Correction
Flots

Exercice 1 | Exercice 2 | Exercice 3 | Exercice 4

Exercice 1 : écriture dans un fichier

Exercice n°28 (pages 74 et 239) de l'ouvrage *C++ par la pratique*.

```cpp
#include <string>
#include <fstream>
#include <iostream>
#include <limits> // pour numeric_limits
using namespace std;

const string nom_fichier("data.dat"); // le nom du fichier

int main()
{
  ofstream fichier(nom_fichier);  // le flot à destination du fichier

  // on teste si l'ouverture du flot s'est bien réalisée
  if (fichier.fail()) {
    cerr << "Erreur: le fichier " << nom_fichier
         << " ne peut être ouvert en écriture !" << endl;
  } else {

    string nom;         // la donnée "nom" à lire depuis le clavier
    unsigned int age;   // la donnée "age" à lire depuis le clavier

    // itération sur les demandes à entrer
    do {
      cout << "Entrez un nom (CTRL+D pour terminer) : ";

      if (cin >> nom) {

        // L'utilisateur a bien saisi un nom, on peut donc lui demander
        // de saisir l'âge.

        cout << "âge : ";
        if (cin >> age) {
          // écriture dans le fichier
          fichier << nom << ' ' << age << endl;
        } else {
          // on n'a pas réussi à lire un entier
          cout << "Je vous demande un âge (nombre entier positif) pas "
               << "n'importe quoi !" << endl;
          cout << "Cet enregistrement est annulé." << endl;

          // On essaye de remettre le flot cin dans un état correct :
          // on "annule" l'état d'erreur
          cin.clear();
          // et on lit ce qui reste dans le flot
          cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
      } else {
        cout << endl; // purisme pour aller à la ligne à la fin
      }
    } while (not cin.fail()); // et on continue tant que cin est lisible.

    fichier.close(); // fermeture du flot fichier
  }
```

```
  return 0;
}
```

Exercice 2 : lecture depuis un fichier

> Exercice n°29 (pages 75 et 240) de l'ouvrage *C++ par la pratique*.

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

constexpr string nom_fichier("data.dat"); // le nom du fichier

int main()
{
  // le flot d'entrée en provenance du fichier
  ifstream fichier(nom_fichier);

  if (fichier.fail()) {
    cerr << "Erreur: le fichier " << nom_fichier
         << " ne peut etre ouvert en lecture !" << endl;
  } else {
    // si l'ouverture s'est bien produite...

    string nom;       // les données à lire dans le fichier...
    unsigned int age; // ...pas nécessaire de les initialiser

    unsigned int nb(0); // variables nécessaires aux différents calculs
    unsigned int age_max(0);
    unsigned int age_min(300); // je pense que 300 ans est assez large
    double total(0.0);

    // On commence par l'affichage du cadre
    cout << "+"
         << setfill('-') << setw(18) << "+"
         << setfill('-') << setw(6)  << "+" << endl
         << setfill(' ');

    /* Et on boucle directement sur la condition de lecture correcte
     * du couple <nom,age> (en fait, sur la condition de lecture
     * correcte de 'age', mais comme il n'est pas possible de lire
     * 'age' si la lecture de 'nom' a echoué...
     */

    while (fichier >> nom >> age) {
      // mise à jour des variables utilisées pour les calculs finaux
      ++nb;
      total += age;
      if (age_min > age) age_min = age;
      if (age_max < age) age_max = age;

      // ------------------------------------------------
      // AFFICHAGE
      // ------------------------------------------------

      cout.setf(ios::left);  // on définit un alignement à gauche;

      // le nom sur 15 caractères, en chaine de taille fixe
      cout << "| " << setw(15) << nom;

      /*  Pour l'affichage de l'age, on enleve le modificateur
       * 'alignement à gauche'
```

```
         */
        cout.unsetf(ios::left);

        // affiche l'age sur 3 caractères
        cout << " | " << setw(3) << age << " |" << endl;
    }

    // ------------------------------------------------
    // Partie finale
    // ------------------------------------------------

    fichier.close();  // ne pas oublier de fermer le fichier

    cout << "+"
         << setfill('-') << setw(18) << "+"
         << setfill('-') << setw(6)  << "+" << endl
         << setfill(' ');

