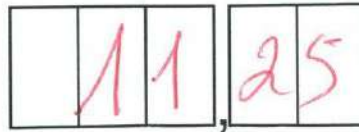


NOM GAUTIER
Prénom Arthur
Promo 2018
Date 13/05/2015



GAUTIER Arthur
L2 - 2014

MATIERE Programmation en C++

I/

1)

a = 7

a = 10

x = 2

y = 13

2) void exercice_references()

{

int a = 7;

int b = 27;

int x = 2;

int y = 3;

int &ref = a;

cout << "a = " << p << endl;

p = p + y;

cout << "a = " << p << endl;

~~p = x;~~

cout << "x = " << p << endl;

y = y + 5 * (p)

cout << "y = " << y << endl

}

1

II) musicien.cpp

1) Musicien (Instrument a-instrument, Date a-naissance, string a-nom):
Instrument (a-instrument), Date (a-naissance)

```
{  
    a-nom = strdup(a-nom);  
}
```

2) Musicien Copie (const Instrument & a-instrument, const Date & a-naissance,
const string & a-nom)

```
{  
    this->a-instrument = a-instrument;  
    this->a-naissance = a-naissance;  
    this->a-nom = strdup(a-nom);  
    return this;  
}
```

3/a) void Musicien Exchange (Musicien & m1, Musicien & m2)

```
{  
    Instrument tmp;  
    tmp = Instrument Copie (m1.Instrument);  
    m1.Instrument = m2.Instrument;  
    m2.Instrument = tmp;  
}
```

= A ex p l i q u e

0.5/1

b) ~~galerich~~ Swap Instrument (Musicien & m)

void SwapInstrument (Musicien & m1, Musicien & m2)

{

Instrument tmp;

tmp = Instrument::Instrument Copie (Musicien::m1.a_instrument);

Musicien::m1.a_instrument = Musicien::m2.a_instrument;

Musicien::m2.a_instrument = tmp;

}

found?

0.5/1

/

~~1) anke.h~~

1) class Anke

{

private:

int a_taille;
int a_age;

public:

Anke (int a_taille = 175, int a_age = 20)

{

this->a_taille = a_taille;
this->a_age = a_age;

}

void AfficheAnke (Anke a)

LL

{

cout << "(" << a.a_taille << " " << a.a_age << ")" << endl;

}

1/1

1/1

0.5/2

Anbre Geten Anbre (Anbreha, m¹ a-taille : 175, int a-age = 20)

{

cout << " Taille : " << a. a-taille << endl;

cout << " Age : " << a. a-age << endl;

if (this->a-taille != a. a-taille)

{

a. a-taille = this->a-taille;

cout << " Nouvelle taille " << a. a-taille << endl;

}

if (this->a-age != a. a-age)

{

a. a-age = this->a-age;

cout << " Nouvel age " << a. a-age << endl;

}

return a

}

0/2

HORS

Sujet

a) Il n'est pas nécessaire de définir un constructeur de copie car la classe a des attributs très simples à initialiser.

b) Il n'est pas nécessaire de définir une surcharge de l'opérateur d'affectation car on travaille avec une classe dont les attributs ont déjà un opérateur d'affectation existant.

c) Le destructeur étant automatiquement défini par le programme, il n'est pas nécessaire dans ce cas.

5/5

NOM GAUTIER
Prénom Arthur
Promo 2018
Date 13/05/2015

MATIÈRE C++

III) 2)

```
ArbreTest.cpp  
#include <arbre.h>  
int main  
{
```

```
    Arbre a = Arbre();  
    Arbre* b = new Arbre(180, 21);  
    b->a-age = 23;  
    AfficherArbre(a);  
    AfficherArbre(*b);  
    return 0;
```

```
}
```

IV)

musicien.cpp

1) Musicien (const char* a_nom, const int a_naissance, const int a_mort,
const char* a_instrument)

```
{
```

```
    this->a_nom = strdup(a_nom);  
    this->a_naissance = a_naissance;  
    this->a_mort = a_mort;  
    this->a_instrument = strdup(a_instrument);  
    return this;
```

```
}
```

②

15/15

2/4

1/1

2) Il n'est pas nécessaire de définir un constructeur de copie car les attributs sont ~~simples~~ à initialiser. ~~Not char *~~ 0/1

3) Il n'est pas nécessaire de surcharger l'opérateur d'affectation car les attributs sont ~~simples~~ à initialiser. 0/1

4) Il est nécessaire de définir le destructeur afin de libérer proprement les char *

~ Musicien()

{

if (a-nom != null)
{
delete(a-nom);

}

if (a-instrument != null)
{
delete(a-instrument);

}

}

1/1

II

Une classe abstraite est une classe dont on ne peut pas créer d'instances. Pour en créer une, il faut que la classe contienne une méthode virtuelle pure.

05/05

VI)

Le polymorphisme est le fait de redéfinir dans une classe héritée des fonctions de la classe de laquelle elle hérite. Le compilateur saura instinctivement quelle fonction appeler.

Exemple!!

0.25/1

