

Ponavljanje, stringovi

Objektno programiranje - 1. vježbe (1. dio)

Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

8. ožujka 2023. godine



Ulaz/izlaz: Zadatak za početak

Zadatak 0. Kompajlirajte i pokrenite sljedeći program (program učitava dva cijela broja i ispiše njihov zbroj).

```
#include <iostream>
int main() {
    std::cout << "Unesite dva broja:  " << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    std::cout << "Zbroj brojeva " << v1 << " i " << v2
              << " je " << v1 + v2 << std::endl;
    return 0;
}
```



Općenite informacije o vježbama

- ▶ dva sata tjedno (srijedom od 17:15h u Praktikumu 2)
- ▶ konzultacije po dogovoru mailom (uživo ili online)
 - ▶ mail: sebastijan.horvat (at) math.hr
 - ▶ ured: A306/III

Domaće zadaće (ima ih **ukupno 6**)

- ▶ **maksimalno** moguće ostvariti **70 bodova**
- ▶ **početni plan** - objava zadaća na Merlinu u svakom parnom tjednu nastave (2,4,6 itd.) nakon vježbi i rok od 10 dana za rješavanje
- ▶ Na vježbama popisujemo, no **dolasci se ne boduju!**

VAŽNO: obavezno je prisustvovati minimalno 70% predavanja i vježbi (u sumi) - bez toga gubi se pravo na polaganje kolegija!



Još o domaćih zadaćama...

- ▶ **predaja zadaće moguća samo preko Merlina unutar roka**
- ▶ nadoknada jedne domaće zadaće (u zadnjem tjednu nastave) moguća samo uz liječničku dokumentaciju
- ▶ ⇒ ne čekajte zadnji dan - nestanak električne energije, zaboravljanje na rok, razne nepredviđene situacije **neće biti opravdanje za nepredaju zadaće!**
- ▶ **sankcioniranje prepisivanja** (na obje strane) s –20 bodova
- ▶ treba **predati barem 4 zadaće za prolaz** kolegija (predati znači ostvariti barem 40% bodova na toj zadaći)
- ▶ za zadaće koje se **ne kompiliraju 0 bodova** (u okolini kao u Pr2)
- ▶ većina bodova daje se za
 - ▶ čitljivost koda,
 - ▶ smislenu upotrebu tehnika uvedenih na nastavi,
 - ▶ ispravno korištenje struktura i algoritama iz STL-a,
 - ▶ klasu algoritamske složenosti itd.
 - ▶ takvi zahtjevi bit će specificirani u tekstu zadatka



Imajte na umu ako ne želite gubiti bodove:

Tekst preuzet sa stranice s uputama za laboratorijske vježbe iz kolegija *Operacijski sustavi*¹:

Izvorne tekstove programa pisati prema uobičajenim pravilima. Posebice paziti na strukturu koda, 'uvlačenje', razmake te lomljenje preduge linije!). Primjer stila pisanja koda u C-u prikazan je u <https://www.kernel.org/doc/Documentation/process/coding-style.rst>. Jedan od razloga lošeg izgleda koda jest što studenti miješaju korištenje tabulatora i razmaka za ostvarenje uvlaka. Takav program u drugim okolinama (npr. koje koriste druge dimenzije za tabulator) izgleda nečitljivo. KONZISTENTNO koristiti ILI samo tabulatore (preporučeno) ILI samo razmake za uvlake.



¹<http://www.zemris.fer.hr/~leonardo/os/math/labosi/>

Prethodni kod, ali za "nagradne" bodove

(tj. za bodove s negativnim predznakom ;)

```
#include <iostream>
int main() {
    std::cout<<"Unesite dva broja:  "
    << std::endl; int v1=0,v2=0;
    std::cin>>v1>>v2;
    std::cout<<"Zbroj brojeva "<<v1<<" i "<<v2
    <<" je "<<v1+v2<<std::endl;return 0; }
```

Ne morate proučavati pravila s linka na prethodnom slajdu - dovoljno je uočiti da primjeri iz prezentacija nisu napisani poput koda s ovog slajda!



