Uvod u programiranje

- predavanja -

rujan 2025.

1. Algoritam, program, programiranje

Algoritam

- Algoritam je skup pravila kojim se opisuje kako riješiti neki problem. Posjeduje sljedeća svojstva:
  - **konačan:** mora završiti nakon konačnog broja koraka
  - **potpuno određen:** svaki korak mora biti precizan i jednoznačan
  - **djelotvoran:** sve operacije su elementarne u mjeri koja omogućuje da se mogu obaviti točno i u konačnom vremenu
  - algoritmom je definiran **ulazni skup objekata** (može biti prazan)
  - algoritmom je definiran **izlazni skup objekata**
- Algoritam se može opisati:
  - prirodnim jezikom
  - pseudo-kôdom
  - dijagramom toka
  - programskim jezikom

Uobičajeni elementi algoritama

- preuzimanje podataka
  - čitanje ulaznih vrijednosti iz nekog vanjskog izvora (npr. tipkovnice)
- izračunavanje
  - obavljanje aritmetičkih i logičkih operacija, usporedbe, ...
- selekcija
  - odabir između dva ili više sljedova akcija, na temelju ulaznih podataka, izračunatih rezultata, itd.
- iteracija
  - uzastopno ponavljanje skupa operacija, unaprijed utvrđeni broj puta ili dok je neki logički uvjet zadovoljen (ili dok nije zadovoljen)
- dostavljanje rezultata
  - obavještavanje korisnika o rezultatima, npr. ispis na zaslon ili upis u datoteku

Primjer

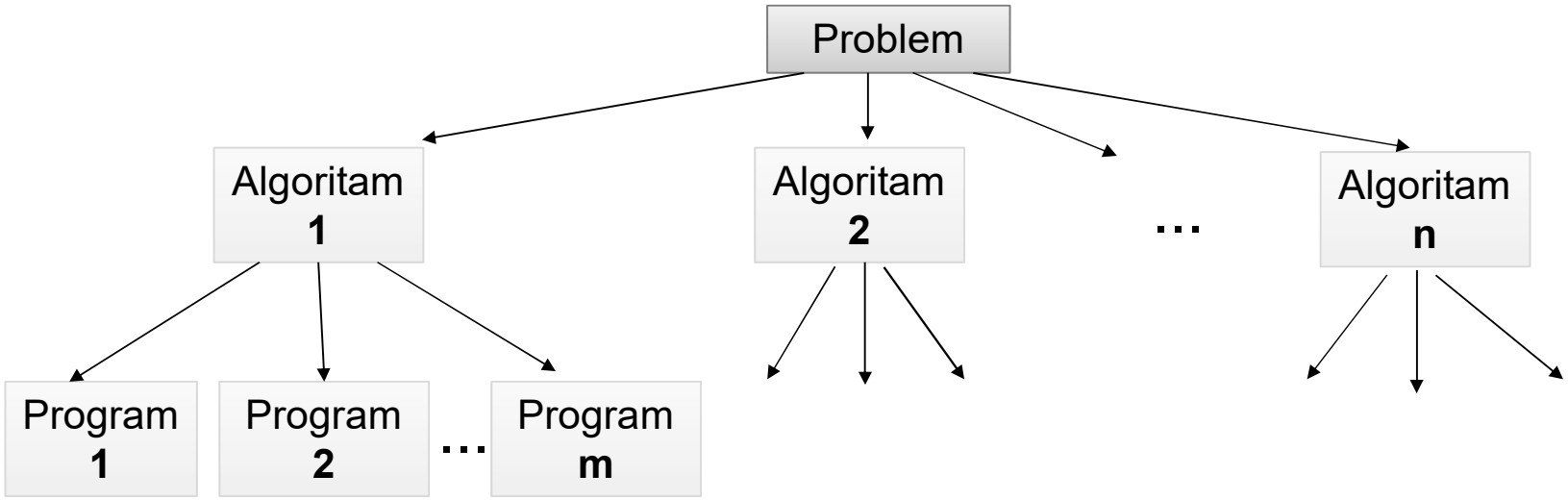
- Kiseljenje krastavaca
- Početni objekti: 5 kg krastavaca, 1 l octa (9%), 30 dag šećera, 10 dag soli, kopar, papar
  - krastavce i kopar oprati i posložiti u čiste staklenke
  - u 2 l vode dodati ocat, šećer, sol i papar
  - zakuhati uz miješanje
  - vruću otopinu uliti u staklenke
  - staklenke zatvoriti celofanom i gumicom
  - *složiti staklenke u široki lonac napunjen vodom do grla staklenki*
  - *ako je toplomjer raspoloživ*
    - *zagrijati vodu do 80 stupnjeva*
  - *inače*
    - *zagrijavati dok se s dna ne počnu dizati mjehurići zraka*
  - ostaviti stajati 24 sata
- Završni objekti: kiseli krastavci á la FER

Programski jezik, programiranje, program

- **Programski jezik**
  - rječnik i skup gramatičkih pravila kojima se računalu opisuje kako obaviti neki posao
- **Programiranje**
  - proces opisivanja algoritma nekim od programskih jezika
- **Program**
  - opis algoritma u nekom programskom jeziku kojim se računalu jednoznačno određuje koje operacije treba obaviti

Problem, algoritam, program

- u pravilu je za svaki problem moguće definirati više različitih algoritama, a za svaki algoritam napisati različite programe, korištenjem istog ili različitog programskog jezika



