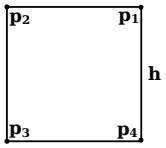
PROG2 : Programmation avancée et C++

Examen du 11 mai 2022 (2 h)

Tous documents interdits à l'exception d'un A4 recto verso manuscrit de votre main. Calculatrices, téléphones, ordinateurs et montres connectées interdites. La correction tiendra compte de la qualité de l'argumentaire et de la présentation. Un code illisible et/ou incompréhensible est un code faux.

★ **Exercice 1 :** Classes `Point` et `Carre` (3pts).

Écrivez une classe `Point` permettant de représenter un point **constant** du plan, sous forme de deux doubles x et y. Écrivez une classe `Carre` permettant de représenter un carré **constant** dans le plan. La classe `Carre` sera dotée de deux constructeurs. L'un prendra un point (correspondant à p_1 dans la figure ci-contre) ainsi que longueur du coté du carré, tandis que l'autre prendra deux points opposés p_1 et p_3.



Vous doterez votre classe `Carre` d'une méthode `aire()` ainsi que de la méthode `translate()` permettant de calculer le carré résultant de l'application de la translation donnée en paramètre sous forme de deux double x et y. Vous doterez aussi votre classe de l'opérateur permettant de trouver un point représentant l'angle p_2 du carré c avec la notation `c[2]`, ainsi que l'opérateur permettant d'afficher un carré sous forme textuelle.

★ **Exercice 2 :** Surcharge et redéfinition de méthodes (5pts).

On compile le programme ci-dessous avec `g++` avec les paramètres `-Wall -Wextra`. Attention, ce n'est pas exactement le code étudié en cours. Les lignes 1 à 15 compilent et s'exécutent sans erreur.

```
1  #include <iostream>
2
3  struct A {
4    virtual void f1()  { std::cout << "A::f1\n"; }
5  };
6  struct B : public A {
7    void f1() override { std::cout << "B::f1\n"; }
8  };
9
10 struct UU {
11    virtual A foo()  { return A();     }
12    virtual A* bar() { return new A(); }
13    virtual void qux(A a)   { a.f1();  }
14    virtual void xyz(A* a) { a->f1(); }
15 };
16 struct DD : public UU {
17    B foo()  { return B(); }
18    B* bar() { return new B(); }
19    void qux(B b)  { b.f1();  }
20    void xyz(B* b) { b->f1(); }
21 };
```

```
22 int main() {
23    UU u;
24    DD d;
25    A a;  B b;
26
27    u.foo().f1();
28    d.foo().f1();
29    u.bar()->f1();
30    d.bar()->f1();
31
32    u.qux(a);
33    u.qux(b);
34    u.xyz(&a);
35    u.xyz(&b);
36
37    d.qux(a);
38    d.qux(b);
39    d.xyz(&a);
40    d.xyz(&b);
41    return 0;
42 }
```

▷ **Question 1 :** Pour chacune des lignes 17 à 20, indiquez (en justifiant très brièvement) si elle compile ou non (rappel : le mot-clé *override* est optionnel en C++).

Dans la suite de l'exercice, on supposera que les lignes ne compilant pas sont commentées.

▷ **Question 2 :** Pour chacune des lignes 27 à 40, indiquez (en justifiant très brièvement vos réponses) :

1. si elle compile ou non, en explicitant les éventuelles erreurs et warning produits ;
2. le message affiché à l'exécution (si la compilation est possible).

▷ **Question 3 :** Expliquez la différence entre *surcharge de méthode* et *redéfinition de méthode* en utilisant des exemples dans le code présenté.

▷ **Question 4 :** Expliquez le fonctionnement des différents mécanismes de sélection du code à exécuter (*code binding*) en C++. La réponse attendue est de l'ordre de la dizaine de lignes d'explications, éventuellement accompagnée de schémas explicatifs.

★ **Exercice 3 :** Design de code orienté objet (6pts).

Vous trouverez sur l'autre feuille le code du jeu de UNO écrit en C++ [1], mais en suivant la philosophie du C. Ce code, relativement bien documenté, tire partie des conteneurs et algorithmes standards du C++, mais aucune classe d'objet n'est définie, il n'y a pratiquement pas de fonction pour découper et abstraire le code. L'objectif de cet exercice est de le refactorer pour le rendre plus lisible.

