

Object oriented programming – Elektrijada 2012 (Kranevo)

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

Task1. What is the output for the following program?

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

class OOP
{
    virtual int f( int );
public:
    int x;
    OOP( int a ) : x(a) { cout<<"-";}
    double f( double );
};

int OOP::f( int a )
{
    return a+x;
}

#define KUB(a) a*a*a*a
#define swap(a, b) a=a*b;b=a/b;a=a/b;
double OOP::f( double a )
{
    cout << "X-X";
    return KUB(a*x-4) + f(5);
}

class Delphi : public OOP
{
    int x;
    int f( int );
public:
    Delphi ( int a, int b ) : OOP(a), x(b) {cout << "-ZZ-";}
    double f( double );
};

int Delphi::f( int a )
{
    cout << "WXZ-";
    return KUB(a-x);
}

double Delphi::f( double a )
{
    cout << f(3) << "-ZXW";
    swap(a,x);
    return a/x+f(5);
}

int main()
{
    OOP* pb = new Delphi(3,7);
    cout << pb->f( 3.4 ) << endl;
    Delphi d(2,4);
    cout << d.f( 2.5 ) << endl;
    OOP b(3);
    cout << b.f( 1.6 ) << endl;
    cout << KUB(2.5-4);
    return (int)0;
}
```

Task2. What is the output for the following program?

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

static int b;

class X {
    int *pi;

public:
    int plaza, sladoled, more;

    X() {b=0; plaza=3; more = 5*b++; sladoled=more++;};
    X(const X &x) : pi(new int(*x.pi)){b*=2; plaza=-4; more = -3*b++;
sladoled=more++;}
    X(int l) : pi(new int(l)), plaza(l), more(l++), sladoled(l++) { b++;}
    X& operator= (const X&);
    ~X() {cout << "b:"<< b << "plaza:"<< plaza << "sladoled:"<<sladoled <<
"more:"<<more << endl;delete pi;}
};

X& X::operator= (const X& x) {
    if (this != &x) {
        delete pi;
        pi = new int(*x.pi);
    }
    return *this;
}

X funF(X& x)
{
    X Xnew(5);
    x=Xnew;
    Xnew.plaza++; Xnew.more+= ++b; Xnew.sladoled=b;
    return x;
}

void g() {

    X xa=3, xb=1;
    X xc = xa;
    xa = funF(xb);
    xc = xa;
    X a(2);
    a.sladoled++;
    b+=a.sladoled;
}

int main(int argc, char* argv[])
{
    X d();
    g();
    return (int)0;
}
```

Task3. What is the output for the following program?

```
#include "stdio.h"
#include "stdlib.h"

#define making(x,y,z) z++; \
    printf(y, GlassOfWater(), Sugar::TwoCubesOfSugar(), x); z++;
#define boil(x,y) TwoCubesOfSugar()+Water::x+Sugar::x+(int)x
#define swap(x,y) x=x+y; y=x-y; x=x-y;

char *s = new char[100];
int i=0;
char* JustDoIt(char* c);

class Sugar
{
private:
    int x;
public:
    Sugar()      {      x= 1; s[i]='$';   s[i+1]='d'; }

    int TwoCubesOfSugar()  {      return x;   }

    friend class Coffee;
};

class Water : public Sugar
{
private:
    int x;
public:
    Water()      {      x=-2; making(this, JustDoIt(s), i); }

    Water(int X)      {      x=X;   }

    int GlassOfWater()      {      return x;   }

    friend class Coffee;
};

class Coffee : public Water, public Sugar
{
private:
    unsigned int x;
public:
    Coffee()      {      x=Water::x; making(this, JustDoIt(s), i); }

    int AddMoreWater(Water* w)
    {
        printf("%d",w->TwoCubesOfSugar()+Water::x+Sugar::x+x);
        swap(i,x);
        return i;
    }

    int AddMoreSugar(Sugar* r)
    {
        printf("%d",r->boil(x,this));
        swap(i,x);
        return i;
    }
};
```

```

char* JustDoIt(char* c)
{
    char* cup = new char[i+2];
    int j;
    for (j=0; j<i+1; j++)
        if ((j+1)%2)
        {
            cup[j]=c[j]+1;
        }
        else
        {
            cup[j]=c[j]|0x11;
        }
    cup[i+1]='\0';
    return cup;
}

int main(int argc, char* argv[])
{
    Coffee c;
    Water w(3);
    c.AddMoreSugar((Sugar*)&w);
    return int(0);
}

```

Task4. What is the output for the following program?

