

Object oriented programming – Elektrijska 2018 (Budva)

Task1 (14):				Task5 (14):			
Task2 (14):				Task6 (14):			
Task3 (14):				Task7 (16):			
Task4 (14)							
Task1	Task2	Task3	Task4	Task5	Task6	Task7	Sum()

Task 1. What is the output for the following program?

```
#include <iostream>
using namespace std;

class Base
{
protected:
    int m;
public:
    Base(int mm = 4) { m = mm; }
    virtual int Process(int a, int b) {
        cout << "Base " << m << endl;
        return a >= b ? a - b - m : a;
    }
    int Change(int k) {
        return m += (m - k) ? k : Change(k / 2);
    }
};

class Derived : public Base
{
    double x;
public:
    Derived(int mm) : Base(mm) { x = 0; }
    Derived(int mm, double xx) : Derived(mm) { x = xx; }
    int Process(int a, int b) {
        int c = Base::Process(a, b);
        cout << "Derived " << m << endl;
        return (c >= a) ? a * (1 + x) >= b ? a - b : a : c;
    }
};

class Container
{
    Base* c;
    Base* Create(int a = 010) {
        if (a < 10)
            return new Base();
        else
            return new Derived(a, 0.25);
    }
public:
    Container() { c = Create(); }
    Container(int a) { c = Create(a); }
    ~Container() { Clear(); }
    void Clear() {
        if (c != NULL) {
            delete c;
            c = NULL;
        }
    }
    Container& operator =(const Container& cc) {
        c = Create();
        c->Change(10);
        return *this;
    }
    int Change(int k, int a = 1) {
        if (c) return c->Change(k);
        return 0;
    }
}
```

```

    int Process(int a, int b) {
        int r = 0;
        if (c) {
            r = c->Process(a, b);
            cout << a << " ";
        }
        else
            cout << "Cannot find c! ";
        return r;
    }
};

class Client
{
    int a;
    Container *c;
public:
    Client(int aa, Container *cc) {
        a = aa; c = cc;
    }
    ~Client() { c->Clear(); }
    void Modify(Container &cc) { c = &cc; }
    void Process(int b) {
        a = c->Process(a, b);
        cout << a << endl;
    }
};

void Process(Client c1, int j)
{
    c1.Process(j);
    Client c2 = c1;
    c2.Process(j);
}

int main()
{
    int i = 100, j = 10, k = 4;
    Container *pb1 = new Container;
    Client c1(i, pb1);
    c1.Process(j);
    Container b2 = *pb1;
    b2.Change(k);
    Client c2(i, &b2);
    pb1->Change(k);
    c1.Process(j);
    Process(c2, j);
    Container w3 = Container(20);
    c1.Modify(w3);
    c1.Process(j);
    Client c3 = c2;
    b2.Change(k);
    c2.Process(j);
    c3.Process(j);
    exit(0);
}

```

Task 2. What is the output for the following program?

```
#include <iostream>
using namespace std;

class A {
public:
    int Val;
    A* n[4];
    A(int w) { Val = w; for (int i = 0; i < 4; i++) n[i] = NULL; }
    virtual void Go(A* a);
    void Show() {
        cout << Val << " -> ";
        for (int i=0; i<4; i++) if (n[i] != NULL) cout<<n[i]->Val<<" "; cout<<endl;
    }
    virtual void Add(A* a);
};

void A::Add(A* a) {
    int m = Val^a->Val;
    switch (m) {
        case 1: n[0] = a; break;
        case 2: n[1] = a; break;
        case 4: n[2] = a; break;
        case 8: n[3] = a; break;
        default:
            for (int i = 0; i < 4; i++) {
                if (n[i] != NULL &&
                    (a->Val&(1 << i))) {
                    n[i]->Add(a); return;
                }
            }
            break;
    }
}

void A::Go(A* a) {
    Show();
    for (int i = 0; i < 4; i++) {
        if (n[i] != NULL)
            n[i]->Go(this);
    }
}

int g(int n, int i) {
    if (n == 1) return i;
    if (i < (2 << n - 2)) return g(n - 1, i);
    else return (1 << n - 1) | g(n - 1, (2 << n - 1) - 1 - i);
}

int main()
{
    A *t = new A(0); A *p = NULL;
    for (int i = 1; i < 8; i++) {
        p = new A(g(4, i));
        t->Add(p);
    }
    for (int i = 15; i >= 8; i--) {
        p = new A(g(4, i));
        t->Add(p);
    }
    t->Go(NULL);
    return 0;
}
```

