Correction TP de programmation n°1

Cours de programmation modulaire

—Licence MI/IM - Info 3—

Structures

La présence aux séances de TPs est obligatoire, et l'assiduité sera prise en compte dans l'évaluation pour le contrôle continu.

Les exercices un peu plus difficiles sont signalés par le symbole . Pour bien suivre la progression des travaux pratiques, il faut au moins avoir fait les exercices non marqués comme difficiles. Si vous n'avez pas eu le temps de les finir en TP, vous devez les finir par vous-même avant le prochain TP.

1 Exercices sur feuille

▸ **Exercice 1.**

Le créateur d'un jeu souhaite pouvoir modéliser des personnages participants à des combats. Un personnage aura un nom, des points de vie, et un bouclier lui permettant d'arrêter les coups. La structure de données choisie pour modéliser un personnage est donc la suivante :

```
1 struct Guerrier {
2   string nom;
3   int ptVie;
4   int bouclier;
5 };
```

La force des coups qu'il portera à son adversaire sera égale à son nombre de points de vie - 1 et il sera blessé s'il reçoit un coup d'une force supérieure à la force du bouclier qu'il porte. Un personnage blessé se verra retirer un nombre de point de vie égal à la différence entre la force du coup reçu et la force de son bouclier.

1. Spécifier et réaliser la procédure **afficheGuerrier** permettant d'afficher l'état d'un guerrier, i.e. son nom, et ses différentes propriétés.

```
1 /** affiche toutes les infos sur un guerrier
2  *  @param g de type Guerrier
3  **/
4 void afficheGuerrier(Guerrier g) {
5     cout << "personnage : " << g.nom << endl;
6     cout << "ptdeVie : " << g.ptVie << endl;
7     cout << "bouclier : " << g.bouclier << endl;
8 }
```

2. Écrire la fonction **créeGuerrier** qui permet d'initialiser un personnage, sachant qu'il doit disposer au départ d'exactement 10 points à répartir entre les points de vie du personnage (au moins deux points) et la force de son bouclier. On devra pouvoir saisir le nom du personnage, par exemple «Elisa», et lui donner des points de vie, par exemple 7 points. Dans ce cas, la fonction **créeGuerrier** devra lui affecter automatiquement un bouclier d'une force égale à $3 = (10 - 7)$.

   Vous pouvez utiliser, sans la redéfinir ni la réaliser, la fonction ci-dessous :

   `int litValeurBornee(string texteAEcrire, int min, int max);`

   Cette fonction permet, après avoir affiché le `texteAEcrire` explicitant la valeur attendue, de saisir une valeur auprès de l'utilisateur, de vérifier qu'elle est bien comprise entre un `Min` et un `Max`, puis de la renvoyer quand elle respecte les bornes données.

```
/** lit une valeur en entier comprise entre min et max
 *  @param texte de type chaine ed caracteres
 *  @param min de type entier
 *  @param max de type entier
 *  @return un entier compris entre min et max
 **/

int litValeurBornee(string texte, int min, int max) {
   int res;
   cout << texte << endl;
   do {
        cout << "entrez un entier compris entre " << min << " et " << max << ": " ;
        cin >> res ;
        if (res < min) {
            cout << "recommencez  en donnant une valeur plus elevee" << endl;
        } else if (res>max)  {
             cout << "recommencez  en donnant une valeur plus basse" << endl; }
        }
     while ( ( res < min) or (res > max) );
   return res;
 }

/** crée un guerrier
 *  @return un guerrier de type Guerrier
 **/
Guerrier creeGuerrier() {
    Guerrier g;
    cout << "donner le nom de votre personnage"<< endl;
    cin >> g.nom ;
    g.ptVie = litValeurBornee("nbre de point de vie", 2, 10);
    g.bouclier = 10 - g.ptVie;
    return g;
}
```

3. Spécifier et réaliser la fonction **forceCoup** qui permet de renvoyer la force du coup porté par un guerrier, calculée comme égale à son nombre de points de vie - 1.

```
1  /** retourne la force d'un guerrier
2   *  @param g de type Guerrier
3   *  @return un entier
4   **/
5  int forceCoup(Guerrier g) {
6      return g.ptVie - 1;
7  }
```

