EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

**Faculté Informatique et Communication**
Cours d'Informatique à la section SV
Sam J.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# PROGRAMMATION II

# Examen Semestre II

**Instructions :**

- Vous disposez de une heure quarante cinq minutes pour faire cet examen (8h15 - 10h).
- Nombre maximum de points: 100
- Toute documentation est autorisée, hormis les corrigés des anciens tests ;
- Répondez sur les feuilles qui vous sont distribuées à cet effet (utilisez aussi le verso des feuilles si nécessaire). **Ne répondez pas sur l'énoncé**.
- Vous pouvez répondre aux questions en français ou en anglais.
- Veillez à **ne traiter *qu'un* exercice par feuille**.
- L'examen compte 3 exercices indépendants.

  **Vous pouvez commencer par celui que vous souhaitez**
  - Exercice 1: **28** points.
  - Exercice 2: **57** points.
  - Exercice 3: **15** points.

SUJET À LA PAGE SUIVANTE

## Exercice 1 : Questions de cours [30 points]

Répondez clairement et succinctement aux questions suivantes :

① **[5 points]** Soit le programme suivant :

```
0.  class A
1.  {
2.  public:
3.      A(int unA) :a(unA) {}
4.      ~A() {}
5.
6.  private:
7.      int a;
8.  };
9.
10. class B : public A
11. {
12. public:
13.     B(int unA, int unB) : a(unA), b(unB) {}
14.
15. private:
16.    int b;
17. };
```

La ligne 13 fait émettre plusieurs messages d'erreurs au compilateur dont celui-ci :

```
heritage.cc:13:37: error: no matching function for call to ‘A::A()’
```

(a) Expliquez pourquoi ce message d'erreur est émis.

(b) Selon vous quelle autre faute le compilateur signale t-il sur cette ligne ?

(c) Comment rendre le programme compilable en ne retouchant que la ligne 13 ?

② **[4 points]** Le programme suivant signale une erreur à la compilation (on suppose que toutes les inclusions nécessaires sont faites) :

```
class C
{
   void operator-(const C& c1, const C& c2);
};
```

(a) Quelle est cette erreur ?

(b) Comment la corriger ?

**suite au verso ☞**

③ **[8 points]** Soit le programme suivant :

```
0.  #include <iostream>
1.  using namespace std;
2.
3.  class X
4.  {
5.  public:
6.      X(int unX=3) : x(unX) {}
7.
8.      virtual void m() const
9.      {
10.         cout << "<<  " << endl;
11.         cout << x      << endl;
12.     }
13.
14. private:
15.     int x;
16. };
17.
18. class Y : public X
19. {
20. public:
21.     virtual void m() const
22.     {
23.         X::m();
24.         cout << " >>" << endl;
25.     }
26. };
```

```
27. void m(const X& x)
28. {
29.     x.m();
30. }
31.
32. int main()
33. {
34.     Y y;
35.     m(y);
36.
37.     return 0;
38. }
```

(a) Qu'affiche t-il ? Justifiez brièvement.

(b) Qu'afficherait-il si :

- on supprime le mot clé `virtual` dans la classe `X` ?
- on supprime le mot clé `virtual` dans la classe `Y` ?
- Peut-on supprimer sans faute de compilation le `const` dans la classe `X` ? si oui indiquez ce que le programme affiche dans ce cas, sinon expliquez la faute de compilation.
- Peut-on supprimer sans faute de compilation le `const` dans la classe `Y` ? si oui indiquez ce que le programme affiche dans ce cas, sinon expliquez la faute de compilation.

④ **[5 points]** Dans quelles situations est-il souhaitable de définir une destructeur virtuel pur ? Donnez un exemple concret simple d'une situation où vous le feriez (réponse en français ou avec du code C++)

**suite ☞**

⑤ [**7 points**] Soit le programme suivant (on suppose que toutes les inclusions nécessaires sont faites) :

```
0.  class A
1.  {
2.  public:
3.      virtual void m() const = 0;
4.  };
5.
6.  class B : public A
7.  {
8.  public:
9.      B(size_t n) : collection(n) {
10.         for (auto& val : collection)
11.             val = new double(10.0);
12.     }
13.
14.     void m() const {
15.       for (const auto& val : collection) {
16.         if (val != nullptr)  cout << *val << endl;
17.       }
18.     }
19.
20.     ~B() {
21.       cout << "Clear" << endl;
22.       for (auto& val : collection) {
23.         delete val;
24.       }
25.     }
26.
27. private:
28.     vector<double*> collection;
29. };
30.
31. int main()
32. {
33.     A* a(new B(5));
34.     a->m();
35.     delete a;
36.
37.     return 0;
38. }
```

(a) Indiquez ce qu'il affiche.

(b) Il comporte un problème. Lequel et comment le corriger ?

