### Question 1 : Quel suffixe dois-je utiliser pour être sûr qu'un littéral numérique déclaré soit de type double ?

b A

B If

C f et le nombre doit être décimal

D aucun, il suffit que le nombre soit décimal.

# Question 2 : Soit un short int x et un char y. Combien de conversions l'expression int f = x + y + 3; engendre-t-elle?

A 4

В 0

 $C_{2}$ 

D 5

E 1

F 3

#### Question 3 : L'opérateur new peut être utilisé pour effectuer l'allocation :

A d'un objet avec appel au constructeur.

B d'un tableau d'objets sans appel au constructeur.

C d'un tableau d'objets avec appel au constructeur.

D d'un objet sans appel au constructeur.

E sous sa forme operator, d'une zone de mémoire en précisant sa taille (i.e. comme malloc) sans appel à aucun constructeur.

# Question 4 : Soit le patron de fonction : template <class U> void fun(U x) $\{ ... \}$ . Alors, l'écriture template <> void fun<int> (int x) $\{ ... \}$ où ... représente le code de la fonction:

A n'est pas syntaxiquement correcte.

B est une spécialisation totale de fun.

C est une spécialisation partielle de fun.

#### **Question 5 : La composition de A avec B traduit la relation :**

A que A possède ou est constitué de B, mais sans idée de possession exclusive. B que A est un B.

C que A possède ou est constitué de B, avec une idée de possession exclusive.

#### Question 6 : Un objet est une entité qui possède :

A une cardinalité.

B un état.

C une navigabilité.

D une identité.

E une dépendance.

F un comportement.

#### Question 7 : Soit un entier n. L'écriture : A \*a = new A(n) réserve :

A cette écriture produit une erreur de compilation.

B une zone mémoire permettant de stocker n objets de type A.

C une zone mémoire permettant de stocker un objet de type A initialisé avec le constructeur A(int) où cet entier est n.

# Question 8 : En général, pour une classe T, lorsque l'on utilise le constructeur par déplacement T(T&& t):

A on ne déplace qu'une propriété.

B les champs de t sont copiés dans \*this.

C le destructeur sur t est appelé

D les champs de t sont échangés avec \*this.

E l'objet construit prend l'adresse de t.

#### **Question 9 : Un fonctor:**

A peut toujours se substituer à un pointeur de fonction

B ne peut avoir qu'une seule instance

C peut passer une instance par référence

D peut possèder des états internes

E s'appelle comme une fonction

#### Question 10 : Soit le code: $\{A *a = new A[10]; \}$

A a et la mémoire allouée par A[10] sont automatiquement libérées en fin de portée.

B a est automatiquement libérée en fin de portée, la mémoire allouée par A[10] doit explicitement être libérée.

C a et la mémoire allouée par A[10] doivent toutes les deux être explicitement libérées.

# Question 11 : Soit une fonction dont le prototype est void fun(const int \*p). cette fonction accepte qu'on lui passe en paramètre un:

A const int\*.

B int\*.

C int.

D int&.

E const int.

F const int&.

#### Question 12: La composition est un couplage plus fort que l'agrégation :

A vrai

B faux

### Question 13 : Parmi les relations suivantes entre classes, quelle est celle qui introduit le couplage le moins fort :

A la composition

B l'association

C l'agrégation

D l'héritage

### Question 14 : Soit une fonction dont le prototype est const int fun(int p). cette fonction accepte que la valeur de retour soit affectée à un:

A const int&.

B int.

C const int\*.

D int\*.

E const int.

Fint&.

### Question 15 : Si un membre (=un champs) d'une classe T est déclaré public, alors si l'objet est const, son accès est:

#### A pour les méthodes ou fonctions, en lecture seule

B pour les méthodes ou fonctions, en lecture/écriture

C pour les méthodes, en lecture/écriture; pour les fonctions, en lecture seule

D pour les méthodes, en lecture seule; pour les fonctions, inaccessible

### Question 16 : Dans une classe T, on définit un constructeur de la forme: T(const &S). Ce constructeur permet:

A de convertir explicitement un objet de type S en un objet de type T

B de construire un objet de type T à partir d' un objet de type S

C de convertir implicitement un objet de type S en un objet de type T

D de construire un objet de type S à partir d' un objet de type T

#### Question 17 : Le stack est la zone mémoire dans laquelle est stockée:

A la mémoire allouée dynamiquement

B Les variables globales ou les variable statiques

C La pile d'appel des fonctions

D Les variables locales des fonctions

#### Question 18 : En C++, l'agrégation de A avec B peut s'écrire:

A A est détruit quand B est détruit.

