Nom	Cocogne
Prénom	Romain

Pour cette évaluation, les livres, les notes et internet sont autorisés. Vous devez travailler seul, sans échange avec d"autres personnes, en répondant directement sur le pdf. Dès la fin de l'examen, vous devez me faire parvenir ce pdf par email (cpp.elec4@gmail.com).

1 Echauffement

1.1 Entourer la ou les affirmations exactes

A propos du language C++:

1.1.a ☑ Le C++ est un langage compilé.

1.1.b ☐ Le C++ est un langage interprété.

1.1.c ☐ Le C++ est un langage dérivé du langage C#.

1.1.d ☐ Le C++ n"existe que depuis le début des années 2000.

1.2 Entourer la ou les affirmations exactes

Le code exécutable issu de la compilation d"un fichier source C++ est

1.2.a ☑ du code binaire, dépendant de la machine physique.

1.2.b ☐ du code assembleur pour une machine virtuelle.

1.2.c ☐ toujours un code binaire pour les processeurs Intel de la famille x86.

1.2.d ☐ exclusivement généré par le compilateur g++.

1.3 Entourer la ou les affirmations exactes

On reconnait un fichier source C++ qui applique les bonnes pratiques du standard 2011/2014 si en inspectant le code on trouve :

1.3.a ☐ Un nombre important de pointeurs.

1.3.b ☐ Des objets créés sur le tas plutôt que sur la pile.

1.3.c ☐ Un nombre important de new et delete.

1.3.d ☑ L"utilisation de fonctions de la STL, par exemple std::max.

1.4 Entourer la ou les affirmations exactes

Un projet avec de nombreux fichiers sources C++ utilise généralement un fichier Makefile.

1.4.a ☐ La syntaxe du fichier Makefile fait partie du standard C++.

1.4.b ☑ Un fichier Makefile exécute une concaténation des fichiers sources.

1.4.c ☑ Un fichier Makefile décrit la dépendance temporel entre des fichiers et les actions nécessaires à prendre si un fichier source a été modifié après un fichier objet.

1.4.d ☐ Un fichier Makefile est uniquement utilisé par l'environment Qt.

1.5 Entourer la ou les affirmations exactes

Une classe abstraite est une classe qui :

1.5.a ☐ n'a aucun membre.

1.5.b ☐ a toutes ses méthodes virtuelles.

1.5.c ☑ possède au moins une méthode virtuelle pure.

1.5.d ☐ a toutes ses données membres en private.

1.6 Entourer la ou les affirmations exactes

Objet vs. Classe

1.6.a ☐ Il n"y a pas de différence entre un objet et une classe.

1.6.b ☐ Un objet est le schéma de construction d"une classe.

1.6.c ☑ Une classe définit un nouveau type.

1.6.d ☑ On dit qu"un objet est une instance d"une classe.

1.7 Entourer la ou les affirmations exactes

1.7.a ☐ Il est obligatoire de définir un destructeur dans une classe.

1.7.b ☑ L'encapsulation consiste à restreindre l'accès de certaines données dans une classe.

1.7.c ☐ Il est préférable qu"un constructeur dans une classe ait l'attribut private.

1.7.d ☑ Il est possible de créer une class dérivée de std::vector¡T¿.

1.8 Entourer la ou les affirmations exactes

Pour une classe A quelconque :

1.8.a ☐ Les membres de la classe sont toujours private.

1.8.b ☑ Il est possible que les constructeurs soient private.

1.8.c ☐ Les méthodes ne peuvent jamais łtre private.

1.8.d ☐ Il faut toujours déclarer explicitement un destructeur.

1.9 Entourer la ou les affirmations exactes

Pour une classe B dérivée de faon publique d”une classe A :

1.9.a ☐ Les membres private de la classe A sont accessibles dans B.

1.9.b ☐ Les membres protected de la classe A deviennent des membres private dans B.

1.9.c ☑ Les membres protected de la classe A restent protected dans B.

1.9.d ☐ Il faut que tous membres de la classe A soient toujours protected plutôt que private.