Task 3. What is the output for the following program?

```
#include <iostream>

using namespace std;

static int id = 0;

class A
{
public:
    A() { cout << "A(default)" << endl; id = 1; }
    A(double &x) { cout << "A(double) " << x++ * 2 << endl; id++; }
    A(int x) { cout << "A(int) " << x * 2 << endl; id--; }
    A(const A& x) { cout << "A(A) " << (id += 5) << endl; }
};

A a = 5.7;
void f(int x) { cout << "f(int) " << x << endl; };
void f(double x) { cout << "f(double) " << x-- << endl; };
void f(const A &x) { cout << "f(A)" << endl; };

int main()
{
    int i = 1;
    double j = 2.5;
    A a;

    f(i ? j : a);
    f(true ? 1 : 2.5);
    f(i ? 2.5 : a);
    f(i ? a : 2.5);

    cout << j << " " << id << endl;
    return 0;
}
```

Task 4. What is the output for the following program?

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

#define CLASS_FOO_II(X, F, FC) (F::*X)(int, int)=&F::FC
#define SCLAS_FOO_II(X, F, FC) (*X)(int, int)=&F::FC
#define N 11

class Comparator {
public:
    Comparator(int val)
    {
        this->comp_val = val;
    }
    static int cmp(int a, int b)
    {
        return a - b;
    }
    static int cmp(double a, double b)
    {
        return b - a;
    }
    int bcmp(int a, int b)
    {
        return (a & this->comp_val) - (b & this->comp_val);
    }
    int bcmp(double a, double b)
    {
        int bi = (int)b;
        int ai = (int)a;
        return (bi | this->comp_val) - (ai | this->comp_val);
    }
private:
    int comp_val;
};

template <class T> class Sorter
{
public:
    Sorter(std::vector<T> v)
    {
        this->original = v;
        this->p = 0;
        std::cout << "I am immutable sorter!" << std::endl;
        this->print(this->original);
    }

    void sort(Comparator a, int (Comparator::*f)(T, T))
    {
        std::cout << "Use me!" << std::endl;
        std::vector<T> sorted = this->original;

        for (int i = 0; i < sorted.size(); i++) {
            T tmp = sorted[i];
            int j = i;
            while (j > 0 && (a.*f)(tmp, sorted[j - 1]) >= 1) {
                sorted[j] = sorted[j - 1];
                j--;
            }
            sorted[j] = tmp;
        }
        this->print(sorted);
    }
}
```

```

void sort(Comparator a, int(*f)(T, T))
{
    std::cout << "This shouldn't be used..." << std::endl;
    std::vector<T> sorted = this->original;

    for (int i = 0; i < sorted.size(); i++) {
        T tmp = sorted[i];
        int j = i;
        while (j > 0 && (*f)(tmp, sorted[j - 1]) >= 1) {
            sorted[j] = sorted[j - 1];
            j--;
        }
        sorted[j] = tmp;
    }
    this->print(sorted);
}

void print(std::vector<T> v)
{
    int i = 0;
    for (i = 0; i < v.size() - 1; i++) {
        if (this->p = !this->p) {
            std::cout << v[i] << ", ";
        }
    }
    std::cout << v[i] << std::endl;
}

private:
    std::vector<T> original;
    int p;
};

int main(int argc, char const *argv[])
{
    double arr[N] = { 12.2, 16.5, 123, 256, 121, 23.04, 5.2, 4, 23.3, 25.639, 1 };
    std::vector<int> v;

    for (int i = 0; i < N; i++) {
        v.push_back(arr[i]);
    }

    Comparator comp(0xF);
    Sorter<int> s(v);

    int SCLAS_FOO_II(f, Comparator, cmp);
    int CLASS_FOO_II(g, Comparator, bcmp);

    s.sort(comp, f);
    s.sort(comp, g);

    return 0;
}

```

Task 5. What is the output for the following program?