B B fait partie des champs de A.

CB a un pointeur vers A.

D A hérite de B.

#### Question 19: template <class U, class V> T fun(U &u) { return V(u); }

A L'utilisation de ce template exigera de spécifier explicitement le type de U et V au moment de l'appel.

B Ce type de template n'est pas autorisé car le compilateur n'a pas moyen de déterminer le type de retour.

C Ce type de template est autorisé.

# Question 20 : Pour la classe complex définissant les nombres complexes, parmi les opérateurs suivants, indiquer ceux qui produiront nécessairement des objets temporaires:

A operator=

B operator+

C operator+=

D operator==

#### Question 21 : La gestion du déplacement doit être nécessairement effectuée dans une classe lorsque:

A la classe ne possède que des champs privées interne

B sa classe mère implémente déjà le déplacement

C la classe possède des champs partagées allouées dynamiquement (de type shared\_ptr).

D la classe possède des champs privées allouées dynamiquement (de type unique ptr).

E la classe implémente le constructeur ou l'assignation par déplacement

### Question 22 : Soit la structure suivante: struct S { double a; char b; int c; }. La taille en octet de cette structure est:

A 9 octets

B 13 octets

C 24 octets

# Question 23 : Soit la structure suivante: struct S { double a; char b; int c; }. Cette structure est alignée sur une adresse multiple de:

A 16 octets

B 8 octets

C 1 octets

D 2 octets

E 4 octets

# Question 24 : Soit x une variable de type short int. Quels sont les types vers lequel ce type peut être converti sans perte de précision?

A char

B short unsigned int

C int

D unsigned char

E unsigned int

F short int

#### Question 25: Lors de l'utilisation du type unique\_ptr,

A lors de la déclaration d'une variable locale, la mémoire pointée par le unique\_ptr est automatiquement allouée.

B lors de l'allocation d'un objet ayant comme membre un unique\_ptr, la mémoire pointée est automatiquement allouée.

C si je déclare membre d'une classe, alors la mémoire pointée est automatiquement libérée lorsque l'objet est détruit.

D si je déclare une variable locale de ce type, alors la mémoire pointée est automatiquement libérée lorsque l'on arrive en fin de portée de la variable.

## Question 26 : Si une classe A hérite publiquement d'une classe B alors les membres et méthodes protégés de B sont:

A publics

B protégés

C privés

#### **Ouestion 27: Un cast du C:**

A n'autorise pas la modification du type d'un pointeur.

B laisse à la responsabilité à l'utilisateur de savoir si la conversion est possible et si la valeur convertie représentera quelque chose.

C n'est pas plus dangereux qu'un static\_cast lorsque l'on convertit des valeurs numériques.

D n'échoue jamais à la compilation, mais peut échouer à l'exécution.

# Question 28 : Parmi les relations suivantes entre classes, quelle est celle qui introduit le couplage le plus fort:

A l'association

B l'héritage

C la composition

D l'agrégation

# Question 29 : On voudrait construire un nombre plus grand qu'un int et le stocker dans un long long int. On peut écrire: long long int x =

A 2 \* numerical\_limit<int>max());

B 1 << 48

C 1000000000000LL:

D numerical\_limit<int>max()) + 1LL;

#### Question 30 : L'agrégation de A avec B traduit la relation:

A que A possède ou est constitué de B, mais sans idée de possession exclusive

B que A est un B.

C que A possède ou est constitué de B, avec une idée de possession exclusive

#### Question 31 : L'opérateur delete peut être utilisé pour effectuer la desallocation:

A d'un objet avec appel au destructeur.

B sous sa forme operator, d'une zone de mémoire en précisant le pointeur de début de bloc (comme malloc) sans appel à aucun destructeur.

C d'un tableau d'objets sans appel au destructeur.

D d'un tableau d'objets avec appel au destructeur.

E d'un objet sans appel au destructeur.

#### Question 32 : En programmation orienté objet, une interface A:

A dont le sizeof(A) est égal à 0.

B doit obligatoirement être héritée.

C ne contient aucun membre.

D est une classe abstraite.

E ne contient que des méthodes virtuelles.

# Question 33 : Une classe T définit l'assignation par copie et le constructeur T(const S&) où S est une autre classe. Indiquer les opérations que rendent précisément possibles ces déclarations :

A S a = b; où b est de type T

B a = b; où a est de type T, b de type S

C a = b; où a est de type S, b de type T

D T a = b; où b est de type S

#### Question 34 : Lors de l'allocation mémoire d'un tableau d'objets de type A avec l'opérateur new:

A si le constructeur par défaut n'existe pas, aucun constructeur n'est lancé et les objets ne sont pas initialisés.