(c) Quel autre problème se pose si l'on remplace le constructeur de la classe `B` par ce code :

```
B(size_t s) : collection(s, new double(10.0))
```

**suite au verso ☞**

## Exercice 2 : Conception de programme et programmation [57 points]

**Il est important de parcourir l'exercice dans son intégralité avant de commencer à l'analyser plus en détail et répondre aux questions**.

Il s'agit dans cet exercice de simuler l'évolution, en fonction des saisons, des surfaces cultivables sur des territoire ressemblant à ceci :

Un territoire est représenté au moyen d'un grille ou chaque case contient une couche minérale (sol) qui peut être couverte par une couche d'herbe, par une couche d'eau ou par les deux. La couche minérale existe toujours.

Les cases bleues sont celles ayant une quantité d'eau dépassant un seuil donné `water_treshold` (ce seuil et donné dans un fichier de configuration de type .json similaire à ce que vous avez utilisé dans votre projet: voir les contraintes de conception plus loin).

Les cases vertes sont celles contenant de l'herbe et où l'eau est présente en quantité inférieure à `water_treshold`. **La surface cultivable d'un territoire est donnée par le nombre de cellules vertes.**

Un territoire est caractérisé par la *zone climatique* à laquelle il appartient, tropicale ou tempérée, ainsi que la *saison* pendant laquelle il est simulé.

Il doit être possible de simuler les quatre saisons: automne, hiver, printemps et été.

Les fonctionnalités souhaitées sont de pouvoir:

1. Initialiser un territoire en fournissant en paramètre son climat et un nom de fichier. Ce fichier contient les données permettant d'initialiser la grille à savoir ses dimensions (vous supposerez que c'est un carré) et la nature de chaque case de la grille. La saison simulée est par défaut l'été.
2. Simuler l'évolution du territoire sur un pas de temps dt (méthode `update`).
3. Modifier la saison pour laquelle on souhaite simuler l'évolution territoire.
4. Faire chuter un volume donné `v` de pluie sur le territoire. Chaque case reçoit alors une quantité d'eau calculée en fonction de `v`. Les modalités précises du calcul ne nous intéressent pas.
5. Connaître la surface cultivable (à n'importe quel moment de la simulation)

**Zones climatiques** Les zones climatiques seront représentées au moyen de classes. Le service principal offert par ces classes est le calcul de la *température moyenne en fonction d'une saison* : `computeAverageTemp(Saison`

On distingue deux catégories de zones climatiques les *tempérées* et les *tropicales*. Pour chacune de ces catégories la calcul de la température moyenne se fait différemment. Les modalités exactes des calculs ne nous intéressent pas.

Vous considérerez qu'il n'est pas possible de définir ce calcul pour une zone climatique quelconque.

**Cellules** Comme mentionné plus haut, chaque case de la grille est la superposition d'au plus trois types de *cellules* (sol, eau, herbe). Chaque cellule a une position sur la grille (un `CellCoord` analogue à celui de votre projet).

Les *cellules d'eau* sont caractérisées par la quantité d'eau qui y est présente. Les *cellules d'herbe* par la quantité d'herbe.

La quantité d'eau présente sur une case de la grille est donc la quantité d'eau présente dans sa cellule d'eau. En cas de pluie, la quantité d'eau reçue s'ajoute à la cellule «eau» (s'il n'y en avait pas, il faut donc en créer une).

Les cellules d'eau évoluent au cours du temps en diffusant 20% de leur quantité d'eau sur les cases qui sont immédiatement ses voisines sur la grille. Si la température moyenne du territoire est supérieure à un certain seuil `evaporation_treshold` (donné par fichier de paramétrage), elle s'évapore d'une certaine quantité dépendant du temps écoulé. Vous doterez ce type de cellules d'une méthode `evaporate` chargée de simuler l'évaporation de l'eau à proprement parler. S'il n'y a plus d'eau, la cellule d'eau disparaît.

Les cellules d'herbe voient leur quantité d'herbe diminuer ou augmenter en fonction du temps écoulé, de la température moyenne et de la quantité d'eau présente sur la case correspondant à la cellule. Vous doterez ce type de cellules d'une méthode `grow` chargée du calcul de l'augmentation ou de la diminution de la quantité d'herbe. S'il n'y a plus d'herbe (quantité nulle), la cellule d'herbe disparaît.

## Question 1 : Conception [52 points]

1. on vous demande d'écrire le prototype de la (ou des) classe(s) permettant de représenter le problème ci-dessus en un programme C++ orienté objets ; **On ne vous demande pas d'écrire de programme complet, ni le corps des méthodes**;
2. les prototypes devront contenir les membres nécessaires pour mettre en oeuvre toutes les fonctionnalités souhaitées, **qu'elles soient explicitement demandées ou suggérées par les spécifications de l'énoncé** ;
3. pour les méthodes autres que les getters/setter et constructeurs/destructeurs, **vous noterez en commentaire la signification du type de retour et ce qu'elles font dans les grandes lignes**.
4. **ne négligez ni les constructeurs/destructeurs, ni les droits d'accès**.