---

4. Spécifier et réaliser une fonction **reçoitCoup** qui prend en paramètre un guerrier et la force d'un coup et qui renvoie le guerrier après avoir reçu le coup.

```
1  /** retourne un guerrier modifie
2   *  @param g de type Guerrier
3   *  @param f de type entier
4   *  @return un Guerrier
5   **/
6  Guerrier recoitCoup(Guerrier g, int f) {
7      int temp;
8      if (f > g.bouclier) {
9          temp = g.ptVie + g.bouclier - f;
10         if (temp >= 0) g.ptVie = temp;
11         else g.ptVie = 0;
12     }
13     return g;
14 }
```

---

5. On suppose déjà réalisée la fonction

   `Guerrier initialiseGuerrierAutomatique(Guerrier G1);`

   qui permet au concepteur de créer le guerrier G_2 qui combattra le guerrier G_1 créé par l'utilisateur du jeu. Réaliser le programme principal qui crée les deux guerriers, les affiche, enchaîne les coups qu'ils se portent jusqu'à ce que l'un des deux ne soit plus en état de se battre puis affiche le nom du gagnant et ses points de vie.

```
int main() {
  Guerrier g1, g2;
  g1 = creeGuerrier();
  g2 = initialiseGuerrierAutomatique(g1);
  while (g1.ptVie > 1 and g2.ptVie > 1) {
    afficheGuerrier(g1);
    afficheGuerrier(g2);
    g2 = recoitCoup(g2, ForceCoup(g1));
    if (g2.ptVie > 1) g1 = recoitCoup(g1, ForceCoup(g2));
  }
  cout << "Gagnant : ";
  if (g1.ptVie > g2.ptVie) afficheGuerrier(g1);
  else                     afficheGuerrier(g2);
}
```

---

2 Exercices sur machine

▸ **Exercice 2. (Environnement de travail)**

Comme au premier semestre, nous utilisons un environnement de travail sur MyDocker accessible depuis chez vous. De même, vous récupérerez chaque semaine les fichiers du TP grâce à un script. Si l'accès à gitlab est rétabli au cours du semestre, vous pourrez aussi déposer votre travail.

À faire avant de commencer : Suivez les instructions "Environnement de travail en TP" qui sont données sur Ecampus pour configurer votre espace de travail et récupérer les fichiers de la première semaine. A la fin, vous devez avoir un répertoire 'ProgMod' qui contient un répertoire 'Semaine1' contenant lui même 3 fichiers ('bonjour.cpp' 'convertHMS.cpp' et 'sujet.pdf')

Compilateur Un compilateur est nécessaire pour créer un fichier exécutable à partir d'un code source. Puisque nous développons en `C++`, ce semestre nous conseillons le compilateur `clang++`. Il est accessible en ligne de commande (terminal). Ce compilateur est installé sur l'environnement Jupyter Lab (ainsi que sur les machines de TP).

La syntaxe à utiliser pour compiler un fichier est la suivante

```
clang++ -std=c++11 -Wall fichier.cpp -o nom_programme
```

Voici quelques explications :

- `-std=c++11` : Pour obliger le compilateur à suivre la norme `ISO C++11`
- `-Wall` : Pour afficher les messages d'avertissement (Warnings) durant la compilation
- `fichier.cpp` : Le code source c++ à compiler
- `-o nom_programme` : Pour spécifier un nom à l'exécutable

Note : en cas de problème, on peut remplacer le compilateur `clang++` par `g++` mais il a tendance à avoir de moins bon messages d'erreur.

Si vous avez des messages d'erreur à la compilation, il faut lire ces messages d'erreur en commençant par le premier pour voir comment corriger votre code ou votre commande de compilation. Vous pouvez consulter et compléter le document partagé qui contient des explications des messages.

À faire pour vérifier que tout est ok :

1. Vérifier que tout va bien en compilant le programme `bonjour.cpp` fourni dans l'archive grâce à la commande :
   ```
   clang++ -std=c++11 -Wall bonjour.cpp -o bonjour
   ```
2. Exécuter le programme précédemment compilé (commande `./bonjour`).