    // les infos 'finales' ...
    cout.setf(ios::left);
    cout << setw(18) << "  âge minimum" << ": ";
    cout.unsetf(ios::left);
    cout << setw(3) << age_min << endl;

    cout.setf(ios::left);
    cout << setw(18) << "  âge maxmimal" << ": ";
    cout.unsetf(ios::left);
    cout << setw(3) << age_max << endl;

    cout << setw(2) << nb << " personnes, âge moyen : "
         << setw(4) << setprecision(3) << (total / nb) << " ans"
         << endl;
  }
  return 0;
}
```

Exercice 3 : statistiques sur un fichier

> Exercice n°30 (pages 76 et 242) de l'ouvrage *C++ par la pratique*.

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

```
#include <iostream>
#include <array>
#include <iomanip>
#include <fstream>
using namespace std;

// ===== CONSTANTES =====

// nombre maximum de demandes en cas d'erreur
constexpr unsigned short int NB_DEMANDES(3);

// taille maximum d'une Statistique : au plus 256 car il n'y a pas plus
// que 256 char - attention, avec des entiers signés, on ne peut aller
// au-dela du 127è caractère (sinon, les indices sont négatifs).
constexpr unsigned short int TAILLE(256);

// bornes sur les caractères à prendre en compte
constexpr char start(' ');
constexpr char stop('}');

// ===== DEFINITIONS DE TYPES ======
```

```cpp
typedef array<unsigned long int, TAILLE> Statistique;

// ===== FONCTIONS ======
bool demander_fichier(ifstream& fichier,
        unsigned short int max_demandes = NB_DEMANDES);

void initialise_statistique(Statistique& a_initialiser);

unsigned long int collecte_statistique(Statistique& a_remplir,
                                       ifstream& fichier_a_lire);

void affiche(const Statistique& a_afficher, unsigned long int total = 0,
      unsigned short int taille = TAILLE);

// ============================================================================
int main()
{
  ifstream f;
  if (not demander_fichier(f)) {
    cout << "=> j'abandonne !" << endl;
  } else {
    Statistique stat;
    initialise_statistique(stat);
    affiche(stat, collecte_statistique(stat, f), stop-start+1);
    f.close();
  }

  return 0;
}

/* ============================================================================
 * Fonction demander_fichier
 * ----------------------------------------------------------------------------
 * In: Un ifstream (par référence) à ouvrir et le nombre maximum de
 *     demandes (par défaut NB_DEMANDES).
 * Out:  Ouvert ou non ?
 * What: Demande à l'utilisateur (au plus max fois) un nom de fichier
 *       et essaye de l'ouvrir.
 * ============================================================================ */
bool demander_fichier(ifstream& f, unsigned short int max)
{
  string nom;
  unsigned short int nb(0);

  do {
    ++nb;

    // demande le nom du fichier
    do {
      cin.clear();
      cout << "Nom du fichier à lire : ";
      cin >> ws;
      getline(cin, nom);
    } while (cin.fail());

    // essaye d'ouvrir le fichier
    f.open(nom);

    // est-ce que ça a marché ?
    if (f.fail()) {
      cout << "-> ERREUR, je ne peux pas lire le fichier "
           << nom << endl;
    } else {
      cout << "-> OK, fichier " << nom << " ouvert pour lecture."
           << endl;
    }
  } while (f.fail() and (nb < max));
```

```
  return not f.fail();
}


/* =====================================================================
 * Fonction initialiser_statistique
 * ---------------------------------------------------------------------
 * In:   Une Statistique (par référence) à initialiser.
 * What: Initialiser tous les éléments d'une Statistique à zéro.
 * ===================================================================== */
void initialise_statistique(Statistique& stat)
{
  for (auto& nb : stat) {
    nb = 0;
  }
}

/* =====================================================================
 * Fonction collecte_statistique
 * ---------------------------------------------------------------------
 * In:   Une Statistique (par référence) à remplir et le fichier à lire.
 * Out:  Le nombre d'éléments comptés dans la Statistique.
 * What: Lit tous les caractères dans le fichier et compte dans la
 *       Statistique combien de fois chaque caractère apparait dans le
 *       fichier.
 * ===================================================================== */
unsigned long int collecte_statistique(Statistique& stat, ifstream& f)
{
  char c;  // le caractère lu
  unsigned long int nb(0); // le nombre d'éléments comptés

  while (f.get(c)) {
    if ((c >= start) and (c <= stop)) {
      ++(stat[c-start]); // Non portable !! Voir remarque ci-dessous.
      ++nb;
    }
  }

