ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Одсеци РТИ, ОТ, ОС и ОГ Београд, 25. 11. 2017.

Колоквијум из Објектно оријентисаног програмирања 1

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

   а) Које наредбе су неисправне (преписати програм и подвући их):

```
int main() {int i=1; int *p1=new int(10); int &r1=i; int &r2=p1;
            int *p2=p1; int *p3=r1; int *p4=&r1; int &r3=r2; int &rn[10]; }
```

   б) Да ли је дозвољено из методе класе `A` позване за објекат `A a1` приступити: (1) приватном члану објекта `A a2`; (2) јавном члану објекта `B b1`; (3) приватном члану објекта `A a1`; (4) приватном члану објекта `B b2`?

   в) Које особине стандардних оператора се подразумевају при њиховом преклапању?

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):

   - *(15 поена)* ***Карта*** се задаје знаком (*PIK, TREF, KARO, HERC*) и бројем (*K1=1, K2, K3, K4, K5, K6, K7, K8, K9, K10=10, ZANDAR=12, DAMA=13, KRALJ=14*). Може да се одреди вредност карте према следећем критеријуму: жандар, дама, краљ, 1 и 10 = 1 поен, 2 треф = 1 поен, 10 каро = 2 поена, док све остале карте вреде 0 поена. Могуће је упоредити број на две карте (`karta1>karta2`), као и вредност коју носе две карте (`karta1>>karta2`). Карту је могуће уписати у излазни ток (`it<<karta`) у облику **`karta(`***број*`,` *знак***`)`**.
   - *(40 поена)* ***Шпил*** се састоји од произвољног броја карата. Ствара се празан после чега се карте додају једна по једна на крај шпила (`spil+=karta`). Могуће је узети карту са краја шпила (`spil--`), при чему било каква грешка приликом узимања карте доводи до прекида програма. Може да се одреди укупан број карата у шпилу, укупна вредност свих карата у шпилу (`spil()`) и да се дохвати карта са највећим бројем. Два шпила је могуће упоредити на основу укупне вредности свих карата у њима (`spil1>spil2`) или на основу карте са највећим бројем (`spil1>>spil2`) – први шпил је већи од другог ако је највећа карта у првом шпилу већа од највеће карте у другом шпилу. Шпил се у излазни ток исписује (`it<<spil`) тако што се у првој линији испише **`spil(`***број_карата***`)`**, а затим се у засебним линијама исписују појединачне карте из шпила.
   - *(5 поена)* ***Тест*** се састоји од два шпила који се задају приликом стварања. Могуће је узети карту (`test--`) из првог шпила, док се у случају да је први шпил празан, карта узима из другог шпила. Тест је завршен када оба шпила постану празна. Може се проверити да ли је тест завршен.

(10 поена) Написати на језику C++ програм који створи један шпил, дода неколико карата у њега и затим га испише. Потом се направи нови шпил као копија претходног и у њега дода још једна карта, а из постојећег шпила се избаци једна карта са краја шпила. Оба шпила је потребно исписати. Након тога се направи тест помоћу претходна два шпила и редом узимају карте уз исписивање на стандардном излазу све док се не дође до краја теста. Није потребно ништа учитавати 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 Karta {
public:
  enum Znak { PIK, TREF, KARO, HERC };
  enum Broj { K1 = 1, K2, K3, K4, K5, K6, K7, K8,
              K9, K10, ZANDAR = 12, DAMA, KRALJ };
  Karta(Znak zz, Broj bb) :z(zz), b(bb) {}
  int vrednost()const {
       return b == K2 && z == TREF ? 1 :
       b == K10 && z == KARO ? 2 : b >= 10 ?1:0;
  }
  friend bool operator>(const Karta &k1, const
              Karta &k2) { return k1.b > k2.b;}
  friend bool operator>>(const Karta &k1, const
       Karta &k2){
       return k1.vrednost() > k2.vrednost();
  }
  friend ostream& operator<<(ostream &ot, const
       Karta &k) {
   string znakovi[]={"pik","tref","karo","herc"};
   string brojevi[]={"zandar", "dama","kralj"};
   ot << "karta(";  if (k.b < 12) ot << k.b;
        else ot << brojevi[k.b - 12];
   return ot << ", " << znakovi[k.z] << ")";
  }
private: Znak z; Broj b;
};
```

