ÔN TẬP THI KẾT THÚC MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG IT002

Thời gian: 60 phút – 50 câu trắc nghiệm online

(SV không được dùng tài liệu và phải sử dụng Safe Exam Browser trong thời gian thi)

NỘI DUNG 1: CƠ BẢN OOP

❖ Chương 2: Tổng quan về Lập trình hướng đối tượng

1. Có các phương pháp lập trình nào?
2. Lập trình hướng đối tượng: các khái niệm cơ bản, các đặc điểm quan trọng, các ưu điểm của OOP.
3. Mục tiêu của việc thiết kế phần mềm là gì?
4. Trình bày thứ tự các bước khi phân tích hướng đối tượng.

❖ Chương 3: Lớp và đối tượng
Chương 3: Đối tượng và Lớp đối tượng (tham khảo)
Chương 4: Lớp và đối tượng – Một số vấn đề liên quan

1. Đối tượng: khái niệm, khai báo, con trỏ, cấp phát bộ nhớ.
2. Đối tượng hằng và phương thức hằng: tính chất, cách khai báo.
3. Dữ liệu tĩnh và phương thức tĩnh.
4. Hàm thành phần: khai báo, cài đặt, cách gọi hàm, phân loại.
5. Con trỏ this: khái niệm, vai trò, cách dùng.
6. Hàm tạo/Hàm dựng/Hàm thiết lập (constructor): khái niệm, cú pháp, phân loại (default constructor, parameterized constructor, copy constructor), cách sử dụng. Đối tượng có khả năng tự khởi động trong các trường hợp nào?
7. Hàm hủy (destructor): khái niệm, cách viết, cách sử dụng.
8. Hàm bạn: khái niệm, cú pháp, cách sử dụng.
9. Lớp bạn: khái niệm, cú pháp, cách sử dụng.
10. Tổ chức file của class: header file (.h) và source file (.cpp).
11. Khác nhau giữa class và struct?
12. Một số dạng câu hỏi trắc nghiệm có source code:

    a. *Phạm vi truy cập:*

    Các đối tượng dta và dtb truy cập được các biến nào của lớp A?

```
class A {
private:
    int a;
public:
    int b;
};
class B {
public:
    A dta;
};
class C {
private:
    A dtb;
};
```

Khi chạy đoạn chương trình sau sẽ xảy ra hiện tượng gì?

```
#include <iostream>
using namespace std;
class A {
private:
    int a;
};
class B {
public:
    A dta;
    void Nhap() { cout << "a = "; cin >> dta.a; }
};
void main()
{
    B dtb;
    dtb.Nhap();
}
```

b. Cho biết kết quả các chương trình sau:

```
#include <iostream>
using namespace std;
class Test {
public: Test() { cout << "Goi ham thiet lap\n"; }
      ~Test() { cout << "Goi ham huy \n"; }
};
int main() {
    for (int i = 0; i <= 1; i++)
        Test t;
    return 0;
}
```

```
#include <iostream>
using namespace std;
class Test {
public:
    Test() {}
    Test(const Test& Test) { cout << "a"; }
    Test& operator=(const Test& Test) { cout << "b"; return *this; }
};
void  main()
{
    Test t1, t2, t3;
    Test t4(t1 = t2 = t3);      b b a
    Test t5 = t4 = t3;          b a
}
```

```cpp
#include <iostream>
using namespace std;
class T {
private:
    int t;
public:
    T(int t = 0) : t(5)
    {
        this->t = t;
    }
    int GetSet(int tt = 0)
    {
        t = (tt == 0) ? t : tt;
        return t + 1;
    }
};
int main()
{
    T a, b(7);
    T c(a.GetSet(b.GetSet()));
    cout << a.GetSet() << b.GetSet() << c.GetSet();
    return 0;
}
```

9 8 10

```cpp
#include <iostream>
using namespace std;
class T {
public:
    static int i;
    T() { i++; }
    T(int) {}
    ~T() { i--; }
};
int T::i = 0;
int main()
{
    T* a, * b, c, d, e;
    a = new T();
    b = new T(5);
    delete (a);
    cout << T::i << endl;
    //delete (b);
    //cout << T::i << endl;
    return 0;
};
```

