ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за СИ

Београд 02.12.2017.

Други колоквијум из
Објектно оријентисаног програмирања I

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

- *(10 поена)* ***Поље*** се задаје колоном (једнословна ознака која може имати вредност: *A, B, C, D, E, F, G, H*) и редом (цео број у опсегу од 1 до 8). Могуће је одредити да ли су два поља једнака (`polje1==polje2`), да ли се налазе у истом реду или истој колони (`polje1+polje2`) и да ли се налазе на истој дијагонали (`polje1/polje2`).
- *(30 поена)* ***Фигура*** се задаје бојом (*BELA*, *CRNA*) и пољем на којем се налази. Може да се одреди да ли се фигура са текућег поља може у једном потезу померити на задато поље (није потребно узимати у обзир препреке приликом померања), као и да ли фигура може у једном потезу да „поједе“ другу задату фигуру – фигуру је могуће појести уколико је у датом потезу могуће стати на поље на коме се она налази и уколико је фигура друге боје. Фигура се може померити на задато поље (`figura(polje)`) при чему грешка приликом померања доводи до завршетка програма. Може се дохватити боја, поље, једнословна ознака, као и целобројна вредност фигуре. Две фигуре је могуће упоредити на основу вредности (`figura1>figura2`). Фигура се у излазни ток исписује (`it<<figura`) у облику *једнословна_ознака_фигуре*, при чему се за беле фигуре исписује велико, а за црне мало слово.
- ***Топ*** је фигура која се може кретати само вертикално (по колони) или хоризонтално (по реду). Ознака за топа је *T*, а његова вредност је 5. ***Ловац*** је фигура која се може кретати само дијагонално. Ознака за ловца је *L*, а његова вредност је 3. ***Скакач*** је фигура која се може кретати тако да се у једном потезу помери за тачно две колоне и један ред или једну колону и два реда, произвољним редоследима. Ознака за скакача је *S*, а његова вредност је 2.
- *(30 поена)* ***Шаховска табла*** садржи матрицу (8 x 8) показивача на фигуре. Ствара се празна након чега се фигуре додају једна по једна (`tabla+=figura`). Могуће је померити фигуру са једног на друго задато поље (`tabla(sa_polja,na_polje)`), при чему је повратна вредност индикатор да ли је „поједена“ нека фигура. Може се дохватити показивач на фигуру која се налази на задатом пољу (`tabla[polje]`). Табла се у излазни ток исписује (`it<<tabla`) у 8 редова, при чему сваки ред садржи 8 карактера који представљају једнословне ознаке фигура или знак **_** у случају да на задатом пољу нема фигуре.

Приложена је главна функција која створи једну шаховску таблу, дода неколико фигура на таблу, испише таблу, а затим одигра неколико потеза, при чему се после сваког потеза испише да ли је поједена нека фигура.

НАПОМЕНЕ:

а) За израду задатка на располагању је **100** минута.

б) Сваку класу стављати у засебне датотеке (обавезно `.h`, по потреби и `.cpp`)

в) Рад се предаје на мрежном диску `Rad(L:)`.

г) На располагању је документација на *Web*-у **на адресама: http://www.cplusplus.com/ и http://en.cppreference.com**

д) Није дозвољено имати поред себе друге материјале, нити уз себе имати електронске уређаје, без обзира да ли су укључени или искључени.