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

int dat=1;
class StariVek;

class B {
public:
    B() { cout<<"2012"<<endl; };
    virtual void f(int d=5)=0;
    ~B(){ cout <<"day"<<endl;};
};

class A {
    unsigned i;
public:
    int pod;
    A& operator++() { cout << "A " << pod-- << dat++ << endl; A* b = new A;
return *b;}
    A& operator++(int) { cout << "B " << pod++ << dat-- << endl; A* b = new A;
return *b;}
    A& operator--() { cout << "C " << dat++ << ++pod << endl; A* b = new A;
return *b;}
    A& operator--(int) { cout << "D " << dat--<< --pod << endl; A* b = new A;
return *b;}
    void operator+(A& b) { pod++;}
    friend void B::f(int);
    A() : i(0), pod(dat) {pod++; cout << "Kranev" << endl;}
    ~A(){pod++; cout << "Have ";}
};

const StariVek& returnPraistorija(const StariVek& p) { return p; }

class StariVek : public A, public B {
public:
    void f(int d=2){ A c; c++; pod*=d--;}
    StariVek(){cout << "EL" << endl;}
    ~StariVek(){cout << "a nice ";}
};

StariVek returnStariVek(StariVek s){ return s; }

class ModernoDoba : public StariVek {
public:
    ModernoDoba(){cout << "enjoy" << endl;}
    ~ModernoDoba(){cout << "party time" << endl;}
};

StariVek returnModernoDoba(StariVek s) { return s; }

int main(int argc, char* argv[])
{
    ModernoDoba x;
    A v,w;
    B *c = new StariVek;
    --++v--++w--;
    ++--++v+++w--;
    returnModernoDoba(returnStariVek(returnModernoDoba(returnPraistorija(x))));
    return int(0);
}
```

Task5. What is the output for the following program?

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

class Seed {
protected:
    char data;
public:
    virtual int Length() = 0;
    virtual int Count(int &min, int &max) = 0;
    virtual char Process() = 0;
};

class Starter : public Seed
{
public:
    Starter(char d) { data = d; };
    virtual int Length() { return 1; };
    virtual int Count(int &min, int &max) {
        min = data & 1 ? 0 : 1;
        max = data & 1 ? 1 : 0;
        return 0;
    };
    virtual char Process() { return '0'; };
};

class Block : public Seed {
protected:
    Seed *seed;
public:
    virtual int Length() { return seed->Length()+2; };
};

class BlockDefault : public Block {
public:
    BlockDefault(Seed *s, char d) { seed = s; data = d; };
    virtual int Count(int &min, int &max) {
        int cnt = seed->Count(min, max);
        min++; max++;
        return cnt;
    };
    virtual char Process() { return seed->Process() + (data & 1 ? 0 : 1); };
};

class BlockComplex : public Block {
public:
    BlockComplex(Seed *s, char d) { seed = s; data = d; };
    virtual int Count(int &min, int &max) {
        int cnt = seed->Count(min, max);
        if (data & 1) {
            max += 2;
            return cnt+min;
        } else {
            min += 2;
            return cnt+max;
        }
    };
    virtual char Process() { return seed->Process(); };
};
```

```

class Creator {
    bool status;
    Seed *seed;
public:
    Creator(char c) { status = true; seed = new Starter(c); };
    bool GetState() { return status; };
    int GetLength() { return seed->Length(); };
    void ChangeStatus() {
        int min, max;
        status = !status;
        cout << seed->Process() << " " << seed->Count(min, max) << endl;
    };
    void Modify(char cc, char cp) {
        if (!cp) { status = false; return; }
        if (cc & 0x1 ^ cp & 0x1)
            seed = new BlockDefault(seed, cc);
        else
            seed = new BlockComplex(seed, cc);
    };
};

class Sequence {
    int lenght;
    int count;
    char *arSeq;
    Creator **arCreator;
public:
    Sequence(char arr[], int n) {
        arSeq = new char[lenght=n];
        strcpy(arSeq, arr);
        arCreator = new Creator*[lenght];
        count = 0;
    };
    void Add(Creator *cr) { arCreator[count++] = cr; };
    void Remove(Creator *cr) {
        int i;
        for (i=0; i<count && arCreator[i] != cr; i++);
        arCreator[i]->ChangeStatus();
        for (count--; i<count; i++)
            arCreator[i] = arCreator[i+1];
    };
    void Step(int ind) {
        for (int i=0; i<count; i++) {
            int indS = ind - arCreator[i]->GetLength() - 1;
            arCreator[i]->Modify(arSeq[ind], indS < 0 ? 0 : arSeq[indS]);
        }
    };
};

int main(int argc, char* argv[]) {
    char arr[] = "0101001110001011";
    Sequence seq(arr, strlen(arr));
    Creator **aCreators = new Creator*[strlen(arr)];
    for (int i=0; i<strlen(arr); i++) {
        seq.Step(i);
        for (int j=0; j<i; j++)
            if (!aCreators[j]->GetState())
                seq.Remove(aCreators[j]);
        aCreators[i] = new Creator(arr[i]);
        seq.Add(aCreators[i]);
    }
    return (int)0;
}

