

Commencé le vendredi 9 décembre 2022, 09:30

État Terminé

Terminé le vendredi 9 décembre 2022, 11:59

Temps mis 2 heures 29 min

Note 20,00 sur 20,00 (100%)

Description

Marquer la question

- Pour un examen, le **plagiat** inclut le fait de demander de l'aide à quelqu'un pour répondre aux questions et le copier-coller venant de sources dont vous ne possédez pas tous les droits (i.e. TP fait en équipe, notes de cours, site Internet), sans en citer la source. Toute personne qui aide un autre étudiant pendant l'examen commet une **fraude**.
- En cas de doute sur le sens d'une question, énoncez clairement toutes suppositions que vous faites, soit en commentaires dans le code, soit dans la dernière question de cet examen (qui est un espace pour écrire vos commentaires). Nous ne répondrons pas aux questions.
- En cas de réel problème, vous pouvez demander aux surveillants.
- Vous avez droit à toutes les notes de cours, à cppreference.com (peut aussi être téléchargé de Moodle) et, pour les questions de «programmation» (où vous devez écrire un programme) à un compilateur, mais CodeRunner est suffisant pour répondre et les questions n'ont pas été particulièrement faites pour l'utilisation d'un autre compilateur. Vos programmes doivent passer les tests CodeRunner.
- Ce qui est indiqué C++20 dans les notes de cours fonctionne dans CodeRunner, mais il se peut qu'autres choses de C++20 ne soit pas supporté; C++17 devrait être entièrement supporté.
- La sauvegarde se fait automatiquement lorsque vous changez de page, elle est déclenchée quand le bouton « Suivant » dans le bas de la page est utilisé ou lorsque vous changez de page en utilisant le bloc «Navigation du test». Cette sauvegarde permet de limiter la perte d'information en cas de coupure avec Internet ou autre problème technique et de garder la session active. Après deux heures d'inactivité (c.-à-d. aucune interaction avec le serveur), vous êtes automatiquement déconnecté(e) de Moodle.
- En cas de perte de connexion à la fin de l'examen, la tentative est envoyée automatiquement.
- L'examen est sur 20 points, le barème de chaque exercice est indiqué dans le bloc «Navigation du test».

Bon travail !

Description

Marquer la question

Indiquez l'ordre de complexité de temps des algorithmes ci-dessous. On considère l'ordre moyen et/ou amorti. S'il y a plusieurs possibilités, indiquez l'ordre qui est le plus petit possible.

La fonction f a un temps en $O(1)$ et $range$ n'ajoute pas de complexité par rapport à une boucle faite à la main.

Question 1

Correct

Note de 1,00
sur 1,00

Marquer la question

```
map<int, int> v;
for (int i : range(n)) | Est O(   ✓ )
```

Votre réponse est correcte.

Question 2

Correct

Note de 1,00
sur 1,00

Marquer la question

```
deque<int> v;
for (int i : range(n))
    v.insert(v.begin() + v.size() / 2, f(i)); | Est O(   ✓ )
```

Votre réponse est correcte.

Question 3

Correct

Note de 1,00
sur 1,00[Marquer la question](#)

```
set<int> v;
for (int i : range(n)) | Est O( n · log(n) ) ✓ )
    v.insert(f(i));
```

Votre réponse est correcte.

Question 4

Correct

Note de 1,00
sur 1,00[Marquer la question](#)Choisissez dans la liste **TOUTES** les affirmations **VRAIES** au sujet de l'architecture modèle-vue-controlleur

- a. Toutes les mises à jour de la vue doivent passer par le modèle.
- b. Le contrôleur doit vérifier la validité des actions de l'utilisateur. ✓
- c. La vue doit continuellement vérifier l'état du modèle en cas de changement.

Votre réponse est correcte.

Description

[Marquer la question](#)