Literatura za 1. vježbe

Stanley B. Lippman, Josée Lajoie, Barbara E. Moo: C++ Primer, Fifth Edition, Addison Wesley Professional, 2013.

- ▶ Poglavlje 1. - *Getting started* (str. 1 - 28)
- ▶ Poglavlje 3. - *Strings, Vectors, and Arrays* (str. 81 - 132)
- ▶ iz 9. poglavlja cjelina 9.5. - *Additional string operations* (str. 360 - 368)

Ostala literatura:

- ▶ ukoliko će se o nekoj temi koja je spomenuta na slajdu moći više saznati na nekoj web stranici, tada će link na tu stranicu biti podcrtan i obojan plavom bojom



Podsjetnik: Kompajliranje i pokretanje

```
1  #include <iostream>
2  int main(){
3      std::cout << "Unesite dva broja: " << std::endl;
4      int v1 = 0, v2 = 0;
5      std::cin >> v1 >> v2;
6      std::cout << "Zbroj brojeva " << v1 << " i " << v2
7          << " je " << v1 + v2 << std::endl;
8      return 0;
9  }
```

Spremljeno kao `prvi.cpp` (na gornjoj slici korišten je *Atom*). Nakon pozicioniranja u terminalu u mapu sa spremljenom datotekom:

```
seh@DES:/m...$ g++ prvi.cpp -std=c++11 -o prog
seh@DES:/m...$ ./prog
```

```
Unesite dva broja:
14
15
Zbroj brojeva 14 i 15 je 29
```



Kompajliranje i pokretanje na vlastitom računalu

Ako na računalu imate instaliran:

- ▶ **Linux** - može se koristiti isti uređivač teksta i kompajlirati programe iz komandne linije kao u praktikumu,
- ▶ **Windows** - postizanje okruženja poput praktikumskog:
 - ▶ **Windows Subsystem for Linux** - link s uputama: <https://web.math.pmf.unizg.hr/nastava/rp1/wsl.php>
 - ▶ **virtualne mašine** s GNU/Linuxom - link s uputama: <https://web.math.pmf.unizg.hr/nastava/rp1/virtualbox.php>

Prezentacija: korišten *Windows Subsystem for Linux* i g++ 7.5.0



Ubuntu 18.04.5 on Windows

Aplikacija

```
sehorva@DESKTOP-S92RB8H:~$ g++ --version
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

9/40

Ulaz/izlaz (IO)

```
#include <iostream>
```

```
int main() {
    std::cout << "Unesite dva broja:  " << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    std::cout << "Zbroj brojeva " << v1 << " i " << v2
                << " je " << v1 + v2 << std::endl;
    return 0;
}
```

- ▶ standardna (nema .h!) biblioteka **iostream** omogućuje rad s IO
- ▶ dva tipa: **istream** (objekt **cin**) i **ostream** (objekti **cout**, **cerr**, **clog** - [Zašto nemamo samo cout?](#))
- ▶ **output operator** << (slično >> za **input**)
 - ▶ lijevi operand tipa **ostream** (**cout**) i desni vrijednost (string literal)
 - ▶ rezultat lijevi operand (tipa **ostream**) ⇒ možemo ulančavati <<
- ▶ ispis posebne vrijednosti **endl** (tzv. manipulator)
 - ▶ završava trenutnu liniju i prazni spremnik (**buffer**) - važno kod debugiranja

10/40

Prefiks `std::`