2 Encore Facile

2.1 Erreur 1

Le programme suivant doit afficher la somme des carrés d"un tableau d"entiers Le programme compile correctement mais donne une erreur lors de son exécution :

```
1  #include <iostream>
2  int main() {
3    int *a = new int[100];
4    // array initialization
5    for(int i = 1; i <= 100; ++i) a[i] = i * i;
6    // sum of the array
7    int sum = 0;
8    for(int i = 1; i <= 100; ++i) sum += a[i];
9
10   std::cout << SUM =  << sum << std::endl;
11   delete [] a;
12   return 0;
13 }
```

Expliquer pourquoi ce programme donne une erreur:

1. SUM = n'est pas une entree valide pour l'operateur <<.
2. De plus, même si ça ne crée pas d'erreur, il y a un problème d'indices dans les boucles.
3. Il ne faut pas commencer à 1 mais à 0, et il ne faut pas finir a 100 mais a 99.

Corriger en ré-écrivant dans les champs ci-dessus la ou les lines correctes:

1. std::cout << "SUM =" << sum << std::endl;
2.
3.

2.2 Erreur 2

Le programme suivant ne peut pas compiler :

```
1  class Point {
2    int     x_;
3    int     y_;
4    Point(int x, int y) : x_{x}, y_{y} {}
5  };
6
7  int main() {
8   ...
9   Point my_point(14, 57);
10  ...
11 }
```

Expliquez pourquoi et proposez une correction

1. Si rien n'est précisé, ici le constructeur est privé.
2.
3. public:
4. Point(int x, int y) : x_{x}, y_{y} {}

2.3 Erreur 3

Le programme ci-dessous semble correct

```
 1 #include <iostream>
 2 class Rect {
 3   int width_;
 4   int height_;
 5  public:
 6   Rect(int width, int height) :
 7     width_{width}, height_{height} {}
 8   int area() const  { return width_ * height_; }
 9 };
10 int main() {
11  Rect my_rect();
12  std::cout << my_rect.area() << std::endl;
13  return 0;
14 }
```

Mais ce programme ne compile pas, voici le message d"erreur:

```
prog.cpp: In function 'int main()' at line 12:
prog.cpp: error: request for member 'area' in 'my_rect',
  which is of non-class type 'Rect()'
  std::cout << my_rect.area() << std::endl;
                       ^
```

Expliquez pourquoi et corrigez.

1. A la ligne 11, en écrivant Rect my_rect(), le compilateur pense
2. voir une déclaration de fonction. Or nous voulons déclarer un objet de type Rect.
3. Il faut donc enlever les parenthèses ou préciser des valeurs
4. pour appeler soit le constructeur par défaut soit le constructeur de la ligne 6.
5.
6.

2.4 Erreur 4

Le programme est modifié, tout semble ok cette fois ci

```
 1 #include <iostream>
 2 class Rect {
 3   int     width_;
 4   int     height_;
 5  public:
 6   Rect(int width, int height) :
 7     width_{width}, height_{height} {}
 8   int area() const  { return width_ * height_; }
 9 };
11 int main() {
12   Rect my_rect;
13   std::cout << my_rect.area() << std::endl;
14   return 0;
15 }
```

Mais le programme ne compile toujours pas, voici le message d'"erreur :

```
prog.cpp: In function 'int main()':
prog.cpp: error: no matching function for call to 'Rect::Rect()'
  Rect my_rect;
       ^
}
```

1) Expliquez pourquoi

1. On essaye d'appeler le constructeur par défaut.
2. Or avec le constructeur de la ligne 6, le compilateur ne va pas ajouter le constructeur par défaut automatiquement.
3. Il faut donc le réécrire explicitement dans la classe:
4. Rect(): width_{0}, height_{0}{}
5. Par exemple.
6.

2) Phase à compléter : On dit que la méthode area(), qui ne peut pas modifier les membres de la classe est un

5. Par exemple.

2.5 Amélioration 1

```
 1 #include <iostream>
 2 #include <iomanip>
 3
 4 class Time {
 5  private:
 6   int h_;
 7   int m_;
 8   int s_;
 9  public:
10   void set_hour(int h) { h_ = h; }
11   void set_minute(int m) { m_ = m; }
12   void set_second(int s) { s_ = s; }
13   void print(std::ostream &os) const {
14     os
15       << std::setfill('0') << std::setw(2) << std::right << h_ << :
16       << std::setfill('0') << std::setw(2) << std::right << m_ << :
17       << std::setfill('0') << std::setw(2) << std::right << s_;
18   }
19 };
20 int main() {
21   Time t;
22   t.set_hour(11);
23   t.set_minute(34);
24   t.set_second(6);
25   t.print(std::cout);
26 }
```

On souhaite modifier les méthodes `set_hour`, `set_minute`, `set_second` pour que l"on puisse écrire

```
1 int main() {
2   Time t;
3   t.set_hour(11).set_minute(34).set_second(6);
4   t.print(std::cout);
5 }
```

Ecrire la solution pour `set_hour` uniquement

1. Time set_hour(int h) {
2. h_ = h;
3. return *this;
4. }
5.
6.

2.6 Amélioration 2

On souhaite maintenant pouvoir écrire :

```
1 int main() {
2   Time t;
3   t.set_hour(11).set_minute(34).set_second(6);
4   std::cout << t << std::endl;
5 }
```

Il faut ajouter une nouvelle fonction. Comment nomme-t-on ce type de fonction ?