Računalo i računalni sustav (computer system)

- računalo je uređaj koji može obavljati aritmetičke i logičke operacije na temelju instrukcija definiranih u računalnom programu
- računalni sustav obuhvaća hardver i softver koji čine funkcionalnu cjelinu sposobnu za preuzimanje ulaznih podataka, njihovu obradu, pohranu i prezentaciju rezultata

Hardver

- fizičke komponente računala
  - procesor (*Central Processing Unit*)
    - upravlja izvršavanjem strojnih instrukcija i izvršava aritmetičke i logičke operacije
  - memorija
    - **primarna memorija**: brža, skuplja, manjeg kapaciteta (tipično RAM). Privremena pohrana programa i podataka - programi i podaci se prije izvršavanja i obrade moraju iz sekundarne memorije učitati u primarnu memoriju (*program load*)
    - **sekundarna memorija**: sporija, jeftinija, većeg kapaciteta (tipično HDD, SSD). Trajna pohrana podataka i programa.
  - ulazno/izlazne jedinice
    - tipkovnica, zaslon, miš, mrežna kartica, itd.

Softver - izvršni kôd

- softver je skup podataka i računalnih instrukcija kojima je definiran niz operacija koje računalo treba izvršiti
- skup podataka i instrukcija u strojnom jeziku (strojni kôd) koje računalo stvarno obavlja naziva se izvršni kôd (*executable code*)
  - binarni kôd
  - instrukcije su specifične za određenu vrstu procesora (izvršni kôd je neprenosiv, *nonportable*)

Softver - simbolički strojni jezik

- simbolički strojni jezik (*assembly language*) koristi ljudima razumljive simbole za opisivanje instrukcija strojnog jezika
  - u instrukcije strojnog jezika prevodi se asemblerom
  - programiranje je sporo, teško, vjerojatnost pogreške je velika
  - programi su neprenosivi - za svaki tip procesora treba napisati novi program
  - u odnosu na više programske jezike, rezultira efikasnijim strojnim kôdom

primjer simboličkog strojnog kôda

```
...
movl    %esp, %ebp
.cfi_def_cfa_register 5
andl    $-16, %esp
subl    $32, %esp
call    ___main
leal    24(%esp), %eax
movl    %eax, 4(%esp)
...
```

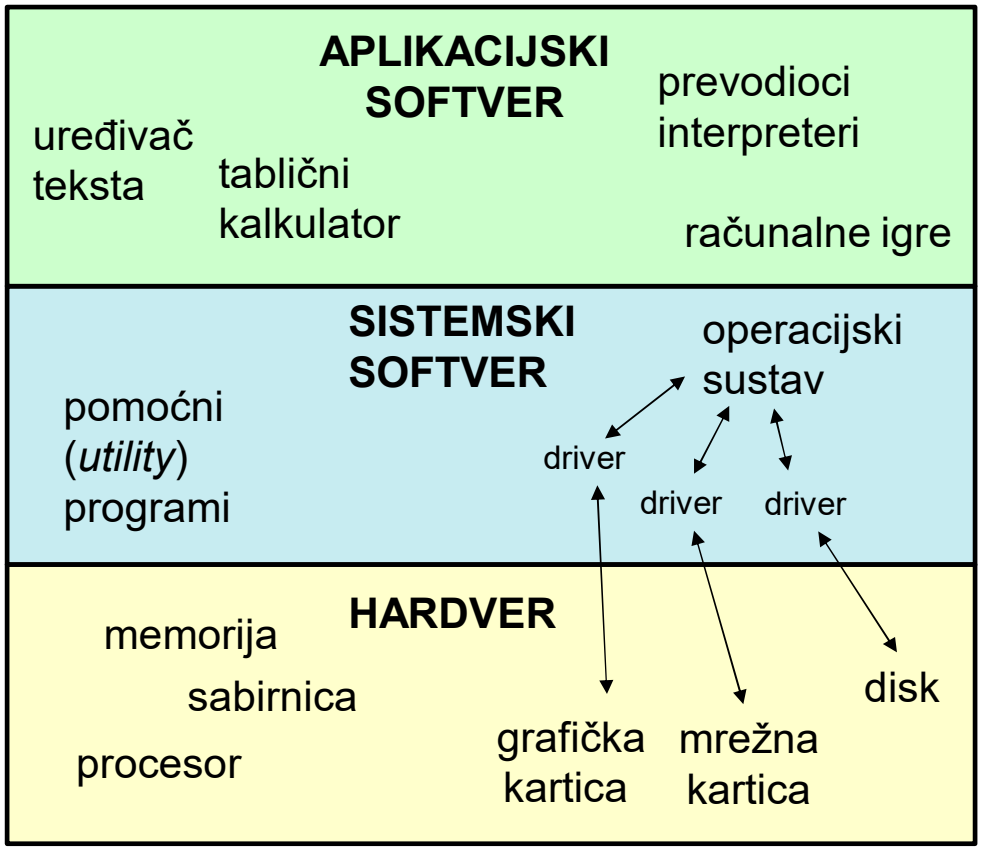
Softver - viši programski jezici

- programi se najčešće pišu u višim programskim jezicima (*high-level programming language*)
  - C, C++, C#, Java, Python, PHP, Javascript, Fortran, Pascal, ...
  - instrukcije (naredbe) izražavaju se uglavnom engleskim riječima i razumljivim dodatnim simbolima - programiranje je znatno olakšano u odnosu na programiranje u simboličkom strojnom jeziku
  - kôd je (uglavnom) prenosiv (uz eventualno ponovno prevođenje)
  - program napisan u višem programskom jeziku (ali i simboličkom strojnom jeziku) naziva se izvorni program (*source program*)
    - izvorni programski kôd (*source code*)
  - izvorni programski kôd se mora prevesti u strojni kôd pomoću specijaliziranog programa
    - unaprijed, kao cjelina (prevodilac, *compiler*) ili
    - u trenutku izvršavanja, naredba po naredba (*interpreter*)

