HE2B ESI

# Langage C++
# Examen de seconde session

/40

R. Absil (abs)

23 août 2017

## Consignes générales

1. Rédigez vos réponses en français correct : soignez votre orthographe et votre grammaire.
2. Justifiez chacune de vos réponses de façon complète et concise : une réponse incorrecte justifiée avec de bons arguments peut rapporter quelques points, alors qu'une bonne réponse non justifiée sera systématiquement comptée comme nulle.
3. L'examen est à cahier fermé.
4. L'examen dure 30 minutes.

**Question 1.** On souhaite que les codes suivants produisent l'affichage souhaité. Si ce n'est pas le cas, corrigez chacun de ces codes pour qu'ils produisent sans erreur l'affichage souhaité. Dans tous les cas, *justifiez également* les raisons pour lesquelles l'affichage souhaité apparaît ou non. /10

Afin de produire le comportement souhaité, vous ne pouvez modifier que la partie « définition » des instructions ou des prototypes de variables et de fonctions, à moins qu'il n'y ait pas d'autre choix. Cela signifie entre autres que vous ne pouvez pas arbitrairement supprimer des instructions pour produire le comportement désiré, ou modifier le type des paramètres ou du retour d'une fonction s'il existe une autre solution. Vous pouvez, en revanche, ajouter des prototypes entiers de fonctions, se cela est nécessaire.

Remarques :

— ces codes peuvent ne pas compiler ;
— ces codes ne comprennent que des erreurs sémantiques et / ou de logique (ne cherchez donc pas un « `;` » ou un « `std::` » manquant) ;
— il est possible qu'il soit impossible de produire l'affichage souhaité, si tel est le cas, justifiez pourquoi.

## Code 1

```
1  #include <iostream>
2  using namespace std;
3
4  class A
5  {
6    public:
7      void f(int n) {cout << "A::integer " << n << endl; }
8      void f(char n) {cout << "A::character " << n << endl; }
9  };
10
11 class B : public A
12 {
13   public:
14     void f(int n, int m) { cout << "B::integers " << n << " " << m << endl; }
15 };
16
17
18 int main()
19 {
20   int n = 1;
21   char c = 'a';
22   B b;
23   b.f(n);
24   b.f(c);
25   b.f(n, c);
26 }
```

## Affichage souhaité du Code 1

```
1  A::integer
2  A::character
3  B::integers
```

## Affichage produit par le code 1

```
1  Erreur de compilation
```

**Code 2**

```
1  #include <iostream>
2
3  int f() {
4    static int i { 4 };
5    return --i;
6  }
7
8  int main()
9  {
10   std::cout << f() << std::endl;
11   while (f()) { }
12   std::cout << f() << std::endl;
13 }
```

**Affichage souhaité du Code 2**

```
1  4
2  -1
```

**Affichage produit par le code 2**

```
1  3
2  -1
```

## Code 3

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Integer
6  {
7    unsigned i;
8    bool positive;
9
10   public:
11     Integer(unsigned i, bool positive = true) : i(i), positive(positive) {}
12     Integer operator +(Integer a)
13     {
14       if(positive && a.positive)
15         return Integer(i + a.i);
16       else if(positive && !a.positive)
17         return Integer(i - a.i);
18       else if(!positive && a.positive)
19         return Integer(a.i - i);
20       else
21         return Integer(-i - a.i);
22     }
23
24     void print()
25     {
26       if(!positive)
27         cout << "-";
28       cout << i << endl;
29     }
30 };
31
32 int main()
33 {
34   Integer a1(2u);
35   Integer a2(3u,false);
36
37   Integer s = a1 + a2;
38   s.print();
39 }
```

## Affichage souhaité du Code 3

```
1  -1
```

## Affichage produit par le code 1

```
1  4294967295
```

## Code 4

```
1  #include <iostream>
2  using namespace std;
3
4  class exceptA {};
5  class exceptB : public exceptA {};
6  class exceptC : public exceptB {};
7
8  void f() {   throw exceptB(); }
9
10 int main()
11 {
12   try
13   {
14     f();
15   }
16   catch(exceptA& e)
17   {
18     cout << "I caught an A" << endl;
19   }
20   catch(exceptB& e)
21   {
22     cout << "I caught a B" << endl;
23   }
24   catch(exceptC& e)
25   {
26     cout << "I caught a C" << endl;
27   }
28   catch(...)
29   {
30     cout << "I caught something" << endl;
31   }
32 }
```

## Affichage souhaité du Code 4