▸ **Exercice 3. (Conversion Heures/Minutes/Secondes d'une durée)**

On représente une durée heure, minute, seconde sous la forme d'un type `C++` structuré `Duree` où les heures et les minutes seront des entiers et les secondes un réel pour avoir des fractions de seconde. Par exemple : 5h 20min et $10,25$s sera représenté par la structure

```
{ .heures = 5; .minutes = 20; .secondes = 10.25 }
```

1. Ouvrir avec un éditeur de texte le fichier `convertHMS.cpp` fourni.

2. Compléter la définition du type `Duree`. Attention ! les champs `heures`, `minutes` et `secondes` sont au pluriels. Ne pas oublier le «s» à la fin.
3. Écrire une fonction `egalHMS` qui prend deux durées et qui retourne si elles sont égales.
4. Tester cette fonction en complétant les tests de la fonction `testEgalHMS`.

   On utilisera cette fonction dans la suite pour les tests.
5. Écrire une fonction

   ```
   float convertHMS2S(Duree hms)
   ```

   qui pour une durée exprimée sous la forme « heures, minutes, secondes » retourne la durée correspondante exprimée en secondes. Ajouter 3 autres tests pour cette fonction. Compiler et exécuter votre programme pour le vérifier ; ceci est à faire à chaque question dans tous les TPs, on ne le rappellera plus dans les énoncés, à vous de bien penser à le faire systématiquement.
6. Écrire une fonction

   ```
   Duree convertS2HMS(float d)
   ```

   qui pour une durée exprimée en secondes retourne la durée correspondante exprimée sous la forme « heures, minutes, secondes ». Ajouter 3 autres tests pour cette fonction.
7. Écrire une fonction

   ```
   void testHMS(Duree hms)
   ```

   qui vérifie qu'une durée est bien une durée sous la forme « heures, minutes, secondes » correcte. Une structure `hms` contient une heure correcte si $0 \leq$ `hms.heures` < 24, $0 \leq$ `hms.minutes` < 60 et $0 \leq$ `hms.secondes` < 60. On écrira les vérifications en utilisant la commande `CHECK` fournie. Note : `CHECK(condition)` permet de tester n'importe quelle condition. C'est juste un raccourci pour `if (not (condition)) cout << "Test failed ...";`
8. On va tester à grande échelle nos deux fonctions de conversion. Pour ceci, écrire un programme qui pour toutes les durées de 0 à 80000 secondes, en avançant de $0,25$s à chaque fois,
   — convertit la durée sous la forme « heures, minutes, secondes » ;
   — vérifie grâce à la fonction `testHMS` que le résultat obtenu est bien correct ;
   — re-convertit la durée en seconde et vérifie (avec `CHECK`) que l'on retrouve bien la durée initiale.

✂ ----------

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  /** Infrastructure minimale de test **/
6  #define CHECK(test) if (!(test)) cout << "Test failed in file " << __FILE__ \
7                                        << " line " << __LINE__ << ": " #test << endl
8
9  struct Duree {
10   int heures;
11   int minutes;
12   float secondes;
13 };
14
15 /** égalité des durées
16  *  @param d1, d2 deux Duree
```

```
17   *  @return true/false selon si d1 et égal à d2
18   **/
19  bool egalHMS(Duree d1, Duree d2) {
20    return (d1.heures == d2.heures &&
21            d1.minutes == d2.minutes &&
22            d1.secondes == d2.secondes);
23  }
24
25  void testEgalHMS() {
26      CHECK( egalHMS({10, 10, 2}, {10, 10, 2}));
27      CHECK( not egalHMS({10, 10, 2}, {10, 11, 2}));
28  }
29
30  /** Converti une durée en Heures, Minutes, Secondes en secondes
31   *  @param hms de type Duree encodant la durée h:m:s
32   *  @return la conversion de hms en secondes
33   **/
34  float convertHMS2S(Duree hms) {
35    return hms.secondes + 60*hms.minutes + 3600*hms.heures;
36  }
37
38  void testConvertHMS2S() {
39    CHECK( convertHMS2S({0, 1, 15}) == 75 );
40    CHECK( convertHMS2S({10, 10, 2}) == 36602 );
41  }
42
43  /** Converti une durée en secondes en Heures, Minutes, Secondes
44   *  @param d un nombre de secondes
45   *  @return la conversion de d en Heures, Minutes, Secondes
46   *          dans une structure de type Duree
47   **/
48  Duree convertS2HMS(float d) {
49    Duree hms;
50    hms.heures = int(d/3600);
51    hms.minutes = int((d - hms.heures*3600)/60);
52    hms.secondes = d - hms.heures*3600 - hms.minutes*60;
53    return hms;
54  }
55
56  void testConvertS2HMS() {
57    Duree d;
58    d = convertS2HMS(36602);
59    CHECK(egalHMS(d, {10, 10, 2}));
60    d = convertS2HMS(75);
61    CHECK(egalHMS(d, {0, 1, 15}));
62  }
63
64
65  void testHMS(Duree hms) {
66    CHECK( hms.secondes >= 0 );
67    CHECK( hms.secondes < 60 );
68    CHECK( hms.minutes >= 0 );
69    CHECK( hms.minutes < 60 );
70    CHECK( hms.heures >= 0 );
71    CHECK( hms.heures < 24 );
72  }
73
74  int main() {
75    Duree hms;
76    testConvertHMS2S();
```