```
class Spil {
public: Spil() = default;
  Spil(const Spil &s) { kopiraj(s); }
  Spil(Spil &&s) { premesti(s); }
  ~Spil(){ brisi(); }
  Spil& operator=(const Spil &s) {
       if (this != &s) { brisi(); kopiraj(s); }
       return *this;}
  Spil& operator=(Spil &&s) {
       if (this != &s) { brisi(); premesti(s); }
       return *this;}
  Spil& operator+=(const Karta &k);
  Karta operator--(int i);
  int brojKarata()const;
  int operator()()const;
  Karta najveca()const;
  friend bool operator>(const Spil &s1, const
     Spil &s2) { return s1() > s2();}
  friend bool operator>>(const Spil &s1, const
     Spil &s2) {
       return s1.najveca() > s2.najveca();}
  friend ostream& operator<<(ostream &ot, const
     Spil &s) {
    ot << "spil(" << s.brojKarata() << ")\n";
    for (Elem*tek = s.prvi; tek; tek = tek->sled)
       ot << tek->k<<endl;
    return ot;
  }
private: void kopiraj(const Spil &s);
  void premesti(Spil &s) {
    prvi = s.prvi; posl = s.posl;
    s.prvi = s.posl = nullptr;}
  void brisi();
  struct Elem {
    Karta k; Elem *sled;
    Elem(const Karta &kk):k(kk),sled(nullptr){}
  };
  Elem *prvi = nullptr, *posl=nullptr;
};
void Spil::kopiraj(const Spil &s) {
  prvi = posl = nullptr;
  for (Elem* tek = s.prvi; tek; tek = tek->sled)
    *this += tek->k;
}
void Spil::brisi() {
  while (prvi) { Elem *stari = prvi;
    prvi = prvi->sled; delete stari;}}
```

```
Spil& Spil::operator+=(const Karta &k) {
  posl=(!prvi?prvi:posl->sled) = new Elem(k);
  return *this;
}
Karta Spil::operator--(int i) {
  if (prvi == nullptr) exit(2);
  Karta k = posl->k;
  Elem *tek = prvi, *pret = nullptr;
  while(tek->sled){pret = tek;tek = tek->sled;}
  delete tek;
  if (!pret) { prvi = posl = nullptr;}
  else { pret->sled=nullptr; posl=pret;}
  return k;
}
int Spil::brojKarata()const {
  int br = 0;
  for(Elem*tek=prvi;tek;tek=tek->sled)br++;
  return br;
}
int Spil::operator()()const {
  int vr = 0;
  for(Elem*tek = prvi; tek; tek = tek->sled)
    vr += tek->k.vrednost();
  return vr;
}
Karta Spil::najveca()const {
  if (prvi == nullptr) exit(3);
  Karta najveca = prvi->k;
  for(Elem*tek=prvi->sled;tek;tek=tek->sled)
    if (tek->k > najveca) najveca = tek->k;
  return najveca;
}
```

```
class Test {
public:Test(Spil &p,Spil &d):prvi(p), drugi(d){}
  Karta operator--(int i) {
    if (zavrsen()) exit(4);
    return prvi.brojKarata()>0?prvi--:drugi--;
  }
  bool zavrsen()const { return prvi.brojKarata()
    == 0 && drugi.brojKarata() == 0; }
private: Spil prvi, drugi;
};
```

```
int main() {
  Spil prvi;
  Karta k1(Karta::Znak::HERC, Karta::Broj::DAMA),
        k2(Karta::Znak::PIK, Karta::Broj::K2),
        k3(Karta::Znak::TREF, Karta::Broj::K7);
  cout << ((prvi += k1) += k2) << endl;
  Spil drugi(prvi); drugi += k3;
  prvi--; cout << prvi << endl << drugi;
  Test i(prvi, drugi); cout << endl;
  while (!i.zavrsen()) cout << i-- << endl;
  return 0;
}
```

```
spil(2)
karta(dama, herc)
karta(2, pik)

spil(1)
karta(dama, herc)

spil(3)
karta(dama, herc)
karta(2, pik)
karta(7, tref)

karta(dama, herc)
karta(7, tref)
karta(2, pik)
karta(dama, herc)
```