Softver - klasifikacija prema namjeni

- Softver se prema namjeni svrstava u sljedeće kategorije:
  - Sistemski softver
    - izravno upravlja hardverom osiguravajući osnovne funkcionalnosti računalnog sustava (npr. čitanje podataka iz datoteke)
      - operacijski sustavi (pokretanje aplikacijskih programa, datotečni sustav, ljuska operacijskog sustava, alokacija memorije, ...)
      - driveri (upravljanje konkretnim tipom uređaja)
      - pomoćni (utility) programi za konfiguraciju, optimizaciju i održavanje računalnog sustava
  - Aplikacijski softver
    - obavljanje funkcija specifičnih za neku namjenu
      - editori i uređivači dokumenata, tablični kalkulatori
      - prevodioci, interpreteri
      - integrirani razvojni alati
      - ...

Računalni sustav



Primjer: od algoritma do programa

- Programski zadatak
  - s tipkovnice učitati cijeli broj, zatim izračunati apsolutnu vrijednost učitanog broja, zatim na zaslon ispisati učitanu vrijednost i njegovu apsolutnu vrijednost

Primjer: algoritam

- Algoritam izražen prirodnim jezikom
  - učitati cijeli broj. Ako je broj manji od nule, apsolutnu vrijednost izračunati promjenom predznaka učitanog broja, inače, apsolutna vrijednost je jednaka učitanom broju. Učitani broj i izračunatu apsolutnu vrijednost ispisati na zaslon

Pseudo-kôd

- algoritam opisan prirodnim jezikom je uglavnom preopširan, bez jasno raspoznatljive strukture i inherentno dvosmislen
- pseudo-kôd koristi konvencije o programskim strukturama uobičajenim u programskim jezicima, ali istovremeno izbjegava specifične detalje programskih jezika
  - pridonosi uklanjanju dvosmislenosti prirodnog jezika
  - u odnosu na neki konkretni programski jezik, razumljiviji je ljudima koji ne poznaju detalje tog programskog jezika
- na prikladnim mjestima u pseudo-kôdu dopušteno je koristiti prirodni jezik ili matematičku notaciju
- iako neki elementi pseudo-kôda mogu u grubo podsjećati na uobičajene elemente nekog konkretnog programskog jezika, ne postoji opće prihvaćena sintaksa za pseudo-kôd

Primjer: pseudo-kôd

- Pseudo-kôd - oblik s manje detalja, sličniji prirodnom jeziku

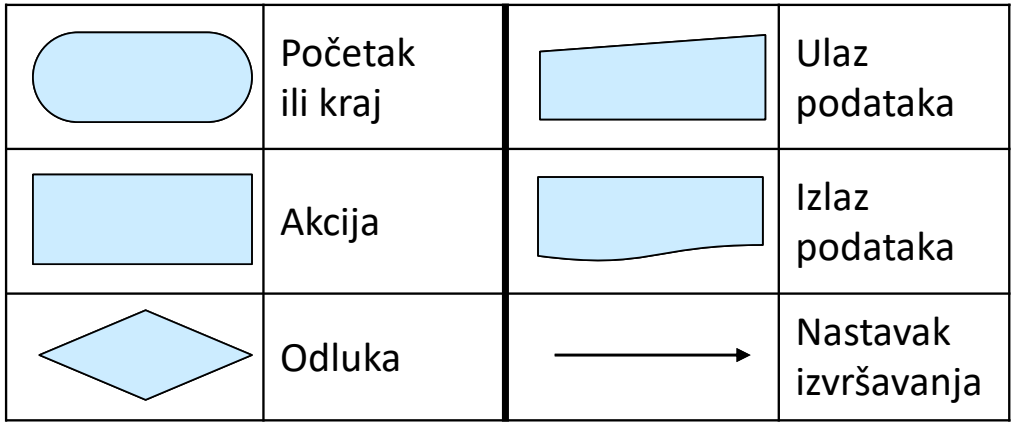
```
učitaj cijeli broj
izračunaj apsolutnu vrijednost učitanog broja
ispiši učitanu i izračunatu vrijednost
```

- Pseudo-kôd - s više detalja, uz korištenje dodatnih, unaprijed dogovorenih simbola

```
učitaj (n)

{ izračunaj apsolutnu vrijednost }
ako je n < 0 tada
   rez := - n
inače
   rez := n

ispiši (n, rez)
```