```
#include <iostream>
int main() {
    std::cout << "Unesite dva broja:  " << std::endl;
    int v1 = 0, v2 = 0;
    std::cin >> v1 >> v2;
    std::cout << "Zbroj brojeva " << v1 << " i " << v2
        << " je " << v1 + v2 << std::endl;
    return 0;
}
```

- ▶ kaže da su imena `cout`, `cin` i `endl` definirana u imeničkom prostoru (*namespace*) `std` (koristimo operator dosega `::`)
- ▶ izbjegavamo slučajne kolizije između imena koja definiramo i upotrebe imena iz biblioteka
- ▶ jednostavnije pisanje uz `using` deklaraciju



Using deklaracija

- ▶ uz npr. `using std::cin;` pisali bi samo `cin` umjesto `std::cin` - za sva imena nekog imeničkog prostora (npr. `std`):

```
#include <iostream>
using namespace std;
int main() {
    cout << "Unesite dva broja:  " << std::endl;
    int v1 = 0, v2 = 0;
    cin >> v1 >> v2;
    cout << "Zbroj brojeva " << v1 << " i " << v2
        << " je " << v1 + v2 << std::endl;
    return 0;
}
```

Sitnim slovima: Uočite da operandi u izrazu prije `return 0;` nisu istog tipa - to nije problem jer biblioteka definira verzije `<< i >>` operatora za različite tipove operanada



Pitanje: Što radi sljedeći kod?

```
#include <iostream>
using namespace std;
int main() {
    int zbr = 0, br = 0;
    while (cin >> br)
        zbr += br;
    cout << zbr << endl;
    return 0;
}
```

- ▶ input operator vraća lijevi operand (`std::cin`)
- ▶ provjerava se je li ispravan (da nije došlo do greške)
- ⇒ while petlja se prekida ako neispravan unos ili došli do *end-of-file*

Napomena: *end-of-file* se s tipkovnice unosi kao `Ctrl+Z` (i `Enter` za Windowse) ili `Ctrl+D` (najčešće za ne-Windowse)



Primjeri unosa i ispisa za prethodni kod

```
seh@DESK:/m.../Desktop$ ./prog
12
4
-67
j
-51
seh@DESK:/m.../Desktop$ ./prog
3
5
8
```

- ▶ u drugom primjeru korišten `Ctrl+D` (nakon što je uneseno 5)



Važan fokus u dizajnu jezika C++: Klase

```
#include <iostream>
int main() {
    int a, b = 2023;
    std::cin >> a;
    std::cout << a + b << std::endl;
    return 0;
}
```

- ▶ omogućavanje definiranja tipova koji se prirodno ponašaju kao i ugrađeni tipovi (npr. `int`)
- ▶ klasa definira tip (ime tipa = ime klase) zajedno s kolekcijom operacija za taj tip (u upotrebi nas zanimaju više od detalja implementacije)
- ▶ za korištenje klase treba nam: kako se zove, gdje je definirana, koje operacije podržava



Primjer 1. (podaci o prodaji zadane knjige)

```
#include <iostream>
#include "Knjiga.h" //nije standardna pa " "
int main() {
    Knjiga zadana; //Knjiga def. u Knjiga.h
    if(std::cin >> zadana) {
        Knjiga trenutna;
        while(std::cin >> trenutna) {
            if(trenutna.isbn() == zadana.isbn())
                zadana += trenutna; //tip=>značenje +=
            std::cout << zadana << std::endl;
        }
    }
    return 0;
}
```

- ▶ funkcije članice (metode) - dio klase (operator `.` daje funkciju)
- ▶ autor klase određuje sve operacije koje se mogu koristiti na objektima tipa te klase



Stringovi i vektori

- ▶ uz osnovne tipove (`int`, `char`, ...), C++ pruža mehanizam za definiranje vlastitih tipova
- ▶ biblioteke to koriste za definiranje složenijih tipova
- ⇒ imamo bogate biblioteke **apstraktnih tipova podataka**
- ▶ među najvažnijima **stringovi** (nizovi znakova varijabilne duljine) i **vektori** (nizovi objekata zadanog tipa varijabilne duljine)
- ▶ za pristup elementima stringa i vektora koristimo **iteratore**
- ▶ tipovi `string` i vektor definirani u biblioteci su apstrakcije ugrađenih **polja** (*array*)



Stringovi - Primjer 2. (definicija i inicijalizacija)