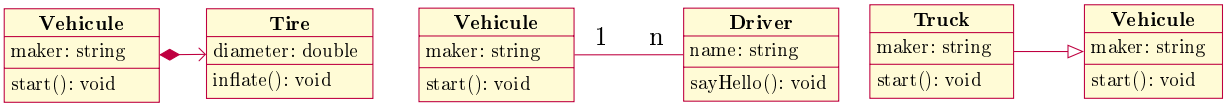
Exemple de partie

```
Joueur 1, à vous.
Carte sur la défausse: 7 bleu;
Votre jeu: a:3 rouge | b:4 rouge | c:7 vert | d:9 vert | e:change de sens vert | f:2 jaune | g:Joker |
Quelle carte jouer ('z' pour tirer une nouvelle carte)? g


Joueur 2, à vous.
Carte sur la défausse: Joker;
Votre jeu: a:+2 rouge | b:passe ton tour vert | c:3 jaune | d:2 jaune | e:+2 jaune | f:0 bleu | g:7 bleu |
Quelle carte jouer ('z' pour tirer une nouvelle carte)? a


2 cartes de pénalité pour le joueur 1!!
Carte sur la défausse: +2 rouge;
Votre jeu: a:3 rouge | b:4 rouge | c:9 rouge | d:7 vert | e:9 vert | f:change de sens vert | g:2 jaune | h:5 jaune |
Quelle carte jouer ('z' pour tirer une nouvelle carte)? ^C
```

▷ **Question 1 :** On souhaite modifier ce code pour que la main du joueur soit triée lorsque vient son tour. Indiquer la ou les ligne(s) à ajouter, ainsi que l'endroit du code où l'ajout doit avoir lieu.

▷ **Question 2 :** Proposez une fonction pour séparer et abstraire proprement les lignes 57, 58 et 59. Proposez un nom, donnez son implémentation et indiquez comment la fonction `main()` doit être modifiée pour utiliser cette nouvelle fonction.

On rappelle les notations UML pour les relations classiques entre classes.



Composition *Association* *Héritage*

▷ **Question 3 :** Proposez un découpage en classes de ce problème. Vous donnerez les responsabilités de chaque classe (au sens CRC) éventuellement avec les lignes du code existant correspondant à chaque responsabilité, avant de dessiner le diagramme UML liant les classes. Justifiez ce qui doit l'être.

▷ **Question 4 :** Écrivez le code de la boucle principale en supposant que toutes les autres méthodes de vos classes sont écrites. Si vous avez correctement découpé le projet, la boucle principale ne devrait pas dépasser une quinzaine de lignes.

★ **Exercice 4 :** Comprendre et modifier du code C++ (reprise du TP5 – 6pts)

Observez le code intitulé *Code mystère* en dernière page. Les entêtes et la méthode `sol_to_str()` sont omises, mais le code complet compile et s'exécute sans problème. Attention, ce n'est probablement pas exactement le code que vous avez utilisé dans le TP 5.

▷ **Question 1 :** Expliquez le principe général de ce code. L'objectif n'est pas d'expliquer l'algorithme permettant de résoudre le problème, mais plutôt d'expliquer les constructions C++ utilisées : précisez le statut de `tplate()`, son fonctionnement et son utilisation dans `main()`.

▷ **Question 2 :** Expliquez les lignes 38 et 39 de ce code.

▷ **Question 3 :** Expliquez la ligne 45 de ce code.

▷ **Question 4 :** Replacez les lignes 22 à 31 par autre chose pour éviter la fuite mémoire liée au `new` de la ligne 24, dont la mémoire n'est jamais libérée dans l'état actuel.

▷ **Question 5 :** Écrivez la fonction `sol_to_str()` dont le prototype est donné ci-dessous. Pour obtenir tous les points de la question, vous devez l'écrire en utilisant une écriture aussi fonctionnelle que possible. Pour cela, vous aurez probablement besoin de `ranges::zip_view`, `ranges::iota_view(0)` et `ranges::views::filter()`, définis dans la bibliothèque range-v3. Pourquoi utiliser cette bibliothèque ?

```
1 std::string sol_to_str(std::vector<bool>& sol, int value);
```