3

3

```cpp
#include <iostream>
using namespace std;
class Lop1 {
    int a = 10;
    friend class Lop2;
};
class Lop2 {
  public:
    void HienThi(Lop1 &l1) {
        cout << "Gia tri cua a la: "<< l1.a;
    }
};
int main() {
    Lop1 l1;
    Lop2 l2;
    l2.HienThi(l1);
    return 0;
}
```

Gia tri cua a la: 10

NỘI DUNG 2: OPERATOR OVERLOADING

❖ Chương 5 – Overload toán tử

1. Toán tử, toán hạng, biểu thức: khái niệm, cú pháp overload toán tử.
2. Xây dựng lớp Phân số: phương thức thiết lập (mặc định, truyền tử số, truyền tử số và mẫu số, gán giá trị mặc định), overload các toán tử + - * / (hàm thành phần, hàm friend) chuyển kiểu, >> <<, sự nhập nhằng giữa Phân số + Số (chuyển Phân số sang số thực hay chuyển Số sang Phân số? => chuyển kiểu tường minh), ++ -- (phân biệt dạng tiền tố và hậu tố của toán tử bằng tham số giả, xây dựng phương thức, cách gọi phương thức).
3. Các toán tử ép kiểu trong C++.
4. Các toán tử có thể và không thể overload được.
5. Một số dạng câu hỏi trắc nghiệm có source code:
    a. *Phương thức sau có chức năng gì?*

```cpp
PHANSO operator++(int) {
    PHANSO ps = *this;
    tuso += mauso;
    return ps;
}
```

b. Cho biết kết quả thực hiện của chương trình sau (sửa các lỗi sai nếu có):

```
115  class PHANSO {
116  private:
117      int tuso, mauso;
118  public:
119      PHANSO(int tuso = 0, int mauso = 1) { this->tuso = tuso; this->mauso = mauso; }
120      friend PHANSO operator+(PHANSO ps1, PHANSO ps2) {
121          return PHANSO(ps1.tuso * ps2.mauso + ps2.tuso * ps1.mauso, ps1.mauso * ps2.mauso);
122      }
123      operator float() { return (float)tuso / mauso; }
124      friend ostream& operator<<(ostream& os, PHANSO ps) {
125          os << ps.tuso << "/" << ps.mauso;
126          return os;
127      }
128  };
129  void main() {
130      PHANSO ps1(1,2);
131      cout << "A = ps1 + 2 = " << ps1 << " + 2 = " << ps1 + 2 << endl;
132  }
```

```
class T {
private:
    int t;
public:
    T(int tt) :t(tt) {}
    friend ostream& operator << (ostream& os, const T& t) {
        os << t.t << endl;
        return os;
    }
};
void main() {
    T ttt(7);
    cout << ttt;                    7
    operator<<(cout, ttt);          7
}
```

```
class T {
public:
    int t;
    T(int tt = 0) : t(tt) {};
    int& operator[](int tt) { cout << "#"; return t; }
    int operator[](int tt) const { cout << "*"; return t; }
};
void xuat(const T& a) { int tam = a[5]; }
void main() {
    T a(7);
    xuat(a);                *##
    a[3] = 9;
    int tam = a[5];
    xuat(tam);              *
}
```

```
class SoPhuc {
private:
    float thuc;
    float ao;
public:
    SoPhuc(float t = 0, float a = 0): thuc(t), ao(a){}
    SoPhuc operator ()(float t, float a) {
        thuc += t;
        ao += a;
        return *this;
    }
    SoPhuc operator() (float t) {
        thuc += t;
        return *this;
    }
    void xuat() {
        cout << "(" << thuc << ", " << ao << ")" << endl;
    }
};
void main() {
    SoPhuc sp1(5.9, 3.7), sp2;
    sp2 = sp1(0.1);
    sp2(1.9, 2.6);
    cout << "sp1 = "; sp1.xuat();      (6, 3.7)
    cout << "sp2 = "; sp2.xuat();      (7.9, 6.3)
}
```

```
class Test {
private:
    int t;
public:
    Test() { t = 0; }
    Test(int tt) :t(tt) {}
    Test(const Test& t1) { t = t1.t; }
    Test operator + (const Test& t1) const { return Test(this->t + t1.t); }
    Test operator = (const Test& t1) { t = t1.t; return *this; }
    int Get_t() { return t; }
};
ostream& operator << (ostream& os, const Test& te) {
    os << te.Get_t() << " " << endl;
    return os;
}
void main() {                5
    Test t2(5);
    cout << t2;
}
```

