ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсеци ИР, ОТ, ОС и ОГ Београд 24.11.2018.

Први колоквијум из Објектно оријентисаног програмирања 1

1) *(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)* ***Воћка*** има назив, реалан годишњи приход и целобројан животни век изражен у годинама који се задају при стварању и тренутну старост воћке која је при стварању нула. Могу да се дохвате годишњи приход и преостале године живота. Воћка може да остари годину дана (`vocka++`). Воћку је могуће уписати у излазни ток (`it<<vocka`) у облику *назив* **(***тренутна_старост* **/** *животни_век***) :** *годишњи_приход*.
- (*30 поена*) ***Воћњак*** садржи аутоматски генерисан јединствен целобројан идентификатор који може да се дохвати и матрицу показивача на воћке. Ствара се празан, задатих димензија матрице, после чега се воћке додају једна по једна (`vocnjak+=vocka`) на прву слободну локацију претражујући матрицу по редовима. Покушај додавања у попуњену матрицу се игнорише. Могућ је једногодишњи прогрес воћњака (`vocnjak++`) током којег све воћке у воћњаку остаре годину дана, а затим се из воћњака избаце све воћке које су достигле крај животног века. Могуће је одредити укупан годишњи приход воћњака као збир годишњих прихода свих воћки у воћњаку. Два воћњака је могуће упоредити на основу годишњег прихода (`vocnjak1<vocnjak2`). Воћњак се у излазни ток исписује (`it<<vocnjak`) у облику **V_***идент* **:** *годишњи_приход* након чега се појединачне воћке исписују у матричном формату тако што се у засебним редовима исписују све воћке из одговарајућег реда матрице. За празна поља матрице треба исписати **`prazno_mesto`**. Воћке у појединачним редовима треба да буду одвојене једним знаком за табулацију.
- *(15 поена)* ***Плантажа*** се састоји из произвољног броја воћњака и не може да се копира ни на који начин. Ствара се празна, након чега се воћњаци додају један по један (`plantaza+=vocnjak`). Могуће је извршити предикцију *k*-годишњег прихода свих воћњака на плантажи (`plantaza(k)`) при чему се на стандардном излазу, у засебним редовима за сваку годину, исписује идентификатор и годишњи приход воћњака са највећим годишњим приходом за ту годину.

(10 поена) Написати на језику C++ **програм** који формира воћњак и у њега дода неколико воћки. Затим ствара нов воћњак као копију претходног и у оба воћњака дода још неколико различитих воћки. Након тога испише оба воћњака на стандардни излаз. Коначно ствара једну плантажу у коју се додају претходна два воћњака и симулира предикцију прихода од неколико година. Није потребно ништа учитавати с главног улаза.