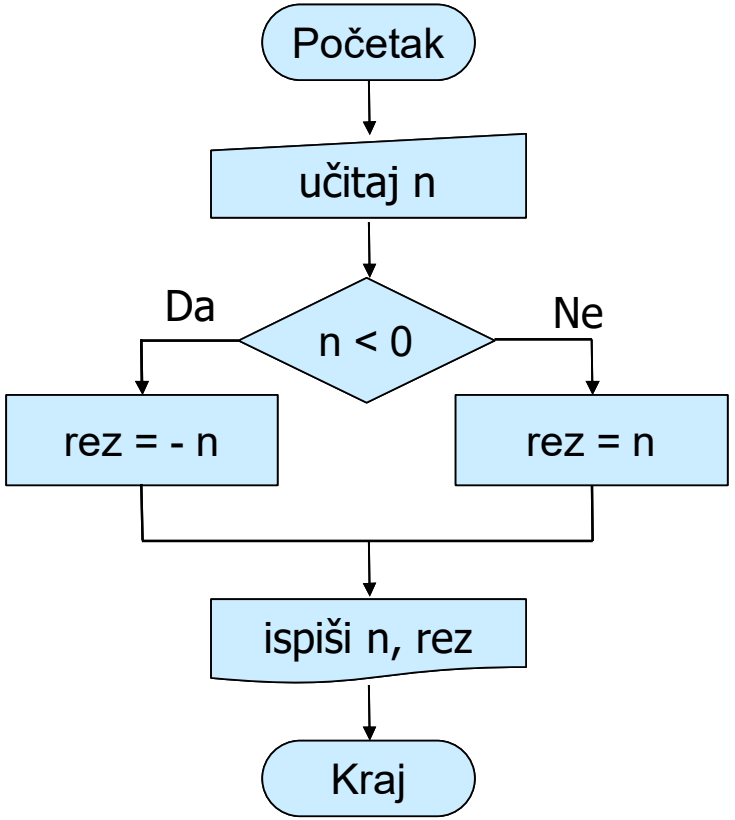
Dijagram toka

- **Dijagram toka** - grafički opis algoritma ili, općenito, funkcioniranja nekog sustava
  - nedvosmisleno opisuje algoritam
  - relativno standardiziran
  - nezgrapan kod izmjena
- Najčešće korišteni simboli su:



Primjer: dijagram toka

- Algoritam opisan dijagramom toka



Primjer: programski kôd

- Algoritam opisan (ili implementiran) u programskom jeziku **C**
  - **izvorni kôd**, napisan s pomoću editora teksta (npr. Notepad, vi) pohranjuje se u datoteku s nastavkom .c (prema konvenciji)

datoteka prog1.c

```
#include <stdio.h>

int main(void) {
   int n, rez;
   scanf("%d", &n);

   // izracunaj apsolutnu vrijednost
   if (n < 0) {
      rez = -1 * n;
   } else {
      rez = n;
   }

   printf("Ulaz: %d Rezultat: %d", n, rez);
   return 0;
}
```

Primjer: programski kôd

- Za svaki algoritam u pravilu je moguće (čak i korištenjem istog programskog jezika) napisati različite programe. Najčešće ne postoji samo jedno ispravno rješenje ili prema svim kriterijima najbolje rješenje.

datoteka prog1.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   int n;
   scanf("%d", &n);

   // izracunaj i ispisi
   printf("Ulaz: %d Rezultat: %d", n, abs(n));
   return 0;
}
```

Primjer: programski kôd

- Algoritam opisan u programskom jeziku Python
  - izvorni kôd u datoteci s nastavkom .py (prema konvenciji)

datoteka prog1.py

```
n = int(input());

# izracunaj apsolutnu vrijednost
if (n < 0):
   rez = -1 * n
else:
   rez = n

print("Ulaz:", n, "Rezultat:", rez)
```

Primjer: programski kôd

- Algoritam opisan u programskom jeziku **Java**
  - izvorni kôd u datoteci s nastavkom .java

datoteka Prog1.java

```
import java.util.Scanner;
public class Prog1 {
   public static void main(String argv[]) {
      int n, rez;
      Scanner scanner = new Scanner(System.in);
      n = scanner.nextInt();

      // izracunaj apsolutnu vrijednost
      if (n < 0)
         rez = -1 * n;
      else
         rez = n;

