ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ

Београд 29.11.2013.

Други колоквијум из
Објектно оријентисаног програмирања I

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

а) Да ли се у језику C++ може дефинисати операторска функција као метода класе `X`, са следећом декларацијом: `X operator/();`? Образложити.

б) Који је редослед извршавања деструктора при деструкцији објекта изведене класе, ако и изведена и основна класа садрже атрибуте (податке чланове)?

в) Шта је операнд, а шта резултат оператора `typeid`?

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

- *(30 поена)* ***Домина*** садржи два целобројна поља (*a*,*b*) – подразумевано (0,0). Може да се дохвати вредност првог и другог поља домине, да се изврши међусобна замена поља (`~dom`), да се одреди да ли су две домине једнаке (`dom1==dom2`) не водећи рачуна о редоследу поља и да се домина упише у излазни ток (`it<<dom`) у облику **(***a*`,`*b***)**.
- Апстрактан ***скуп*** домина садржи произвољан број различитих домина. Ствара се празан после чега се домине додају појединачно и смештају у скуп по неком непознатом редоследу (`skup+=dom`). Повратна вредност при додавању представља индикатор успеха. Може да се испита да ли се нека домина налази у скупу и да се скуп упише у излазни ток уписивањем садржаних домина.
- *(30 поена)* ***Кутија*** је скуп домина код које се домине додају на крај скупа. Може да се напуни свим могућим доминама, по случајном редоследу, с вредностима поља у опсегу од 0 до $n-1$ (број таквих домина је $n(n+1)/2$, пуњење може да се врши по систему покушаја стављања у кутију случајно генерисане домине док се кутија не напуни) и да се узме домина са почетка кутије (`uzmi(dom)`; повратна вредност је индикатор успеха). Библиотечка функција `rand()` при сваком позивању даје случајан цео број у опсегу `[0,RAND_MAX]`.
- ***Табла*** је скуп домина код којег се домине додају на почетак или крај тако да суседна поља суседних домина буду једнака (тј. да друго поље претходне домине буде једнако првом пољу наредне домине).

(10 поена) Написати на језику *C++* програм који направи једну кутију и једну таблу, напуни кутију за фиксно одабрани параметар *n*, испише кутију на главном излазу, узима домине из кутије и ставља их на таблу све док може и испише завршни садржај табле на главном излазу.

НАПОМЕНЕ: **а)** Колоквијум траје **100** 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 <cstdlib>
using namespace std;

class Domina { int a, b;
public:
  Domina() { a = b = 0; }
  Domina(int p, int q) { a = p; b = q; }
  int A() const { return a; }
  int B() const { return b; }
  Domina& operator~()
    { int c = a; a = b; b = c;
      return *this; }
  bool operator==(const Domina& d) const
    { return a==d.a && b==d.b ||
             b==d.a && a==d.b;   }
  friend ostream& operator<<(ostream& it,
                             const Domina& d)
    { return it << '(' << d.a << ',' << d.b
                << ')'; }
};
```

```
class Skup {
protected:
  struct Elem {
    Domina dom; Elem* sled;
    Elem(const Domina& d, Elem* s=0)
      : dom(d) { sled = s; }
  };
  Elem *prvi, *posl;
  void kopiraj(const Skup& s);
  void brisi();
public:
  Skup() { prvi = posl = 0; }
  Skup(const Skup& s) { kopiraj(s); }
  virtual ~Skup() { brisi(); }
  Skup& operator=(const Skup& s) {
    if (this != &s) { brisi(); kopiraj(s); }
    return *this;
  }
  bool ima(const Domina& d) const;
  virtual bool operator+=
    (const Domina& d) =0;
  friend ostream& operator<<(ostream& it,
                             const Skup& s);
};

void Skup::kopiraj(const Skup& s) {
  prvi = posl = 0;
  for (Elem* tek=s.prvi; tek; tek=tek->sled)
    posl = (!prvi ? prvi : posl->sled)
         = new Elem(tek->dom);
}

void Skup::brisi() {
  while (prvi) { Elem* stari = prvi; prvi
    = prvi->sled; delete stari; }
}

bool Skup::ima(const Domina& d) const {
  for (Elem *tek=prvi; tek; tek=tek->sled)
    if (d == tek->dom) return true;
  return false;
}

ostream& operator<<(ostream& it,
                    const Skup& s) {
  for (Skup::Elem* tek=s.prvi; tek;
       tek=tek->sled) it << tek->dom;
  return it;
}
```

```
class Kutija: public Skup {
public:
  bool operator+=(const Domina& d) {
    if (ima(d)) return false;
    posl = (!prvi ? prvi : posl->sled)
         = new Elem(d);
    return true;
  }

  Kutija& napuni (int n) {
    prvi = posl = 0;
    int m = n * (n+1) / 2;
    for (int i=0; i<m;)
      if (*this += Domina
        ((int)(rand()/(RAND_MAX+1.)*n),
         (int)(rand()/(RAND_MAX+1.)*n)))
        i++;
    return *this;
  }

  bool uzmi(Domina& d) {
    if (!prvi) false;
    d = prvi->dom;
    Elem* stari = prvi;
    prvi = prvi->sled;
    delete stari;
    if (!prvi) posl = 0;
    return true;
  }
};
```

```
class Tabla: public Skup {
public:
  bool operator+=(const Domina& d) {
    if (ima(d)) return false;
    if      (!prvi || d.B()==prvi->dom.A())
      naPocetak(d);
    else if (d.A() == posl->dom.B())
      naKraj(d);
    else {
      Domina e(d); ~e;
      if      (e.B() == prvi->dom.A())
        naPocetak(e);
      else if (e.A() == posl->dom.B())
        naKraj(e);
      else
        return false;
    }
    return true;
  }
private:
  void naPocetak(const Domina& d) {
    prvi = new Elem (d, prvi);
    if (!posl) posl = prvi;
  }

  void naKraj(const Domina& d) {
    posl = (!prvi ? prvi : posl->sled)
         = new Elem(d);
  }
};
```

```
int main() {
  int n = 4;
  Kutija k; k.napuni(n);
  cout << "Kutija: " << k << endl;
  Tabla t; Domina d;
  while (k.uzmi(d) && (t+=d));
  cout << "Tabla : " << t << endl;
}
```

```
Kutija: (2,0)(3,0)(1,2)(3,1)(2,3)(2,2)(0,1)(0,0)(1,1)(3,3)
Tabla : (2,2)(2,3)(3,1)(1,2)(2,0)(0,3)
```