Soit le code suivant qui représente une situation d'étudiants qui suivent un cours en ligne. Le code ne compile bien sûr pas et des commentaires sont mis à certains endroit pour remplacer le code fonctionnel. Ce code est la base pour la question qui suit.

```
1. struct ErreurCoursEnLigne : std::logic_error {
2.     using logic_error::logic_error;
3. };
4.
5. struct ErreurIntraTropDifficile : ErreurCoursEnLigne {
6.     using ErreurCoursEnLigne::ErreurCoursEnLigne;
7. };
8.
9. struct ErreurRienCompris : ErreurIntraTropDifficile {
10.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
11. };
12.
13. struct ErreurPasAssezEtudié : ErreurIntraTropDifficile {
14.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
15. };
16.
17. struct ErreurTwitch : ErreurCoursEnLigne {
18.     using ErreurCoursEnLigne::ErreurCoursEnLigne;
19. };
20.
21. struct ErreurColocsDuChargéOntRebranchéLeRouteur : ErreurTwitch {
22.     using ErreurTwitch::ErreurTwitch;
23. };
24.
25.
26. class Étudiant {
27. public:
28.     void étudier(bool pourVrai) {
29.         if (pourVrai) {
```

```

9. struct ErreurRienCompris : ErreurIntraTropDifficile {
10.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
11. };
12.
13. struct ErreurPasAssezEtudié : ErreurIntraTropDifficile {
14.     using ErreurIntraTropDifficile::ErreurIntraTropDifficile;
15. };
16.
17. struct ErreurTwitch : ErreurCoursEnLigne {
18.     using ErreurCoursEnLigne::ErreurCoursEnLigne;
19. };
20.
21. struct ErreurColocsDuChargéOntRebranchéLeRouteur : ErreurTwitch {
22.     using ErreurTwitch::ErreurTwitch;
23. };
24.
25.
26. class Étudiant {
27. public:
28.     void étudier(bool pourVrai) {
29.         if (pourVrai) {
30.             // Refaire ses TD par soi-même et refaire les exemples faits en classe. :D
31.             prêtPourExamen_ = true;
32.         } else {
33.             // Meh, relire les notes de cours une fois la veille de l'exam. :(
34.             prêtPourExamen_ = false;
35.         }
36.     }
37.
38.     void assisterAuCours() {
39.         // Assister au stream sur Twitch.
40.         bool coursFini = false;
41.         while (not coursFini) {
42.             if (not streamTwitchActif())
43.                 throw ErreurColocsDuChargéOntRebranchéLeRouteur("Ah come on!");
44.
45.             coursFini = /* Est-ce que le chargé a fini de monologuer et/ou commence à avoir faim? */;
46.         }
47.     }
48.
49.     void faireIntra() {
50.         double note = /* Est-ce que ça c'est bien passé? */;
51.         if ( note < 10/20.0 ) {
52.             if (not prêtPourExamen_)
53.                 throw ErreurPasAssezEtudié("Oups! Les questions demandaient de comprendre... ");
54.             else
55.                 throw ErreurRienCompris("De toute façon, les examens sont juste des constructions sociales.");
56.         } else if (note <= 15/20.0) {
57.             cout << "Pas pire, pas pire." << "\n";
58.         } else if (note <= 20/20.0) {
59.             cout << "Yay!" << "\n";
60.         }
61.     }
62.
63. private:
64.     bool prêtPourExamen_;
65. };

```

Question 5

Correct

Note de 1,75
sur 1,75Marquer la
question

Soit le code suivant :

```

1. int main () {
2.     Étudiant steveFleury;
3.     try {
4.         steveFleury.assisterAuCours();
5.         steveFleury.étudier(false);
6.         steveFleury.faireIntra();
7.     } catch (ErreurPasAssezÉtudié& e) {
8.         // Catch 1
9.     } catch (ErreurRienCompris& e) {
10.        // Catch 2
11.    } catch (ErreurIntraTropDifficile& e) {
12.        // Catch 3
13.    } catch (ErreurCoursEnLigne& e) {
14.        // Catch 4
15.    }
16.
17.    // Fin du programme
18. }
```

Qu'arrive-t-il si le stream est interrompu (`streamTwitchActif()` retourne faux) et qu'on présume que l'élève obtiendrait une note de 8/20?

Veuillez choisir une réponse.

- a. Une exception est levée dans `assisterAuCours()`, aucun catch ne peut attraper le `ErreurColocsDuChargéOntRebranchéLeRouteur` et le programme plante à cause de l'exception non gérée.
- b. Une exception est levée dans `faireIntra()`, le catch 2 est exécuté, puis le programme se termine normalement.
- c. Une exception est levée dans `assisterAuCours()`, le catch 4 est exécuté et le programme se termine normalement.✓

Votre réponse est correcte.

Question 6

Correct

Note de 1,00
sur 1,00Marquer la
questionChoisissez dans la liste **TOUTES** les bonnes raisons d'attraper les exceptions par référence dans les blocs `try-catch`

Veuillez choisir au moins une réponse.