      System.out.println("Ulaz: " + n + " Rezultat: " + rez);
      scanner.close();
   }
}
```

Prevođenje ili interpretiranje programa

- Procesor može izvršavati isključivo strojni (binarni) kôd. Naredbe (instrukcije) napisane u višem programskom jeziku stoga se moraju:
  - **unaprijed prevesti** u strojni kôd, odnosno samostalni izvršni program kojem je za izvršavanje na računalu dovoljan operacijski sustav

  ili

  - **interpretirati naredbu po naredbu** u trenutku izvršavanja programa, za što je pored operacijskog sustava potreban poseban program - interpreter

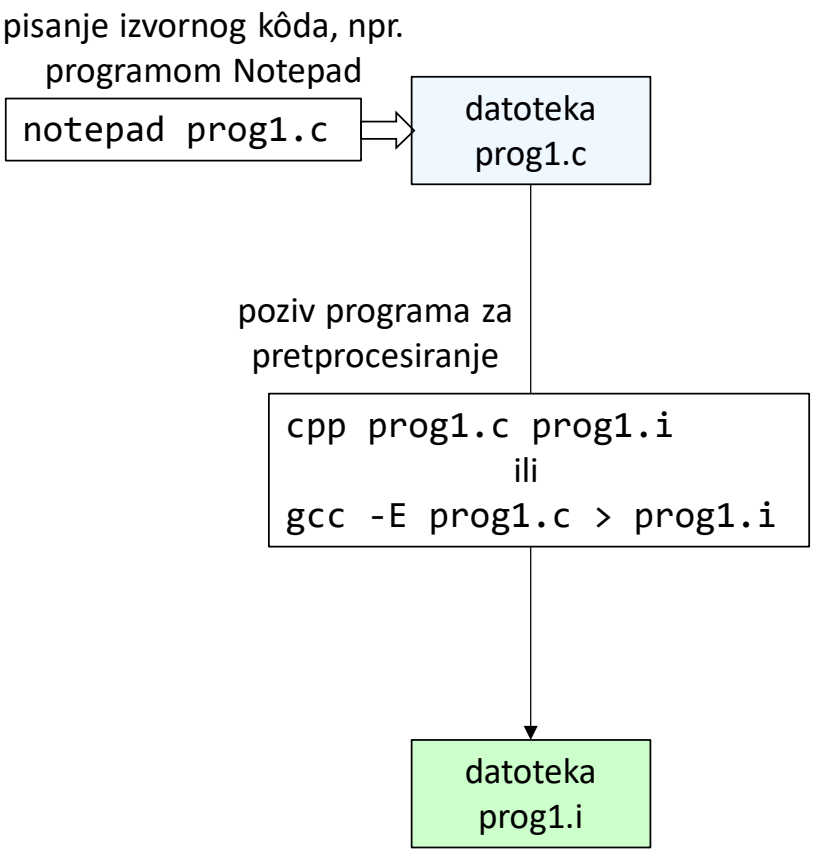
  ili

  - unaprijed prevesti u **posebni oblik programskog kôda** (***bytecode***) koji se može vrlo efikasno interpretirati ili prevesti u strojni kôd neposredno u trenutku izvršavanja programa. Pored operacijskog sustava potreban je poseban program za interpretiranje i/ili prevođenje

Prevođenje C programa

- Prevođenje C programa obavlja se u nekoliko faza:
  - **Pretprocesor (*preprocessor*):** dopunjava i prepravlja izvorni programski kôd. Npr. direktivu `#include <stdio.h>` zamjenjuje programskim kôdom iz datoteke `stdio.h`, eliminira komentare, itd.
  - **C prevodilac (*compiler*):** programski kôd dobiven iz prethodnog koraka prevodi u simbolički strojni kôd (*assembly code*)
  - **Asembler (*assembler*):** simbolički strojni kôd prevodi u objektni kôd (*object code*)
    - objektni kôd je strojni kôd koji još nije spreman za izvršavanje jer, između ostalog, nije povezan sa strojnim kôdom unaprijed pripremljenih programskih biblioteka
  - **Povezivač (*linker*):** objektni kôd povezuje sa strojnim kôdom iz programskih biblioteka (npr. strojnim kôdom za ispis na zaslon) i eventualno objektnim kôdom drugih već prevedenih modula čime nastaje izvršni programski kôd

Primjer: pretprocesor



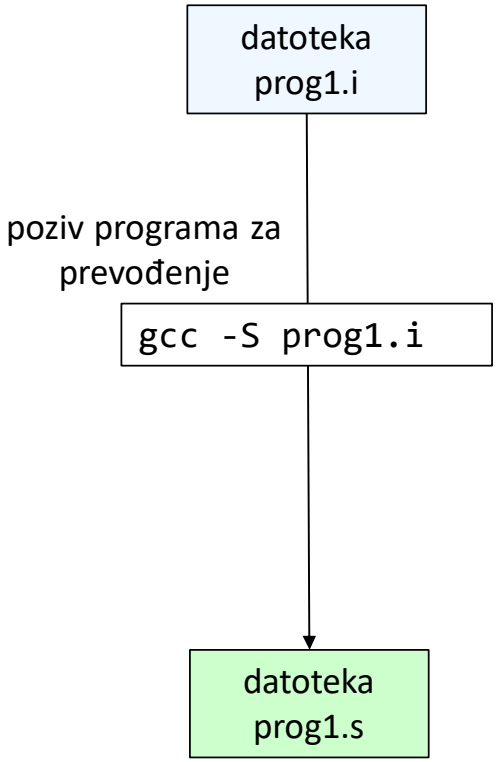
izvorni kôd

```
#include <stdio.h>

int main(void) {
   int n, rez;
   scanf("%d", &n);

   // izracunaj apsolutnu vrijednost
   if (n < 0) {
      rez = -1 * n;
   } else {
      rez = n;
   }

   printf("Ulaz: %d Rezultat: %d", n, rez);
   return 0;
}
```

pretprocesirani izvorni kôd

```
# 1 "prog1.c"
# 1 "<built-in>"
...
# 62 "c:\\mingw\\include\\sys/types.h" 3
  typedef long __off32_t;
  typedef __off32_t _off_t;...
# 3 "prog1.c"
...
int main(void) {
   int n, rez;
   scanf("%d", &n);