B si le constructeur par défaut n'existe pas, la compilation échoue.

C s'il existe, le constructeur par défaut de A est utilisé sur chaque élément du tableau.

#### Question 35 : Si une classe A hérite d'une classe B, alors la partie B de l'objet A est construite:

A en utilisant un des constructeurs de B dans la chaine d'initialisation.

B en appelant implicitement et obligatoirement le constructeur par défaut de B.

C en initialisant les membres de B dans la chaine d'initialisation.

### Question 36 : Soit le patron de fonction: template <class U> void fun(U &x). Alors, l'appel fun(iPtr) ou iPtr est de type int\*:

A n'est pas pris en charge par ce patron de classe.

B instancie ce patron avec U=int\*.

C instancie ce patron avec U=int.

# Question 37 : Soit le patron de fonction template <class T> const T std::max(const T a, const T b) { return (b<a?a:b); }. Alors ce template ne peut s'instancier que pour une classe U pour laquelle est définie:

A Le constructeur par copie.

B L'assignation par copie.

C L'opérateur <

#### **Question 38 : Un lien peut posséder:**

A une cardinalité.

B un comportement.

C une dépendance.

D une identité.

E une navigabilité.

F un état.

# Question 39 : Parmi les littéraux flottants double précision suivants, donner ceux qui peuvent être convertis en float (simple précision) sans perte de précision:

A 1.0/8.0

B 1.0/3.0

C 1.0

D M\_PI

#### Question 40 : Dans l'expression suivante: a = fun(b+4) + c;, indiquez les rvalues:

A fun(b+4)

Ва

Сс

 $D_{b+4}$ 

Εb

F fun(b+4) + c

#### **Question 41 :** La surcharge de l'opérateur = pour une classe T définie par l'utilisateur s'effectue:

A on peut choisir parmi toutes les autres possibilités

B nécessairement comme méthode dans la classe T

C nécessairement comme une fonction extérieure à la T et friend avec T

D nécessairement comme une fonction extérieure à la T sans nécessairement être friend avec T

#### Question 42: L'upcasting d'une classe A vers une classe B

A est implicitement effectuée dans tous les contextes où elle est possible.

B doit être explicitement effectué (avec un cast) et peut échouer.

C se définit comme la conversion d'une classe fille vers une classe mère.

D se définit comme la conversion d'une classe mère vers une classe fille.

#### Question 43 : Soit le patron de fonction: template <class U> void fun(U &x). Alors, elle est compilée:

A seulement pour les types U des arguments avec lesquels la fonction a pu être appelée dans le code.

B seulement pour tous les types U utilisés dans le code.

C tous les types standards (BIT), tous les types définis dans les headers inclus, et tous les nouveaux types définis dans le code.

### Question 44 : Soit une fonction dont le prototype est int& fun(int &p). cette fonction accepte que la valeur de retour soit affectée à un:

A const int.

B const int\*.

C const int&.

D	int*.
Е	<mark>int.</mark>
F	int&.

#### Question 45 : Soit A \*a = new A, alors pour l'objet alloué:

A a = this

Ba = &this.

C this = &a.

### Question 46 : Il est possible de faire du polymorphisme entre les classes d'un patron de classe pour les différentes instanciations de type.

A faux

B vrai

#### Question 47: Le constructeur par copie d'une classe T a pour prototype:

A T(T)

B T(const T&)

C T(T&)

D T(const T)

#### Question 48 : Si une méthode d'une classe A est virtuelle, alors:

A elle est virtuelle dans toute classe héritée de A, même si la méthode est spécialisée.

B le pointeur dans la VTABLE n'est utilisé que si le compilateur n'est pas en mesure de déterminer le type exact de l'objet utilisé.

C pour un objet de type A, elle utilise systématiquement le pointeur associée dans la VTABLE.

D pour un pointeur sur un objet de type A, elle utilise systématiquement le pointeur associée dans la VTABLE.

#### **Question 49 : Un shared ptr a la sémantique suivante:**

A il peut être copié sans desallocation ou nouvelle allocation mémoire.

B il peut être réaffecté sans jamais aucune desallocation ou nouvelle allocation mémoire.

C il peut être déplacé sans jamais aucune desallocation ou nouvelle allocation mémoire.

#### Question 50 : Si une classe A hérite d'une classe B, alors je peux:

A passer un objet de type B à une fonction qui attend un objet de type A B les deux autres réponses sont fausses.

C passer un objet de type A à une fonction qui attend un objet de type B

# Question 51 : Soit x une variable de type unsigned short int. Quels sont les types vers lequel ce type peut être converti sans perte de précision?