Ecrire votre solution : c'est possible en 4 lignes.

```
1.
2. friend std::ostream& operator << (std::ostream& f, Time& t){
3. t.print(f);
4. return f;
5. }
6.
```

Indice : Dans du code fourni en classe, tu trouveras ton "ami" et ne pas écrire plus d"une fois l"opérateur $<<$.

3 Plus Compliqué

3.1 Influence des caches

Le temps d"accès a une donnée est directement lié à sa zone de stockage : depuis un accès très rapide en cache L1 jusqu"à un accès très lent sur le disque. Le programme ci-dessous calcule le total de toutes les données. Les deux variantes proposées sont identiques d"un point de vue fonctionnelle, mais laquelle allez-vous choisir et pourquoi ?

```
struct MyData {
 public:
  MyData() {   // this constructor initialize the data values
    for (int i = 0; i < 64; ++i) {
      values_[i] = i;
    }
  }
  int values_[64];
};
main() {
  vector<MyData> shapes(1000000);  // one million shapes
  int total = 0;
  //variant1 or variant2 goes here
  std::cout << Total is    << total << std::endl;
}
```

Variante 1: Variante 2:

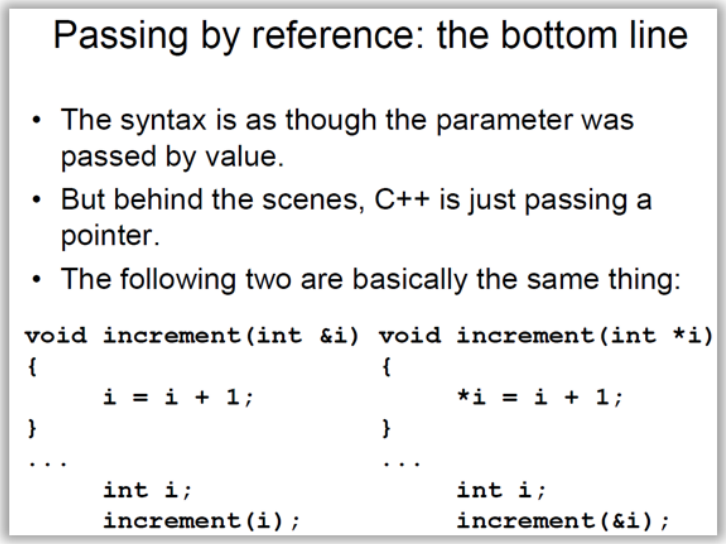
```
  for (int i = 0; i < 64; ++i) {
    for (int j = 0; j < 1000000; ++j) {
      total += shapes[j].values_[i];
    }
 }
```

```
  for (int i = 0; i < 64; ++i) {
    for (int j = 0; j < 1000000; ++j) {
      total += shapes[j].values_[i];
    }
  }
```

1. A chaque accès d'une nouvelle shapes[i], on va charger en mémoire les 64 valeurs.
2. Il est donc plus intéressant d'utiliser la variante 2 car on ne charge une nouvelle
3. shapes[i] qu'après avoir lu toute les données de la précédente.
4.

3.2 Erreur Web

J'ai trouvé sur le web un cours C++ avec la planche suivante :



Passing by reference: the bottom line

- The syntax is as though the parameter was passed by value.
- But behind the scenes, C++ is just passing a pointer.
- The following two are basically the same thing:

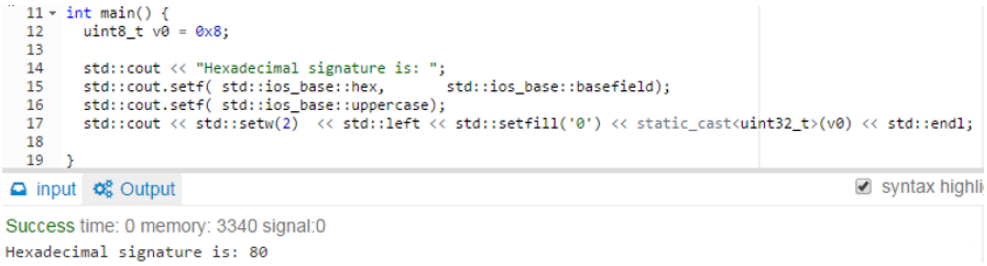
```
void increment(int &i)          void increment(int *i)
{                               {
     i = i + 1;                      *i = i + 1;
}                               }
...                             ...
     int i;                          int i;
     increment(i);                   increment(&i);
```

Il y a une erreur grossière, à vous de corriger en écrivant correctment la ligne.