НАПОМЕНЕ:
а) Колоквијум траје **120** минута. Тест траје 15 минута, а израда задатка 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: http://rti.etf.bg.ac.rs/rti/ir2oo1/index.html/

```
#include <iostream>
#include <string>
using namespace std;
class Vocka {
public:
  Vocka(string n, int zv, double gp) :naziv(n),
       zivVek(zv), godPrihod(gp), starost(0) {}
  double dohvGodPrihod()const {return godPrihod;}
  int dohvPreostalo() const
    {return zivVek - starost;}
  Vocka operator++(int){ Vocka v = *this;
                         starost++; return v;}
  friend ostream& operator<<(ostream& os, const
    Vocka& v)
    {return os << v.naziv << '(' << v.starost
     << '/' << v.zivVek << "): " << v.godPrihod;}
private:
  string naziv; int zivVek;
  int starost; double godPrihod;
};
```

```
class Vocnjak {
public:
  Vocnjak(int m, int n);
  Vocnjak(const Vocnjak& p) { kopiraj(p); }
  Vocnjak(Vocnjak&& p) { premesti(p); }
  ~Vocnjak() { brisi(); }
  Vocnjak& operator=(const Vocnjak& p) {
    if (this != &p) { brisi(); kopiraj(p); }
    return *this;}
  Vocnjak& operator=(Vocnjak&& p) {
    if (this != &p) { brisi(); premesti(p); }
    return *this;}
  int dohvId() const { return id; }
  Vocnjak& operator+=(const Vocka&);
  Vocnjak operator++(int);
  double prihod() const;
  friend bool operator<(const Vocnjak& p1, const
   Vocnjak& p2){return p1.prihod()<p2.prihod();}
  friend ostream& operator<<(ostream& os, const
    Vocnjak& p);
private:
  void kopiraj(const Vocnjak&);
  void premesti(Vocnjak& p) { mat = p.mat;
    brRed = p.brRed; brKol = p.brKol;
    p.mat = nullptr; p.brKol = p.brRed = 0;}
  void brisi();
  static int posId; int id = ++posId;
  int brRed, brKol; Vocka*** mat;
};
Vocnjak::Vocnjak(int m, int n):brRed(m),brKol(n){
  mat = new Vocka**[brRed];
  for (int i = 0; i < brRed; i++){
    mat[i] = new Vocka*[brKol];
       for (int j = 0; j < brKol; j++)
         mat[i][j] = nullptr;}}
Vocnjak& Vocnjak::operator+=(const Vocka &v){
  for (int i = 0; i < brRed; i++)
   for (int j = 0; j < brKol; j++)
     if (!mat[i][j]){ mat[i][j] = new Vocka(v);
       return *this;} return *this;}
Vocnjak Vocnjak::operator++(int){
  Vocnjak v = *this;
  for (int i = 0; i < brRed; i++)
    for (int j = 0; j < brKol; j++)
      if (mat[i][j]){ (*(mat[i][j]))++;
        if (mat[i][j]->dohvPreostalo() <= 0){
         delete mat[i][j]; mat[i][j] = nullptr;}}
  return v;}
double Vocnjak::prihod() const{
  double broj = 0;
  for (int i = 0; i < brRed; i++)
    for (int j = 0; j < brKol; j++)
      if (mat[i][j])
        broj += mat[i][j]->dohvGodPrihod();
  return broj;}
```

```
void Vocnjak::kopiraj(const Vocnjak &p){
  brRed = p.brRed; brKol = p.brKol;
  mat = new Vocka**[brRed];
  for (int i = 0; i < brRed; i++) {
    mat[i] = new Vocka*[brKol];
    for (int j = 0; j < brKol; j++)
      mat[i][j] = p.mat[i][j] ?
      new Vocka(*p.mat[i][j]) : nullptr;}}
void Vocnjak::brisi(){
  for (int i = 0; i < brRed; i++) {
    for (int j = 0; j < brKol; j++) {
      delete mat[i][j];}
    delete [] mat[i];}
  delete [] mat; mat = nullptr;}
ostream& operator<<(ostream& os, const
                    Vocnjak& p){
  os << "V_" << p.id << ':' << p.prihod() <<endl;
  for (int i = 0; i < p.brRed; i++) {
    for (int j = 0; j < p.brKol; j++)
      if(p.mat[i][j])os << *(p.mat[i][j]) <<'\t';
      else os << "prazan_red" << '\t';
    os << endl;}
  return os;}
int Vocnjak::posId = 0;
```

```
class Plantaza {
public:
  Plantaza() = default;
  Plantaza(const Plantaza&) = delete;
  Plantaza& operator=(const Plantaza&) = delete;
  ~Plantaza();
  Plantaza& operator+=(Vocnjak &v) {
    posl = (!prvi ? prvi : posl->sled)
         = new Elem(v); return *this;}
  void operator()(int k) const;
private:
  struct Elem {
    Vocnjak* voc; Elem* sled;
    Elem(Vocnjak& v) :voc(&v), sled(nullptr) {}
  };
  Elem* prvi = nullptr, *posl = nullptr;
};
Plantaza::~Plantaza(){
  while(prvi) {Elem* stari=prvi; prvi=prvi->sled;
               delete stari;} posl = nullptr;}
void Plantaza::operator()(int k) const {
  if(prvi){
    for (int i = 0; i < k; i++) {
      double max=0; int id = prvi->voc->dohvId();
      for (Elem* tek = prvi; tek; tek=tek->sled){
        Vocnjak* cpy = new Vocnjak(*tek->voc);
        for (int j= 0; j < i; j++)(*cpy)++;
        if (max<cpy->prihod()){max=cpy->prihod();
          id = tek->voc->dohvId();}
        delete cpy;}
      cout << i+1 <<". "<< id << ':' << max
           << endl;}}}
```

```
int main() {
  Vocka v1("Jabuka", 4, 10);
  Vocka v2("Kruska", 3, 22);
  Vocka v3("Visnja", 2, 25);
  Vocnjak voc(2, 2); voc += v1;
  Vocnjak vocCpy(voc); vocCpy += v3; voc += v2;
  cout << voc << endl << vocCpy << endl;
  Plantaza pred; (pred += voc) += vocCpy;
  pred(3); return 0;
}
```

```
V_1:32
Jabuka(0/4): 10  Kruska(0/3): 22
prazno_mesto     prazno_mesto

V_2:35
Jabuka(0/4): 10  Visnja(0/2): 25
prazno_mesto     prazno_mesto

1. 2:35
2. 2:35
3. 1:32
```