Qualité logicielle Devoir n°1

par Thierry Vaira © v.1.00

Ce sujet comprend 20 questions pour un total de 30 points.
Aucun document autorisé
Durée : 2 h

Nom : _____

La Qualité logicielle regroupe les savoirs suivants :
- Gestion des erreurs/exceptions
- Documentation des programmes (dont le suivi des versions (*versioning*))
- Intégration logicielle
- Outils de déverminage (débogage)
- Procédures de test et de stress

Gestion des erreurs/exceptions

Question 1 (1 point)
Dans un développement orienté objet, qu'est-ce que permet la gestion d'exceptions ?
☐ la facilité d'écrire un code sans erreurs
☐ la facilité d'écrire un code robuste

Question 2 (1 point)
Quelle est l'instruction qui permet de lever une exception ?
☐ le bloc `try-catch` ☐ `exception` ☐ `range_error` ☐ `throw`

Question 3 (1 point)
Quelle est l'instruction qui permet d'intercepter (capturer) une exception ?
☐ le bloc `try-catch` ☐ `exception` ☐ `range_error` ☐ `throw`

Soit le code source suivant :

```
try
{
   vector<int> tableau( 10 ); // crée un tableau de taille 10
   tableau.at( 10 );
}
catch ( const std::bad_alloc & )
{
   cerr << "Erreur : mémoire insuffisante !\n";
}
```

```
catch ( const std::out_of_range & )
{
   cerr << "Erreur : débordement de mémoire !\n";
}
catch ( const std::exception & e )
{
   cerr << "Erreur !\n";
}
catch ( ... )
{
   cerr << "Erreur inconnue !\n";
}
```

Question 4 (1 point)
Si on exécute le code ci-dessus, quel message s'affichera-t-il ?
☐ "Erreur : mémoire insuffisante !"
☐ "Erreur : débordement de mémoire !"
☐ "Erreur !"
☐ "Erreur inconnue !"
☐ aucun

Soit le code source suivant :

```
class ErreurRatio : public exception
{
   private:
      string cause;

   public:
      ErreurRatio(string c) throw() : cause(c) {}
      ~ErreurRatio() throw() {}
      const char* what() const throw() { return cause.c_str(); }
};

/*
 * La classe Ratio code un nombre rationnel de la forme n/d
 * en stockant indépendamment son numérateur (n) et
 * son dénominateur (d) qui sont tous les deux des entiers.
*/
class Ratio
{
   private:
      int numerateur, denominateur;

   public:
      Ratio(int num = 0, int den = 1) throw (ErreurRatio);
      ~Ratio();
};
```

Question 5 (1 point)
Dans la définition du constructeur de la classe `Ratio`, on doit lever une exception `ErreurRatio` si le dénominateur est nul. Quelle sera alors l'instruction à ajouter dans le code du constructeur ?
☐ `try { if (denominateur == 0) return; } catch (ErreurRatio & e) { throw ErreurRatio("Denominateur nul !") };`
☐ `if (denominateur == 0) throw ErreurRatio("Denominateur nul !");`
☐ `if (denominateur == 0) cerr « "Denominateur nul !";`
☐ `if (denominateur == 0) throw "Denominateur nul !";`

Documentation des programmes : le suivi des versions (*versioning*))

Question 6 (1 point)
Qu'est-ce qu'un système de gestion de version VCS (*Version Control System*) comme Subversion ?
☐ un outil de déploiement d'applications
☐ un outil pour maintenir l'ensemble des versions des fichiers d'un logiciel
☐ un outil de création de versions d'un logiciel
☐ un outil garantissant la qualité d'un projet
☐ un système de communication entre les membres d'une équipe de développement

Question 7 (1 point)
Dans une gestionnaire de version comme Subversion, que représentent les différents numéros de révision ?
☐ la priorité des modifications apportées par le gestionnaire
☐ les numéros de version du logiciel attribués par le développeur
☐ un indice attribué pour toute modification prise en compte par le gestionnaire
☐ les numéros de version du gestionnaire

Question 8 (1 point)
Dans la convention `nom.majeur.mineur` de nommage d'une *release*, que signifie le numéro `majeur` ?
☐ il signifie une évolution majeure qui la rend incompatible avec la version précédente
☐ il signifie un changement d'équipe de développeurs
☐ il signifie une correction majeure d'anomalies ou bugs
☐ il signifie que le numéro mineur a atteint son maximum et qu'il faut changer de numéro majeur

Question 9 (1 point)
Dans la convention `nom.majeur.mineur` de nommage d'une *release*, que signifie le numéro `mineur` ?
☐ il signifie une évolution mineure qui la rend incompatible avec la version précédente
☐ il signifie un changement d'équipe de développeurs
☐ il signifie une correction d'anomalies ou bugs tout en restant compatible avec la version majeure en cours
☐ il signifie qu'une modification importante a été apportée

Intégration logicielle

Une bibliothèque logicielle est fournie avec un ensemble de fichiers. Généralement, on trouve :
– Un répertoire `include`, qui contient des fichiers ayant une extension « `.h` »
– Un répertoire `lib`, qui contient des fichiers ayant comme extension « `.lib` »

Lors du développement d'un projet, un ensemble de fichiers est nécessaire à la génération de l'application (partie logicielle). On vous demande de préciser le contenu des fichiers manipulés (on demande de distinguer deux types de fichier, les fichiers textes et les fichiers binaires). On vous demande également d'indiquer la phase durant laquelle ces fichiers sont utilisés lors de la génération de l'application.

