ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци за ИР, ОТ и ОС Београд 15.11.2015.

Колоквијум из
Објектно оријентисаног програмирања I

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

а) Које од следећих наредби су исправне: (1) `int &x=5;` (2) `int &&y=3;`
(3) `const int &z=0;` (4)`int n[2]{1,2};` (5) `(true?int a:int b)=5;`

б) Да ли се деструктор позива аутоматски при уништавању: (1) статичког, (2) аутоматског, (3) динамичког, (4) привременог објекта?

в) У чему је разлика између иницијализације `X x=1;` и доделе вредности `x=1;` ?

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

- *(20 поена)* Понуђени ***одговор*** се састоји од текста одговора (`string`), који се задаје при стварању (подразумевано `""`), информације да ли је тачан и процентуалног удела поена који тај одговор носи (у опсегу од -100 до 100). Сви подаци се могу дохватити, а информација о тачности и процентуални удео се могу постављати (без провере узајамне конзистентности). Одговор може да се испише на главном излазу (`it<<odgovor`) у облику *текст*`:` *удео*`%`.
- *(20 поена)* ***Питање*** садржи задате: текст, број поена који носи, број понуђених одговора (подразумевано 5), као и понуђене одговоре. Ствара се без понуђених одговора, након чега се они додају појединачно (`pitanje+=odgovor`). Приликом додавања понуђеног одговора рачунају се једнаки процентуални удели свих тачних одговора додатих у дато питање (у збиру дају 100%), док сваки нетачан одговор носи -100%. Може да се одговори на питање, када се задаје низ који садржи редне бројеве понуђених одговора које испитаник сматра тачним и дужина тог низа. Тада се рачуна број освојених поена на питању, при чему је најмањи број поена који се може освојити 0. На главном излазу се исписује (`it<<pitanje`) тако што се у првом реду испише *текст*`:` *поени* (максимални које питање носи), а потом се у засебним редовима испишу садржани понуђени одговори.
- *(20 поена)* ***Тест*** се састоји од произвољног броја питања и не може да се копира ни на који начин. Ствара се празан, после чега се питања додају појединачно (`test+=pitanje`). Може да се дохвати питање са задатим редним бројем, почев од 1 (`test[i]`), при чему дохваћено питање не може да се мења. Може да се на главном излазу испише тест, тако што се у засебним редовима испишу садржана питања (`it<<test`). Може да се одговори на питање задатог редног броја *i*, почев од 1 (`test(i,niz,n)`; где је `test` објекат). Сматрати да ће корисник највише једном одговарати на једно питање. Може да се дохвати укупан број поена освојен на тесту.