```


Task6. What is the output for the following program?

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

#define NN 8
class Link;

template<class T, int n>
class Node {
public:
    T val[n];
    Link *next, *prev;
    Node() {
        for (int i=0; i<n; i++) { val[i] = 0; }
        next = NULL; prev = NULL;
    };
    bool operator <(const Node &node) {
        int i;
        for (i=0; i<n && val[i]<=node.val[i]; i++);
        return i == n;
    };
    void Convert(T data, T base[]) {
        for (int i=0; i<n; i++) {
            val[i] = 0;
            while (!(data % base[i])) {
                val[i]++;
                data /= base[i];
            }
        }
    };
};

class Link {
public:
    int val[2];
    Link *next, *prev, *down, *up;
    Link(int x, int y, Link *nn, Link *pp) {
        val[0]=x; val[1]=y; next = nn; prev = pp;
    };
    void Update(Link *dd, Link *uu) { down = dd; up = uu; };
};

template<class T, int n>
class Collection {
public:
    int nNodes;
    Node<T, n> *arNodes;
    T base[n];
    Collection(T arN[], int nN, T b[]) {
        for (int i=0; i<nN; i++) { base[i] = b[i]; }
        arNodes = new Node<T,n>[nNodes = nN];
        for (int i=0; i<nN; i++) {
            arNodes[i].Convert(arN[i], base);
        }
    };
    void Process() {
        for (int i=0; i<nNodes; i++)
            for (int j=0; j<nNodes; j++)
                if (i != j && arNodes[i] < arNodes[j])
                    Connect(i, j);
    };
};
```

```

void Connect(int u, int v) {
    Link *p = arNodes[u].next, *pp = NULL, *pt;
    for ( ; p!=NULL && p->val[1]<v; pp=p, p=p->next);
    if (pp == NULL) {
        pt = arNodes[u].next = new Link(u, v, p, pp);
    } else {
        pt = pp->next = new Link(u, v, p, pp);
    }
    if (p != NULL)
        p->prev = pt;
    p = arNodes[v].prev; pp = NULL;
    for ( ; p!=NULL && p->val[0]<u; pp=p, p=p->down);
    pt->Update(p, pp);
    if (pp == NULL) {
        arNodes[v].prev = pt;
    } else {
        pp->down = pt;
    }
    if (p != NULL)
        p->up = pt;
};

void Traverse() {
    int arr[NN];
    for (int i=0; i<nNodes; i++)
        arr[i] = i;
    int ind;
    for (int cnt=nNodes; cnt > 0; cnt--) {
        for (ind=0; ind<cnt && arNodes[arr[ind]].prev != NULL; ind++);
        Delete(arr[ind]);
        cout << arr[ind]; (cnt-1)%2 ? cout << " " : cout << endl;
        for (int i=ind; i<cnt-1; i++) {
            arr[i] = arr[i+1];
        }
    }
};

void Delete(int u) {
    Link *p = arNodes[u].next;
    for ( ; p!=NULL; p=p->next) {
        if (p->up == NULL)
            arNodes[p->val[1]].prev = p->down;
        else
            p->up->down = p->down;
        if (p->down != NULL)
            p->down->up = p->up;
    }
};

void Print() {
    for (int i=0; i<nNodes; i++)
        for (Link *p=arNodes[i].next; p!=NULL; p=p->next)
            cout << p->val[0] << " " << p->val[1] << endl;
    cout << endl;
};

