ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за ИР Београд 23.11.2014.

Први колоквијум из
Објектно оријентисаног програмирања I

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

а) Који стандардни оператор се користи за динамичко стварање низа података неког типа `T` (навести пример позива) и шта је резултат операције (у случају успеха, односно неуспеха)?

б) Којег типа треба да буде први параметар (формални аргумент) копирајућег, а којег типа премештајућег конструктора класе `X`? Одговорити речима и навођењем ознаке типа.

в) Која наредба позива које функције класе (навести називе и декларације функција које најбоље одговарају): (а) `X x=5;` (б) [`X x;`…] `x=3;` (в) [`X x1;`…] `X x2(x1);` (г) `X x=X();` Део у угластим заградама објашњава контекст одговарајуће наредбе.

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)* Лист папира има редни број и текст (`string`) који се задају при стварању. Редни број може да се дохвати. Могу да се упореде два листа (`list1<list2`, односно `list1==list2`; упоређују се само редни бројеви листова), да му се дода задати текст (`list+=tekst`) и да се лист упише у излазни ток (`it<<list`) у облику *tekst*`(`*redni_broj*`)`.
- *(20 поена)* Књига има задати назив који може да се дохвати и може да садржи произвољан број листова. Ствара се празна после чега се листови додају појединачно (`knjiga+=list`), по растућем редоследу редних бројева. Додавање листа са редним бројем који већ постоји у књизи, замењује постојећи лист. Може да се дохвати број листова у књизи, да се дохвати лист са задатим редним бројем (`knjiga[i]`; дохваћени лист не може да се мења; редни број листа који не постоји у књизи прекида програм) и да се упише књига у излазни ток (`it<<knjiga`), тако што се у првом реду наведе њен назив, а потом у засебним редовима њени листови.
- *(20 поена)* Полица може да садржи највише 10 књига (по адреси). Ствара се празна, након чега се књиге могу појединачно стављати на њу (`polica+=knjiga`). Са полице се може узети прва пронађена књига задатог назива. Покушај стављања књиге на пуну полицу, као и узимања књиге која није на полици се игнорише. Може да се упише полица у излазни ток (`it<<polica`), тако што се књиге на њој наводе у засебним редовима. Полица не може да се копира ни премешта, ни иницијализацијом ни доделом.

(10 поена) Написати на језику *C++* **програм** који направи једну полицу, неколико књига, дода неколико листова у сваку књигу, потом направљене књиге стави на полицу и на крају испише полицу на главном излазу. 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 <cstdlib>
#include <string>
#include <iostream>
using namespace std;

class List{
  int br;  string tek;
public:
  List(string t, int b){   br = b; tek = t; }
  int dohvBroj() const { return br;}
  friend bool operator<(const List& l1,
    const List& l2) { return l1.br<l2.br; }
  friend bool operator==( const List& l1,
    const List& l2) { return l1.br==l2.br; }
  List& operator+=(string s)
    { tek += s; return *this; }
  friend ostream& operator<<(ostream& it,
                             const List& l)
   {return it << l.tek << '(' << l.br << ')';}
};
```

```
class Knjiga{
  string naziv;
  struct Elem {
    List list;  Elem* sled;
    Elem(const List& l, Elem* s=nullptr)
       : list(l), sled(s) { }
  };
  Elem *prvi=nullptr, *posl=nullptr;
  int   duz=0;
  void  kopiraj(const Knjiga& k);
  void  premesti(Knjiga& k) {
    prvi = k.prvi; posl = k.posl;
    naziv = k.naziv; duz = k.duz;
    k.prvi = k.posl = nullptr; k.duz = 0;
  }
  void  brisi();
public:
  Knjiga(string n) { naziv = n; }
  Knjiga(const Knjiga& k){ kopiraj(k); }
  Knjiga(Knjiga&& k){ premesti(k); }
  ~Knjiga(){ brisi(); }
  Knjiga& operator=(const Knjiga& k) {
    if (this != &k){ brisi(); kopiraj(k); }
    return *this;
  }
  Knjiga& operator=(Knjiga&& k) {
    if (this != &k){ brisi(); premesti(k);}
    return *this;
  }
  string dohvNaziv() const { return naziv;}
  Knjiga& operator+=(const List& l);
  int dohvatiDuz() const { return duz;}
  const List& operator[](int i) const;
  friend ostream& operator<<(ostream& it,
                             const Knjiga& k);
};
void  Knjiga::kopiraj(const Knjiga& k){
  prvi = posl = nullptr; duz = k.duz;
  for (Elem* tek=k.prvi; tek; tek=tek->sled)
      posl = (posl ? posl->sled : prvi) =
                          new Elem(tek->list);
}
void  Knjiga::brisi(){
  while (prvi) { Elem* stari = prvi;
            prvi = prvi->sled; delete stari; }
  posl = nullptr; duz = 0;
}
Knjiga& Knjiga::operator+=(const List& l){
  Elem *pret=nullptr, *tek=prvi;
  while(tek && tek->list.dohvBroj() <
       l.dohvBroj()){ pret=tek;tek=tek->sled;}
  if(tek && tek->list.dohvBroj() ==
                 l.dohvBroj()) tek->list = l;
    else { (pret ? pret->sled : prvi) =
           (tek ? new Elem(l,tek) : posl =
            new Elem(l,tek));
    duz ++;}
    return *this;
}
const List& Knjiga::operator[](int i) const{
  Elem *tek=prvi;
  while(tek && tek->list.dohvBroj() != i)
    tek = tek->sled;
  if (!tek) exit(1);
  return tek->list;
}
ostream& operator<<(ostream& it,
     const Knjiga& k){ it << k.naziv << endl;
  for(Knjiga::Elem *tek=k.prvi; tek;
       tek = tek->sled){it<<tek->list<<endl;}
  return it;
}
```

```
const int N = 10;
class Polica{
  Knjiga* k[N];  int pop = 0;
public:
  Polica()
   { for(int i=0; i<N; i++) k[i]=nullptr; }
  Polica(const Polica& p)= delete;
  Polica& operator=( const Polica&)= delete;
  Polica& operator+=(Knjiga& kk);
  Knjiga* uzmi(string naziv);
  friend ostream& operator<<(ostream& it,
                             const Polica& p);
};
Polica& Polica::operator+=(Knjiga& kk){
  if(pop == N) return *this;
  int i = 0; while(k[i]) i++;
  k[i] = &kk; pop++;
  return *this;
}
Knjiga* Polica::uzmi(string naziv){
  int i = 0;
  while(i<N){
     if(k[i]&&k[i]->dohvNaziv()==naziv) break;
     i++;
  }
  if(i>=N) return nullptr;
  Knjiga* knjiga = k[i]; k[i]=nullptr; pop--;
  return knjiga;
}
ostream& operator<<(ostream& it,
                    const Polica& p){
  for(int i=0; i<N; i++){
     if(p.k[i]) it << *p.k[i] << endl;
  }
  return it;
}
```

```
int main() {
  Polica p;
  Knjiga k1("k1"), k2("k2"), k3("k3");
  List l1("list1", 1), l2("list2", 2),
       l3("list3", 3), l4("list4", 3),
       l5("list5", 5), l6("list6", 6);
  k1+=l1; k1+=l3; k1+=l2; k2+=l5; k3+=l6;
  p+=k1;   p+=k3;   p+=k2; cout << p;
  return 0;
}
```

```
k1
list1(1)
list2(2)
list3(3)

k3
list6(6)

k2
list5(4)
```