```
77    testConvertS2HMS();
78
79    for (float d=0; d < 8000; d = d + 0.25) {
80      hms = convertS2HMS(d);
81      testHMS(hms);
82      CHECK( d == convertHMS2S(hms) );
83    }
84  }
```

▸ **Exercice 4.(Calcul avec les dates).**

Il faut d'abord télécharger le fichier d'exemple du cours sur les dates qui se trouve dans la section Exemples sur le site du module (page du chargé de cours). Ce fichier sera nommé `date.cpp` et devra être déposé dans votre répertoire `ProgMod/Semaine1`. Pour que ce fichier soit pris en compte par le système d'archivage, il faut ensuite taper dans le répertoire du fichier la commande

```
git add date.cpp
```

Cette commande est à noter et à retenir. Elle vous sera en effet utile dès que vous voulez ajouter un fichier par rapport à ceux qui vous ont été fournis.

On vous demande d'ajouter et de tester les fonctions suivantes :

1. `bool avantDate(Date d1, Date d2)` qui dit si la date `d1` est avant la date `d2` ;
2. `Date ajouteDate(Date d, int n)` qui calcule la date qui suit `d` de `n` jours ;
3. `int differenceDate(Date d1, Date d2)` qui calcule le nombre de jours écoulés entre les dates `d1` et `d2` ;
4. Sachant que le 1er janvier 2000 était un samedi, ecrire une fonction `int jourDate(Date d)` qui retourne le jour de la semaine d'une date (renvoie 0 pour lundi, 1 pour mardi, ...).

```
1   #include<iostream>
2   #include<iomanip>
3   using namespace std;
4
5   /** Infrastructure minimale de test **/
6   #define CHECK(test) if (!(test)) cout << "Test failed in file " << __FILE__ \
7                                         << " line " << __LINE__ << ": " #test << endl
8
9   struct Date {
10    int jour, mois, annee;
11  };
12
13  /** Affiche une Date sous le format jj/mm/aaaa
14  * @param[in] d : la date
15  **/
16  void afficheDate(Date d) {
17    cout << setfill('0') << setw(2) << d.jour << "/"
18         << setfill('0') << setw(2) << d.mois << "/" << d.annee;
19  }
20
21  /** Teste si une année est bissextile
```