- a. Pour utiliser les move semantics.
- b. Pour éviter les copies inutiles d'objets d'exception.✓
- c. Pour pouvoir utiliser les méthodes statiques des objets d'exception.
- d. Pour garder le polymorphisme des exceptions lancées.✓

Votre réponse est correcte.

Question 7

Correct

Note de 2,00
sur 2,00Marquer la
question

Cette question devrait être faite sans utiliser un compilateur.

On désire déterminer l'ordre de construction des différents objets/sous-objets pour une déclaration donnée. On ne listera pas dans cet ordre la "construction" des types fondamentaux, qui n'ont pas de constructeur.

Soit les définitions de classes suivantes (dans des fichiers séparés et on suppose que les "include" sont faits correctement pour que ça compile):

```

class A : public B, public D {
public:
    A() {}
private:
    D att1_;
    B* att2_;
};
```

Question 7

Correct

Note de 2,00
sur 2,00[Marquer la question](#)**Cette question devrait être faite sans utiliser un compilateur.**

On désire déterminer l'ordre de construction des différents objets/sous-objets pour une déclaration donnée. On ne listera pas dans cet ordre la "construction" des types fondamentaux, qui n'ont pas de constructeur.

Soit les définitions de classes suivantes (dans des fichiers séparés et on suppose que les "include" sont faits correctement pour que ça compile):

```
class A : public B, public D {  
public:  
    A() {}  
private:  
    D att1_;  
    B* att2_;  
};  
  
class B {  
public:  
    B() {}  
};  
  
class C : public D {  
public:  
    C() { att2_ = new A(); }  
private:  
    B att1_;  
    A* att2_;  
};  
  
class D {  
public:  
    D() {}  
};
```

Soit le programme suivant:

```
int main() {  
    C x;  
}
```

On veut savoir l'ordre de construction lorsqu'on exécute ce programme. Indice: il y a 7 objets/sous-objets construits.

Indiquez uniquement les noms des types dans le bon ordre, exemple: A B C D A B C

Réponse : (régime de pénalités : 0 %)

Indiquez l'ordre de début de construction pour les classes ci-haut:

C D B A B D D

Indiquez l'ordre de début d'exécution du corps des constructeurs pour les classes ci-haut:

D B C B D D A

Votre réponse	Devrait être	
C D B A B D D	C D B A B D D	✓
D B C B D D A	D B C B D D A	✓

Réponse bien enregistrée: CDBABDD, DBCBDDA

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 2,00/2,00.

Question 8

S'identifier à un site fait avec utilisant

Question 8

Correct

Note de 1,50
sur 1,50[Marquer la question](#)**Cette question devrait être faite sans utiliser un compilateur.**

Soit le programme suivant. Dans chaque case écrivez une seule valeur ou expression de manière à ce que l'exécution du programme corresponde à l'affichage.

INDICE : vous devez utiliser un adaptateur d'itérateur qui fait l'insertion.

```
bool estInf(int i)
{
    return (i<20);
}

int main() {
    vector<int> v1 = { 100, 19, 25, 33, 17 };
    vector<int> v2;

    copy_if(v1.rbegin(), v1.rend(), back_inserter(v2), estInf);

    v1.clear();
    int i = 0;
    for (auto it = v2.rbegin() ; it != v2.rend() ; ++it)
    {
        cout << *it << " ";
        *it = ++i;
    }
}
```

Affichage :

19 ✓ 17

Question 9

Correct

Note de 2,00
sur 2,00[Marquer la question](#)Nous avons besoin d'une classe **MatierePrecieuse** qui servira comme classe de base pour plusieurs matières précieuses.

Cette classe contiendra les éléments suivants:

- Un constructeur et un destructeur par défaut.
- Un attribut de type **réel à double précision** qui représente le prix d'un gramme de cette matière.
- Les méthodes non virtuelles **getPrixParGramme** et **setPrixParGramme** qui permettent d'obtenir et de modifier l'attribut du prix, respectivement.
- Une méthode virtuelle pure **estSynthetisable** qui retourne un booléen indiquant s'il est possible de fabriquer cette matière de manière synthétique à partir de matières moins précieuses.
- Une méthode virtuelle pure **getNom** qui retourne le nom de la matière sous forme d'une chaîne de caractères (par valeur).

Donnez la définition de cette classe. Si des méthodes sont à définir, vous pouvez le faire dans ou à l'extérieur de la classe, comme vous voulez.