   if (n < 0) {
      rez = -1 * n;
   } else {
...
```

Primjer: prevodilac



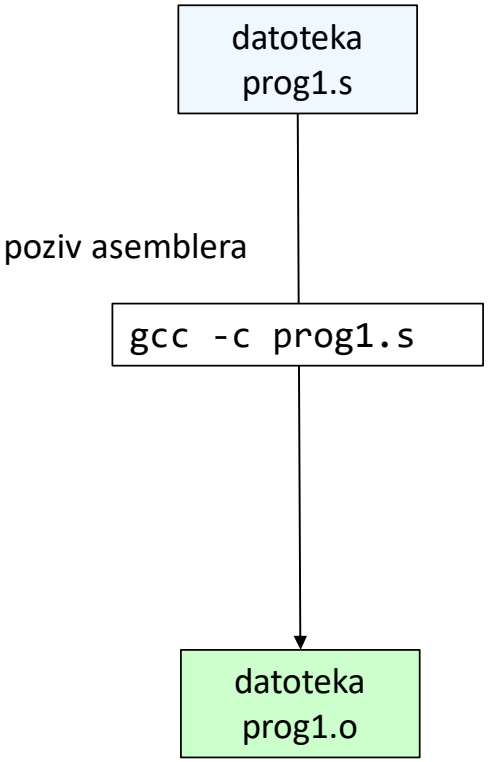
pretprocesirani izvorni kôd

```
# 1 "prog1.c"
# 1 "<built-in>"
...
# 62 "c:\\mingw\\include\\sys/types.h" 3
  typedef long __off32_t;
  typedef __off32_t _off_t;...
# 3 "prog1.c"
...
int main(void) {
   int n, rez;
   scanf("%d", &n);

   if (n < 0) {
      rez = -1 * n;
   } else {
...
```

simbolički strojni kôd

```
...
_main:
LFB10:
       .cfi_startproc
       pushl %ebp
       .cfi_def_cfa_offset 8
       .cfi_offset 5, -8
       movl  %esp, %ebp
       .cfi_def_cfa_register 5
       andl  $-16, %esp
       subl  $32, %esp
       call  ___main
       leal  24(%esp), %eax
...
```

Primjer: asembler



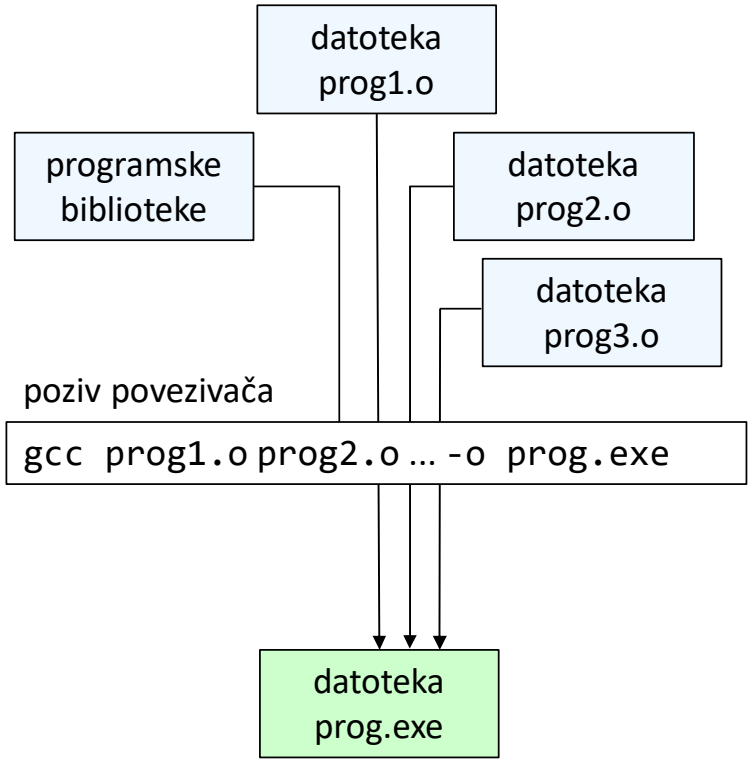
simbolički strojni kôd

```
...
_main:
LFB10:
        .cfi_startproc
        pushl %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $32, %esp
        call  ___main
        leal  24(%esp), %eax
        movl  %eax, 4(%esp)
        movl  $LC0, (%esp)
...
```

objektni kôd: strojni kôd i reference na strojni kôd iz programskih biblioteka i ostalih modula

```
...
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 01110000 01110010 01101111 01100111 00110001
00101110 01100011 00000000 01101101 01100001 01101001
01101110 00000000 01011111 01011111 01101001 01110011
01101111 01100011 00111001 00111001 01011111 01110011
01100011 01100001 01101110 01100110 00000000 01110000
...
```

Primjer: povezivač



objektni kôd