1. Les règles sont légèrement simplifiées par rapport au vrai jeu de UNO, mais l'esprit demeure.

Jeu de UNO en pseudo-C

```
std::string to_str(int card) { // Représentation textuelle d'une carte identifiée par son num.
  std::string couleur[] {"rouge", "vert", "jaune", "bleu"};
  if (card <= 103) {
    int v = card % 26;
    if (v > 12)
      v -= 13; // Toutes les cartes sont en double
    if (v < 10)
      return std::to_string(v % 10) + " " + couleur[card / 26];
    if (v == 10)
      return "+2 " + couleur[card / 26];
    if (v == 11)
      return "change de sens " + couleur[card / 26];
    if (v == 12)
      return "passe ton tour " + couleur[card / 26];
  } else if (card <= 107)
    return "Joker";
  return "Super joker +4";
}
int main() {
  srand(time(nullptr)); // Initialize the pseudo-random
  std::vector<int> deck, defausse;
  for (int i = 0; i < 112; i++)
    deck.push_back(i);
  std::random_shuffle(deck.begin(), deck.end()); // permutation aléatoire
  std::vector<int> hand, other;  // jeu de chaque joueur (hand: joueur actuel; other: l'autre)
  for (int i = 0; i < 7; i++) {
    hand.push_back(deck.back());     deck.pop_back();
    other.push_back(deck.back());    deck.pop_back();
  }
  defausse.push_back(deck.back());   deck.pop_back(); // Place une carte dans la défausse
  int player = 1;
  while (true) { // La partie continue
    int idx;
    int penalite = (defausse.back() % 26 == 10 || defausse.back() % 26 == 23) ? 2
                 : ((defausse.back() > 107) ? 4 : 0);
    if (penalite > 0) {
      std::cout << "\n\n" << penalite << " cartes de pénalité pour le joueur "<<player<<"!!\n";
      for (int i = 0; i < penalite; i++) {
        hand.push_back(deck.back());
        deck.pop_back();
    } } else std::cout << "\n\nJoueur " << player << ", à vous.\n";
    while (true) { // Attend une carte valide
      std::cout << "Carte sur la défausse: " << to_str(defausse.back()) << ";\nVotre jeu: ";
      for (unsigned i = 0; i < hand.size(); i++)
        std::cout << static_cast<char>('a' + i) << ":" << to_str(hand[i]) << " | ";
      std::cout << "\nQuelle carte jouer ('z' pour tirer une nouvelle carte)? ";
      char choix;
      std::cin >> choix;
      if (choix == 'z' || (choix >= 'a' && choix - 'a' < hand.size())) {
        idx = (choix == 'z') ? 0 : choix - 'a' + 1;
        if (idx == 0 // le joueur tire une carte
            || defausse.back() > 103 || hand[idx - 1] > 103 // l'une est joker
            || defausse.back() / 26 == hand[idx - 1] / 26 // même couleur
            || (defausse.back() % 26)%13 ==  (hand[idx - 1] % 26)%13) // même valeur
          break; // choix validé
    } }
    if (idx == 0) {
      hand.push_back(deck.back());      deck.pop_back();
    } else {
      defausse.push_back(hand[idx - 1]);
      hand.erase(hand.begin() + idx - 1); // Retire hand[idx] du vecteur hand
    }
    if (hand.empty()) {
      std::cout << "LE JOUEUR " << player << " A GAGNÉ!!\n";
      exit(0);
    }
    if (deck.empty()) {
      std::swap(deck, defausse); // inverse les deux piles
      std::random_shuffle(deck.begin(), deck.end()); // mélange
      defausse.push_back(deck.back());       deck.pop_back();
    }
    std::vector<int> pt{11, 12, 24, 25}; // Cartes qui passent son tour, dans chaque couleur
    if (std::none_of(pt.begin(), pt.end(),
                     [&defausse](auto i) { return defausse.back() % 26 == i; })) {
      player = (player == 1 ? 2 : 1);
      std::swap(hand, other);
} } }
```

Code mystère

```
std::function<X(Y, Z)> tplate(std::function<X(Y, Z)> a){
  auto* b = new std::map<std::pair<Y, Z>, X>();
  return [a, b](Y y, Z z) -> X {
    auto k = std::make_pair(y, z);
    if (b->find(k) == b->end())
      (*b)[k] = a(y, z);
    return (*b)[k];
  };
}

int main()
{
  std::function<std::pair<int, std::vector<bool>>(std::vector<int>, int)> fn =
    tplate<std::pair<int, std::vector<bool>>, std::vector<int>, int>
      ([&fn](std::vector<int> pp, int nn) {

        if (nn == 0)
          return std::make_pair(0, std::vector<bool>(pp.size(), false));

        int max_value = INT_MIN;
        std::vector<bool> best_solution;
        for (int i = 1; i <= nn; i++) {
          auto [cost, solution] = fn(pp, nn - i);

          if (pp[i - 1] + cost > max_value) {
            max_value     = pp[i - 1] + cost;
            best_solution = solution;
            best_solution[i] = true;
          }
        }

        return std::make_pair(max_value, best_solution);
      });

  std::vector<int> p {2, 5, 9, 9, 14, 17, 17, 25, 20, 22, 23, 2, 4, 6};
  auto [cost, sol] = fn(p, p.size());

  std::cout << "Best solution for n=" << p.size() << ": " << sol_to_str(sol, cost);
  return 0;
}
```