```
#include <iostream>
using namespace std;
int g = 0;
class Foo {
public:
    Foo(){ cout << "Foo" << endl; }
    Foo(int a){ cout << "Foo:" << a << endl; }
    Foo(double a){ cout << "Foo" << a << endl; }
};
class Bar : public Foo { };

template<class T, class U, int i> class A {
public:
    A(){ cout << "A:" << i << endl; }
    A(int a){ cout << "A:" << a << endl; }
    A(double a){ cout << "A:" << a << endl; }
    template<class V>
    void sum(V a, V b){
        g--;
        cout << "V:" << g << endl;
    }
    void sum(U a, U b){
        cout << "U:" << g << endl;
    }
    void sum(int a, double b){
        g += a;
        cout << "A:" << a*b << endl;
    }
    void sum(Foo a, double b){
        cout << "A:" << b << endl;
    }
};

template<class T> class A<T, Foo, 2> {
public:
    void sum(int a, double b) {
        g++;
        cout << "A2:" << a + b + 1 << endl;
    }
};

class B : public A<double, int, 2> {
public:
    void sum(int a, double b){
        Foo foo(a);
        cout << "B:" << a + b - 1 << endl;
    }
};

class C : public A<double, Foo, 2> { };

int main() {
    A<double, Foo, 2> foo;
    foo.sum(2.5, 2.5);
    A<double, Bar, 2> bar;
    bar.sum(2.5, 2.5);
    B b;
    b.sum(2.5, 2.5);
    C c;
    c.sum(2, 2);
    c.sum(g, g);
    bar.sum(g, 2.5);
    Foo d(2), f(g);
    bar.sum(d, f);
    return 0;
}
```


Task 6. What is the output for the following program?

```
#include <iostream>
using namespace std;

class Base;
typedef int (Base::*PF)(int);

class Base {
protected:
    int a;
public:
    Base() : a(2) {}
    Base(int aa) : a(aa) {}
    Base& operator =(const int &aa) { a = aa + 1; }
    virtual int f(int x) { x = x << a + 1 | a;
        cout << "Base::f() " << x << endl; return x; }
    virtual int g(int x) { x = x << a + 1 & a;
        cout << "Base::g() " << x << endl; return x; }
    PF h() { return &Base::f; }
};

class Derived : public Base {
public:
    Derived(int aa) : Base(aa) {}
    int f(int x) { x = x << a + 1 | a; cout << "Derived::f() " << x << endl; return x; }
private:
    int g(int x) { x = x << a + 1 & a; cout << "Derived::g() " << x << endl; return x; }
};

class Specific : public Derived {
public:
    Specific(int aa) : Derived(aa) {}
    int f(int x) { x = x << a + 1 | a; cout << "Specific::f() " << x << endl; return x; }
};

int main() {
    int x = 1;
    Base b = 1;
    Derived d = 2;
    Specific s = 3;
    Base *pB = new Derived(2);
    Derived *pD = new Specific(3);
    Base *pBB = new Specific(3);

    x = b.f(x);
    x = b.g(x);
    x = d.f(x);
    x = s.f(x);
    x = pB->f(x);
    x = pB->g(x);
    x = pD->f(x);
    x = pBB->f(x);
    x = pBB->g(x);

    PF pBf = b.h();
    (b.*pBf)(x);
    (d.*pBf)(x);
    (s.*pBf)(x);
    (pB->*pBf)(x);
    (pD->*pBf)(x);
    (pBB->*pBf)(x);
    return 0;
}
```

Task 7. What is the output for the following program?

```
#include <iostream>
using namespace std;

template <class Tip, int n>
class Node
{
public:
    Tip key, value, weigth;
    Node<Tip, n>(Tip k, Tip v, Tip w) {
        key = k; value = v; weigth = w;
    }
    virtual ~Node<Tip, n>() {}
    virtual bool Insert(Tip t, bool b = false) {
        return weigth == 0 ? (weigth = t) != 0 : false;
    }
    virtual int Search(Tip t) {
        cout << std::hex << key << " " << value << " " << weigth << endl;
        return 1;
    }
};

