MOOC Init Prog C++
Corrigés semaine 6

Les corrigés proposés correspondent à l'ordre des apprentissages : chaque corrigé correspond à la solution à laquelle vous pourriez aboutir au moyen des connaissances acquises jusqu'à la semaine correspondante.

Exercice 18 : générateur automatique de lettres

> Cet exercice correspond à l'exercice n°16 (pages 54 et 217)
> de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Première version du code :

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

void genereLettre()
{
  cout
    << "Bonjour chère Mireille,"
    << "Je vous écris à propos de votre cours."
    << "Il faudrait que nous nous voyons le 18/12 pour en discuter."
    << "Donnez-moi vite de vos nouvelles !"
    << "Amicalement, John."
}

int main()
{
  genereLettre();
  return 0;
}
```

Seconde version du code :

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

void genereLettre(bool masculin, string destinataire, string sujet,
                  unsigned int jour, unsigned int mois,
                  string politesse, string auteur)
```

```
{
  cout << "Bonjour ";
  if (masculin) cout << "cher";
  else cout << "chère";
  cout << " " << destinataire << "," << endl;

  cout
    << "Je vous écris à propos de " << sujet << endl
    << "Il faudrait que nous nous voyons le " << jour << "/" << mois
    << " pour en discuter." << endl
    << "Donnez-moi vite de vos nouvelles !" << endl
    << politesse << ", " << auteur << endl;
}

int main()
{
  genereLettre(false, "Mireille", "votre cours" , 18, 12,
               "Amicalement", "John");
  cout << endl;
  genereLettre(true, "John", "votre demande de rendez-vous", 16, 12,
               "Sincèrement", "Mireille");
  return 0;
}
```

Exercice 19 : nombres complexes

Cet exercice correspond à l'exercice n°22 (pages 60 et 226) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le code fourni ici est en C++11. Pour une version compilant avec l'ancien standard (C++98) voir ci-dessous.

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

struct Complexe {
  double x;
  double y;
};

// Solution simple
void affiche(Complexe z)
{
  cout << "(" << z.x << "," << z.y << ")";

  // autre solution :  cout << z.x << "+" << z.y << "i";
}

// Solution plus complexe mais plus élégante
void affiche2(Complexe z)
{
  if ((z.x == 0.0) and (z.y == 0.0)) {
    cout << "0";
    return;
  }

  if (z.x != 0.0) {
    cout << z.x;
    if (z.y > 0.0)
      cout << "+";
  }
  if (z.y != 0.0) {
    if ((z.x == 0.0) and (z.y == -1.0))
      cout << "-";
    else if (z.y != 1.0)
      cout << z.y;
    cout << "i";
  }
}
```

```
Complexe addition(Complexe z1, Complexe z2)
{
  return { z1.x + z2.x, z1.y + z2.y };
}

Complexe soustraction(Complexe z1, Complexe z2)
{
  return { z1.x - z2.x, z1.y - z2.y };
}

Complexe multiplication(Complexe z1, Complexe z2)
{
  return { z1.x * z2.x - z1.y * z2.y ,
           z1.x * z2.y + z1.y * z2.x };
}

Complexe division(Complexe z1, Complexe z2)
{
  const double r(z2.x*z2.x + z2.y*z2.y);
  return { (z1.x * z2.x + z1.y * z2.y) / r ,
    (z1.y * z2.x - z1.x * z2.y) / r };
}

int main()
{
  Complexe un = { 1.0, 0.0 };
  Complexe i  = { 0.0, 1.0 };

  affiche(un); cout << " + "; affiche(i); cout << " = ";
  Complexe j(addition(un, i));
  affiche(j); cout << endl;

  affiche(i); cout << " * "; affiche(i); cout << " = ";
  affiche(multiplication(i,i)); cout << endl;

  affiche(j); cout << " * "; affiche(j); cout << " = ";
  Complexe z(multiplication(j,j));
  affiche(z); cout << endl;

  affiche(z); cout << " / "; affiche(i); cout << " = ";
  affiche(division(z,i)); cout << endl;

  z= { 2.0, -3.0 };
  affiche(z); cout << " / "; affiche(j); cout << " = ";
  affiche(division(z,j)); cout << endl;

  return 0;
}
```

La seule différence en C++98, c'est que la syntaxe d'initialisation n'est pas permise en affectation. En clair, les expression du type :

```
return { z1.x + z2.x, z1.y + z2.y };
```

doivent être remplacées par une initialisation de variable :

```
Complexe z = { z1.x + z2.x, z1.y + z2.y };
return z;
```

De même dans le `main()`

```
z= { 2.0, -3.0 };
```

doit être remplacé par :