NỘI DUNG 3: KẾ THỪA

- ❖ **Chương 6 – Dẫn xuất và thừa kế**
- ❖ **Chương 5 – Kỹ thuật thừa kế (tham khảo)**

1. Quan hệ giữa các lớp đối tượng: mối quan hệ liên kết Has-A (bao hàm/bộ phận – toàn thể) và mối quan hệ kế thừa Is-A.
   a. Mối quan hệ liên kết Has-A: quan hệ một một (1-1), quan hệ một nhiều (1-n), quan hệ nhiều nhiều (n-n). Cho ví dụ.
   b. Mối quan hệ kế thừa Is-A: quan hệ đặc biệt hóa – tổng quát hóa. Cho ví dụ.
2. Lớp cơ sở và lớp dẫn xuất: khái niệm, cách khai báo, các kiểu dẫn xuất, phạm vi truy xuất (truy xuất theo chiều dọc, truy xuất theo chiều ngang).

Lớp cơ sở	DX private	DX protected	DX public
Thành phần **private**	-	-	-
Thành phần **protected**	private	protected	protected
Thành phần **public**	private	protected	public

3. Thừa kế: các loại thừa kế, thừa kế thuộc tính, thừa kế phương thức, các hàm thành phần nào trong lớp cơ sở không được thừa kế trong lớp dẫn xuất?
4. Trong tất cả các lớp có quan hệ thừa kế, các hàm thiết lập / hủy bỏ sẽ được thực hiện theo thứ tự nào theo phân cấp kế thừa?
5. Một số dạng câu hỏi trắc nghiệm có source code:
   a. Lớp Test2 dưới đây có thể truy xuất đến các biến thành phần nào của lớp Test?

```
class Test
{
private:
    int t1;
public:
    int t2, t3;
protected:
    int t4, t5, t6;
};
class Test1 : protected Test {};
class Test2 : public Test1 {};
```

b. *Cho biết kết quả khi thực hiện chương trình sau:*

```
class A {
public:
    A() { cout << "Goi ham thiet lap A" << endl; }
    ~A() { cout << "Goi ham huy A " << endl; }
};
class B : public A {
public:
    B() { cout << "Goi ham thiet lap B " << endl; }
    ~B() { cout << "Goi ham huy B " << endl; }
};
class C : public B {
public:
    C() { cout << "Goi ham thiet lap C " << endl; }
    ~C() { cout << "Goi ham huy C " << endl; }
};
void main() {
    C obj;
}
```

Goi ham thiet lap: A -> B -> C
Goi ham huy: C -> B -> A

```
class X {
public:
    int x1, x2;
    void in()   {
        cout << "x1 = "; cin >> x1;
        cout << "x2 = "; cin >> x2;
        out();
    }
    void  out() {
        cout << "Sum = " << x1 + x2;
    }
};
class  Y : public X {
    void  out() {
        cout << "Sub = " << x1 - x2;
    }
};
void main() {
    Y *y = new Y();
    y->in();
}
```

```
class X {
protected:
    int x1;
public:
    X() { x1 = 5; }
    void Out() { cout << "x1 = " << x1 << endl; }
};
class Y {
protected:
    int y1;
public:
    Y() { y1 = 10; }
    void Out() { cout << "y1 = " << y1 << endl; }
};
class Z : X, Y {};
void main()
{
    Z z;
    z.Out();
}
```

c. *Các biến x, y, z, t có thể truy xuất được các biến thành phần nào trong các lớp T1, T2, T3, T4?*

```
class T1 {
private:
    int t1;
public:
    int t2;
};
class T2 : public T1 {
public:
    int t3;
};
class T3 : private T1 {
protected:
    int t4;
};
class T4 : protected T1, T2, T3 {
public:
    int t5;
};
void main() {
    T1 x;
    T2 y;
    T3 z;
    T4 t;
}
```

