université PARIS DIDEROT PARIS 7

EXAMEN — LOAV — C++ — 07/01/2019 08^{30}–10^{30}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé.
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Éteignez impérativement vos mobiles.

1 Exercice

Soit le code suivant :

```
1  void tri_insertion(int tableau[],int longueur) {
2    int i, memory, compt, marqueur;
3    for(i=1; i<longueur; i++) {
4      memory=tableau[i];
5      compt=i-1;
6      do {
7        marqueur=false;
8        if (tableau[compt]>memory) {
9            tableau[compt+1]=tableau[compt];
10           compt--;
11           marqueur=true;
12         }
13       if (compt<0) marqueur=false;
14     } while(marqueur);
15     tableau[compt+1]=memory;
16   }
17 }
```

Dans les questions qui suivent, aucune modification du code n'est autorisée à partir de la ligne 4 (y compris celle-ci).

1. Modifier le code en introduisant un type `Tableau` permettant d'éviter de manipuler séparément les éléments et leur nombre (*i.e.* créer un vrai type tableau). Réécrire les 3 premières lignes de sorte que l'algorithme ne soit pas impacté par la modification mais continuent néanmoins de «tourner» à l'identique.
2. Proposer une version générique du type tableau ainsi que de la fonction afin de pouvoir trier des tableaux de n'importe quel type primitif arithmétique
3. Cet algorithme de tri impose des contraintes sur le type des éléments à trier, lesquelles ?
4. Proposer une implémentation minimale d'un type non-standard pouvant être utilisé comme élément du tableau générique pour l'algorithme de tri.

2 Exercice

On souhaite implémenter un jeu qui fonctionne par tour, chaque tour permettant à chaque joueur présent dans le tour de jouer un coup (échecs, belote, 421, etc). On souhaite obtenir en fin de partie la liste des coups joués, mais aussi à tout instant dans le jeu de revenir en arrière, soit d'un coup, soit d'un tour (en milieu de tour on revient au début du tour courant). Dans la suite penser à bien faire apparaître de quoi enregistrer les coups et revenir en arrière...

1. Écrire un bout de code qui implémente un tour.
2. Écrire un bout de code qui implémente une partie.
3. Proposer un modèle UML des classes qui apparaissent et en particulier celles liées à l'historique.
4. Implémenter les classes de façon le plus minimal possible, mais quelles soient fonctionnelles (instanciables et utilisables).

3 Exercice

Soit l'extrait de code suivant :

```
class Perso {
private:
  int pVie;
public:
  Perso(int v) { pVie = v; };
  virtual int rapidite() = 0;
  int vie() { return pVie; };
  virtual int force() = 0;
  virtual int resistance() = 0;
  void porteCoup(Perso &adversaire) {
    cout << this << " porte coup " << force() << " a " << &adversaire << endl;
    adversaire.recoitCoup(force());
  }
  void recoitCoup(int dommages) {
    if (dommages<resistance()) return;
    pVie -= dommages-resistance();
  }
};
void combat(Perso &p1, Perso &p2) {
  srand(time(NULL));
  do {
    int v = rand()%(p1.rapidite()+p2.rapidite());
    if (v>=p1.rapidite()) p2.porteCoup(p1);
    else                  p1.porteCoup(p2);
  } while (p1.vie()>0 && p2.vie()>0);
}
int main() {
  Personnage<PasFort,PasResistant,PasRapide> p1(10);
  Personnage<PasFort,Resistant,Rapide> p2(10);
  combat(p1,p2);
  if (p1.vie()<=0) cout << "p1 a perdu" << endl;
  if (p2.vie()<=0) cout << "p2 a perdu" << endl;
}
```

1. Pourquoi les méthodes `porteCoup`, `recoitCoup`, `vie` de la classe `Perso` ne sont-elles pas virtuelles ?
2. Quel type de passage de paramètre est employé par la fonction `combat` ? Pourquoi ?
3. Donnez une implémentation des classes `Personnage`, `PasFort`, `PasRapide` et `PasResistant`. Indice, une bonne solution ne fait définir qu'une et une seule méthode pour chacune de ces classes.
4. Réécrire la classe `Perso` de sorte que soient séparés déclarations et définitions.