ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ Београд 17.11.2013.

Колоквијум из
Објектно оријентисаног програмирања I

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) Попунити следећу табелу знаком + за дозвољено и знаком – за недозвољено:

дефиниција	пре и декларације и дефиниције класе	после декларације, пре дефиниције класе	после дефиниције класе
објекта			
показивача на објекат			

б) Да ли се статичка (заједничка) метода може позвати пре него што се направи први објекат одговарајуће класе и зашто?

в) Да ли је механизам преклапања операторских функција у језику C++ статички или динамички и зашто?

2) *(укупно 70 поена)* Написати на језику *C++* следеће класе (класе опремити оним конструкторима, деструктором и оператором доделе који су потребни за безбедно коришћење класа):

- *(30 поена)* ***Вектор*** се задаје помоћу три реалне компоненте (*x*, *y*, *z*), подразумевано (0, 0, 0). Може да се упореди са задатим вектором (`vekt1<vekt2`), односно (`vekt1==vekt2`), при чему се пореде интензитети вектора. Може да се одреди збир два вектора (`vekt1+vekt2`) и производ скалара и вектора (`skalar*vekt`), да се вектору дода други вектор (`vekt1+= vekt2`) и да се вектор упише у излазни ток (`it<<vekt`) у облику **(***x*`,`y`,`z**)**.
- ***Честица*** има задате векторе положаја и брзине (подразумевано честица у координатном почетку која се не креће) који могу да се дохвате. Може да промени положај на основу протеклог реалног времена (`cest<<=vreme`, нов положај се рачуна као збир старог положаја и производа брзине и протеклог времена). Честица може да се упише у излазни ток (`it<<cest`) у облику **[***положај***|***брзина***]**.
- *(30 поена)* ***Облак*** честица има задато име (ниска) и може да садржи задат број честица (подразумевано 10) уређених по неопадајућем интензитету њихових вектора положаја. Ствара се празан, после чега се честице додају појединачно (`oblak+=cest`; препуњавање облака прекида програм). Може да се дохвати број честица у облаку, да се дохвати честица са задатим редним бројем из облака (`oblak[i]`; дохваћена честица не може да се помера; вредност индекса ван дозвољеног опсега прекида програм), да се помере све честице у облаку у зависности од протеклог времена (`oblak<<=vreme`) и да се облак упише у излазни ток (`it<<oblak`), тако што се у првом реду испише име, а у наредним редовима честице у облаку.

(10 поена) Написати на језику *C++* **програм** који направи један облак честица, дода неколико честица у облак и испише облак на главном излазу у тренуцима 0 и 10. Користити фиксне параметре – није потребно ништа учитати с главног улаза.

НАПОМЕНЕ: **а)** Колоквијум траје **120** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Решење задатка не треба раздвајати у датотеке. Довољно је за сваку класу навести дефиницију класе и одмах иза ње евентуалне дефиниције метода које нису дефинисане у самој класи.

д) Резултати колоквијума биће објављени на *Web*-у на адреси: `home.etf.rs/~kraus/` (одреднице: *настава* | <име предмета> | *оцене* | *колоквијуми*).

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

class Vekt {
  double x, y, z;
public:
  explicit Vekt(double a=0, double b=0,
                double c=0)
    { x = a; y = b; z = c; }
  friend bool operator<(const Vekt& v1,
const Vekt& v2) {
  return v1.x*v1.x + v1.y*v1.y + v1.z*v1.z
    < v2.x*v2.x + v2.y*v2.y + v2.z*v2.z; }
  friend bool operator==(const Vekt& v1,
const Vekt& v2) { return v1.x==v2.x &&
     v1.y==v2.y && v1.z==v2.z; }
  friend Vekt operator+(const Vekt& v1,
      const Vekt& v2) { return
    Vekt(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);}
  friend Vekt operator*(const Vekt& v,
                        double s)
    { return Vekt(v.x*s, v.y*s, v.z*s); }
  Vekt& operator+=(const Vekt& v2)
    { return *this = *this + v2; }
  friend ostream& operator<<(ostream& it,
                             const Vekt&v)
    { return it << '(' << v.x << ','
        << v.y << ',' << v.z << ')'; }
};
```