```
22  * @param[in] annee : un entier
23  * @return le booléen correspondant au test
24  **/
25  bool estBissextile(int annee) {
26    return (annee % 4 == 0 and annee % 100 != 0) or (annee % 400 == 0);
27  }
28
29  /** Le nombre de jours qu'il y a dans un mois
30  * @param[in] annee : un entier
31  * @param[in] mois : un entier entre 1 et 12
32  * @return le nombre de jours du mois
33  **/
34  int nbJourMois(int mois, int annee) {
35    switch (mois) {
36    case 1 : case 3 : case 5 :
37    case 7 : case 8 : case 10 : case 12 : return 31;
38    case 4 : case 6 : case 9 : case 11  : return 30;
39    case 2 :
40      if (estBissextile(annee)) return 29;
41      else                      return 28;
42    default: return -1;
43    }
44  }
45
46  /** Teste si une Date est correcte
47  * @param[in] d : une Date
48  * @return le booléen correspondant au test
49  **/
50  bool estCorrecteDate(Date d) {
51    if (d.mois <= 0 or d.mois > 12) return false;
52    if (d.jour <= 0) return false;
53    return d.jour <= nbJourMois(d.mois, d.annee);
54  }
55
56  /** Demande une Date à l'utilisateur
57  *   Si la date n'est pas correcte, une nouvelle date est demandée
58  * @return une Date
59  **/
60  Date lireDate() {
61    Date res;
62    bool erreur;
63    do {
64      cout << "jour ? "; cin >> res.jour;
65      cout << "mois ? "; cin >> res.mois;
66      cout << "annee ? "; cin >> res.annee;
67      erreur = not estCorrecteDate(res);
68      if (erreur) cout << "Date incorrecte !" << endl;
69    } while (erreur);
70    return res;
71  }
72
73  /** Le bug de l'an 2000
74  * @return le premier janvier 2000
75  **/
76  Date bugday() {
77    return {1, 1, 2000};
78  }
79
80  /** Le bug de l'an 2000
81  * @param[out] d reçoit le bug day
```

```
82  **/
83  void bugdayRef(Date &d) {
84    d = {1, 1, 2000};
85  }
86
87  /** Lendemain d'une Date
88  *  @param[in] d : une date
89  **/
90  Date lendemain(Date d) {
91    d.jour++;
92    if (d.jour > nbJourMois(d.mois, d.annee)) {
93      d.jour = 1;
94      d.mois++;
95      if (d.mois == 13) {
96        d.mois = 1;
97        d.annee++;
98      }
99    }
100   return d;
101 }
102
103 bool avantDate(Date d1, Date d2) {
104   if (d1.annee < d2.annee) return true;
105   if (d1.annee == d2.annee) {
106     if (d1.mois < d2.mois) return true;
107     if (d1.mois == d2.mois) {
108       if (d1.jour < d2.jour) return true;
109     }
110   }
111   return false;
112 }
113
114 void testAvantDate() {
115   CHECK(avantDate({1, 1, 2000}, {1, 1, 2001}) == true);
116   CHECK(avantDate({1, 1, 2001}, {1, 1, 2000}) == false);
117 }
118
119 /**
120 Solution facile, mais pas optimale
121
122 Date ajouteDate(Date d, int n){
123     Date res = d;
124     for (int i = 0; i<n; i++){
125         res = lendemain(res);
126     }
127     return res;
128 }
129 */
130
131 Date ajouteDate(Date d, int n){
132         d.jour += n;
133         while (d.jour > nbJourMois(d.mois, d.annee)){
134                         d.jour -= nbJourMois(d.mois, d.annee);
135                         d.mois ++;
136                         if (d.mois > 12){
137                                 d.annee ++;
138                                 d.mois = 1;
139                         }
140         }
141         return d;
```

```
142 }
143
144 bool operator==(const Date& d1, const Date& d2)
145 {
146     return (d1.jour == d2.jour && d1.mois==d2.mois && d1.annee==d2.annee);
147 }
148
149 void testAjouteDate() {
150     Date d1 = {1,1,2000};
151     CHECK(ajouteDate({1, 1, 2000}, 0) == d1);
152
153     Date d2 = {1,2,2000};
154     CHECK(ajouteDate({1, 1, 2000}, 31) == d2);
155
156     Date d3 = {1,3,2000};
157     CHECK(ajouteDate({1, 1, 2000}, 60) == d3);
158
159     Date d4 = {31,12,2000};
160     CHECK(ajouteDate({1, 1, 2000}, 365) == d4);
161
162     Date d5 = {1,1,2001};
163     CHECK(ajouteDate({1, 1, 2000}, 366) == d5);
164
165     for (int i=0; i<10000; i++){
166         CHECK(estCorrecteDate(ajouteDate({1, 1, 2000}, i)));
167     }
168 }
169
170 int differenceDate(Date d1, Date d2){
171     int nbDeJourDeDifference = 0;
172     while (avantDate(d1, d2)) {
173       d1 = lendemain(d1);
174       nbDeJourDeDifference++;
175     }
176     while (avantDate(d2, d1)) {
177       d2 = lendemain(d2);
178       nbDeJourDeDifference++;
179     }
180     return nbDeJourDeDifference;
181 }
182
183 void testDifferenceDate() {
184   CHECK( differenceDate({1, 1, 2000}, {1, 1, 2001}) == 366 );
185   CHECK( differenceDate({1, 1, 2000}, {5, 1, 2000}) == 4 );
186 }
187
188 int jourDate(Date d){
189   int diffJour = differenceDate(d, {1, 1, 2000});
190   int resteDelaDivision = diffJour % 7;
191   switch(resteDelaDivision){
192     case 0 : return 5; //"samedi";
193     case 1: return 6; //"dimanche";
194     case 2: return 0; //"lundi";
195     case 3: return 1; //"Mardi";
196     case 4: return 2; //"Mercredi";
197     case 5: return 3; //"Jeudi";
198     case 6: return 4; //"Vendredi";
199     default: return -1;
200   }
201 }
```