```
1  I caught a B
```

## Affichage produit par le code 4

```
1  I caught an A
```

## Code 5

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  class A
7  {
8    public:
9      int i;
10     A(int i) : i(i) {}
11     void print() { cout << i << "␣"; }
12 };
13
14 int main()
15 {
16   vector<A> v(5);
17   for(int j = 0; j < v.size(); j++)
18     v[j].i = j;
19   for(A a : v)
20     a.print();
21 }
```

## Affichage souhaité du Code 5

```
1  0 1 2 3 4
```

## Affichage produit par le code 5

```
1  Erreur de compilation
```

**Code 6**

```
1  #include <iostream>
2  using namespace std;
3  class A {
4    public:
5      A() { cout << "+A" << endl; }
6      A(const A& a) { cout << "rA" << endl; }
7      virtual ~A() { cout << "-A" << endl; }
8  };
9  class B {
10   public:
11     B() { cout << "+B" << endl; }
12     B(const B& b) { cout << "rB" << endl; }
13     virtual ~B() { cout << "-B" << endl; }
14 };
15 class C : public A, public virtual B {
16   public:
17     C() { cout << "+C" << endl; }
18     C(const C& c) { cout << "rC" << endl; }
19     virtual ~C() { cout << "-C" << endl; }
20 };
21
22 void f(A a) {}
23
24 int main() {
25   C c; cout << endl;
26   f(c); cout << endl;
27   C * cc = new C(); cout << endl;
28   delete cc; cout << endl;
29 }
```

**Affichage souhaité du Code 6**

```
1  +A   rA   +A   -C   -C
2  +B   -A   +B   -B   -B
3  +C        +C   -A   -A
```

**Affichage produit par le code 6**

```
1  +B   rA   +B   -C   -C
2  +A   -A   +A   -A   -A
3  +C        +C   -B   -B
```

**Remarque 1.** Les affichages ci-dessus sont à lire en colonnes, de haut en bas et de gauche à droite.

**Code 7**

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7    unsigned i = 5;
8    while(i >= 0)
9    {
10     if(i != 0)
11       cout << i << endl;
12     i--;
13   }
14   cout << "BOOM" << endl;
15 }
```

**Affichage souhaité du Code 7**

```
1 5
2 4
3 3
4 2
5 1
6 BOOM
```

**Affichage produit par le code 7**

```
1 Boucle infinie
```

**Code 8**

```
1  #include <iostream>
2
3  void swap(int * i, int * j)
4  {
5    int * tmp = i;
6    i = j; j = tmp;
7  }
8
9  int main()
10 {
11   int i = 2; int j = 3; swap(&i, &j);
12   cout << i << "␣" << j << endl;
13 }
```

**Affichage souhaité du Code 8**

```
1  3 2
```

**Affichage produit par le code 8**

```
1  2 3
```

**Code 9**

```
#include <iostream>
class A {};
class B {
  A * a;
  public:
    B() : a(new A()) {}
    ~B() { delete a; }
    void print() {  cout << "Get to the choppaaa" << endl;  }
};

void f(B b) { std::cout << "You have no respect for logic" << std::endl; }

int main()
{
  B b;
  f(b);
  b.print();

  B bb;
  bb = b;
  bb.print();

  bb = bb;
  bb.print();
}
```

**Affichage souhaité du Code 9**

```
You have no respect for logic
Get to the choppaaa
Get to the choppaaa
Get to the choppaaa
```

**Affichage produit par le code 9**

```
Erreur de segmentation
```

### Code 10

```
#include <iostream>
#include <list>

int main()
{
  std::list<int> l;
  for(int i = 0; i <= 6; i++)
    l.push_back(i * i - i);
  for(int i = 0; i < l.size(); i++)
  {
    auto j = l.rend();
    std::cout << *j << " ";
    l.pop_back();
  }
}
```

### Affichage souhaité du Code 10