```
#include <iostream>
#include <string> ← treba uključiti iako nekad radi i bez toga (vidi ovdje)
using namespace std; ← inače za string treba using std::string;
int main() {
    string s1, ← defaultna inicijalizacija na prazan string
           s2 = s1,
           s3 = "pmf",
           s4(10, 'c');
    cout << "Ispis zadanih stringova (sv"
          << "aki u svoj red):" << endl;
    cout << s1 << endl << s2 << endl
         << s3 << endl << s4 << endl;
    return 0;
}
```

- ▶ `s2` je **kopija** od `s1`
- ▶ u `s3` iskopirani svi (osim null znaka) iz literala "pmf"
- ▶ `s4` sadrži 10 kopija znaka 'c'



Ispis za prethodni primjer

```
Ispis zadanih stringova (svaki u svoj red):
```

```
pmf  
CCCCCCCCC
```

Uočite:

- ▶ uzastopni string literali "Ispis zadanih stringova (sv" i "aki u svoj red) : " spojeni (konkatenirani) u jedan literal



Inicijalizacije

- ▶ **direktne** inicijalizacije (bez korištenja =)

```
string s("matka");  
string t(10, 'c');
```

- ▶ inicijalizacije **kopiranjem** (uz korištenje =)

```
string p = "pmf";
```

Primjer 3. Neizravna inicijalizacija kopiranjem:

```
string s = string(10, 'c');
```

ekvivalentna je stvaranju privremenog objekta za kopiranje:

```
string temp(10, 'c');  
string s = temp;
```



Primjer 4. (Ostali konstruktori)

- ▶ Definirajmo dva polja znakova (samo prvo završava s '\0'):

```
const char *cp = "abeceda";  
char neNull[] = {'a', 'b'};
```

- ▶ Ne možemo kopirati drugo polje (jer ne završava s '\0'):

```
string s1(cp);      ✓  
string s2(neNull);  ✗
```

- ▶ No, možemo kopirati prva 2 znaka:

```
string s3(neNull, 2); ✓
```

Greška ako > 2 znaka!

- ▶ Za stringove, 2 bi značio poziciju od koje se kopira:

```
string s4(s1, 2);    ✓ //kopirali "eceda"  
string s5(s1, 10);  ✗ //out_of_range iznimka
```

- ▶ Pozicija od koje se kopira + koliko znakova:

```
string s6(s1, 2, 4); ✓ //kopirali "eced"  
string s7(s1, 2, 10); ✓ //"eceda"
```

- ▶ u posljednjem primjeru, ako je br. znak. prevelik, kop. se do kraja

Miješanje C++ stringova i C-ovskih stringova

Nužno je da C++ ima sučelje prema C programima!

Lippman, Lajoie, Moo, *C++ Primer*, 5. izdanje, str. 122:



Although C++ supports C-style strings, they should not be used by C++ programs. C-style strings are a surprisingly rich source of bugs and are the root cause of many security problems. They're also harder to use!

- ▶ C → C++:

```
string s("niz znakova"); ✓
```

- ▶ obratno nema izravnog načina, nego preko `c_str`:

```
char *str = s;      ✗  
const char *str = s.c_str(); ✓
```

- ▶ posljednje vraća samo pokazivač ⇒ promjenom stringa `s` može se dogoditi da ne možemo više koristiti `str` (zato bi za daljnje korištenje trebalo kopirati C-string `str`)

Operacije na stringovima: `string` IO operatori

Primjer 5.

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main(){
7      string s1, s2;
8      cin >> s1 >> s2;
9      cout << "Prvi: " << s1 << endl
10         << "Drugi: " << s2 << endl;
11         return 0;
12 }
```



