valeur n doit être demandée à l'utilisateur. À chaque itération, le programme doit afficher u_n.

exemple :

```
Entrez la valeur de n : 16
u0=0
u1=1
u2=1
u3=2
u4=3
u5=5
u6=8
u7=13
u8=21
```

```
u9=34
u10=55
u11=89
u12=144
u13=233
u14=377
u15=610
u16=987
```

Exercice 8

On peut montrer que la suite (u_n), $n \in \mathbb{N}$ vérifie $u_n \underset{+\infty}{\sim} \phi^n$, où ϕ est le nombre d'or. Ainsi, $\lim_{n \to +\infty} \frac{u_n}{u_{n-1}} = \phi$. Faites une copie de votre programme et modifiez-la pour tester cette propriété (utilisez le type `double`). À chaque itération, le programme doit afficher u_n et le rapport $\frac{u_n}{u_{n-1}}$.

Remarque : $\phi = \frac{1+\sqrt{5}}{2} \cong 1{,}61803398874989484820458683436 56$.

exemple :

```
Entrez la valeur de n : 20
u0=0
u1=1
u2=1
  u2/u1=1.000000000000000
u3=2
  u3/u2=2.000000000000000
u4=3
  u4/u3=1.500000000000000
u5=5
  u5/u4=1.666666666666667
u6=8
  u6/u5=1.600000000000000
u7=13
  u7/u6=1.625000000000000
u8=21
  u8/u7=1.615384615384615
u9=34
  u9/u8=1.619047619047619
u10=55
  u10/u9=1.617647058823529
```

```
u11=89
  u11/u10=1.618181818181818
u12=144
  u12/u11=1.617977528089888
u13=233
  u13/u12=1.618055555555556
u14=377
  u14/u13=1.618025751072961
u15=610
  u15/u14=1.618037135278515
u16=987
  u16/u15=1.618032786885246
u17=1597
  u17/u16=1.618034447821682
u18=2584
  u18/u17=1.618033813400125
u19=4181
  u19/u18=1.618034055727554
u20=6765
  u20/u19=1.618033963166706
```

Exercice 9

Faites une copie de votre programme et modifiez-la pour déterminer quel est le rang n à partir duquel ϕ est estimé avec une précision de 10^{-10}. À chaque itération, le programme doit afficher u_n, $\frac{u_n}{u_{n-1}}$ et l'erreur commise.

exemple :

```
u0=0
u1=1
u2=1
u2/u1=1.000000000000000
erreur : -0.618033988740000
u3=2
u3/u2=2.000000000000000
erreur : 0.381966011260000
u4=3
u4/u3=1.500000000000000
erreur : -0.118033988740000
u5=5
u5/u4=1.666666666666667
erreur : 0.048632677926667
u6=8
u6/u5=1.600000000000000
erreur : -0.018033988740000
```

```
u7=13
u7/u6=1.625000000000000
erreur : 0.006966011260000
u8=21
u8/u7=1.615384615384615
erreur : -0.002649373355385
u9=34
u9/u8=1.619047619047619
erreur : 0.001013630307619
u10=55
u10/u9=1.617647058823529
erreur : -0.000386929916470
u11=89
u11/u10=1.618181818181818
erreur : 0.000147829441818
...
```

Remarque : la fonction `fabs`, contenue dans la bibliothèque mathématique `math.h`, permet d'obtenir la valeur absolue d'un réel.

6 Devinette

On veut programmer un jeu simple, dont les règles sont les suivantes :

- D'abord, le programme choisit un nombre entre 1 et 100 au hasard ;
- Puis, il demande à l'utilisateur de deviner cette valeur.
- Si l'utilisateur a bien deviné, le jeu s'arrête.
- Sinon, le programme indique à l'utilisateur si la valeur proposée est trop grande ou trop petite. Puis, l'utilisateur doit proposer une nouvelle valeur, et on recommence ainsi jusqu'à ce que la valeur du programme soit trouvée.

Le nombre de tentatives n'est pas limité : le jeu s'arrête quand l'utilisateur a deviné la valeur choisie par le programme. Le score de l'utilisateur correspond au nombre de tentatives. Bien sûr, le but du jeu est de trouver la valeur en un minimum de tentatives.

Exercice 10

Écrivez un programme qui implémente ce jeu.

exemple :

```
J'ai tiré un nombre au hasard : essayez de le deviner !
  Premiere tentative ? 28
  La valeur 28 est trop grande.
  Tentative 2 ? 5
  La valeur 5 est trop petite.
  Tentative 3 ? 8
  Oui, c'est bien ca (8) ! Bravo, vous avez gagne !
Votre score est : 3
```

Remarque : pour tirer au hasard un entier n entre 1 et 100, utilisez le code source suivant (qui sera expliqué lors d'un TP futur). Vous avez besoin d'inclure la bibliothèque `time.h`.

```
srand(time(NULL));
int n = 1 + rand()%100;
```

À votre avis, quelle stratégie le joueur doit-il adopter pour obtenir le meilleur score (i.e. le plus petit score) ?