A unsigned int

B int

C short int

D unsigned char

E char

F short unsigned int

#### Question 52 : Le heap (ou free memory) est la zone mémoire dans laquelle est stockée:

A La pile d'appel des fonctions

B Les variables locales des fonctions

D la mémoire allouée dynamiquement

# Question 53 : soit une fonction dont le prototype est void fun(int &p). cette fonction accepte qu'on lui passe en paramètre un:

A const int.

B int&.

C const int&.

D int\*.

E int.

F const int\*.

### Question 54 : Soit le patron de fonction : template <class U> const U& fun(U &x, U &y) { ... }. Alors l'appel fun<int>(3.2f,4.5f):

A force l'instanciation du type en entier, et convertira les paramètres en entier à l'appel de la fonction.

B n'est pas pris en charge par ce patron de classe car cette conversion est impossible.

### Question 55 : Si l'on veut passer une fonction en paramètre d'une autre fonction, quels sont les types qui permettent de prendre une fonction en paramètres :

A un pointeur de fonction

B un prédicat

C un template

D un fonctor

E un opérateur

F une lambda-expression

#### Question 56: La composition de A avec B exprime:

A que A est une spécialisation B.

B que la durée de vie de A dépend de celle de B.

C que A contient B.

#### **Question 57 : L'héritage de A par B exprime :**

A que A est une spécialisation B.

B que A contient B.

C que la durée de vie de A dépend de celle de B.

### Question 58 : Soit une fonction dont le prototype est int\* fun(int \*p). cette fonction accepte que la valeur de retour soit affectée à un:

A const int.

B int\*.

C const int&.

D const int\*.

E int.

Fint&.

#### Question 59 : Soit le patron de fonction : template <class U> void fun(U &x). Alors, l'appel fun(4)

A n'est pas pris en charge par ce patron de classe.

B instancie ce patron avec U = int.

C instancie ce patron avec U=const int.

#### Question 60 : Une méthode virtuelle est une méthode qui :

A doit nécessairement être redéfinie à tous les niveaux de la hiérarchie de classe.

B doit nécessairement être définie à tous les niveaux de la hiérarchie de classe.

C peut n'être définie qu'à certains niveaux de la hiérarchie, héritée si non redéfinie, indéfinie sinon.

### Question 61 : Soit le patron de fonction : template <class U> void fun(U &x). Alors, l'appel fun(iPtr) ou iPtr est de type int\* provoque l'instanciation suivante :

A void fun<int\*>(int\* &x).

B n'est pas pris en charge.

C void fun<int\*>(int\* x).

D void fun<int\*>(int& x).

### Question 62 : En UML, dans la boite rectangulaire qui représente une classe, le second compartiment représente:

A les liens

B les surcharges d'opérateur

C les attributs

D le nom de la classe

E les méthodes

#### Question 63 : Dans l'expression suivante : a = b + 4;, indiquez les lvalues:

A 4

B b+4

Cb

D a

# Question 64 : Soit la fonction : int& max(int &x,int &y) { return (a > b ? a : b); }. Dans l'expression suivante: max(u,v) = u+v; où u et v sont deux variables entières, indiquez les lvalues:

A u+v

Вu

C max(u,v)

Dν

#### **Question 65: L'opérateur reinterpret\_cast:**

A permet de convertir tout type de pointeur en tout autre type de pointeur seulement si ces types sont directement incompatible.

B permet de convertir tout type de pointeur en tout autre type de pointeur seulement si ces types sont compatibles par upcasting ou downcasting.

C permet de convertir tout type de pointeur en tout autre type de pointeur même si ces types sont incompatible (=équivalent au cast du C).

#### Question 66 : L'écriture A \*a = new A[10]

A libère automatiquement la totalité mémoire allouée en fin de portée de a.

B utilise dans la mémoire seulement une zone de taille 10\*sizeof(A).

C utilise dans la mémoire l'espace d'un pointeur et une zone de taille 10\*sizeof(A).

# Question 67 : Soit une classe A qui hérite d'une classe B. S'il n'y a pas de constructeur par défaut pour B (y compris par défaut) et que je ne définis pas de constructeur pour A, alors :

A la compilation échoue car le constructeur par défaut de A ne peut être construit.

B le compilateur génère un constructeur par défaut pour A qui ne construit (= n'initialise) ni A ni B

C le compilateur génère un constructeur par défaut de A qui ne construit pas B (=non initialisé).

# Question 68 : Quel suffixe dois-je utiliser pour être sur qu'un littéral numérique entier déclaré soit de type long long int?