```
z.x = 2.0; z.y = -3.0;
```

Encore une illustration des avantages de la nouvelle norme C++11 !

Exercice 20 : QCM

Cet exercice correspond à l'exercice n°24 (pages 61 et 228) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le code fourni ici est en C++11. Pour une version compilant avec l'ancien standard (C++98) voir ci-dessous.

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

struct QCM {
  string question;
  vector<string> reponses;
  unsigned int solution;
};

typedef vector<QCM> Examen;

void affiche(const QCM& question);
unsigned int demander_nombre(unsigned int min, unsigned int max);
unsigned int poser_question(const QCM& question);
Examen creer_examen();

// ======================================================================
int main()
{
  unsigned int note(0);
  Examen exam(creer_examen());

  for (auto question : exam) {
    if (poser_question(question) == question.solution) {
      ++note;
    }
  }

  cout << "Vous avez trouvé " << note << " bonne";
  if (note > 1) cout << 's';
  cout << " réponse";
  if (note > 1) cout << 's';
  cout << " sur " << exam.size() << "." << endl;

  return 0;
}
```

```
// ============================================================================
void affiche(const QCM& q)
{
  cout << q.question << " ?" << endl;
  unsigned int i(0);
  for (auto reponse : q.reponses) {
    cout << "    " << ++i << "- " << reponse << endl;
  }
}

// ============================================================================
unsigned int demander_nombre(unsigned int a, unsigned int b)
{
  /* échange les arguments s'ils n'ont pas été donnés dans *
   * le bon sens.                                          */
  if (a > b) { unsigned int tmp(b); b=a; a=tmp; }

  unsigned int res;
  do {
    cout << "Entrez un nombre entier compris entre "
         << a << " et " << b <<" : ";
    cin >> res;
  } while ((res < a) or (res > b));

  return res;
}

// ============================================================================
unsigned int poser_question(const QCM& q)
{
  affiche(q);
  return demander_nombre(1, q.reponses.size());
}

// ============================================================================
Examen creer_examen()
{
  return {
    // Question 1
    { "Combien de dents possède un éléphant adulte",
      { "32", "de 6 à 10", "beaucoup", "24", "2" },
      2 // réponse
    },

    // Question 2
    { "Laquelle des instructions suivantes est un prototype de foncti
      { "int f(0);"     ,
```

```
        "int f(int 0);" ,
        "int f(int i);" ,
        "int f(i);"     },
      3 // réponse
    },

    // Question 3
    { "Qui pose des questions stupides",
      { "le prof. de math",
        "mon copain/ma copine",
        "le prof. de physique",
        "moi",
        "le prof. d'info",
        "personne, il n'y a pas de question stupide",
        "les sondages" } ,
      6 // réponse
    }
  };
}
```

La principale différence en C++98 est que la syntaxe d'initialisation n'est pas permise ni pour les `vectors`, ni en affectation pour les `structs`. Cela change pas mal la fonction `creer_examen` :

```
Examen creer_examen()
{
  QCM q;
  Examen retour;

  q.question = "Combien de dents possède un éléphant adulte";
  q.reponses.clear();
  q.reponses.push_back("32");
  q.reponses.push_back("de 6 à 10");
  q.reponses.push_back("beaucoup");
  q.reponses.push_back("24");
  q.reponses.push_back("2");
  q.solution=2;
  retour.push_back(q);

  q.question = "Laquelle des instructions suivantes est un prototype
  q.reponses.clear();
  q.reponses.push_back("int f(0);");
  q.reponses.push_back("int f(int 0);");
  q.reponses.push_back("int f(int i);");
  q.reponses.push_back("int f(i);");
  q.solution=3;
  retour.push_back(q);
```

```
  q.question = "Qui pose des questions stupides";
  q.reponses.clear();
  q.reponses.push_back("le prof. de math");
  q.reponses.push_back("mon copain/ma copine");
  q.reponses.push_back("le prof. de physique");
  q.reponses.push_back("moi");
  q.reponses.push_back("le prof. d'info");
  q.reponses.push_back("personne, il n'y a pas de question stupide");
  q.reponses.push_back("les sondages");
  q.solution=6;
  retour.push_back(q);

  return retour;
}
```

L'autre changement est que les boucle `for(:)` (*range-based `for`*) n'existent pas en C++98. Il faut les remplacer par des boucles «à la C» :

```
int main()
{
  //...
  for (unsigned int i(0); i < exam.size(); i++) {

    if (poser_question(exam[i]) == exam[i].solution) {

      ++note;
    }
  }
  //...
}

// ============================================================
void affiche(const QCM& q)
{
  cout << q.question << " ?" << endl;
  for (unsigned int i(0); i < q.reponses.size(); ++i) {

    cout << "    " << i+1 << "- " << q.reponses[i] << endl;

  }
}

// ...
```