

Uvod, ponavljanje

Objektno programiranje (C++) - 1. vježbe (1. dio)

dr. sc. Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

6. ožujka 2024. godine



Općenite informacije o vježbama

- ▶ **vježbe:** dva sata tjedno (srijedom od 12:15h u Praktikum 1)
- ▶ **konzultacije:** po dogovoru mailom (uživo ili online)
 - ▶ mail: sebastijan.horvat (at) math.hr
 - ▶ ured: A306/III

VAŽNO: obavezno je prisustvovati na barem 70% od ukupnog broja sati predavanja i vježbi, te na barem 50% sati predavanja.

- ▶ na vježbama popisujemo, no **dolasci se ne boduju**



Napomena o bodovanju domaćih zadaća

- ▶ za zadaće koje se **ne kompiliraju 0 bodova** (u okolini kao na vježbama u Praktikum 1)
- ▶ 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*²:

*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.*



Kod 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!

Napomena: Gradivo kolegija Računarski praktikum 1

- ▶ predmet prethodnik za ovaj kolegij je **Računarski praktikum 1** (web.math.pmf.unizg.hr/nastava/rp1)
- ▶ nećemo ponavljati svo gradivo tog kolegija
- ▶ tako se na nekim slajdovima nalazi sljedeća oznaka:



- ▶ dio gradiva uz koje se nalazi ta oznaka preskočit ćemo jer se to gradivo može naći na [slajdovima vježbi](#) kolegija *Računarski praktikum 1*

Iz predgovora (*preface*) knjige *C++ Primer* (2012)

Stanley B. Lippman, Josée Lajoie, Barbara E. Moo: *C++ Primer*, Fifth Edition, Addison Wesley Professional, 2012.

- ▶ ...o C++ programskom jeziku i standardu C++11:
*It's focus, and that of its programming community, has widened from looking mostly at **machine efficiency** to devoting more attention to **programmer efficiency**.*
- ▶ ...o pristupu (učenje o spremnicima, bibliotekama itd. prije klasa):
*By using the abstract facilities defined by the library, you will become more comfortable with using high-level programming techniques. The library facilities are themselves abstract data types that are usually written in C++. The library can be defined using the same class-construction features that are available to any C++ programmer. Our experience in teaching C++ is that **by first using well-designed abstract types, readers find it easier to understand how to build their own types.***



Podsjetnik: Kompajliranje i pokretanje

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;
}
```



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
```

11/19

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:
web.math.pmf.unizg.hr/nastava/rpl/wsl.php
 - ▶ *virtualni stroj* s GNU/Linuxom - link s uputama:
web.math.pmf.unizg.hr/nastava/rpl/virtualbox.php

U prezentacijama se koristi *Windows Subsystem for Linux* i *g++* verzija 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.
```

12/19

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**)
- ▶ *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



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

- ▶ omogućavanje definiranja tipova koji se prirodno ponašaju kao i ugrađeni tipovi (npr. `int`)

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

- ▶ 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