1. *i = *i + 1;
2.
3.

3.3 Impression en Hexadécimal 1

J'ai aussi trouvé le code ci-dessous :



```
11 int main() {
12   uint8_t v0 = 0x8;
13
14   std::cout << "Hexadecimal signature is: ";
15   std::cout.setf( std::ios_base::hex,       std::ios_base::basefield);
16   std::cout.setf( std::ios_base::uppercase);
17   std::cout << std::setw(2)  << std::left << std::setfill('0') << static_cast<uint32_t>(v0) << std::endl;
18
19 }
```

input Output syntax highli

Success time: 0 memory: 3340 signal:0

```
Hexadecimal signature is: 80
```

Le résultat est incorrect, on attendait en fait

`Hexadecimal signature is: 08`

Les lignes 15 & 16 permettent de forcer la sortie console en base 16 pour toute impression de nombre.

Il y a un bug à la ligne 17, à vous de corriger Pour vous aider, voici les effets de std::left, sur 4 données, avec `std::setfill(''_'')`

Adjustment	width()	-42	0.12	"Q"	'Q'
left	6	-42___	0.12__	Q_____	Q_____
right	6	___-42	__0.12	_____Q	_____Q
internal	6	-___42	__0.12	_____Q	_____Q

1. il faut changer std::left par
2. std::right
3.

3.4 Constructeur

Dans le code ci-dessous, si on de-commente l'"un des 3 constructeurs, le code compile.

```
class Player {
 private:
  std::string name_;
  double score_;
 public:
// constructor 1
// Player(std::string name, double score) {
//   name_ = std::move(name);
//   score_ = score;
// }
// constructor 2
// Player(const std::string &name, double score) {
//   name_ = name;
//   score_ = score;
// }
// constructor 3
//  Player(std::string &name, double score) {
//    name_ = std::move(name);
//    score_ = score;
//  }
  void print() const {
    std::cout << Name =  << name_ <<  Score =  << score_ << std::endl;
  }
};
main(int argc, char *argv[]) {
  std::string name(argv[1]);
  Player player1(name, 34);
  player1.print();
  std::cout << ARGUMENT is  << name << std::endl;
}
```

Mais l'"exécution donne des surprises dans certains cas : parmi les 3 constructeurs possibles dans le code ci-dessus, le ou lesquels choisissez-vous ? Le ou lesquels allez-vous supprimer ? Expliquez pourquoi ? Donnez votre réponse ci-dessous pour chaque possibilités:

Choix 1. On peut garder le choix 1 parce que la copie de name se fait par passage de l'argument. en faisant le move sur name, on ne déplace qu'une copie de la variable.

Choix 2. On peut garder le choix 2 parce que le constructeur de copie du std::string fonctionne très bien pour ne copier que la valeur de name. Cette méthode est peut être plus lente de la 1.

Choix 3. On ne garde pas le choix 3 parce que on fait un move sur la variable name définie dans le main. Après la construction du player1, name sera vide et on risque d'avoir des erreurs si on considère que la variable contient encore le nom.

3.5 Amélioration 1

On trouve parfois des reliquats de code C++ antérieur à C++11 qui sont largement améliorables.

```
#include <string>
#include <iostream>
#include <vector>

class Employee {
 private:
  std::string name_;
  std::string address_;
 public:
  // constructor is not shown in full, but assume it is correct
  Employee(...) { ... }

  std::string get_name() const {
    return name_;
  }
  std::string get_address() const {
    return address_;
  }
};

std::string find_address(std::vector<Employee> emps,
                         std::string name) {
  for (std::vector<Employee>::iterator i = emps.begin();
       i != emps.end();
       i++) {
    if (i->get_name() == name) {
      return i->get_address();
    }
  }
  return \¨;
```