  return nb;
}

/* =====================================================================
 * Fonction affiche
 * ---------------------------------------------------------------------
 * In: La Statistique à afficher, le nombre par rapport auquel on
 *     affiche les pourcentages (si 0 recalcule ce nombre comme la
 *     somme des éléments) et la taille utilisée du tableau.
 * What: Affiche tous les éléments d'une Statistique (valeurs absolue
 *       et relative).
 * ===================================================================== */
void affiche(const Statistique& stat, unsigned long int nb,
      unsigned short int taille)
{
  if (nb == 0) {
    for (auto s : stat)
      nb += s;
  }

  const double total(nb);

  cout << "STATISTIQUES :" << endl << setprecision(3);
  for (unsigned short int i(0); i < taille; ++i) {
    if (stat[i] != 0) {
      cout << char(i+start) << " : " << setw(11) << stat[i] << " - "
           << setw(5) << 100.0 * stat[i] / total << "%" << endl;
    }
  }
}
```

Version C++98, les différences avec la version C++11 sont mises en évidence :

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

// ===== CONSTANTES =====

// nombre maximum de demandes en cas d'erreur
const unsigned short int NB_DEMANDES(3);

// taille maximum d'une Statistique : au plus 256 car il n'y a pas plus
// que 256 char - attention, avec des entiers signés, on ne peut aller
// au-dela du 127è caractère (sinon, les indices sont négatifs).
const unsigned short int TAILLE(256);

// bornes sur les caractères à prendre en compte
const char start(' ');
const char stop('}');

// ===== DEFINITIONS DE TYPES ======

typedef unsigned long int Statistique[TAILLE];


// ===== FONCTIONS ======
bool demander_fichier(ifstream& fichier,
        unsigned short int max_demandes = NB_DEMANDES);

void initialise_statistique(Statistique& a_initialiser);

unsigned long int collecte_statistique(Statistique& a_remplir,
           ifstream& fichier_a_lire);

void affiche(const Statistique& a_afficher, unsigned long int total = 0,
      unsigned short int taille = TAILLE);

// ============================================================================
int main()
{
  ifstream f;
  if (not demander_fichier(f)) {
    cout << "=> j'abandonne !" << endl;
  } else {
    Statistique stat;
    initialise_statistique(stat);
    affiche(stat, collecte_statistique(stat, f), stop-start+1);
    f.close();
  }
  return 0;
}

/* ============================================================================
 * Fonction demander_fichier
 * ----------------------------------------------------------------------------
 * In:   Un ifstream (par référence) à ouvrir et le nombre maximum de demande
 *       (par défaut NB_DEMANDES).
 * Out:  Ouvert ou non ?
 * What: Demande à l'utilisateur (au plus max fois) un nom de fichier et
 *       essaye de l'ouvrir
 * ============================================================================ */
bool demander_fichier(ifstream& f, unsigned short int max)
{
  string nom;
  unsigned short int nb(0);
```

```
  do {
    f.clear(); ++nb;

    // demande le nom du fichier
    do {
      cin.clear();
      cout << "Nom du fichier à lire : ";
      cin >> nom;
    } while (cin.fail());

    // essaye d'ouvrir le fichier
    f.open(nom.c_str());

    // est-ce que ça a marché ?
    if (f.fail()) {
      cout << "-> ERREUR, je ne peux pas lire le fichier "
    << nom << endl;
    } else {
      cout << "-> OK, fichier " << nom << " ouvert pour lecture."
    << endl;
    }
  } while (f.fail() and (nb < max));

  return (not f.fail());
}


/* =============================================================================
 * Fonction initialiser_statistique
 * -----------------------------------------------------------------------------
 * In:   Une Statistique (par référence) à initialiser.
 * What: Initialiser tous les éléments d'une Statistique à zéro.
 * ============================================================================= */
void initialise_statistique(Statistique& stat)
{
  for (int i(0); i < TAILLE; ++i) {
    stat[i] = 0;
  }
}

/* =============================================================================
 * Fonction collecte_statistique
 * -----------------------------------------------------------------------------
 * In:   Une Statistique (par référence) à remplir et le fichier à lire.
 * Out:  Le nombre d'éléments comptés dans la Statistique.
 * What: Lit tous les caractères dans le fichier et compte dans la Statistique
 *       combien de fois chaque caractère apparait dans le fichier.
 * ============================================================================= */
unsigned long int collecte_statistique(Statistique& stat, ifstream& f)
{
  char c;  // le charactère lu
  unsigned long int nb(0); // le nombre d'éléments comptés

  while (f.get(c)) {
    if ((c >= start) and (c <= stop)) {
      ++(stat[c-start]); // Non portable !! Voir remarque ci-dessous.
      ++nb;
    }
  }

  return nb;
}

/* =============================================================================
 * Fonction affiche
 * -----------------------------------------------------------------------------
 * In:   La Statistique à afficher, le nombre par rapport auquel on affiche
 *       les pourcentages (si 0 recalcule ce nombre comme la somme des
```

```
 *       éléments) et la taille utilisée du tableau.
 * What: Affiche tous les éléments d'une Statistique (valeurs absolue et
 *       relative).
 * ================================================================== */
void affiche(const Statistique& stat, unsigned long int nb,
      unsigned short int taille)
{
  if (nb == 0) {
    for (unsigned short int i(0); i < taille; ++i)
      nb += stat[i];
  }

  const double total(nb);

  cout << "STATISTIQUES :" << endl << setprecision(3);
  for (unsigned short int i(0); i < taille; ++i) {
    if (stat[i] != 0) {
      cout << char(i+start) << " : " << setw(11) << stat[i] << " - "
    << setw(5) << 100.0 * stat[i] / total << "%" << endl;
    }
  }
}
```