(10 поена) Написати на језику *C++* **програм** који формира један тест и у њега дода неколико питања са понуђеним одговорима. Затим испише тест на главном излазу, одговори на сва питања у тесту и испише освојени број поена. 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:
`http://rti.etf.bg.ac.rs/rti/ir2oo1/index.html/`

```
#include <iostream>
#include <string>
using namespace std;
class Odgovor {
  string tks; bool tac; double proc;
public:
  Odgovor(string tst="") : tks(tst) {}
  string dohTekst() const { return tks; }
  bool dohTac() const { return tac; }
  void postTac(bool t) { tac = t; }
  void postProc(double p) {
    if (p<-100 || p>100) exit(1);    proc = p;
  }
  double dohProc() const { return proc; }
  friend ostream& operator<<(ostream& it,
                          const Odgovor& o) {
    return it << o.tks <<": " << o.proc <<"%";
  }
};
```

```
class Pitanje {
  string tks; int poeni; Odgovor* odg;
  int kap, duz, tac;
  void racunajProcente();
  void kopiraj(const Pitanje& p);
  void premesti(Pitanje& p) {
    odg=p.odg; p.odg=nullptr; kap=p.kap;
    duz=p.duz; poeni=p.poeni; tac=p.tac;
    tks=p.tks;
   }
  void brisi() { delete[] odg; }
public:
  Pitanje(string t, int p, int k = 5):
                           poeni(p), tks(t) {
    odg = new Odgovor[kap=k]; duz = tac = 0;
  }
  Pitanje(const Pitanje& p) { kopiraj(p); }
  Pitanje(Pitanje&& p) { premesti(p); }
  ~Pitanje() { brisi(); }
  Pitanje& operator=(const Pitanje& p) {
    if (this != &p) { brisi(); kopiraj(p); }
    return *this;
  }
  Pitanje& operator=(Pitanje&& p) {
    if (this != &p) { brisi(); premesti(p);}
    return *this;
  }
  Pitanje& operator+=(const Odgovor& o);
  double odgovori (int niz[], int br) const;
  friend ostream& operator<<(ostream& it,
                           const Pitanje& p);
};
void Pitanje::kopiraj(const Pitanje &p) {
  odg = new Odgovor[kap = p.kap]; duz=p.duz;
  poeni=p.poeni; tks=p.tks; tac = p.tac;
  for (int i=0; i<duz; i++) odg[i] = p.odg[i];
}
Pitanje& Pitanje::operator+=(
                          const Odgovor& o) {
  if (duz == kap) exit(1);
  if (o.dohTac()) tac++; odg[duz++] = o;
  racunajProcente(); return *this;
}
ostream& operator<<(ostream& it,
                          const Pitanje& p) {
  it << p.tks << ": " << p.poeni << endl;
  for (int i = 0; i < p.duz; i++)
    it << p.odg[i] << endl;
  return it;
}
void Pitanje::racunajProcente() {
  double procTac;
  procTac = (tac==0) ? 0 : 100./tac;
  for (int i = 0; i<duz; i++)
    odg[i].dohTac()? odg[i].postProc(procTac):
                        odg[i].postProc(-100);
}
```

```
double Pitanje::odgovori (int niz[],
                                  int br) const {
  double poeni = 0;
  for (int i = 0; i < br; i++)
    poeni += this->poeni*odg[niz[i]-1]
                                  .dohProc()/100;
  return poeni>0 ? poeni : 0;
}
```

```
class Test {
  struct Elem {
    Pitanje pit; Elem* sled;
    Elem(const Pitanje& p,Elem* s = nullptr)
                        : pit(p), sled(s) {}
  };
  Elem* prvi, *posl; double poeni = 0;
public:
  Test() { prvi = posl = nullptr; }
  Test(const Test& t) = delete;
  Test& operator=(const Test& t) = delete;
  ~Test();
  Test& operator+=(const Pitanje& p);
  const Pitanje& operator[](int ind) const;
  friend ostream& operator<<(ostream& it,
                                const Test& t);
  Test& operator()(int ind,int niz[],int br);
  double rezultat() const { return poeni; }
};
Test::~Test() {
  while (prvi) {
    Elem* stari = prvi; prvi = prvi->sled;
    delete stari;
  }
}
Test& Test::operator+=(const Pitanje& p) {
  Elem *tek = prvi;
  posl=(!prvi?prvi:posl->sled)=new Elem(p);
  return *this;
}
ostream& operator<<(ostream& it,
                              const Test& t) {
  for(Test::Elem* p=t.prvi; p; p=p->sled)
    it << p->pit << endl;
  return it;
}
const Pitanje& Test::operator[](int ind)const{
  Elem* tek = prvi; int br = 1;
  while(tek && br<ind) { tek=tek->sled; br++;}
  return tek->pit;
}
Test& Test::operator()(int ind, int niz[],
                                       int br){
  poeni += (*this)[ind].odgovori(niz, br);
  return *this;
}
```

```
int main() {
  Pitanje p1("U C++ postoje", 5, 3),
                    p2("U C-u postoje", 4, 3);
  Odgovor o1("Klase"),o2("Unije"),
                                o3("Strukture");
  o1.postTac(true); o2.postTac(true);
  o3.postTac(true);((p1 += o1) += o2) += o3;
  o1.postTac(false); ((p2 += o1) += o2) += o3;
  Test t; cout << ((t += p1) += p2);
  int niz1[] = {2}, niz2[] = {1, 3};
  t(1, niz1, 1)(2, niz2, 2);
  cout << t.rezultat() << endl; return 0;
}
```

```
U C++ postoje: 5
Klase: 33.3333%
Unije: 33.3333%
Strukture: 33.3333%

U C-u postoje: 4
Klase: -100%
Unije: 50%
Strukture: 50%

1.66667
```