template <class Tip, int n>
class List : public Node<Tip, n>
{
public:
    Node<Tip, n>* arr[n];
    List<Tip, n>(Tip k, Tip v, Tip w) : Node<Tip, n>(k, v, w) {
        for (int i = 0; i < n; i++)
            arr[i] = NULL;
    }
    List<Tip, n>(Node<Tip, n> &node)
        : List<Tip, n>(node.key, node.value, node.weigth) {}
    ~List<Tip, n>() {
        for (int i = 0; i < n; i++)
            if (arr[i] != NULL)
                delete arr[i];
    }
    int Find(Tip t) {
        int kk = this->key;
        int mode = 0, shift = 0;
        int mask = 0xF;
        while (t != 0) {
            int bit = (t & mask) >= (kk & mask | this->value >> 1);
            mode |= bit << shift++;
            t >>= 0x4;
            kk >>= 0x4;
        }
        return mode;
    }
    int FindUnoccupied(int mode) {
        int val = 0, shift = 0;
        int mask = 0xF;
        int mm = mode;
        while (mm != 0) {
            int bit = mm & 0x1 ? this->value >> 1 : 0;
            val |= bit << shift;
            mm >>= 0x1;
            shift += 0x4;
        }
        return val;
    }
};
```

```

bool Insert(Tip t, bool b = false) {
    bool bOk = false;
    int mode = Find(t);
    if (arr[mode] == NULL) {
        int k = FindUnoccupied(mode);
        arr[mode] = new Node<Tip, n>(this->key + k, this->value >> 1, t);
        bOk = b ? Insert(this->weigth, false) : true;
    }
    else {
        bOk = arr[mode]->Insert(t);
        if (!bOk) {
            Node<Tip, n>* ptr = arr[mode];
            arr[mode] = new List<Tip, n>(*arr[mode]);
            delete ptr;
            bOk = arr[mode]->Insert(t, true);
        }
    }
    return bOk;
}

int Search(Tip t) {
    int count = 1;
    int mode = Find(t);
    for (int i = 0; i < n; i++)
        if (arr[mode^i] != NULL)
            count += arr[mode^i]->Search(t);
    return count;
}

};

template <class Tip, int n> class Container {
protected:
    Node<Tip, n> *start;
    int count;
public:
    Container<Tip, n>() {
        start = NULL;
        count = 0;
    }
    void Insert(Tip t) {
        if (start == NULL)
            start = new Node<Tip, n>(0, n, t);
        else
            if (!start->Insert(t)) {
                start = new List<Tip, n>(*start);
                start->Insert(t, true);
            }
    }
    int Search(Tip t) {
        return start->Search(t);
    }
};

int main() {
    int n = 10;
    int arr[]={0x111, 0x351, 0x517, 0x131, 0x515, 0x333, 0x575, 0x735, 0x153, 0x373};
    class Container<int, 8> container;
    for (int i = 0; i < n; i++)
        container.Insert(arr[i]);
    int count = container.Search(0x0);
    cout << std::dec << count << endl;
    count = container.Search(0x555);
    cout << std::dec << count << endl;
    return 0;
}

```

Answers: Object oriented programming – Elektrijada 2018 (Budva)

Task1.

```
Base 4
100 86
Base 16
86 60
Base 16
100 74
Base 16
74 48
Base 20
Derived 20
60 30
Cannot find c! 0
Cannot find c! 0
```

Task2.

```
0 -> 1 2 4 8
1 -> 3 5 9
3 -> 7 11
7 -> 15
15 ->
11 ->
5 -> 13
13 ->
9 ->
2 -> 6 10
6 -> 14
14 ->
10 ->
4 -> 12
12 ->
8 ->
```

Task3.

```
A(int) 10
A(default)
A(double) 5
f(A)
f(double) 1
A(int) 4
f(A)
A(A) 6
f(A)
3.5 6
```

Task4.

```
I am immutable sorter!
12, 123, 121, 5, 23, 1
This shouldn't be
used...
256, 121, 23, 16, 5, 1
Use me!
12, 121, 23, 5, 1, 256
```

Task5.

```
A2:5.5
A:2
V:0
A:2
Foo:2
B:3.5
A2:5
A2:3
A:5
Foo:2
Foo:4
V:3
```

Task6.

```
Base::f() 5
Base::g() 0
Derived::f() 2
Specific::f() 35
Derived::f() 282
Derived::g() 0
Specific::f() 3
Specific::f() 51
Derived::g() 0
Base::f() 1
Derived::f() 2
Specific::f() 3
Derived::f() 2
Specific::f() 3
Specific::f() 3
```

Task7.

```
0 2 111
20 2 131
222 2 333
42 2 153
240 2 351
262 2 373
404 2 515
406 2 517
624 2 735
444 4 575
14
444 4 575
404 2 515
406 2 517
624 2 735
240 2 351
262 2 373
42 2 153
222 2 333
20 2 131
0 2 111
14
```