Réponse : (régime de pénalités : 0 %)

[Réinitialiser la réponse](#)

```
1 class MatierePrecieuse {
2     //TODO: Définition de la classe
3 public:
4     MatierePrecieuse() : prixParGramme_()
5     {}
6
7     double getPrixParGramme() const
8     {
9         return prixParGramme_;
10    }
11
12    void setPrixParGramme(double nouveauPrixParGramme)
13    {
14        prixParGramme_ = nouveauPrixParGramme;
15    }
16
17    virtual std::string getNom() const = 0;
18
19    virtual bool estSynthetisable() const = 0;
20
21    virtual ~MatierePrecieuse() = default;
22
23 private:
24     double prixParGramme_;
25};
```

	Test	Résultat attendu	Résultat obtenu	
✓	-			✓
✓	cout << is_abstract_v<MatierePrecieuse> << endl;	1	1	✓
✓	// L'objet a la bonne taille	1	1	✓
✓	unique_ptr<MatierePrecieuse> precieux; cout << permet_unique_ptr_MatierePrecieuse ;	1	1	✓
✓	// test non montré			✓
✓	// test non montré			✓
✓	// Réussi à instancier une classe dérivée et cout << has_virtual_getNom_v<MatierePrecieuse>; cout << has_virtual_estSynthetisable_v<MatierePrecieuse>;	11	11	✓
✓	const MatierePrecieuse* memeMatiere = uneMatiere; uneMatiere->setPrixParGramme(1.25); cout << memeMatiere->getPrixParGramme() << endl; uneMatiere->setPrixParGramme(3.5); cout << memeMatiere->getPrixParGramme() << endl;	1.25 3.5	1.25 3.5	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 2,00/2,00.

Question 10

Correct

Note de 1,50
sur 1,50

Marquer la
question

Nous voulons utiliser la classe précédente afin de définir une nouvelle classe `Or24k` qui héritera de `MatierePrecieuse`.

Donnez la définition et l'implémentation de la classe `Or24k` tel que:

- Le nom de la matière est "Or".
- Il n'est pas possible de la synthétiser.
- `Or24k` n'est pas une classe abstraite.

L'implémentation des méthodes peut être à l'intérieur ou à l'extérieur de la classe.

NOTE: Cette question n'utilise pas votre version de la classe précédente et peut donc être répondue sans avoir réussi l'autre question.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```

1 class Or24k : public MatierePrecieuse
2 {
3     // TODO: Définition de la classe
4 public:
5     Or24k() : MatierePrecieuse()
6     {}
7
8     bool estSynthetisable() const override
9     {
10         return false;
11     }
12
13     std::string getNom() const override
14     {
15         return "Or";
16     }
17
18 };
19 // TODO: Implémentation des méthodes (peut aussi être dans la classe)

```

	Test	Résultat attendu	Résultat obtenu	
✓	-			✓
✓	Or24k orPur; cout << is_base_of_v<MatierePrecieuse, Or24k>;	true	true	✓
✓	Or24k* d = nullptr; MatierePrecieuse* c = d; cout << "ok";	ok	ok	✓
✓	Or24k orPur; MatierePrecieuse& matiere = orPur; cout << matiere.getNom() << "\n";	Or	Or	✓
✓	Or24k orPur; MatierePrecieuse& matiere = orPur; cout << matiere.estSynthetisable() << "\n";	false	false	✓
✓	// test non montré	true	true	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,50/1,50.

Description

Marquer la question

Les questions de cette page utilisent la classe **MatierePrecieuse** de la page précédente; vous pouvez l'ouvrir dans un autre onglet de votre navigateur pour y accéder facilement.

On conserve l'inventaire des quantités de matières précieuses dans une classe, classifiées par nom, et pour aider la lisibilité un type **NomMatiere** est défini:

```
using NomMatiere = std::string;

class Inventaire {
public:
    Inventaire();
    ~Inventaire();

    double prixTotal(const std::map<NomMatiere, MatierePrecieuse*> matieres) const;