```
30 20 12 6 2 0 0
```

### Affichage produit par le code 10

```
30 20 12 6
```

### Code 11

```
#include <iostream>

class A
{
  int i;
  public:
    A operator +(int i) { std::cout << "For Maccrage we march" << std::endl; }
    A operator +(A a) { std::cout << "and we shall know no fear" << std::endl; }
};

int main()
{
  int i1; A a1; A a2;
  i1 + a1 + a2; //vous ne pouvez pas changer cette ligne
}
```

### Affichage souhaité du Code 11

```
For Maccrage we march
and we shall know no fear
```

### Affichage produit par le code 11

```
Erreur de compilation
```

**Code 12**

```
1  #include <iostream>
2
3  template<class T> class Brol
4  {
5    public:
6      T & t; int & i;
7      Brol(T & t) { this-> t = t; }
8      Brol(int & i) { this-> i = i; }
9      void print() { std::cout << t << " " << i << std::endl;}
10 };
11
12 int main()
13 {
14   Brol<int> b(2);    b.t = 3;
15   b.print();
16 }
```

**Affichage souhaité du Code 12**

```
1  3 2
```

**Affichage produit par le code 12**

```
1  Erreur de compilation
```

## Code 13

```
#include <iostream>
#include <list>
#include <algorithm>

class MyList
{
  std::list<int> l;
  public:
    MyList(int i = 0) : l(std::list<int>(i)) {}
    void add(int i) { l.push_back(i); }
    void print()
    {
      for(int i : l)
        std::cout << i << " ";
      std::cout << std::endl;
    }
};

int main()
{
  std::list<MyList<int>> list(5);
  for(auto it = list.begin(), int n = 4; it != list.end(); it++, n--)
  {
    MyList<int> l;
    for(int i = n; i >= 0; i--)
      l.add(i);
    *it = l;
  }
  std::sort(list.begin(), list.end());//trie les listes par taille croissante
  for(MyList<int> l : list)
    l.print();
}
```

## Affichage souhaité du Code 13

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

## Affichage produit par le code 13

```
Erreur decompilation
```

**Code 14**

```
#include <iostream>

class A
{
  protected:
    int i;
  public:
    A(int i = 0) : i(i) {}
    void print() { std::cout << i << std::endl; }
};

class B : public A
{
  int j;
  public:
    B(int i = 0) : A(i + 2), j(i + 3) {}
    void print() { std::cout << (i + j) << std::endl; }
};

int main() { A a = B(); a.print(); }
```

**Affichage souhaité du Code 14**

```
5
```

**Affichage produit par le code 14**

```
2
```

**Code 15**

FICHIER A.CPP

```
#ifndef A_H
#define A_H
#include <iostream>
#include "B.cpp"

class A
{
  public:
    B b;
    A() {}
    void print()
    {
      std::cout << "A with B" << std::endl;
    }
};
#endif
```

FICHIER B.CPP

```
#ifndef B_H
#define B_H
#include <iostream>
#include "A.cpp"

class B
{
  public:
    A a;
    B() {}
    void print()
    {
      std::cout << "B with A" << std::endl;
    }
};
#endif
```

FICHIER ABMAIN.CPP

```
#include <iostream>

using namespace std;

int main()
{
  B b;
  A a;
  a.b = b;
  b.a = a;
  a.print();
  b.print();
}
```

**Affichage souhaité du Code 15**

```
A with B
B with A
```

**Affichage produit par le code 15**

```
Erreur de compilation
```

**Code 16**

FICHIER A.CPP

```cpp
#include <iostream>

class B;

class A
{
  public:
    B * b;
    A() : b(nullptr) {}
    void print()
    {
      std::cout << "A_with_B" << std::endl;
    }
};
```

FICHIER B.CPP

```cpp
#include <iostream>

#include "A.cpp"

class B
{
  public:
    A a;
    B() {}
    void print()
    {
      std::cout << "B_with_A" << std::endl;
    }
};
```

FICHIER ABMAIN.CPP

```cpp
#include <iostream>

using namespace std;

int main()
{
  B b;
  A a;
  a.b = &b;
  b.a = a;
  a.print();
  b.print();
}
```

**Affichage souhaité du Code 16**