A aucun

B LL

CI

DL

E 11

## Question 69 : On s'intéresse aux conversions explicites avec le cast du C et le static\_cast du C++. Cocher les assertions vraies.

A toute conversion effectuée par un static\_cast peut également être effectué par un cast.

B dans tous les cas où la conversion est possible pour le C et le static\_cast, les deux conversions donnent exactement le même résultat.

C toute conversion effectuée par un cast peut également être effectué par un static\_cast.

D il existe des cas où le cast est autorisé, mais pas le static cast, et inversement.

#### Question 70 : Si on exclut l'élision de copie, le constructeur par copie sur T est nécessairement utilisé :

A lors de l'appel à cette fonction: T fun(const T&)

B lors de l'appel à cette fonction: const T& fun(T)

C lors de l'écriture a = b; où a et b sont de type T

D lors de l'écriture T a = b; où b est de type T

#### Question 71 : L'élision de copie peut avoir lieu lorsqu'un objet est construit :

A au retour d'une fonction

B au passage du paramètre d'une fonction

C dans le constructeur par copie

# Question 72 : Si une classe A hérite d'une classe B dont le destructeur est défini, alors lorsque je détruis un objet de type A, la destruction s'effectue dans l'ordre :

A l'ensemble des membres de A sont détruits dans le destructeur de A, et le destructeur de B n'est jamais appelé B destruction de la partie spécifique à A, puis destruction de la partie B de A

C destruction de la partie B de A, puis destruction de la partie spécifique à A

#### Question 73 : Une méthode virtuelle est dite pure si :

A il n'est pas possible de donner sa définition dans la classe où elle est définie.

B elle est héritée d'une classe qui ne contient que des méthodes virtuelles.

C tous les niveaux de la hiérarchie donne une définition de cette méthode.

#### Question 74 : En C++, la composition de A avec B peut s'écrire:

A A hérite de B.

B B fait partie des champs de A.

C B a un pointeur vers A.

D A est détruit quand B est détruit.

#### Question 75 : Le destructeur est défini dans une classe T. Il est appelé :

A sur une référence de type T en fin de portée de cette référence

B sur un objet temporaire de type T lorsque celui-ci n'est plus nécessaire

C sur une variable automatique de type T qui arrive en fin de portée

D sur un paramètre de type T passé par copie au retour de la fonction

E sur une référence de type T en fin de portée de la variable à laquelle elle fait référence

### Question 76 : Une conversion d'un int vers un short int se fait-elle toujours avec perte de précision quelque soit l'entier converti ?

A oui

B non

#### Question 77: Le downcasting d'une classe A vers une classe B

A se définit comme la conversion d'une classe mère vers une classe fille.

B doit être explicitement effectué (avec un cast) et peut échouer.

C est implicitement effectuée dans tous les contextes où elle est possible.

D se définit comme la conversion d'une classe fille vers une classe mère.

#### Question 78 : Si je définis mon propre constructeur avec au moins un paramètre alors :

A l'assignation par copie par défaut n'est plus définie

B le constructeur par défaut n'est plus défini

C le destructeur par défaut n'est plus défini

D le constructeur par copie par défaut n'est plus défini

# Question 79 : Si je déclare une fonction friend dans un patron de classe, alors cette fonction friend est nécessairement un patron de fonction

A vrai.

B faux.

#### **Question 80: Soit le code suivant:**

```
class A ...;
class B : public A ...;
class C : public A ...;
class D : public B, public C ...;
```

A la classe A est stockée une seule fois, et l'on y accède depuis D soit par B, soit C.

B ce code ne compile pas, car cette définition crée une ambigüité sur le stockage A.

C la classe A est stockée deux fois dans la classe D, une fois dans B, une fois dans C.

#### **Question 81 : L'écriture A a[10]**

A libère automatiquement la totalité mémoire allouée en fin de portée de a.

B utilise dans la mémoire seulement une zone de taille sizeof(A).

C utilise dans la mémoire, l'espace d'un pointeur et une zone de taille sizeof(A).

#### Question 82 : En C++, l'héritage de A avec B peut s'écrire :

A B a un pointeur vers A.

B B fait partie des champs de A. C A est détruit quand B est détruit. D A hérite de B.

#### Question 83: Une classe A est abstraite si:

A il n'est pas possible de créer un objet type A.

B elle ne contient aucun membre.

C elle ne contient que des méthodes virtuelles.

#### Question 84 : Soit un entier n. L'écriture : A \*a = new[n] réserve :

A cette écriture produit une erreur de compilation.

B une zone mémoire permettant de stocker un objet de type A initialisé avec le constructeur A(int) où cet entier est n.