```
...
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000
00000000 01110000 01110010 01101111 01100111 00110001
00101110 01100011 00000000 01101101 01100001 01101001
01101110 00000000 01011111 01011111 01101001 01110011
01101111 01100011 00111001 00111001 01011111 01110011
01100011 01100001 01101110 01100110 00000000 01110000
...
```

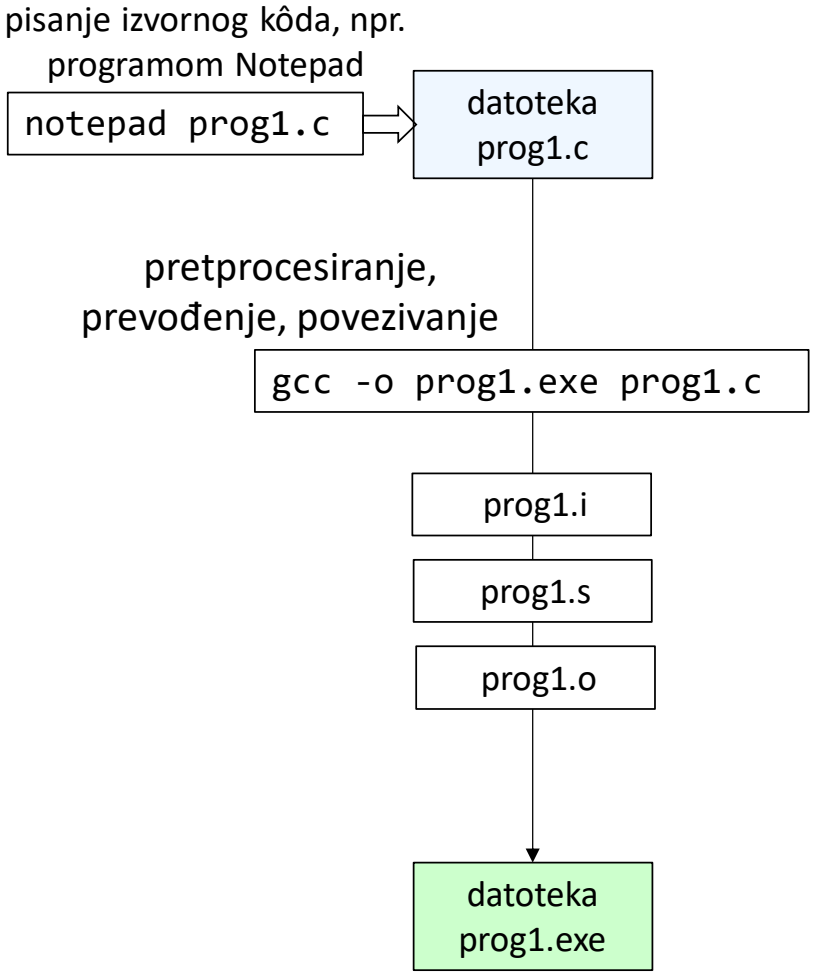
```
01101111 01100011 00111001 00111001 01011111 01110011
01100011 01100001 01101110 01100110 00000000 01110000
...
```