```
A with B
B with A
```

**Affichage produit par le code 16**

```
Erreur de compilation
```

## Code 17

```
1  #include <iostream>
2  using namespace std;
3  class A {
4    public:
5      A() { cout << "+A "; }
6      A(const A& a) { cout << "rA "; }
7      ~A() { cout << "-A "; }
8      A& operator =(const A& a) { cout << "=A " << endl; return *this; }
9  };
10
11 void f(A a) {}
12 void g(A& a) {}
13
14 int main()
15 {
16   A a;
17   f(a);
18   A * aa = new A();
19   aa = &a;
20   g(*aa);
21 }
```

## Affichage souhaité du Code 17

```
1  +A rA -A +A -A -A
```

## Affichage produit par le code 17

```
1  +A rA -A +A -A
```

**Code 18**

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    int a1 = 2.1;
7    cout << a1 << " ";
8    int a2(2.1);
9    cout << a2 << " ";
10   int a3{2.1};
11   cout << a3 << endl;
12 }
```

**Affichage souhaité du Code 18**

```
1  2 2 2
```

**Affichage produit par le code 18**

```
1  Erreur de compilation
```

**Question 2.** Décrivez les différentes classes d'allocations de variables en `C++`, c'est-à-dire, entre autres, /10

- en quoi elles se différencient d'un point de vue gestion de la mémoire,
- comment les mettre en œuvre (mots clés associés),
- la durée de vie, portée, création et destruction de la mémoire allouée,
- les mécanismes que certaines de ces classes rendent possibles, etc.

**Question 3.** Expliquez la manière dont les chaînes de caractères sont mises en œuvre en `C` pur (sans les mécanismes inhérents au `C++`), ainsi que la façon de les manipuler. /10

**Question 4.** Vous devez stocker dans un conteneur 100 000 objets de type `A`. Pour l'application que vous développez, vous serez amenés à régulièrement extraire (obtenir sans supprimer) des éléments à des positions arbitraires de ce conteneur. Sur quel type de conteneur votre choix se porterait-il ? *Justifiez rigoureusement votre réponse.* /10

**Question 5.** Expliquez sommairement le principe de surcharge d'opérateur en `C++`. En particulier, détaillez /10

— les mots-clé associés,
— les limites de ce mécanisme (ce que le langage ne permet pas de faire).

**Question 6.** Expliquez en détail le mécanisme d'inférence de type (comment il est mis en œuvre, à quoi sert-il, les mots-clé associés). /10

**Question 7.** Qu'est-ce que le mécanisme de la liste d'initialisation et dans quels cas ne peut-on pas s'en passer ? /10

**Question 8.** Détaillez comment mettre en œuvre des conversions implicites définies par l'utilisateur dans les cas suivants : /10

— d'un type de base vers un autre type de base,
— d'un type de base vers une classe,
— d'une classe vers un type de base,
— d'une classe vers une autre classe.

**Question 9.** Expliquez l'utilité du mot-clé **virtual**, à la fois dans l'entête d'une classe comme dans l'entête d'une fonction. /10

**Question 10.** Expliquez l'usage des mots clés **const** et constexpr. /10

**Question 11.** Expliquez la différence entre la ligature statique et la ligature dynamique des liens. /10

**Question 12.** Expliquez les différences entre pointeurs et références, entre autres d'un point de vue utilisation que d'un point de vue gestion de la mémoire. /10

**Question 13.** Illustrez les différences de fonctionnement en matière d'allocation mémoire entre `std::vector` et `std::list` dans le cas d'ajout d'éléments à une position arbitraire du conteneur. /10

**Question 14.** Expliquez les différences entre les différents types de conteneurs associatifs de la librairie standard. /10

**Question 15.** Expliquez en quoi l'utilisation de templates permet « d'émuler » le polymorphisme, comme par exemple dans la fonction `std::sort` qui peut à la fois prendre un `std::vector` comme un `std::list` alors que ces classes n'ont pas de superclasse commune. Détaillez les avantages et inconvénients d'une telle pratique. /10

**Question 16.** Expliquez les différences entre /10

- `A a(2);`,
- `A a {2};`,
- `A a = {2};`,
- `A a = 2;`,

ainsi que ce qui est requis dans le prototype de `A` pour que ces instructions fonctionnent correctement.