C une zone mémoire permettant de stocker n objets de type A.

#### Question 85 : Soit l'écriture: template < class U, int N> U \*fun() { return new U[N]; }

A Ce type de template n'est pas autorisé car le compilateur n'a pas moyen de déduire le type de retour.

B Ce type de template est autorisé, mais les deux paramètres templates doivent être spécifiés dans tous les cas.

C Ce type de template n'est pas autorisé car seuls les types peuvent être paramètre d'un template.

### Question 86 : AB représentes-en UML :

A une association

B une dépendance

C une navigabilité

# Question 87 : Soit la classe complex, Combien d'objets temporaires sont générés par l'expression suivante : z = complex(1.f, 2.f) + z + 3.0;

A 5

B 4

C 2

D 3

# Question 88 : Si je déclare à la fois un patron de fonction : template <class U> void fun(Ux) et une fonction void fun(int x), alors:

A c'est toujours le patron de classe qui est appelé (le fonction est ignorée).

B c'est la fonction qui est appelée si le paramètre est entier, et le patron de classe dans tous les autres cas.

C cela crée une ambiguïté provoquant une erreur de compilation.

D c'est toujours la fonction qui est appelée (le template est ignoré).

### Question 89 : Soit le patron de fonction : template <class U> void fun(const U x). Alors, l'appel fun(a) ou a est un entier (int) provoque l'instanciation suivante :

A n'est pas pris en charge.

B void fun<int>(const int x).

C void fun<int>(const int x).

D void fun<const int>(const int x).

# Question 90 : Soit le patron de fonction : template <class U> void fun(U x). Alors, l'appel fun(a) ou a est une référence sur un entier (i.e. son type est int&) provoque l'instanciation suivante:

A void fun<int&>(int &x).

B void fun<int>(int x). C n'est pas pris en charge. D void fun<int>(int &x).

#### Question 91 : Lors de la définition d'une classe, this est:

A dans un constructeur, un pointeur vers l'objet en cours de construction

B dans une méthode, une référence vers l'objet courant

C dans une méthode, un pointeur vers l'objet courant

D dans n'importe quel opérateur, un pointeur vers l'objet sur lequel l'opérateur s'applique

### Question 92 : Si une classe A hérite publiquement d'une classe B alors elle peut accéder aux membres et méthodes de B:

A publique seulement

B privée, protégée et publique

C publiques et protégées

# Question 93 : Soit une méthode d'une classe complex dont le prototype est bool equal(complex&). Indiquer quels types de paramètres peuvent être passés à cette méthode:

A une variable complex

B une référence à une variable complex

C une variable const complex

D un pointeur vers un complex

E un objet complex temporaire

#### Question 94 : Soit un entier n. L'écriture: A \*a = new A[n]:

A provoque un arrêt du programme s'il n'y a plus assez de mémoire disponible.

B échoue à la compilation s'il n'existe pas de constructeur par défaut.

C alloue toujours un bloc contigu dans la mémoire.

### Question 95 : Soit une rvalue qui n'est pas une valeur pure (comme 4). Les propriétés d'une telle rvalue sont les suivantes :

A elle a toujours un nom.

B elle est nécessairement en fin de vie.

C elle peut être le résultat d'une expression numérique

D elle se trouve toujours à droite d'un signe =.

E elle peut être déplacée.

#### Question 96 : Si une classe A hérite d'une classe B, alors A hérite :

A du destructeur de B

B des champs de B

C des relations d'amitiés de B

D des constructeurs de B

E des méthodes de B

# Question 97 : Soit le patron de fonction: template <class U> const U& fun(U&x,U&y) $\{$ ... $\}$ . Alors l'appel fun(3.4.f):

A provoque l'instanciation du template avec U=float.

B ne peut pas être géré par ce patron de classe, y compris en spécifiant explicitement le paramètre du template.

C est ambigu.

D provoque l'instanciation du template avec U=int.

Question 98 : Soit le patron de fonction template <class T> const T& std::max(const T &a, const T &b) { return (b<a?a:b); }. Alors ce template ne peut s'instancier que pour une classe U pour laquelle est définie :

A L'assignation par copie.

B Le constructeur par copie.

C L'opérateur <

### Question 99 : Soit une fonction dont le prototype est void fun(int p). Cette fonction accepte en paramètre un :

A int.

B const int.

C int\*.

D const int\*.

E const int&.

Fint&.

# Question 100 : si l'on veut qu'un template s'adapte à tout type T compatible, en laissant exclusivement le compilateur les déduire du contexte, alors cette bibliothèque est constituée :

A de fichiers d'entête (.h) et de code (.cpp).

