# Zadaća 2
# Objektno orjentirano programiranje

Novembar 22, 2017

## Sadržaj

# 1 Napomena

Rok za završetak zadaće je 06.12.2017.

# 2 Problem 1

Definirati zaglavlja i implementaciju za slijedeće module:

- *IOrijecio.h* Modul za unos i ispis vektora `string`-ova. Modul treba da ima dvije funkcije jednu za unos drugu za ispis vektora.
- *dodavanje.h* Modul koji sadrži dvije funkcije. Prva verzija funkcije uzima string `a`, karakter `b` i cijeli broj `c`. Funkcija nadodaje `c` karaktera `b` na lijevoj stani stringa `a`. Druga verzija funkcije uzima string `a` i cijeli broj `c`. Fukcija nadodaje `c` karaktera `' '` na lijevoj strani stringa `a`. Koristiti overloading ulaznih parametara.
- *MAXrijec.h* Modul sadrži jednu funkciju koja nalazi riječ unutar vektora riječi koja ima maksimalnu dužinu. Funkcija vraća pronađenu riječ.

Za sve funkcije u gornjim modulima odabrati najadekvatniji način za prosljeđivanje parametara. Napisati glavni program koji koristeći gore navedene module čita niz riječi a zatim ispisuje učitani niz sortiran po dužini riječi i poravnat na desnu stranu kao u slijedećem primjeru pozivanja programa.

```
$ ./zad1.exe
adfa
gadgdgda
dsf
adfaffadfafa
adf
ds
^Z
          ds
         dsf
         adf
        adfa
    gadgdgda
adfaffadfafa
```

# 3 Problem 2

Napisati funkciju `translate_each_element` koja će da izvrši transliranje svakog pojedinačnog elementa vektora realnih brojeva. Način transliranja se definiše lambda izrazom koji se prosljeđuje kao parametar funkcije. Nakon poziva funkcije svaki od elemenata vektora realnih brojeva će biti transliran u novi domen. Napisati i odgovarajući program koji će testirati datu funkciju. Primjer poziva:

```
Unesite vektor realnih brojeva:
3.5
3.7
2.8

Translirani vektor funkcijom ( f(x) = 2x+3 ):
9.0
10.4
8.6

Translirani vektor funkcijom ( f(x) = -3x/2 ):
5.25
5.55
4.2
```

# 4 Problem 3

Napisati funkciju koja modificira kontejner tipa `list<int>` na način da briše sve elemente kontejnera koji su neparni. Napisati program koji prvo učitava listu cijelih brojeva tipa `list<int>`, eliminira neparne elemente pozivajući prethodnu funkciju, a zatim ispisuje listu sortiranu od većeg ka manjem elementu.

# 5 Problem 4

**Minesweeper**
Cilj igre je naći gdje se nalaze sve mine unutar $M \times N$ polja. Da bi vam pomogla igra vam pokaže broj unutar kvadrata koji govori koliko se mina nalazi u neposrednoj blizini tog istog kvadrata. Na primjer, pretpostavimo da imamo polje $4 \times 4$ sa 2 mine (pretstavljene sa karakterom '`*`'):

```
*...
....
.*..
....
```

Ukoliko bi pretstavili isto polje prema gore navedenom naputku izgledalo bi:

```
*100
2210
1*10
1110
```

Kao što možete primjetiti, svaki kvadrat može imati najviše 8 susjednih polja.

**Ulaz**

Ulaz se satoji od proizvoljnog broja polja. Prva linija svakog polja sastoji se od dva cijela broja $m$ i $n$ koji pretstavljaju broj linija i kolona polja respektivno. Svaki sigurni kvadrat pretstavljen je karakterom '`.`' (bez navodnika), dok su mine pretstavljene sa karakterom '`*`' (bez navodnika). Prva linija polja u kojoj je $m = n = 0$ pretstavlja kraj unosa i ne bi trebala biti procesirana.

**Izlaz**

Za svako polje potrebno je u zasebnoj liniji ispisati:

```
Polje #x:
```

Gdje $x$ pretstavlja broj polja (počevši od 1). Slijedećih $n$ linija predstavljaju polje u kome je karakter '`.`' zamjenjen sa brojem susjednih mina tog kvadrata. Između svakog polja potrebno je umetnuti praznu liniju.

**Primjer ulaza**

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

**Primjer izlaza**

```
Polje #1:
*100
2210
1*10
1110

Polje #2:
**100
33200
1*100
```

**Napomena**
Problem razbiti na funkcije za:

- parsiranje (sa provjerom validnosti koji baca različite iznimke spram različitih grešaka u parsiranju; provjeravati validnost svake linije unosa),
- riješavanje problema
- ispis riješenja

Potrebno odlučiti koju data stukturu parser treba izbaciti kao izlaz, a koju algoritam za riješenje koristi kao ulaz.

*main* u osnovi treba da izgleda kao:

```
try
{
    ispis(rijesi(parser(cin)))
}
catch
{
}
```