# PROGRAMIRANJE ZA ELEKTROTEHNIKE II

Rešitev 2. testa, 19. 5. 2009

1. (10) Napišite **rekurzivno funkcijo** za izračun fakultete naravnega števila *n* vključno s številom 0.

Glejte na primer prosojnice.

2. (30) Napišite definicijo funkcije `kopiraj_ang` za štetje angleških črk v vhodni tekstovni datoteki in izpisovanje nekoliko spremenjene vsebine vhodne datoteke v izhodno. Funkcija naj deluje tako: Če ne more odpreti vhodne ali izhodne datoteke, naj vrne vrednost `false`, sicer pa vrednost `true`. Če uspe odpreti datoteki, naj iz vhodne datoteke bere in sproti izpisuje vrstice v izhodno datoteko, tako da namesto vsake velike ali male črke q, x, y in w izpiše zvezdico. Sproti naj šteje, koliko je v vhodni datoteki omenjenih angleških črk in število na koncu vrne v tretjem argumentu.

Funkcija naj bo torej logičnega tipa in naj ima tri argumente: ime vhodne datoteke, ime izhodne datoteke ter argument, v katerem naj vrne zapisano število velikih in malih črk q, x, y in w v vhodni datoteki. Imeni vhodne in izhodne datoteke naj bosta tipa `string` (kakor vas je večina lepo naredila v 6. DN).

Primer uporabe funkcije:

```
int main() {
   int stAng = 0;
   if (kopiraj_ang(˝vhod.txt˝, ˝izhod.txt˝, stAng)) {
       if (stAng == 0)
            cout << ˝Izhodne datoteke ni treba
                     popravljati.˝ << endl;
       else cout << ˝Naredi popravke v izhodni datoteki na ˝
                 << stAng << ˝mestih.˝ << endl;
   }
   return 0;
}
```

```
bool kopiraj_ang(string vhdat, string izhdat, int& stAng)
{
   string vrstica;
   ofstream izhodna_datoteka(izhdat.c_str());
   if (!izhodna_datoteka.is_open())
      return false;
   ifstream vhodna_datoteka(vhdat.c_str());
   if (!vhodna_datoteka.is_open())
      return false;
   while (!vhodna_datoteka.eof()) {
      getline(vhodna_datoteka, vrstica, '\n');
      for (int i = 0; i < vrstica.size(); i++) {
         if (toupper(vrstica[i]) == 'Q' ||
             toupper(vrstica[i]) == 'X' ||
             toupper(vrstica[i]) == 'Y' ||
             toupper(vrstica[i]) == 'W')
             {
              vrstica[i] = '*';
              stAng++;
             }
      }
      izhodna_datoteka << vrstica << endl;
   }
   izhodna_datoteka.close();
   vhodna_datoteka.close();
   return true;
}
```

3. (1) Tu napišite vključitveni stavek, nujno potreben za omogočitev dela z datotekami v programu iz prejšnje naloge.

```
#include <fstream>
```

4. (35) V tej nalogi vzemite, da je celotna koda napisana v eni datoteki. Ni treba pisati vključitvenih stavkov ipd., ampak samo zahtevano kodo. Uporaba ključne besede friend ni dovoljena.

Napišite definicijo razreda `Ura`, ki ima privatna atributa `ure` in `minute`. Implementirajte konstruktor za nastavitev ure, tj. števila ur in minut, na izbrano vrednost. Implementiran naj bo kar v definiciji razreda.

Napišite še definicijo razreda `Budilka`, ki naj tudi kar vsebuje implementacije konstruktorjev oziroma metod. Izpeljite ga iz razreda `Ura` (uporabite javno izpeljavo). Izpeljani razred naj ima privatne atribute `ura_bujenja`, `minuta_bujenja` in boolov atribut `zvoni`. Za vsakega definirajte metodo za vračanje njegove vrednosti. Z uporabo nadrazreda implementirajte konstruktor, ki poleg nastavitve trenutnega časa (ure, minute) na izbrano vrednost omogoča še nastavitev časa zvonjenja (ura, minuta) ter vklop (logična vrednost 'pravilno') ali izklop (logična vrednost 'nepravilno') zvonjenja.

Zunaj razreda `Budilka` implementirajte prekrivni operator << za izpis nastavitve časa bujenja in ali je zvonjenje vklopljeno. Operator naj zapovrstjo v eni vrstici izpiše vrednosti ure, minute ter glede na vrednost vklopa besedo VKLOPLJENO ali IZKLOPLJENO, ločene z dvopičji.

Napišite še funkcijo main, ki definira en objekt tipa budilka s po želji, a smiselno nastavljenimi vrednostmi trenutnega časa in vrednostmi v zvezi z zvonjenjem. Slednje naj izpiše z uporabo prekrivnega operatorja.

```
class Ura {
private:
   int ure;
   int minute;
public:
   Ura(int ure, int minute) {
      this->ure = ure;
      this->minute = minute;
   }
};

class Budilka: public Ura {
private:
   int ura_bujenja;
   int minuta_bujenja;
   bool zvoni;
public:
   Budilka(int ure, int minute, int ura_bujenja,
        int minuta_bujenja, bool zvoni): Ura(ure, minute) {
      this->ura_bujenja = ura_bujenja;
      this->minuta_bujenja = minuta_bujenja;
      this->zvoni = zvoni;
   }
   int getUraBuj() {return ura_bujenja;}
   int getMinBuj() {return minuta_bujenja;}
   bool getZvoni() {return zvoni;}
};

ostream& operator<<(ostream& out, Budilka budilka) {
   out << budilka.getUraBuj() << ":"
       << budilka.getMinBuj() << ":";
   if (budilka.getZvoni())
      out << "VKLOPLJENO" << endl;
   else
      out << "IZKLOPLJENO" << endl;
   return out;
}

int main() {
   Budilka budilka1(5, 35, 5, 40, true); // samo se 5 minut!
   cout << budilka1;
   return 0;
}
```