B de fichiers d'entête (.h) et de bibliothèques compilées (.a, .lib, .dll, etc).

C de fichiers d'entête (.h) seulement.

#### Question 101 : Une interface définit :

A une fonctionnalité.

B un comportement.

C une dépendance

D un état.

E une dépendance.

F une navigabilité.

# Question 102 : Soit une méthode d'une classe A dont le prototype est complex& fun(complex&) const;. Que signifie le const à la fin de la définition ?

A cette syntaxe n'existe pas

B la méthode peut être appelée sur un objet constant

C la méthode ne peut retourner que des références constantes

D la méthode ne peut être appelée qu'avec des paramètres constants

# Question 103 : On a unsigned char x = 127;. Soit l'expression x = x + 1;. Quelle est le nombre n de conversions générés par cette expression et quelle est la valeur x?

A n = 1 et x = 0

B n = 1 et x = 128

C n = 0 et x = 128

D n = 0 et x = 0

E n = 2 et x = 128

F n = 2 et x = 0

# Question 104 : Soit x un unsigned int et y un short int. Après avoir écrit y=x, l'entier y obtenu a une valeur différente de x. Quelles sont les explications possibles ?

A x est un nombre positif trop grand

B le résultat de la conversion est indéfini.

C x est trop petit

### Question 105 : Si un membre de la classe T est déclaré avec l'attribut private, alors on peut accéder à ce membre :

A dans toutes les fonctions déclarée friend dans la classe T

B dans toutes les fonctions qui prennent en paramètre un objet de type T

C dans toutes les méthodes de la classe T

D dans toutes les méthodes d'un classe déclarée comme friend dans la classe T

### Question 106 : Cette question devrait vous permettre de ne pas avoir 0, même si vous n'avez pas travaillé.

A cette réponse n'est pas fausse.

B cette réponse est fausse.

C cette réponse est bonne.

D cette réponse n'est pas bonne.

E surtout ne pas cocher cette case.

# Question 107 : En UML, dans la boite rectangulaire qui représente une classe, le premier compartiment représente :

A les surcharges d'opérateur

B les liens

C le nom de la classe

D les méthodes

E les attributs

#### Question 108 : Quelles sont les fonctions cast permettant de convertir des pointeurs ?

A static\_cast

B reinterpret\_cast

C dynamic\_cast

# Question 109 : Soit une méthode d'une classe complex dont le prototype est bool equal(const complex&). Indiquer quels types de paramètres peuvent être passés à cette méthode :

A un objet complex temporaire

B une variable complex

C une référence à une variable complex

D un pointeur constant vers un complex

E une variable const complex

#### Question 110 : Les propriétés d'une lvalue sont les suivantes :

A elle a toujours un nom.

B elle ne se trouve jamais à droite d'un signe =.

C elle peut être une expression numérique

D elle peut être déplacée.

E on peut toujours prendre son adresse.

### Question 111 : La classe B possède une méthode fun, et la classe A hérite de B. Alors, cette méthode fun

A est occultée par toute surcharge de fun dans A.

B ne peut pas être occultée par une surcharge de fun dans A (erreur de compilation)

C n'est occultée que par une surcharge de fun dans A ayant les mêmes paramètres (sinon elle reste directement utilisable dans A).

### Question 112 : AB représente en UML :

#### A une agrégation où B=conteneur et A=contenu

B une composition où B=conteneur et A=contenu C une composition où A=conteneur et B=contenu D une agrégation où A=conteneur et B=contenu E un héritage où B hérite de A

F un héritage où A hérite de B

#### **Question 113 : Une méthode virtuelle est dite pure si :**

A tous les niveaux de la hiérarchie donne une définition de cette méthode.

B il n'est pas possible de donner sa définition dans la classe où elle est définie.

C elle est héritée d'une classe qui ne contient que de méthode virtuelle.

#### Question 114 : Si un patron de classe hérite d'un autre patron de classe :

A je peux ne préciser aucun paramètre du patron de la classe héritée si ceux-ci peuvent être déduit du patron de la classe qui hérite.

B je ne peux préciser qu'une partie des paramètres templates de la classe héritée.

C il est obligatoire d'expliciter tous les paramètres templates de la classe héritée.

#### Question 115 : Un modèle de données LP64 est tel que:

A sizeof(int)=4, sizeof(double)=8, sizeof(int\*)=8

B ce modèle n'existe pas

C sizeof(int)=4, sizeof(long int)=8, sizeof(int\*)=8

D sizeof(int)=4, sizeof(long int)=4, sizeof(int\*)=8

E sizeof(int)=4, sizeof(long int)=4, sizeof(int\*)=4

### Question 116 : Soit le patron de fonction : template <class U> void fun(U x). Alors, l'appel fun(a) ou a est un entier constant (i.e. son type est const int) provoque l'instanciation suivante :

A void fun<const int>(const int x).