Remarques

En C/C++, le type `char` peut être un type signé (prenant valeur dans [-128,127]) ou non-signé ([0,255]), selon les systèmes (la norme laisse le choix au compilateur). Avec la plupart des compilateurs - dont gcc - il est cependant possible de préciser lors de la compilation la convention que l'on souhaite voir appliquer. Par conséquent, si l'on souhaite établir une statistique comprenant l'ensemble des caractères (ou simplement des caractères au-dela du 127^{e}), le code précédent ne fonctionne plus.

Il faut alors soit procéder à une gymnastique compliquée lors des comparaisons des valeurs ou des calculs d'indices, soit déclarer comme **caractères non signés** tous les `char` du programme (c.-à-d. `unsigned char`). Le problème est qu'on ne peut demander à la fonction `get()` d'extraire un caractère signé (`signed char`) ou non signé (`unsigned char`) ; elle ne sait qu'extraire un caractère (`char`). Pour pallier ce manque au niveau de la librairie standard, il n'y a guère d'autre choix que de lire un caractère (signé ou non, celui qui écrit le programme ne pouvant le savoir à l'avance) au moyen d'une variable intermédaire de type `char`, et de convertir ensuite ce caractère en un entier non signé ; idéalement, cela se fait avec l'un des *opérateurs de transtypage* (*cast*)...

Comme nous ne souhaitons pas entrer sur ce terrain (il y aurait beaucoup à dire pour que vous puissiez comprendre les subtilités de ces opérations de transtypage et ce qu'elles impliquent), voici une version «calculatoire» (donc peu efficace relativement à l'autre solution) :

```
// ...

// bornes sur les caractères à prendre en compte
const unsigned char start(' ');
const unsigned char stop('ÿ');

// ...

/* ====================================================================
 * Fonction collecte_statistique
 * --------------------------------------------------------------------
 * In:   Une Statistique (par référence) à remplir et le fichier à lire.
 * Out:  Le nombre d'éléments comptés dans la Statistique.
 * What: Lit tous les caractères dans le fichier et compte dans la Statistique
 *       combien de fois chaque caractère apparait dans le fichier.
 * ================================================================== */
unsigned long int collecte_statistique(Statistique& stat, ifstream& f)
{
  char c_lu;         // le caractère lu (signé ou non, suivant le système)
  unsigned char c;   // le caractère lu dans sa forme non signée
  unsigned long int nb(0); // le nombre d'éléments comptés
```

```
  while (f.get(c_lu)) {
    c = (256 + c_lu) % 256;  // obtenir la version positive de la représentation
    if ((c >= start) and (c <= stop)) {
      ++(stat[c-start]);
      ++nb;
    }
  }

  return nb;
}

// ...
```

Exercice 4 : QCM revisités

> Exercice n°31 (pages 77 et 244) de l'ouvrage *C++ par la pratique*.

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>
#include <fstream>
using namespace std;

// nombre maximum de demandes en cas d'erreur
constexpr unsigned short int NB_DEMANDES(3);

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(ifstream& fichier);
bool demander_fichier(ifstream& fichier,
        unsigned short int max_demandes = NB_DEMANDES);
string& enlever_blancs(string& chaine);

// ============================================================================
int main()
{
  unsigned int note(0);
  ifstream fichier;

  if (not demander_fichier(fichier)) {
    cout << "=> j'abandonne !" << endl;
  } else {
    Examen exam(creer_examen(fichier));

    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(0);