Proposez une implémentation améliorée et plus moderne de `find_address()`. Vous devez utiliser `const`, `&` et `auto` dans votre solution.

```
1. std::string find_address(const std::vector<Employee> &emps, const std::string &name) const {
2. for (auto emp: emps) {
3. if(emp.get_name() == name){
4. return emp.get_adress();
5. }
6. }
7. return "";
8. }
9.
```

3.6 Amélioration 2

On a créé une classe Triangle, comme ci-dessous :

```
class Triangle {
 public:
  Triangle(int color, int x0, int y0, int x1, int y1, int x2, int y2) {
    color_ = color;
    x0_ = x0;
    ...
  }
  virtual ~Triangle() = default;

  virtual void draw() {
    std::cout << Draw a triangle  << std::endl;
  }
  private:
    int color_;
    int x0_;
    ...
};
```

On ajoute une classe dérivée de `Triangle` appelée `TriangleRectangle`. Lors du copier-coller, un certain nombre de caractères et de mots clés ont disparu.

A vous de compléter pour la classe pour que ce code compile correctement. Pour vous aider, j'ai rempli partiellement quelques lignes.

```
1.  class TriangleRectangle: public Triangle {
2.   public:
3.    TriangleRectangle(int color, int x0, int y0, int x1, int y1, int x2, int y2):
4.  Triangle(color, x0, y0, x1, y1, x2, y2){}
5.
6.  void draw() override {
7.  std::cout << "Draw a TriangleRectangle" << std::endl;
8.  }
9.
10. };
```

3.7 Amélioration 3

La technique dite du delegating constructor permet de définir un constructeur à partir d"un constructeur existant

```
class X {
  int a_;
 public:
  X(int a) {  a_ = (a < 0) ? 0 : ((a > 255) ? 255 : a) ; }
  X(): X{42} {}

};
```

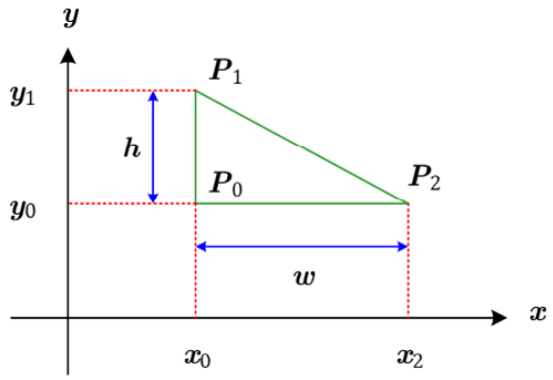
Dans l"exemple ci-dessus, X() qui appelle le constructeur X(int a), est un delegating constructor .

Complétez le code ci-dessous en écrivant un delegating constructor pour la class TriangleRectangle qui a les paramètres suivants et qui utilise le constructeur donnée en page précédente :

```
TriangleRectangle(int color, int x0, int y0, int h, int w) :
```

1. TriangleRectangle(color, x0, y0, x0, y0+h, x0+w, y0){}
2.
3.
4.
5.
6.
7.
8.

pour construire un triangle comme comme ci-dessous.



Note : Le choix de la couleur verte pour le dessin du triangle est arbitraire.

3.8 Le multi-thread

Les années précédentes, certains élèves ont écrit le code ci-dessous pour le TP yuv-viewer en multi-thread :

```
void YuvImage::yuv_to_rgb(size_t group_idx) {
  uint8_t    *y_ptr = y_raw_;
  uint8_t    *u_ptr = u_raw_;
  uint8_t    *v_ptr = v_raw_;
  static int rgb[3];

  // some code removed
  for(int y = y_min; y < y_max; ++y) {
    // some code removed
    for(int x = 0; x < x_max; ++x) {
      // some code removed
      rbg[0] = calcule_composante_rouge();
      rbg[1] = calcule_composante_bleue();
      rbg[2] = calcule_composante_verte();
      setPixel(x,y,qRgb(rgb[0], rgb[1], rgb[2]);
    }
    // some code removed
  }
  // code removed
}
```

En single thread , le programme fonctionne bien, mais dès que l"on active plusieurs threads, on obtient une image incorrecte :

Expliquez pourquoi et proposez une correction. ‘

1. La fonction n'est pas adaptée au multi_thread.
2. En effet, la fonction traite l'image dans son intégralité.
3. Donc si cette fonction est appelée dans plusieurs threads, ils vont écrire simultanément sur la même image,
4. ou essayer de convertir un pixel qui a déjà été converti.
5. Il faut donc segmenter le traitement avec par exemple:
6. int ym = thread*y_max/max_thread;
7. for(int y = y_min; y < ym; ++y) {
8.