B void fun<int>(const int x).

C n'est pas pris en charge.

D void fun<int>(int x).

### Question 117 : AB représentes-en UML :

A une composition où A=conteneur et B=contenu B un héritage où A hérite de B C une agrégation où A=conteneur et B=contenu

c une agregation ou A=conteneur et B=content

D un héritage où B hérite de A

E une composition où B=conteneur et A=contenu

F une agrégation où B=conteneur et A=contenu

# Question 118 : Lorsque l'on écrit une classe qui contient un champs de type pointeur qui possède, obligatoirement et de manière unique, une zone de mémoire sur laquelle il pointe, alors :

A le constructeur et l'assignation par déplacement doivent nécessairement être définis.

B le destructeur doit s'occuper de la désallocation (si celle-ci n'a pas déjà été faite).

C le constructeur et l'assignation par copie doivent nécessairement être définis.

D tout constructeur doit nécessairement allouer de la mémoire.

#### Question 119 : Soit deux entiers n et p. L'écriture : A \*a = new A[p](n) réserve :

A une zone mémoire permettant de stocker n objets de type A initialisés avec le constructeur A(int) où cet entier est p. B une zone mémoire permettant de stocker p objets de type A initialisés avec le constructeur A(int) où cet entier est n.

C cette écriture produit une erreur de compilation.

### Question 120 : Si une classe A hérite d'une classe B dont les constructeurs appropriés sont définis, alors lorsque je construis un objet de type A, la construction s'effectue dans l'ordre :

A l'ensemble des membres de A (y compris ceux hérités de B) sont construits dans le constructeur de A sans qu'aucun constructeur de B ne soit jamais appelé

B construction de la partie spécifique à A, puis construction de la partie B de A

C construction de la partie B de A, puis construction de la partie spécifique à A

#### **Question 121 : Soit la déclaration : T a, b = a; Cette ligne produit l'appel à:**

A un constructeur par défaut et un constructeur par copie.

B un constructeur par défaut et une assignation par copie.

C deux constructeurs par défaut et une assignation par copie.

#### Question 122 : L'agrégation de A avec B exprime :

A que A contient B.

B que la durée de vie de A dépend de celle de B.

C que A est une spécialisation B.

#### Question 123 : Un shared\_ptr fonctionne de la manière suivante :

A il permet à un pointeur de pointer sur plusieurs zones mémoires.

B il désalloue automatiquement la mémoire pointée lorsque plus aucun pointeur n'y fait référence.

C il connait le nombre de pointeurs qui pointent sur lui.

D il connait l'ensemble des pointeurs qui pointent sur lui.

#### Question 124 : Un modèle de données sur un ordinateur dépend de :

A du système d'exploitation seulement

B de l'architecture du processeur et du système d'exploitation

C de l'architecture du processeur

#### Question 125 : Si une classe A hérite d'une classe B alors :

A B est une spécialisation de A

B A est une spécialisation de B

### Question 126 : Une classe A hérite d'une classe B. Si dans un constructeur de A, je ne fais appel à aucun constructeur de B, alors :

A le code ne compile pas et signale une erreur.

B les champs hérités de B doivent être nécessairement initialisés dans le corps du constructeur de A.

#### Question 127: Dans un patron de classe ou de fonction, le mot-clef typename:

A doit nécessairement précéder le nom d'un type si celui-ci dépend de l'un des paramètres du template.

B peut toujours se substituer au mot-clef class dans les arguments d'un template.

C doit nécessairement précéder le nom de tout type déclaré dans le patron.

#### Question 128 : L'héritage de A par B traduit la relation :

A que A est un B.

B que A possède ou est constitué de B, mais sans idée de possession exclusive.

C que A possède ou est constitué de B, avec une idée de possession exclusive.

#### Question 129 : La surcharge de l'opérateur == pour une classe T définie par l'utilisateur s'effectue :

A nécessairement comme méthode dans la classe T

B on peut choisir parmi toutes les autres possibilités

C nécessairement comme une fonction extérieure à la classe T et friend avec T

D nécessairement comme une fonction extérieure à la classe T sans nécessairement être friend avec T

#### Question 130 : Un unique\_ptr possède les sémantiques suivantes :

A il ne peut pas être copié.

B il est le seul pointeur à pointer sur sa zone mémoire.

C il ne peut pas être déplacé.

D il ne peut pas être affecté à une autre zone mémoire.