Primjer unosa i ispisa za prethodni kod

```
sehorva@DESKTOP-S92RB8H:~$ ./prog
      neki      tekst
Prvi:  neki
Drugi: tekst
sehorva@DESKTOP-S92RB8H:~$
```

Pri čitanju:

- ▶ zanemaruju se vodeće bjeline
- ▶ čitaju se znakovi do sljedeće bjeline

Operatori vraćaju lijevi operand \Rightarrow možemo ih ulančavati ✓



getline funkcija

- ▶ argumenti su istream objekt i string
- ▶ čita do **uključivo** prelaska u novi red (ali sam **prelazak se ne sprema nego odbacuje!**) i to sprema u string
- ▶ ako imamo samo prelazak u novi red, dobiveni string je prazan

Pitanje. Što rade sljedeći dijelovi koda:

```
string rijec;
while (cin >> rijec)
    cout << rijec << endl;
```

```
string linija;
while (getline(cin, linija))
    cout << linija << endl;
```



Primjer 6. Operacije sa stringovima

```
string linija;
while (getline(cin, linija)) {
    if (!linija.empty()) {
        auto d = linija.size();
        string poruka = ": " + linija + '\n';
        cout << "Duljina " << d << poruka;
    }
}
```

- ▶ funkcije članice klase string - empty (vraća bool - je li string prazan) i size (vraća duljinu stringa)
- ▶ d je tipa `string::size_type` (auto ⇒ odredio kompajler)
- ▶ = (kopiranje), + (konkatencija - **bar jedan operand je string**)
- ▶ usporedbe: ==, !=, <, <=, >, >=



Pitanja: Koje konkatencije su ispravne?

- (1.) `string s1 = "pmf", s2 = "mo";
s1 += s2;`
- (2.) `string s3 = s1 + "-" + s2 + '\n';`
- (3.) `string s4 = "pmf" + "-mo"`
- (4.) `string s5 = s1 + "-" + "mo";`
- (5.) `string s6 = "pmf" + "-" + s2;`

Primjer 7. (Uspoređivanje stringova)

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main(){
7      string s1 = "pmf", s2 = "pmf-bo",
8          s3 = "PMF-BO", s4 = "PMF-MO";
9      cout << (s1 < s2) << (s2 < s3)
10         << (s3 < s4) << endl;
11     return 0;
12 }
```

(Zbog prioriteta su zagrade oko usporedbe u gornjem primjeru nužne!)

Rad s pojedinim znakovima u stringu

- ▶ biblioteka `naziv.c` iz C-a može se koristiti u C++-u kao `cnaziv`

⇒ C: `#include<ctype.h>` ~> C++: `#include<cctype>`

Kratki pregled `cctype` funkcija:

<code>isalnum(c)</code>	true akko je <code>c</code> slovo ili znamenka
<code>isalpha(c)</code>	true akko je <code>c</code> slovo
<code>iscntrl(c)</code>	true akko je <code>c</code> kontrolni znak
<code>isdigit(c)</code>	true akko je <code>c</code> znamenka
<code>isgraph(c)</code>	true akko je <code>c</code> nije razmak ali je ispisiv
<code>islower(c)</code>	true akko je <code>c</code> malo slovo
<code>isprint(c)</code>	true akko je <code>c</code> ispisiv
<code>ispunct(c)</code>	true akko je <code>c</code> interpunkcija
<code>isspace(c)</code>	true akko je <code>c</code> bjelina
<code>isupper(c)</code>	true akko je <code>c</code> veliko slovo
<code>isxdigit(c)</code>	true akko je <code>c</code> heksadecimalna znamenka
<code>tolower(c)</code>	samo ako je <code>c</code> veliko slovo vraća malo
<code>toupper(c)</code>	samo ako je <code>c</code> malo slovo vraća veliko



Zadatak 1.

U stringu sva mala slova zamijeniti velikima i ispisati broj promijenjenih znakova.

- ▶ pristup pojedinom elementu pomoću **indeksa** (npr. prvi znak stringa `s` je `s[0]`) ili pomoću **iteratora** (kasnije)
- ▶ **Range-Based for** - prolazi svim elementima niza
- ▶ `for` (varijabla za pristup elementima : niz)
...nešto radimo s tim elementima...

Primjer 8. Ispis svih znakova stringa:

