Chapitre 2

Structures itératives

2.1 Boucles simples

EX. 22 Que fait cette suite d'instructions :

```
a=1;
while (a <= 5)
{
  cout << a << " ";
  a+=1;
}
cout << a << endl;
```

EX. 23 Que fait cette suite d'instructions :

```
a=1;
do
{
  cout << a << " ";
  a+=1;
}
while (a <= 5);
cout << a << endl;
```

EX. 24 Que fait cette suite d'instructions :

```
a=0;
while (a < 5)
{
  a+=1;
  cout << a << " ";
}
cout << a << endl;
```

EX. 25 Que fait cette suite d'instructions :

```
a=0;
do
{
  a+=1;
  cout << a;
```

```
}
while(a != 5);
cout << a << endl;
```

EX. 26 Écrire un programme qui lit un nombre `n` sur *input* et qui affiche tous les nombres naturels entre 0 et `n` compris, de manière croissante.

Variantes :

1. Tous les nombres entre 0 et `n` compris, de manière décroissante ;
2. Tous les nombres pairs entre 0 et `n` compris, de amnière croissante ou décroissante ;
3. Tous les nombres entre 0 et `n` bornes non-comprises, de manière coirssante ;
4. *etc.* Inventez-en d'autres !

EX. 27 Écrire un programme qui calcule la moyenne des résultats d'une interrogation, ces notes étant données en *input*, la fin des données étant signalée par la valeur sentinelle -1 (on suppose aussi que la suite des notes contient toujours au moins un élément).

EX. 28 Écrire l'équivalent de l'instruction suivante avec une boucle `while` et puis avec une boucle `do` :

```
int i;

for(i=a;i<=b;++i)
{
  <instruction;>
}
```

EX. 29 Modifier le code ci-dessous en utilisant une boucle `for` :

```
int i,sum=0;

cin >> i ;
while(i>0)
{
  i--;
  sum+=2*i;
}
cout << sum << endl;
```

EX. 30 Modifier le code ci-dessous en utilisant une boucle `while` :

```
double prod;
int i,cpt;

prod=1;
cpt=0;
for(cin >> i;i<100;++i)
{
  cpt++;
  prod*=i;
}
cout << (prod/cpt) << endl;
```

EX. 31 Écrire un programme qui affiche sur output la valeur de $n!$ (où $n! = n \times (n-1) \times (n-2) \cdots 2 \times 1$) avec n lu sur *input*.

EX. 32 Écrire un programme qui lit sur *input* une liste de valeurs entières se trouvant dans l'intervalle $[-max, max]$ et terminée par une valeur sentinelle $> max$ (on peut prendre $max = 100$ pour fixer les idées). Écrire un programme qui affiche le nombre de valeurs < 0 de cette liste de valeurs.

EX. 33 Écrire un programme qui affiche sur *output* les nombres entiers strictement positifs dont le carré est inférieur à un nombre entier n lu sur *input*.

EX. 34 Écrire un programme qui affiche sur output les puissances de 2 à exposants entiers positifs qui sont inférieures à n lu sur *input*.

EX. 35 Écrire un programme qui affiche sur output les 10 premières puissances (à exposants entiers > 0) d'un nombre a lu sur *input*.

EX. 36 Écrire le programme qui lit en *input* un nombre $Max > 0$ et une suite de nombres entiers strictement positifs. On demande d'afficher sur l'*output* les premiers éléments de cette suite, déterminés par la condition que leur somme ne dépasse pas Max. De façon précise, si $a_1, a_2, a_3, \ldots, a_n$ est la suite de nombres, on demande d'afficher les nombres $a_1, a_2, a_3, \ldots, a_k$ tels que $a_1 + a_2 + \cdots + a_k \leq Max$ et si $k < n$, $a_1 + a_2 + \cdots + a_{k+1} > Max$.

Aucune hypothèse n'est faite quant à l'existence d'une solution. La suite de nombres est suivie d'une valeur sentinelle -1.

EX. 37 Écrire le programme qui lit en *input* deux nombres b_i, b_s et puis une liste non vide de valeurs entières triées par ordre croissant. On demande d'afficher sur l'*output* tous les nombres de cette liste appartenant à l'intervalle $[b_i, b_s]$. On suppose que la dernière valeur de la liste est toujours strictement supérieure à b_s.

Avec :

3	25	-7	-5	1	3	12	18	42	51
b_i	b_s	liste							

On imprime :

3 12 18

EX. 38 On suppose que sur l'*input* se trouve une suite d'entiers positifs, terminée par la valeur sentinelle -1. Comparer le premier (noté A) et le dernier élément (noté B) de la suite et afficher un message suivants :

A `est inférieur à` B
A `est strictement supérieur à` B
`La suite est vide`

Si la suite ne compte qu'un élément, on considère que $A = B$.

EX. 39 Ecrire un programme qui calcule F_n le n^{e} nombre de FIBONACCI dont la définition est la suivante : $\forall n \geq 2 : F_n = F_{n-1} + F_{n-2}$ avec $F_0 = 0$ et $F_1 = 1$.

EX. 40 Le *nombre d'or* est la limite de la suite :

$$F_2/F_1\ ,\ F_3/F_2\ ,\ F_4/F_3\ ,\ \ldots\ ,\ F_{n+1}/F_n\ ,\ \ldots$$

où F_n, le nombre de FIBONACCI de rang n, est défini par :

$$F_n = F_{n-1} + F_{n-2} \qquad F_0 = 0 \qquad F_1 = 1$$

Écrire un algorithme qui calcule ce nombre d'or avec une précision ε (ε est lu sur *input*, $1.0E-5$ par exemple).