**Voir la suite avant de répondre ☞**

SUITE DU SUJET À LA PAGE SUIVANTE

Les contraintes à respecter sont :

1. Vous n'utiliserez pas de fonctions globales (mais uniquement des méthodes de classes).
2. Le droit `protected` ne doit pas être utilisé pour les attributs.
3. Votre conception ne doit pas nécessiter de dupliquer du code.
4. Vous supposerez que les classes fournies dans le projet sont à votre disposition: tous les utilitaires(`CellCoord` etc.) ainsi qu'un noyau de simulation analogue la classe `Application`.
5. Vous supposerez que la classe `Application` fournit la fonction `getAppEnv()` qui retourne le territoire simulé. Vous supposerez aussi l'existence de la fonction `getAppConfig()` permettant l'accès aux fichiers de configuration de type `.json`.
6. Les méthodes permettant l'évolution d'un élément au cours du temps auront toutes pour prototype `void update(sf::Time dt)`
7. Il n'est pas nécessaire de vous préoccuper des méthodes de dessin.
8. Il n'est pas nécessaire de préciser les inclusions C++.
9. Vous porterez une attention particulière à la nature des méthodes (normales, const, virtuelles ou virtuelles pures), au liens d'héritage ainsi qu'aux droits d'accès aux attributs et aux méthodes.

### Question 2 : Programmation [5 points]

Donnez le corps de la méthode `update` des cellules d'eau.

**suite au verso ☞**

## Exercice 3 : Déroulement de programme [15 points]

Le programme suivant compile et s'exécute sans erreur. Qu'affiche-t-il ? Justifiez votre réponse en expliquant les points *importants* (ne paraphrasez pas le code, mais montrez que vous avez compris !).

```
0.  #include <iostream>
1.  #include <vector>
2.  using namespace std;
3.
4.  class A {
5.  public:
6.     A(int x = 2, int y = 3) : a(x), b(y) {}
7.
8.     int m1() const { return 2*a; }
9.
10.    virtual int m2() const = 0;
11.
12.    virtual int m3() const {
13.      return b + m1() + m2();
14.    }
15.
16.    int getA() const { return a; }
17.    int getB() const { return b; }
18.    virtual ~A() {}
19.
20. private:
21.    int a;
22.    int b;
23. };
24.
25. class B : virtual public A {
26. public:
27.    B(int x = 3, int y = 4) : A(x,y) { b = 3; }
28.    int m1() const { return b*getA(); }
29.    int m2() const { return 5*getB(); }
30. private:
31.    int b;
32. };
33.
34. class C : virtual public A {
35. public:
36.    C(int x = 2, int y = 3, int z = 5)
37.       : A(x,y), c(z) {}
38.    int m1() const { return  10*getA(); }
39.    int m2() const { return  3*getB()+c; }
40. private:
41.    int c;
42. };
43.
44. class D : public B, public C {
45. public:
46.    int m1() const { return B::m1(); }
47.    int m2() const { return C::m2(); }
48. };
```

```
49. int main() {
50.    unsigned int k(0);
51.    A* pa;
52.
53.    B b;
54.    pa = &b;
55.    cout << ++k << ") " << b.m1()   << endl;
56.    cout << ++k << ") " << b.m2()   << endl;
57.    cout << ++k << ") " << pa->m1() << endl;
58.    cout << ++k << ") " << pa->m2() << endl;
59.    cout << ++k << ") " << pa->m3() << endl;
60.    cout << "@@@@@" << endl;
61.    C c;
62.    pa = &c;
63.    cout << ++k << ") " << c.m1()   << endl;
64.    cout << ++k << ") " << c.m2()   << endl;
65.    cout << ++k << ") " << pa->m1() << endl;
66.    cout << ++k << ") " << pa->m2() << endl;
67.    cout << ++k << ") " << pa->m3() << endl;
68.    cout << "@@@@@" << endl;
69.    D d;
70.    pa = &d;
71.    B* pb(&d);
72.    cout << ++k << ") " << d.m1()   << endl;
73.    cout << ++k << ") " << d.m2()   << endl;
74.    cout << ++k << ") " << pa->m1() << endl;
75.    cout << ++k << ") " << pa->m2() << endl;
76.    cout << ++k << ") " << pb->m1() << endl;
77.    cout << ++k << ") " << pb->m2() << endl;
78.    cout << ++k << ") " << pa->m3() << endl;
79.    cout << ++k << ") " << pb->m3() << endl;
80.    cout << "@@@@@" << endl;
81.    b = d;
82.    pa = &b;
83.    cout << ++k << ") " << b.m1()   << endl;
84.    cout << ++k << ") " << b.m2()   << endl;
85.    cout << ++k << ") " << pa->m1() << endl;
86.    cout << ++k << ") " << pa->m2() << endl;
87.    cout << ++k << ") " << pa->m3() << endl;
88.    cout << "@@@@@" << endl;
89.   return 0;
90. }
```