Algorithmique et programmation - Séance 1
Révisions - Solutions

Les structures conditionnelles et itératives.

Exercice 1.

Ecrire un programme qui lit au clavier les coefficients *a*, *b* et *c* de l'équation du second degré $ax^2 + bx + c = 0$ et affiche à l'écran, si elle(s) existe(nt), la ou les solutions réelles ou complexes de cette équation. Si l'équation n'a pas de solution, le programme le signalera en affichant un message à l'écran. Veillez à envisager tous les cas possibles.

```
#include <cmath>
#include <iostream>
using namespace std;

int main()
{
  float a, b, c, delta;

  cout << "Entrez les coefficients a, b et c." << endl;
  cin >> a >> b >> c;

  if (a == 0)
    if (b == 0)
      if (c == 0)
        cout << "Solutions: tous les nombres complexes." << endl;
      else // a == 0, b == 0 et c != 0
        cout << "Pas de solution." << endl;
    else // a == 0, b != 0
      cout << "Solution: " << -c / b << endl;
  else //a != 0
  {
    delta = b * b - 4 * a * c;
    if (delta == 0)
      cout << "Solution: " << -b / (2 * a) << endl;
    else //delta != 0
      if (delta > 0)
        cout << "Solutions: " << (-b + sqrt(delta)) / (2 * a)
          << " " << (-b - sqrt(delta)) / (2 * a) << endl;
      else //delta < 0
        cout << "Solutions: " << -b / (2 * a) << "+"
          << sqrt(-delta) / (2 * a) << "i "
          << -b / 2 * a << "-"
          << sqrt(-delta) / (2 * a) << "i" << endl;
  }
  return 0 ;
}
```

Exercice 2.

Ecrire un programme qui lit des nombres au clavier et les affiche tant que le dernier nombre lu est la somme des deux précédents.

Solution 1 :

```
#include <iostream>
using namespace std;

int main()
{
  int a, b, c;
```

```
  cin >> a >> b;
  cout << a << endl << b << endl;
  for(cin >> c; c == a + b; cin >> c)
  {
    cout << c << endl;
    a = b;
    b = c;
  }
  return 0;
}
```

Solution 2 :

```
#include <iostream>
using namespace std;

int main()
{
  int a, b, c;
  cin >> a >> b;
  cout << a << endl << b << endl;
  cin >> c;
  while(c == a + b)
  {
    cout << c << endl;
    a = b;
    b = c;
    cin >> c;
  }
  return 0;
}
```

Les fonctions.

Exercice 3.
Ecrire une fonction swap(*a*,*b*) qui échange les valeurs de 2 variables entières *a* et de *b*.

```
//swap.h
//Echange les contenus de a et b
void swap(int &a, int &b);

//swap.cpp
#include "swap.h"

void swap(int &a, int &b)
{
  int tmp = a;
  a = b;
  b = tmp;
}
```

Exercice 4.
Ecrire une fonction qui trie en ordre croissant trois variables entières.

```
#include <iostream>
#include "swap.h"
using namespace std;

//On applique le tri bulle au trois variables
void trier3Var(int &a, int &b, int &c)
{
  if (a > b)
```

```
    swap(a, b);
  //Ici a <= b
  if (b > c)
  {
    //Ici a <= b et b > c
    swap(b, c);
    //Ici c > a et c > b
    if (a > b)
      //Ici c > a > b
      swap(a, b);
    //Ici c > b >= a
  }
  //Ici c >= b >= a
}

int main()
{
  int a, b, c;

  cin >> a >> b >> c;
  trier3Var(a, b, c);
  cout << a << " " << b << " " << c << endl;
  return 0;
}
```

Exercice 5.
Ecrire une librairie contenant les fonctions maximum de deux entiers, minimum de deux entiers et valeur absolue d'un entier. Utiliser cette librairie pour écrire une fonction qui renvoie la valeur absolue du plus petit entier en valeur absolue parmi trois entiers passés en paramètres.

```
//util.h
//Retourne le minimum de a et b
int min(int a, int b);

//Retourne le maximum de a et b
int max(int a, int b);

//Retourne la valeur absolue de a et b
int abs(int a);

//util.cpp
#include "util.h"

int min(int a, int b)
{
  return (a < b) ? a : b;
}

int max(int a, int b)
{
  return (a < b) ? b : a;
}

int abs(int a)
{
  return (a < 0) ? -a : a;
}

//main.cpp
#include <iostream>
#include "util.h"
using namespace std;

int main()
```

```
{
  int a, b, c;

  cin >> a >> b >> c;
  cout << min(abs(a), min(abs(b), abs(c))) << endl;
  return 0;
}
```

Les tableaux.