  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(ifstream& fichier)
{
  QCM q;
  Examen retour;
  bool erreur(false);       // une erreur de format s'est produite
  bool dansquestion(false); // en train de lire une question ?
  string line;              // ligne à lire

  while (getline(fichier, line) and not erreur) {
  // tant qu'on peut lire une ligne

    if ((line.size() > 0) and (line[0] != '#')) {
      // si ce n'est ni une ligne vide ni un commentaire

      if ((line[0] != 'Q') or (line[1] != ':')) {
        // si la ligne ne commence pas par "Q:"

 if (not dansquestion) {
   // Si on n'a pas encore eu de question : qqchose ne va pas !
   cerr << "Mauvais format de fichier : pas de \"Q:\"" << endl;
   erreur = true;
 } else {
   // on a déjà eu une question => c'est donc une ligne de réponse :
   // lecture d'une la réponse à la question

   if ((line[0] == '-') and (line[1] == '>')) {
     // reponse correcte
     line.replace(0,2,""); // supprime le "->" initial
```

```
      if (q.solution != 0) {
        cerr << "Hmmm bizarre, j'avais déjà une réponse correcte"
      << " pour cette question !" << endl;
        cerr << "Q: " << q.question << endl;
      }
      if (enlever_blancs(line) == "") {
        cerr << "??? la réponse indiquée comme correcte est vide !"
      << endl;
        erreur = true;
      } else {
        q.solution = q.reponses.size() + 1;
      }
    }

    // ajoute la réponse courante
    if (enlever_blancs(line) != "") q.reponses.push_back(line);
  }

       } else {
  // ligne de question : "Q: ..."
  if (dansquestion)
    retour.push_back(q); // ajoute la question précédente à
                                // l'examen

  line.replace(0,2,"");  // supprime le "Q:" initial
  if (enlever_blancs(line) == "") {
    cerr << "??? Question vide !!" << endl;
    erreur = true;
  } else {
    q.question = line;      // copie la question
    q.reponses.clear();     // remet à zéro les réponses...
    q.solution = 0;         // ...et la solution
    dansquestion = true;    // on a une question
  }
       }
     }
   }

   // Ne pas oublier la dernière question si elle existe
   if ((not erreur) and dansquestion) retour.push_back(q);

   return retour;
}

/* ========================================================================
 * Fonction demander_fichier
 * ------------------------------------------------------------------------
 * In:   Un ifstream (par référence) à ouvrir et le nombre maximum de
 *       demande (par défaut NB_DEMANDES).
 * Out:  Ouvert ou non ?
 * What: Demande à l'utilisateur (au plus max fois) un nom de fichier
 *       et essaye de l'ouvrir
 * ======================================================================== */
bool demander_fichier(ifstream& f, unsigned short int max)
{
   string nom;
   unsigned short int nb(0);

   do {
     f.clear(); ++nb;

     // demande le nom du fichier
     do {
       cin.clear();
       cout << "Nom du fichier à lire : ";
       cin >> nom;
     } while (cin.fail());

     // essaye d'ouvrir le fichier
```

```
    f.open(nom);

    // est-ce que ça a marché ?
    if (f.fail()) {
      cout << "-> ERREUR, je ne peux pas lire le fichier "
           << nom << endl;
    } else {
      cout << "-> OK, fichier " << nom << " ouvert pour lecture."
           << endl;
    }
  } while (f.fail() and (nb < max));

  return not f.fail();
}

// ============================================================================
string& enlever_blancs(string& chaine)
{ // Supprime les blancs initiaux et finaux d'une chaine

  size_t i;
  for (i = 0; (i < chaine.size()) and (chaine[i] == ' '); ++i);
  if (i > 0) chaine.replace(0, i, "");

  if (chaine != "") {
    for (i = chaine.size()-1; (i != 0) and (chaine[i] == ' '); --i);
    if (i < chaine.size()-1) chaine.replace(i+1, chaine.size(), "");
  }

  return chaine;
}
```

Exemple de fichier d'examen :

```
Q:Combien de dents possède un éléphant adulte
32
-> de 6 à 10
beaucoup
24
2

Q: Laquelle des instructions suivantes est un prototype de fonction
int f(0);
int f(int 0);
-> int f(int i);
int f(i);

Q: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

Q: Quel signe est le plus étrange
 #
 ->
->->##<-
#b
a
```

Pour compiler en C++98 :

1. remplacer constexpr par const

2. remplacer les itérations for (auto ... : ...) (*range-based for*) par les itérations « à la C » :

```
for (size_t  i(0); i < TABLEAU.size(); ++i)
```

3. ajouter .c_str() derrière la string nom dans le open() :

```
f.open(nom.c_str());
```

Dernière mise à jour : 06/09/2021