x: t2
y: t3, t2
z:
t: t5

NỘI DUNG 4: ĐA HÌNH

❖ Chương 7 – Tính đa hình

1. Con trỏ trong thừa kế.
2. Phương thức trùng tên và sự thừa kế.
3. Phương thức tĩnh: cách gọi, hạn chế.
4. Phương thức ảo (pure virtual function): định nghĩa, cú pháp, quy tắc gọi.
5. Liên kết tĩnh (Static binding/Compile-time binding/Early binding) và liên kết động (Dynamic binding/ Runtime binding/Late binding): khái niệm, cách sử dụng, tốc độ.
6. Tính đa hình: khái niệm, cách áp dụng, phân loại: đa hình thời gian biên dịch (compile time polymorphism) và đa hình thời gian chạy (runtime polymorphism).
7. Lớp trừu tượng: khái niệm, phương thức thuần ảo (hàm thuần ảo, hàm ảo thuần túy), có thể sử dụng lớp trừu tượng để tạo đối tượng hoặc con trỏ không? Phương thức thiết lập (constructor) và hủy bỏ (destructor) có thể khai báo là phương thức ảo không?
8. Cơ chế Upcasting và Downcasting.
9. Một số câu hỏi có source code:
    a. *Cho biết kết quả thực hiện của chương trình sau:*

```
1  #include <iostream>
2  using namespace std;
3  class Hoa {
4  public:
5      virtual void Mau() = 0;
6  };
7  class Cuc : public Hoa {
8  public:
9      void Mau() { cout << "Mau: vang" << endl; }
10 };
11 class Hong : public Hoa {
12 public:
13     void Mau() { cout << "Mau: do" << endl; }
14 };
15 class Hue : public Hoa {
16 public:
17     void Mau() { cout << "Mau: trang" << endl; }
18 };
19 void main() {
20
21     Hong hong;
22     Hue* hue = &hong;
23     hue->Mau();
24 }
```

Lỗi biên dịch

```cpp
#include <iostream>
using namespace std;
class Hoa {
public:
    virtual void Mau() { cout << "Mau: ???" << endl; }
};
class Cuc : public Hoa {
public:
    void Mau() { cout << "Mau: vang" << endl; }
};
class Hong : public Hoa {
public:
    void Mau() { cout << "Mau: do" << endl; }
};
class Hue : public Hoa {
public:
    void Mau() { cout << "Mau: trang" << endl; }
};
void main() {
    Hong hong;
    Hoa* hoa = &hong;
    hoa->Mau();
}
```

```cpp
#include <iostream>
using namespace std;
class Hoa {
public:
    virtual void Mau() { cout << "Mau: ???" << endl; }
};
class Cuc : public Hoa {
public:
    void Mau() { cout << "Mau: vang" << endl; }
};
class Hong : public Hoa {};
class Hue : public Hoa {
public:
    void Mau() { cout << "Mau: trang" << endl; }
};
void main() {
    Hong hong;
    Hoa* hoa = &hong;
    hoa->Mau();
}
```

```cpp
#include <iostream>
using namespace std;
class Hoa {
public:
    virtual void Mau() = 0 { cout << "Mau: ???" << endl; }
};
class Cuc : public Hoa {
public:
    void Mau() { cout << "Mau: vang" << endl; }
};
class Hong : public Hoa {};
class Hue : public Hoa {
public:
    void Mau() { cout << "Mau: trang" << endl; }
};
void main() {
    Hong hong;
    Hoa* hoa = &hong;
    hoa->Mau();
}
```

Lỗi biên dịch

```cpp
#include <iostream>
using namespace std;
class Hoa {
public:
    virtual void Mau() = 0 { cout << "Mau: ???" << endl; }
};
class Cuc : public Hoa {
public:
    void Mau() { cout << "Mau: vang" << endl; }
};
class Hong : public Hoa {
public:
    void Mau() { cout << "Mau: do" << endl; }
};
class Hue : public Hoa {
public:
    void Mau() { cout << "Mau: trang" << endl; }
};
void main() {
    Hoa* hoa1 = new Hoa;
    hoa1->Mau();
    Hoa* hoa2 = new Hong;
    hoa2.Mau();
}
```

Lỗi biên dịch

```cpp
#include <iostream>
using namespace std;
class Hoa {
public:
    virtual void Mau() = 0 { cout << "Mau: ???" << endl; }
};
class Cuc : public Hoa {
public:
    void Mau() { cout << "Mau: vang" << endl; }
};
class Hong : public Hoa {
public:
    void Mau() { cout << "Mau: do" << endl; }
};
class Hue : public Hoa {
public:
    void Mau() { cout << "Mau: trang" << endl; }
};
void main() {
    Hoa** hoa;
    hoa[0] = new Hoa;
    hoa[1] = new Cuc;
    hoa[2] = new Hong;
    hoa[3] = new Hue;
}
```

Lỗi biên dịch và thực thi

---------- **HẾT** ----------

(Chúc các bạn thi tốt)