Exercice 6.
Ecrire une fonction qui retourne le maximum d'un vecteur de *n* nombres entiers.

```
#include <iostream>
using namespace std;

int max(int tab[], int n)
{
  int res = tab[0];

  for(int i = 1; i < n; i++)
    if (tab[i] > res)
      res = tab[i];
  return res;
}

int main()
{
  int tab[] = {5, 7, 3, 2, 8, 4, 1};

  cout << max(tab, 7) << endl;
  return 0;
}
```

Exercice 7.
Ecrire une fonction *symetrieVect* qui opère une symétrie sur les éléments d'un vecteur de dimension *n*, c'est-à-dire qui transforme le vecteur (v_1, v_2, ..., v_n) en (v_n, v_{n-1}, ..., v_1).

```
#include <iostream>
#include "swap.h"
using namespace std;

void symetrieVect(int vect[], int n)
{
  int milieu = (n - 1) / 2;

  for(int i = 0; i <= milieu; i++)
    swap(vect[i], vect[n - 1 - i]);
}

int main()
{
  int vect[] = {1, 2, 3, 4, 5, 6};

  symetrieVect(vect, 6);
  for(int i = 0; i < 6; i++)
    cout << vect[i] << endl;
  return 0;
}
```

Exercice 8.
Un polynôme de degré *n* à coefficients entiers, $a_0 + a_1x + a_2x^2 + ... + a_nx^n$, où $a_n \neq 0$, peut être représenté dans un tableau d'entiers dont seuls les *n* premiers éléments sont significatifs, l'élément d'indice *i* (où $i \leq n$) du tableau contenant

le coefficient a_i du polynôme. Le polynôme nul, de degré 0, sera représenté par un tableau d'entiers dont l'unique élément significatif, d'indice 0, est nul.

On supposera que la fonction demandée ne pourra manipuler que des polynômes de degré inférieur ou égal à une constante *DEGRE_MAX*.

Exemple.

Le polynôme $1 + 4x + 7x^3 + 2x^4$ sera représenté par le tableau

1	4	0	7	2			

dont seuls les 5 premiers éléments sont significatifs.

En utilisant cette représentation d'un polynôme, on demande d'écrire une fonction booléenne qui calcule et renvoie le résultat de la multiplication d'un polynôme par le binôme $ax + b$ (la fonction recevra les coefficients *a* et *b* du binôme en paramètre). La fonction retournera la valeur booléenne *false* si le degré du polynôme résultat dépasse *DEGRE_MAX*, *true* sinon. La fonction devra envisager tous les cas possibles : polynôme nul, coefficient *a* du binôme nul, … La fonction ne pourra modifier le tableau représentant le polynôme à multiplier qui lui est passé en paramètre.

```
#include <iostream>
using namespace std;


const int DEGRE_MAX = 20;

void ecrirePolynome(double p[], int degre)
{
  cout << p[0];
  for(int i = 1; i <= degre; i++)
    cout << " + " << p[i] << "x" << i;
  cout << endl;
}

bool multParMonome(double p[], int degre, int a, int b, double pRes[],
  int &degreRes)
{
  if (!a)
  {
    if (!b)
    {
      pRes[0] = 0;
      degreRes = 0;
    }
    else
    {
      degreRes = degre;
      for (int i = 0; i <= degre; i++)
        pRes[i] = b * p[i];
    }
    return true;
  }
  else
    if (degre == DEGRE_MAX)
      return false;
    else
    {
      if (!degre && !p[0])
      {
        pRes[0] = 0;
        degreRes = 0;
      }
      else
```

```
      {
        degreRes = degre + 1;
        pRes[0] = b * p[0];
        for (int i = 1; i <= degre; i++)
          pRes[i] = b * p[i] + a * p[i - 1];
        pRes[degreRes] = a * p[degre];
      }
      return true;
    }
}


int main()
{
  double p[] = {2., 4., 3., 7.}, pRes[DEGRE_MAX+1];
  int degreRes;

  ecrirePolynome(p, 3);
  if (multParMonome(p, 3, 2, 5, pRes, degreRes))
    ecrirePolynome(pRes, degreRes);
  else
    cout << "dépassement de capacité" << endl;
  return 0 ;
}
```