Question 10 (2 points)
Cocher les bonnes réponses concernant ces différents fichiers dans le tableau ci-dessous.

Type de fichier	Contenu du fichier (**texte** : code ascii)	Contenu du fichier (**binaire** : code machine)	Utilisé en entrée de la phase de **compilation séparée**	Utilisé en entrée de la phase **d'édition de lien**
*.cpp				
*.obj				
*.h				
*.lib				
*.exe				

Outils de déverminage (débogage)

Question 11 (1 point)
Qu'est-ce qu'un débogueur ou dévermineur ?
☐ un logiciel capable de détecter les bugs d'un programme
☐ un logiciel capable d'observer et de contrôler l'exécution du programme

Question 12 (1 point)
En exécutant un débogueur ou dévermineur, est-il possible ? (plusieurs réponses possibles)
☐ de modifier le programme en cours d'exécution
☐ de visualiser le contenu d'une variable
☐ de stopper l'exécution en un endroit précis
☐ de corriger les erreurs détectées

Procédures de test et de stress

Question 13 (1 point)
Dans un développement informatique, quel est l'objectif principal d'un test logiciel ?
☐ vérifier que le code fonctionne correctement
☐ permettre d'améliorer et d'optimiser le code
☐ rechercher une anomalie
☐ corriger un bug

Question 14 (1 point)
Dans un développement orienté objet, qu'est-ce que l'on testera lors de la phase des tests unitaires ?
☐ l'ensemble du matériel ☐ un diagramme de séquence
☐ un bout de code important ☐ une méthode ou une classe ☐ le programme

Question 15 (1 point)
Dans un développement informatique, que permet de vérifier un test d'intégration ?
☐ que l'intégration des modules n'a pas altéré leur comportement
☐ que le programme fonctionne normalement lorsqu'on a intégré tous les modules
☐ de s'assurer qu'on a construit le bon produit
☐ de corriger les erreurs d'intégration

Question 16 (1 point)
Quelle méthode est préconisée par la technique de développement de logiciel TDD (*Test Driven Development* ou en français développement piloté par les tests) ?
☐ écrire les tests unitaires avant d'écrire le code source du logiciel
☐ écrire le code source du logiciel avant d'écrire les tests unitaires

On s'intéresse à un programme de gestion de codes postaux dans un système de tri d'objets. Ces codes postaux sont constitués de 5 caractères compris entre '0' et '9' et ils sont stockés dans un objet de type `string`. Le programme doit d'abord vérifier que le code postal est bien constitué de 5 caractères compris entre `0` et `9`. Pour réaliser cela, on a écrit une fonction `verifierCode(string code)` qui retourne `true` si le code est valide sinon `false` :

```
bool verifierCode(string code)
{
   bool codeOK = true;
   int i = 0;
   while (codeOK && i < 5)
   {
      if ((code[i] < '0') || (code[i] > '9'))
            codeOK = false;
      i++;
   }
   return codeOK;
}
```

On désire tester unitairement cette fonction.

Question 17 (4 points)
En utilisant le code source fourni ci-dessus, compléter la fiche de test ci-dessous.

Description	Code en entrée	Résultat attendu	Résultat obtenu
5 caractères compris entre '0' et '9'	`"84000"`	`true`	
Longueur > à 5 caractères compris entre '0' et '9'	`"840006"`	`false`	
Longueur < à 5 caractères compris entre '0' et '9'	`"8400"`	`false`	
5 caractères non compris entre '0' et '9'	`"8A000"`	`false`	

Question 18 (1 point)
Doit-on alors valider la fonction `verifierCode(string code)` ?
☐ oui ☐ non

Les 5 derniers codes postaux entrés seront affichés à l'écran du poste de codage. Pour réaliser cet affichage, les codes sont stockés dans un tableau. La classe `CCodesAffiches` permet de gérer cette liste contenant les codes affichés.

```
const int NB_CODES = 5 ;

class CCodesAffiches
{
   private:
      int nbCodes ;
      string liste[NB_CODES] ;

   public :
      CCodesAffiches() ;
      void ajouterCode( const string &code );
};
```

La méthode `ajouterCode()` ajoute un `code` dans la `liste`. Si le tableau est plein, les données seront décalées d'une case du tableau afin de placer le code ajouté en dernière position.
On désire tester unitairement cette méthode.

Question 19 (3 points)
Proposer un plan de tests permettant de valider la méthode `ajouterCode()` en complétant la fiche ci-dessous.

Description	Code en entrée	Tableau liste	Résultat attendu	Résultat obtenu

Question 20 (5 points)
Définir la méthode `ajouterCode( const string &code )` de la classe `CCodesAffiches`.