```
string s = "Pmf-mo";
for(auto c : s){
    cout << c << endl;
}
```

- ▶ `auto` ⇒ kompajler odredi da je tip od `c` ovdje `char`



Kratko o referencama i pokazivačima

- ▶ **referenca** = drugo ime za objekt
 - ⇒ nije objekt
 - ⇒ mora se inicijalizirati (ne literalom!) i tipovi se moraju podudarati
 - ⇒ ostaje do kraja vezana za objekt

Primjer 9. Što je od sljedećeg ispravno:

```
int br = 123;
int &r1 = br;
int &r2;
r1 = 5; //mijenjamo br
double &r3 = br;
int &r4 = 10;
```

- ▶ **pokazivači** (pointeri) su objekti, ne moraju biti inicijalizirani i mogu se mijenjati tako da pokazuju na druge objekte

Primjer 10.

```
int *p1 = &br, p2;
```

← p2 nije pokazivač!



substr operacija

s.substr(p, n)

- ▶ vraća `string` koji sadrži **n znakova** iz **stringa s** počevši od **indeksa p** (koji ne smije biti veći od veličine stringa)
- ▶ default za **p** je **0**, a za **n** ona vrijednost za koju se kopiraju svi znakovi **do kraja** stringa (isto za prevelik n)

Primjer 11. Odredite sadržaj stringova `s1 – s5`:

```
string s1("praktikum dva");
string s2 = s1.substr(0, 5);
string s3 = s1.substr(6);
string s4 = s1.substr(6, 15);
string s5 = s1.substr(20);
```



Operacije za modificiranje stringova

`s.insert(p, n, c)`

- ▶ ubacuje n znakova c u string s prije indeksa p
- ▶ ima raznih mogućnosti, npr. `s.insert(p, str)` - ubacivanje stringa str

`s.erase(p, n)`

- ▶ briše n znakova u string s počevši od indeksa p

`s.append(str)`

- ▶ dodavanje na kraj stringa s

Primjer 12. Odredite sadržaj stringa s (nakon svake naredbe):

```
string s = "pmfmo";  
s.insert(3, 5, '+');  
s.erase(3, 5);  
s.insert(3, " - ");  
s.append(" 2023");
```



Operacije za modificiranje stringova - replace

```
string s = "Knjiga - 1. izdanje";  
s.replace(9, 2, "drugo");  
cout << s << endl;
```

Primjer 13.

- ▶ počevši od indeksa 9 prvo obrišemo dva znaka, a zatim se od indeksa 9 ubacuje zadani string
- ▶ Odredite što se ispiše!



Operacije za pretraživanje stringova

- ▶ sve imaju povratni tip `string::size_type` - indeks gdje pronašli traženo ili `string::npos`²
- ▶ `find` vraća indeks **prvog** podudaranja
- ▶ `find_first_of` - indeks prvog znaka iz zadanog stringa
- ▶ prvi koji nije iz zadanog stringa - `find_first_not_of`
- ▶ verzije traženja unatrag: `rfind`, `find_last_of`, `find_last_not_of`
- ▶ opcionalan drugi argument predstavlja poziciju od koje počinje potraga

²Konstanta inic. na `-1`, no to je `unsigned` \Rightarrow zapravo je jednako najvećoj mogućoj veličini nekog stringa. Zato je loše koristiti neki `signed` tip za povratnu vrijednost (primjerice `int`).



Primjer 14. Za zadane stringove, što bi se ispisalo?