private:
    std::map<NomMatiere, double> grammesDeMatieres_;
};
```

L'attribut **grammesDeMatieres_** contient la nombre de grammes de différentes matières précieuses, classifiées par nom.

La méthode **prixTotal** reçoit en paramètre une autre map qui contient les pointeurs vers les définitions des matières précieuses, classifiées avec les mêmes noms.

En utilisant des algorithmes et des fonctions (possiblement lambdas), nous voulons implémenter la méthode **prixTotal** qui retourne la valeur totale de l'inventaire selon le prix actuel des matières. Cette question se décompose deux étapes.

Question 11

Écrire une fonction pour calculer le prix d'une certaine quantité d'une matière précieuse.

La fonction reçoit une **map** où les **MatieresPrecieuses** sont placées par nom, ainsi qu'une paire <**NomMatiere**, **double**> qui indique le nom de la matière et le nombre de grammes de cette matière.

La fonction retourne le prix de ce nombre de grammes de cette matière, si la matière est présente dans la **map**.

Dans le cas où la matière n'est pas dans la **map**, la fonction retourne zéro; ceci peut être considéré un cas exceptionnel.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```

1 double evaluerPrixQuantiteMatiere(const map<NomMatiere, MatierePrecieuse*>& matieres, const pair<NomMatiere, double>& grammesDeMatiere)
2 {
3     //TODO: Retourner le prix total pour ce nombre de gramme de la matière nommée, si elle existe.
4     auto it = matieres.find(grammesDeMatiere.first);
5     if (it != matieres.end()) {
6         auto& [cle, valeur] = *it;
7         return valeur->getPrixParGramme() * grammesDeMatiere.second;
8     }
9     //TODO: Retourner zéro si la matière n'existe pas dans la map.
10    return 0.0;
11 }
```

	Test	Résultat attendu	Résultat obtenu	
✓	-			✓
✓	MaMatiere m; m.setPrixParGramme(42); map<string, MatierePrecieuse*> matieres = {"Ma"s, &m}; cout << evaluerPrixQuantiteMatiere(matieres, {"Ma"s, 2.0}) << "\n" << evaluerPrixQuantiteMatiere(matieres, {"Ma"s, 10.0}) << "\n";	84 420	84 420	✓
✓	MaMatiere m1; m1.setPrixParGramme(42); MaMatiere m2; m2.setPrixParGramme(17.25); map<string, MatierePrecieuse*> matieres = {"Ma"s, &m1}, {"Ma2"s, &m2}; cout << evaluerPrixQuantiteMatiere(matieres, {"Ma"s, 10.0}) << "\n" << evaluerPrixQuantiteMatiere(matieres, {"Ma2"s, 4.0}) << "\n";	420 69	420 69	✓
✓	MaMatiere m; m.setPrixParGramme(42); map<string, MatierePrecieuse*> matieres = {"Ma"s, &m}; cout << (evaluerPrixQuantiteMatiere(matieres, {"Ma"s, 1.0}) != 0) << "\n" << evaluerPrixQuantiteMatiere(matieres, {"Rien"s, 0.0}) << "\n";	true 0	true 0	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 2,00/2,00.

Question 12

Correct

Note de 2,25
sur 2,25

Marquer la question

Maintenant que nous avons une fonction pour calculer le prix d'une des matières, nous pouvons assez facilement passer chaque matière d'un inventaire dans cette fonction et faire la somme des résultats pour obtenir le total.

Implémentez la méthode `prixTotal` de la classe `Inventaire` en utilisant la fonction `evaluerPrixQuantiteMatiere` définie précédemment ainsi que l'algorithme `transform_reduce`. On utilise la version qui prend un foncteur binaire et un foncteur unaire (la version vue dans les notes de cours, et la version 3 dans la doc sur cppreference). Vous n'avez évidemment pas le droit d'utiliser `for` ou `while` à la place.

Indice: vous aurez probablement besoin d'écrire un lambda.

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

```

1 double Inventaire::prixTotal(const std::map<NomMatiere, MatierePrecieuse*>& matieres) const
2 {
3     //TODO: Utiliser transform_reduce et evaluerPrixQuantiteMatiere pour calculer le prix total de toutes les matières dans la classe Inventaire.
4     return transform_reduce(grammesDeMatieres_.begin(), grammesDeMatieres_.end(), 0,
5                             [double prix1, double prix2] { return prix1 + prix2; },
6                             [&](const pair<NomMatiere, double>& grammesDeMatiere)
7                             {
8                                 return evaluerPrixQuantiteMatiere(matieres, grammesDeMatiere);
9                             });
10 }
11 }
```

	Test	Résultat attendu	Résultat obtenu	
✓	-			✓
✓	<pre> Inventaire p; Or24k o; o.setPrixParGramme(2); p.ajouterMatiere(o.getNom(), 21); map<string, MatierePrecieuse*> matieres = { {o.getNom(), &o} }; cout << p.prixTotal(matieres) << "\n"; </pre>	42	42	✓
✓	<pre> Inventaire p; MaMatiere m; m.setPrixParGramme(10); Or24k o; o.setPrixParGramme(2); p.ajouterMatiere(m.getNom(), 40); p.ajouterMatiere(o.getNom(), 10); map<string, MatierePrecieuse*> matieres = { {m.getNom(), &m}, {o.getNom(), &o} }; cout << p.prixTotal(matieres) << "\n"; cout << utilise_transform_reduce; </pre>	420 true	420 true	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 2,25/2,25.