ђ) Резултати колоквијума биће објављени на *Web*-у на адреси: `http://rti.etf.rs/rti/ir2oo1/index.html/`

```
#include<iostream>
#include<string>
using namespace std;
class Polje{
public:Polje(char kk,int rr){if(kk>'H'||kk<'A'
      ||rr<1||rr>8)exit(6);r=rr;k = kk;}
  char kolona()const{ return k; }
  int red()const{ return r; }
  friend bool operator+(const Polje &a, const
    Polje &b){return a.k == b.k || a.r == b.r;}
  friend bool operator/(const Polje &a, const
    Polje &b){
      return abs(a.k - b.k) == abs(a.r - b.r);}
  friend bool operator==(const Polje &a, const
    Polje &b){
     return a.r==b.r && a.k==b.k;}
private: char k; int r;
};
```

```
class Figura{
public: enum Boja{ BELA, CRNA };
  Figura(Boja bb, Polje pp) :b(bb), p(pp){}
  virtual ~Figura() {}
  Polje polje()const{ return p; }
  Boja boja()const{ return b; }
  virtual bool mozePolje(Polje p)const = 0;
  bool mozeFiguru(Figura &f){
    return mozePolje(f.p) && b != f.b;}
  void operator()(Polje p){
   if (mozePolje(p)) this->p = p; else exit(1);}
  virtual char oznaka()const = 0;
  virtual int vrednost()const = 0;
  virtual Figura* kopija()const = 0;
  friend bool operator>(const Figura &a, const
    Figura &b){
     return a.vrednost() > b.vrednost();}
  friend ostream& operator<<(ostream &ot, const
    Figura &f){
     return ot << (char)(f.oznaka() + (f.b ==
      Figura::CRNA ? 32 : 0));}
protected: Boja b; Polje p;
};
```

```
class Top :public Figura{
public: Top(Boja b, Polje p) :Figura(b, p){}
  bool mozePolje(Polje p)const override{
   return this->p + p;}
  char oznaka()const override{ return 'T'; }
  int vrednost()const override { return 5; }
  Top* kopija()const override{
    return new Top(*this);}};
```

```
class Lovac :public Figura{
public: Lovac(Boja b, Polje p) :Figura(b, p){}
  bool mozePolje(Polje p)const override{
     return this->p / p;}
  char oznaka()const override{ return 'L'; }
  int vrednost()const override { return 3; }
  Lovac* kopija()const override {
     return new Lovac(*this);}};
```

```
class Skakac :public Figura{
public: Skakac(Boja b, Polje p) :Figura(b, p){}
  bool mozePolje(Polje m)const override{
     return abs(p.red()-m.red())+abs(p.kolona()
    -m.kolona())==3 && p.red()!=m.red() &&
    p.kolona()!=m.kolona();}
  char oznaka()const override{ return 'S'; }
  int vrednost()const override { return 2; }
  Skakac* kopija()const override {
     return new Skakac(*this);}};
```

```
class Tabla{
public:
   Tabla();
   Tabla(const Tabla &t){ kopiraj(t); }
   Tabla(Tabla &&t){ premesti(t); }
   Tabla& operator=(const Tabla &t){
      if (this != &t){ brisi(); kopiraj(t); }
      return *this;}
   Tabla& operator=(Tabla &&t){
      if (this != &t) { brisi(); premesti(t); }
      return *this;}
   ~Tabla(){ brisi(); }
   Tabla& operator+=(Figura &f){
      if ((*this)[f.polje()]) exit(2);
      (*this)[f.polje()] = f.kopija();
      return *this;}
   Figura*& operator[](const Polje p){
    return figure[p.kolona()-'A'][p.red()-1]; }
   Figura* operator[](const Polje p)const {
    return figure[p.kolona()-'A'][p.red()-1]; }
   bool operator()(Polje sa, Polje na){
      Figura *f = (*this)[sa];
      if(!f->mozePolje(na)) exit(3);
      if((*this)[na]&&(*this)[na]->boja()==
          f->boja()) exit(4);
      bool zauzeto = (*this)[na] != nullptr;
      if(zauzeto) delete (*this)[na];
      (*this)[sa] = nullptr;
      (*f)(na);(*this)[na] = f;
      return zauzeto;
   }
   friend ostream& operator<<(ostream &ot,
      const Tabla &t){
      for (int j = 8; j >0; j--){
       for (char i = 'A'; i <= 'H'; i++){
         Polje p = Polje(i, j);
         if (t[p])cout << *t[p];else cout<<'_';
       }ot << endl;}
      return ot;
   }
private:
   void kopiraj(const Tabla &t);
   void premesti(Tabla &t);
   void brisi();
   Figura* figure[8][8];
};
Tabla::Tabla(){
   for (int i = 0; i < 8; i++){
      for (int j = 0; j < 8; j++)
         figure[i][j] = nullptr;}
}
void Tabla::kopiraj(const Tabla &t){
   for (int i = 0; i < 8; i++){
    for(int j=0;j<8;j++)if(t.figure[i][j])
     figure[i][j]=t.figure[i][j]->kopija();
     else figure[i][j] = nullptr;}}
void Tabla::premesti(Tabla &t){
   for (int i = 0; i < 8; i++){
      for (int j = 0; j < 8; j++){
         figure[i][j] = t.figure[i][j];
         t.figure[i][j] = nullptr;}}}
void Tabla::brisi(){
   for (int i = 0; i < 8; i++){
      for (int j = 0; j < 8; j++)
         if (figure[i][j]) {
            delete figure[i][j];}}}
```

```
int main(){
   Tabla t; Polje f7('F', 7), h7('H', 7);
   Lovac lb(Figura::BELA, Polje('C', 2));
   Top tb(Figura::BELA, Polje('F', 2));
   Skakac sb(Figura::BELA, Polje('D', 1));
   Lovac lc(Figura::CRNA, f7);
   Top tc(Figura::CRNA, Polje('H', 8));
   Skakac sc(Figura::CRNA, Polje('B', 8));
   cout<<((((((t+=lb)+=tb)+=sb)+=lc)+=tc)+=sc);
   cout<<endl<<(t(Polje('F',2),f7)?"DA ":"NE ");
   cout<<(t(Polje('H', 8), h7)?"DA ":"NE ");
   cout<<(t(Polje('C', 2), h7)?"DA ":"NE ");
   return 0;}
```

Ispis programa:

```
_s_____t
_____l__
________
________
________
________
__L__T__
___S____
DA NE DA
```