```
class Cest {
  Vekt r, v;
public:
  explicit Cest(const Vekt& rr=Vekt(),
    const Vekt& vv=Vekt()): r(rr), v(vv){}
  Vekt polozaj() const { return r; }
  Vekt brzina() const { return v; }
  Cest& operator<<=(double t)
    { r += v * t; return *this; }
  friend ostream& operator<<(ostream& it,
                             const Cest& c)
    { return it << '[' << c.r << '|'
                << c.v << ']'; }
};
```

```
class Oblak {
  char* ime;
  Cest* niz; int kap, duz;
  void kopiraj(const Oblak& obl);
  void brisi()
    { delete [] ime; delete [] niz; }
public:
  explicit Oblak(const char* i, int k=10){
    ime = new char [strlen(i) + 1];
    strcpy(ime, i);
    niz = new Cest [kap = k]; duz = 0;
  }
  Oblak(const Oblak& obl) {kopiraj(obl);}
  ~Oblak() { brisi(); }
  Oblak& operator=(const Oblak& obl) {
    if (this != &obl)
      { brisi(); kopiraj(obl); }
    return *this;
  }
  int brCest() const { return duz; }
  Oblak& operator+=(const Cest& cest);
  const Cest& operator[](int i) const {
    if (i<0 && i>=duz) exit(2);
    return niz[i];
  }
  Oblak& operator<<=(double t);
  friend ostream& operator<<(ostream& it,
                       const Oblak& obl);
};

void Oblak::kopiraj(const Oblak& obl) {
  ime = new char [strlen(obl.ime) + 1];
  strcpy(ime, obl.ime);
  niz = new Cest [kap = obl.kap];
  duz = obl.duz;
  for (int i=0; i<duz; i++)
    niz[i] = obl.niz[i];
}

Oblak& Oblak::operator+=(const Cest& cest)
{
  if (duz == kap) exit(1);
  int i = duz-1;
  while (i>=0 &&
         cest.polozaj()<niz[i].polozaj())
    { niz[i+1] = niz[i]; i--; }
  niz[i+1] = cest; duz++;
  return *this;
}

Oblak& Oblak::operator<<=(double t) {
  for (int i=0; i<duz; niz[i++]<<=t);
  for (int i=0; i<duz-1; i++)
    for (int j=i+1; j<duz; j++)
      if (niz[j].polozaj() <
          niz[i].polozaj())
      { Cest c = niz[i]; niz[i] = niz[j];
        niz[j] = c; }
  return *this;
}

ostream& operator<<(ostream& it, const
Oblak& obl) {
  it << obl.ime << endl;
  for (int i=0; i<obl.duz;
       it<<obl.niz[i++]<<endl);
  return it;
}

int main() {
  Oblak obl("Oblak");
  obl += Cest(Vekt(3,0,0), Vekt(0,2,0));
  obl += Cest(Vekt(0,1,0), Vekt(0,0,1));
  obl += Cest(Vekt(0,0,2), Vekt(0,1,0));
  obl += Cest(Vekt(3,4,12),
              Vekt(-.3,-.4,-1.2));
  obl += Cest(Vekt(), Vekt(1,0,0));
  cout << obl;
  cout << endl << (obl<<=10);
  return 0;
}
```

```
Oblak
[(0,0,0)|(1,1,1)]
[(0,1,0)|(0,0,2)]
[(0,0,2)|(0,1,0)]
[(3,0,0)|(0,2,0)]
[(3,4,12)|(-0.3,-0.4,-1.2)]

Oblak
[(0,0,0)|(-0.3,-0.4,-1.2)]
[(0,10,2)|(0,1,0)]
[(10,10,10)|(1,1,1)]
[(0,1,20)|(0,0,2)]
[(3,20,0)|(0,2,0)]
```