EX. 41 Que réalise l'algorithme suivant ?

```
#include <iostream>

using namespace std ;

int main()
{
 int d, r;

 cin >> r >> d;
 while ( r >= d )
 {
   r -= d;
 }
 cout << r << endl;
}
```

EX. 42 Que réalise l'algorithme suivant ? Utilisez des exemples pour vous donner l'intuition.

```
#include <iostream>

using namespace std ;

int main()
{
 int a, n, d, r;

 cin >> a >> n;
 r=a;
 d=n;
 while ( r >= d )
 {
   d *= 3;

 }
 while (d != n)
 {
   d /= 3;

   while ( r >= d )
   {
   r -= d;
   }
 }
 cout << r << endl;
}
```

EX. 43 Écrire un programme qui lit en *input* une suite de nombres strictement positifs (sentinelle : 0, la suite comporte au moins deux nombres) et indique si elle est croissante, décroissante, strictement croissante, strictement décroissante ou non triée.

EX. 44 Écrire le code qui calcule la valeur approchée de π sur base de la série suivante (due à Leibniz). Les calculs s'arrêtent lorsque la valeur du dernier terme ajouté est inférieure à un ε

donné, ou lorsque le nombre de termes considérés atteint une borne également donnée.

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

EX. 45 Écrire le code qui calcule la valeur approchée de π sur base de la série suivante. Les calculs s'arrêtent lorsque la valeur du dernier terme ajouté est inférieure à un ε donné, ou lorsque le nombre de termes considérés atteint une borne également donnée.

$$\frac{\pi}{8} = \frac{1}{1 \cdot 3} + \frac{1}{5 \cdot 7} + \frac{1}{9 \cdot 11} + \cdots$$

EX. 46 Écrire le code qui calcule la valeur approchée de π sur base de la série suivante (due à Euler). Les calculs s'arrêtent lorsque la valeur du dernier terme ajouté est inférieure à un ε donné, ou lorsque le nombre de termes considérés atteint une borne également donnée.

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots$$

EX. 47 Le nombre e peut être défini comme la limite d'une série :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!} + \cdots = \sum_{i \geq 0} \frac{1}{i!}$$

Il est connu que l'approximation :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

diffère de e d'au plus deux fois le terme suivant dans la série, c'est-à-dire d'au plus $2 \cdot \frac{1}{(n+1)!}$.

Écrire le code calculant e avec une approximation dont l'erreur est inférieure à une constante donnée (en ignorant les imprécisions dues aux arrondis sur machine).

✍: 25 min.

EX. 48 On peut calculer approximativement le sinus de x en effectuant la sommation des n premiers termes de la série infinie

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

où x est exprimé en radians.
Réécrire cette somme sous la forme $\sin(x) = \sum_{i \geq 0} f(i, x)$. (On vous demande donc de trouver $f(i, x)$). Écrire ensuite le code calculant de cette manière la valeur de $\sin(x)$ où x est lu sur *input*.
Continuer l'addition des termes successifs dans la série jusqu'à ce que la valeur d'un terme devienne inférieure (en amplitude) à une constante ε (exemple : $\varepsilon = 10^{-5}$)

2.2 Boucles imbriquées

EX. 49 Écrire une programme qui lit sur *input* une valeur naturelle `n` et qui affiche à l'écran un carré de `n` caractères `X` de côté, comme suit (pour `n` $= 6$) :

```
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
```

Variantes : afficher uniquement le bord du carré, afficher un rectangle, afficher le triangle supérieur,...

EX. 50 Écrire un programme qui lit sur *input* une valeur naturelle n et qui affiche ensuite à l'écran toutes les paires (i, j) telles que $0 \leq i \leq \mathtt{n}$ et $0 \leq j \leq \mathtt{n}$.

Ces paires seront affichées sur n lignes de la façon suivante :

$$\begin{array}{cccc} (0,0) & (0,1) & \cdots & (0,\mathtt{n}) \\ (1,0) & (1,1) & \cdots & (1,\mathtt{n}) \\ \vdots & \vdots & \vdots & \vdots \\ (\mathtt{n},0) & (\mathtt{n},1) & \cdots & (\mathtt{n},\mathtt{n}) \end{array}$$

EX. 51 Écrire un progamme qui lit sur *input* une valeur naturelle n et qui affiche ensuite à l'écran toutes les paires (i, j) telles que $0 \leq i \leq \mathtt{n}$ et $0 \leq j \leq i$. Ces paires seront affichées sur n lignes de la façon suivante :

$$\begin{array}{cccccc} (0,0) & & & & & \\ (1,0) & (1,1) & & & & \\ \vdots & \vdots & \vdots & \vdots & & \\ (\mathtt{n}-1,0) & (\mathtt{n}-1,1) & \cdots & (\mathtt{n}-1,\mathtt{n}-1) & & \\ (\mathtt{n},0) & (\mathtt{n},1) & \cdots & \cdots & & (\mathtt{n},\mathtt{n}) \end{array}$$

EX. 52 Refaire l'exercice 49 en supposant que n est impaire et en dessinant des 0 sur les deux diagonales principales à la places des X.

Exemple, pour $\mathtt{n} = 7$:

```
0XXXXX0
X0XXX0X
XX0X0XX
XXX0XXX
XX0X0XX
X0XXX0X
0XXXXX0
```