```
string s1("abcAbcAbc"), brojevi("0123456789"),  
        s2("r2d3"), s3("03a714p3");  
auto p = s1.find("Abc");
```

	<code>cout << ... << endl;</code>	ispis
1.	<code>p</code>	
2.	<code>s2.find_first_of(brojevi)</code>	
3.	<code>s3.find_first_not_of(brojevi)</code>	
4.	<code>s2.find_last_of(brojevi)</code>	
5.	<code>s3.find_last_not_of(brojevi)</code>	
6.	<code>s1.find("aBc")</code>	
7.	<code>s1.rfind("bc")</code>	
8.	<code>s1.find("bc",2)</code>	
9.	<code>s3.find_last_not_of(brojevi,4)</code>	



Numeričke konverzije

- ▶ `to_string(a)` - vraća `string` koji predstavlja broj `a`
- ▶ `stoi(?, s, p, b)` - vraća početni podstring od `s` kao broj
 - ▶ broj tipa `?` = `i, l, ul, ll, ull` (`int, long, unsigned long, ...`)
 - ▶ `b` = baza (default je 10)
 - ▶ `p` = pokazivač na `size_t` objekt u koji sprema indeks prvog nenumeričkog znaka u `s` (default je 0 → nema spremanja)
- ▶ `stof(?, s, p), ?` = `f, d, ld` - za `float, double` i `long double`

Primjer 15. Dobiveni broj se ne može prikazati ⇒ `out_of_range`; ako se ne može pretvoriti u broj ⇒ `invalid_argument`.

```
cout << stof("-0.34e5");  
cout << stof("-0.a");  
cout << stof("0x46");  
cout << stof("1.23e100");
```

```
sehorva@DESKTOP-S92RB8H:~$ ./prog  
terminate called after throwing an instance of 'std::out_of_range'  
what(): stof  
Aborted (core dumped)
```



Primjer 16. Što sljedeći dio koda ispiše?

```
int a = 123;  
string s1("234.12m"), s2("pi = 3.14");  
size_t p;  
cout << to_string(a) << endl  
    << stoi(s1, &p) << endl  
    << p << endl  
    << stoi(s1, &p, 5) << endl  
    << stod(s1, &p) << endl  
    << p << endl  
    << stod(s2.substr(s2.find_first_of  
                  ("+- .0123456789"))) << endl;
```



Zadatak 2.

Napišite program koji od korisnika učitava jednu liniju teksta. Ispišite poruku nalazi li se u toj liniji riječ paprika. Ukoliko se nalazi, ispišite i koliko puta se nalazi.

Primjeri unosa i ispisa.

- ▶ Jane je na Dolcu kupila tri paprike.
→ Ispis: paprika se ne javlja u unosu :(
- ▶ Alocirana paprika naziva se punjena paprika, a ne samo paprika.
→ Ispis: paprika se javlja 3 puta



Zadatak 3. (Riješite bez upotrebe regularnih izraza!)

Trgovac John je u liniju teksta upisao podatke o nekom proizvodu i točno jednu cijenu. Neposredno nakon cijene (i samo tamo!) piše HRK. Ispišite cijenu proizvoda (u EUR).³

Primjeri unosa i ispisa.

- ▶ Slanutak u konzervi 11.99HRK za 240g.
→ Ispis: Cijena je 1,59 EUR.
- ▶ Sok 100% jabuka 1L 15.74HRK
→ Ispis: Cijena je 2.09 EUR.
- ▶ 7.53HRK je cijena beskvasnog kruha.
→ Ispis: Cijena je 1.00 EUR.
- ▶ Cijena karte je 50HRK.
→ Ispis: Cijena je 6.64 EUR.

Napomena. Za ispis broja a na točno (fixed) dvije decimale (uz upotrebu `#include <iomanip>`):

```
cout << setprecision (2) << fixed << a;
```

³Fiksni tečaj konverzije: 7,53450 kuna za 1 euro.