Question 13

Correct

Note de 2,00
sur 2,00[Marquer la question](#)

Nous avons la hiérarchie où un **Livre** est un **Document**, un **ManuelPédagogique** est un **Livre**, une **Vidéo** est aussi un **Document** et un **ManuelPédagogiqueMultimédia** est à la fois un **ManuelPédagogique** et une **Vidéo**.

On voit facilement un problème d'héritage en diamant pour **ManuelPédagogiqueMultimédia**. Dans le code ci-dessous, remplissez les boîtes avec le type d'héritage approprié pour que les classes soient compilables et utilisables, et indiquez combien de **Document** sont construits dans la fonction **f**. Si vous avez à le faire, mettez le moins d'héritage virtuel possible.

```
1. class Document {
2. public:
3.     Document(string titre) : titre_(titre) {
4.         cout << "Document(" << getTitre() << ")\\n";
5.     }
6.
7.     virtual ~Document() {
8.         cout << "~Document()\\n";
9.     }
10.    const string& getTitre() const { return titre_; }
11.    virtual void regarder() = 0;
12.
13. private:
14.     string titre_;
15. };
16.
17. class Livre : public virtual Document {
18. public:
19.     Livre(string titre) : Document(titre) {
20.         cout << "Livre(" << getTitre() << ")\\n";
21.     }
22.
23.     ~Livre() override {
24.         cout << "~Livre()" << "\\n";
25.     }
26.
27.     void regarder() override {
28.         cout << "Les livres en informatique ne sont-il pas déjà desuets au moment de l'impression?\\n";
29.     }
30. };
31.
32. class ManuelPedagogique : public Livre {
33. public:
34.     ManuelPedagogique(string titre) : Document(titre), Livre(titre) {
35.         cout << "ManuelPedagogique(" << getTitre() << ")\\n";
36.     }
37.
38.     ~ManuelPedagogique() override {
39.         cout << "~ManuelPedagogique()" << "\\n";
40.     }
41. };
42.
43. class Video : public virtual Document {
44. public:
45.     Video(string titre) : Document(titre) {
46.         cout << "Video(" << getTitre() << ")\\n";
47.     }
48.
49.     ~Video() override {
50.         cout << "~Video()" << "\\n";
51.     }
52.
53.     void regarder() override {
54.     }
```

```
13. }
14.
15. ~Livre() override {
16.     cout << "~Livre()" << "\n";
17. }
18.
19. void regarder() override {
20.     cout << "Les livres en informatique ne sont-il pas déjà desuets au moment de l'impression?\n";
21. }
22. };
23.
24. class ManuelPedagogique : public 1 Livre {
25. public:
26.     ManuelPedagogique(string titre) : Document(titre), Livre(titre) {
27.         cout << "ManuelPedagogique(" << getTitre() << ")\\n";
28.     }
29.
30.     ~ManuelPedagogique() override {
31.         cout << "~ManuelPedagogique()" << "\\n";
32.     }
33. };
34.
35. class Video : public virtual 1 Document {
36. public:
37.     Video(string titre) : Document(titre) {
38.         cout << "Video(" << getTitre() << ")\\n";
39.     }
40.
41.     ~Video() override {
42.         cout << "~Video()" << "\\n";
43.     }
44.
45.     void regarder() override {
46.         cout << "Juste une autre video de chat et je ferme!" << "\\n";
47.     }
48. };
49.
50. class ManuelPedagogiqueMultimedia : public 1 ManuelPedagogique, public 1 Video {
51. public:
52.     ManuelPedagogiqueMultimedia(string titre) : Document(titre), ManuelPedagogique(titre), Video(titre) {
53.         cout << "ManuelPedagogiqueMultimedia(" << getTitre() << ")\\n";
54.     }
55.
56.     ~ManuelPedagogiqueMultimedia() override {
57.         cout << "~ManuelPedagogiqueMultimedia()" << "\\n";
58.     }
59.
60.     void regarder() override {
61.         Video::regarder();
62.     }
63. };
64.
65. void f()
66. {
67.     ManuelPedagogiqueMultimedia monManuel("INF1015"); // Cette ligne construit 1 Document
68. }
```