```
202
203 void testJourDate() {
204   CHECK( jourDate({1, 1, 2000}) == 5 );
205   CHECK( jourDate({8, 1, 2000}) == 5 );
206   CHECK( jourDate({24, 1, 2021}) == 6 );
207 }
208
209
210 int main() {
211         testAvantDate();
212         testAjouteDate();
213         testDifferenceDate();
214     testJourDate();
215 }
```

---

3 Exercice complémentaire (pas forcément sur les structures)

▸ Exercice 5.Calendrier

Le but de cet exercice est d'afficher un calendrier en affichant mois par mois les jours de la semaine. Voici par exemple l'affichage du mois de février 2013 qui commence un vendredi :

```
       Fevrier
lu ma me je ve sa di
 .  .  .  .  1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28
```

Pour ceci, on déclare le nom des mois et leur longueur (dans une année usuelle) en C++ de la manière suivante :

```
vector<string> nom_mois = {
    "Janvier", "Fevrier", "Mars", "Avril", "Mai", "Juin", "Juillet",
    "Aout", "Septembre", "Octobre", "Novembre", "Decembre"
  };

vector<int> long_mois = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Attention : le mois de janvier porte le numéro 0.

1. Dans un premier temps, on suppose que le premier janvier est un lundi. On affichera les jours du mois en revenant à la ligne chaque dimanche sans se soucier des alignements. On sautera une ligne à la fin de chaque mois. On pourra utiliser trois compteurs :
   — `mois` pour stocker le numéro du mois en cours d'affichage ;
   — `j_mois` pour stocker le numéro dans le mois du jour en cours d'affichage ;
   — `j_sem` pour stocker le numéro dans la semaine du jour en cours d'affichage.
2. Écrire un programme qui demande à l'utilisateur le jour du premier janvier sous la forme (0 = lundi, 1 = mardi, 2 = mercredi, etc.) et qui affiche le calendrier de l'année.

✂ ----------

Important : Si le compilateur affiche un message d'erreur sur l'initialisation des deux variables `nom_mois` et `long_mois`, le problème viens très probablement du fait que l'étudiant n'a pas configuré le mode C++11 comme indiqué dans l'introduction du TP.

```
1  #include <iostream>
2  #include <iomanip>
3  #include <vector>
4  using namespace std;
5
6  vector<string> nom_mois =
7    {
8      "Janvier", "Fevrier", "Mars", "Avril", "Mai", "Juin", "Juillet",
9      "Aout", "Septembre", "Octobre", "Novembre", "Decembre"
10   };
11 vector<int> long_mois = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
12
13 int main(void)
14 {
15   int i, j_sem, j_mois, mois;
16   cout << "Quel est le jour du 1er janvier (0=lundi, 1=mardi, ...) ?" << endl;
17   cin >> j_sem;
18   for (mois=0; mois<12; mois++)
19     {
20       cout << "        " << nom_mois[mois] << endl;
21       cout << " lu ma me je ve sa di" << endl;
22       for (i=0; i<j_sem % 7; i++) cout << "  .";
23       for (j_mois=1; j_mois<=long_mois[mois]; j_mois++)
24         {
25           cout << setw(3) << j_mois;
26           j_sem++;
27           if (j_sem % 7 == 0) cout << endl;
28         }
29       cout << endl << endl;
30     }
31 }
```

---------- ✂