};

int main(int argc, char* argv[]) {
    int ar[] = { 50400, 5040, 1035, 7056, 6552, 952, 7425, 18900 };
    int base[] = {2, 3, 5, 7};
    Collection<int, 4> col(ar, NN, base);
    col.Process();
    col.Print();
    col.Traverse();
    return (int)0;
}

```

Task7. What is the output for the following program?

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

#define NN 8

class Item {
public:
    int x, y;

    Item() { x = 0; y = 0; };
    Item(int xx, int yy) { x = xx; y = yy; };

    void Print() { cout << x << " " << y << endl; };

    bool operator <(const Item& it) {
        return x < it.x || y < it.y;
    };

    Item operator +(const Item& it) {
        Item itsum(x + it.x, y + it.y);
        return itsum;
    };
};

template<class D, int n>
class Node {
public:
    D data, ref;
    int count, size;
    Node *link[n];

    Node() {
        data = D(); count = 0;
        for (int i=0; i<n; i++) { link[i] = NULL; }
    };

    Node(D r, int sz) {
        data = D(); ref = r; size = sz; count = 0;
        for (int i=0; i<n; i++) { link[i] = NULL; }
    };

    bool IsEmpty() { return count == 0; };

    void Update(D d) { data = d; count++; };

    Node<D,n>* Find(D d) {
        Node<D,n> *ptr = NULL;
        if (d < ref || ref+Item(size,size) < d)
            ptr = NULL;
        else if (link[0] == NULL)
            ptr = this;
        else
            for (int i=0; ptr==NULL && i<n; i++)
                ptr = link[i]->Find(d);
        return ptr;
    };
};
```

```

void Split() {
    for (int i=0; i<n; i++) {
        Item it( (i & 0x1) * size/2, ((i & 0x2) >> 1) * size/2);
        link[i] = new Node<D,n>(ref+it, size/2);
    }
    Node<D,n>* ptr = Find(data);
    count--;
    ptr->Update(data);
};

void Traverse(int start) {
    if (link[0] == NULL) {
        if (count > 0)
            data.Print();
    } else {
        int delta = (start & 0x1) ? -1 : 1;
        int tmp = start++;
        for (int i=0; i<n; i++, tmp+=delta)
            link[tmp&0x3]->Traverse(start);
    }
};

template<class D, int n>
class Container {
public:
    Node<D, n> *first;
    int size, count;

    Container(int sz) { size = sz; count = 0; first = NULL; };

    void Create(D data) {
        if (first == NULL) {
            first = new Node<D,n>(Item(0,0), size);
            first->Update(data);
        } else {
            Node<D,n> *ptr = first->Find(data);
            while (!ptr->IsEmpty()) {
                ptr->Split();
                ptr = ptr->Find(data);
            }
            ptr->Update(data);
        }
        count++;
    };

    void Traverse(int start) {
        first->Traverse(++start);
    };
};

int main(int argc, char* argv[]) {
    Item ar[] = { Item(1, 2), Item(12,3), Item(11,15), Item(6,2),
                  Item(10,11), Item(5,10), Item(14,15), Item(9,13)};
    int base[] = {2, 3, 5, 7};
    Container<Item, 4> con(2*NN);
    for (int i=0; i<NN; i++) {
        con.Create(ar[i]);
    }
    con.Traverse(0);
    return (int)0;
}

```

Solutions: Object oriented programming – Kranevo 2012

Task1.

```
--ZZ-X-XWXZ--223.2
--ZZ-WXZ--37-ZXWWXZ--24.5
-X-X-48.8
-31.5
```

Task2.

```
b:15plaza:6sladoled:7more:13
b:15plaza:-4sladoled:-42more:-41
b:19plaza:2sladoled:3more:3
b:19plaza:-4sladoled:-12more:-11
b:19plaza:1sladoled:1more:2
b:19plaza:3sladoled:3more:4
```

Task3.

```
429496729442949672941-2
```

Task4.

```
Kranev
2012
EL
enjoy
Kranev
Kranev
Kranev
2012
EL
D 11
Kranev
D 01
Kranev
B 0-1
Kranev
A -1-2
Kranev
C -11
Kranev
D 00
Kranev
B 1-1
Kranev
A -1-2
Kranev
C -11
Kranev
A 10
Kranev
a nice day
Have a nice day
Have a nice day
Have a nice day
Have Have Have party time
a nice day
Have
```

Task5.

0 0
0 1
0 3
1 1
1 3
1 7
2 4
2 6

Task6.

1 0
2 0
2 1
2 6
2 7
4 0
4 1
4 3
5 0
5 1
5 3
5 4
6 7

2 5
4 1
0 3
6 7

Task7.

12 3
1 2
6 2
11 15
9 13
14 15
10 11
5 10