```
01100011 01100001 01101110 01100110 00000000 01110000
...
```

izvršni kôd

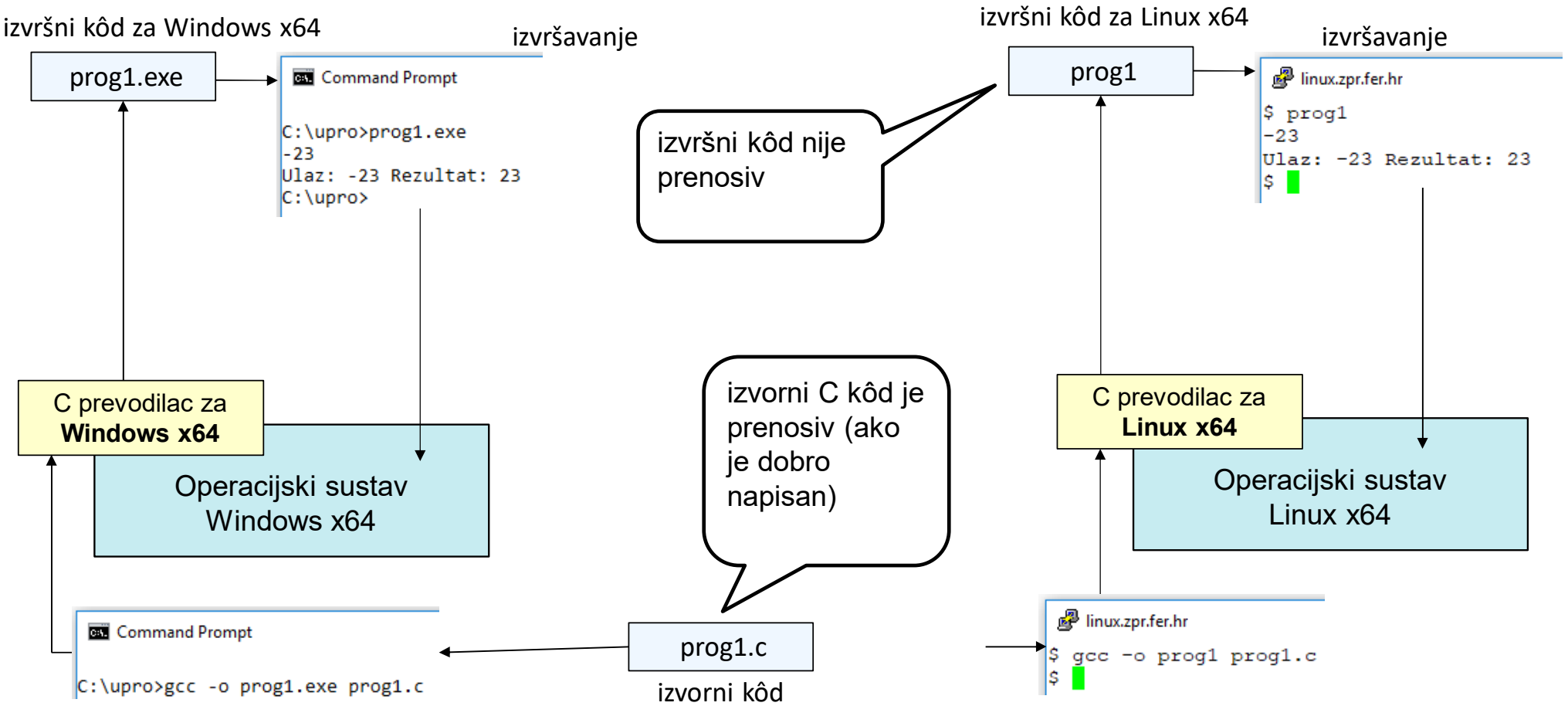
```
...
00000000 01011111 01011111 01100100 01100001 01110100
01100001 01011111 01110011 01110100 01100001 01110010
01110100 00000000 01011111 01011111 01100100 01110011
01101111 01011111 01101000 01100001 01101110 01100100
01101100 01100101 00000000 01011111 01011111 01000100
01010100 01001111 01010010 01011111 01000101 01001110
01000100 01011111 01011111 00000000 01011111 01011111
01101100 01101001 01100010 01100011 01011111 01100011
01110011 01110101 01011111 01101001 01101110 01101001
01110100 00000000 01011111 01011111 01100010 01110011
01110011 01011111 01110011 01110100 01100001 01110010
01110100 00000000 01011111 01011111 01101001 01110011
01101111 01100011 00111001 00111001 01011111 01110011
...
```

Primjer: prevođenje (i ostalo) jednom naredbom

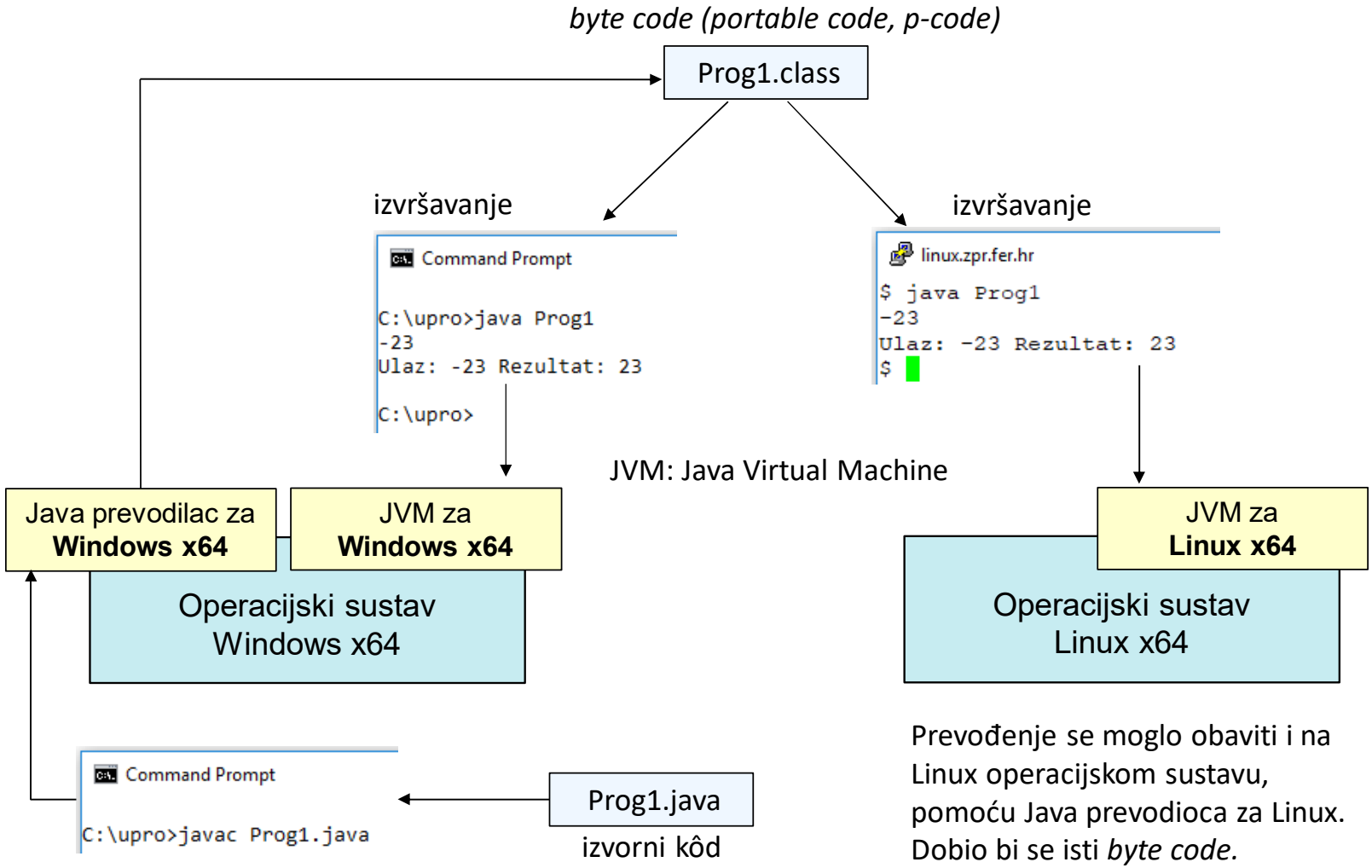


- datoteke prog1.i, prog1.s, prog1.o se automatski brišu
  - žele li se te datoteke zadržati, kôd poziva gcc dodati opciju -save-temps
- vrlo detaljan opis svih koraka prevođenja dobit će se dodavanjem opcije --verbose

Primjer: C, prevođenje i izvršavanje

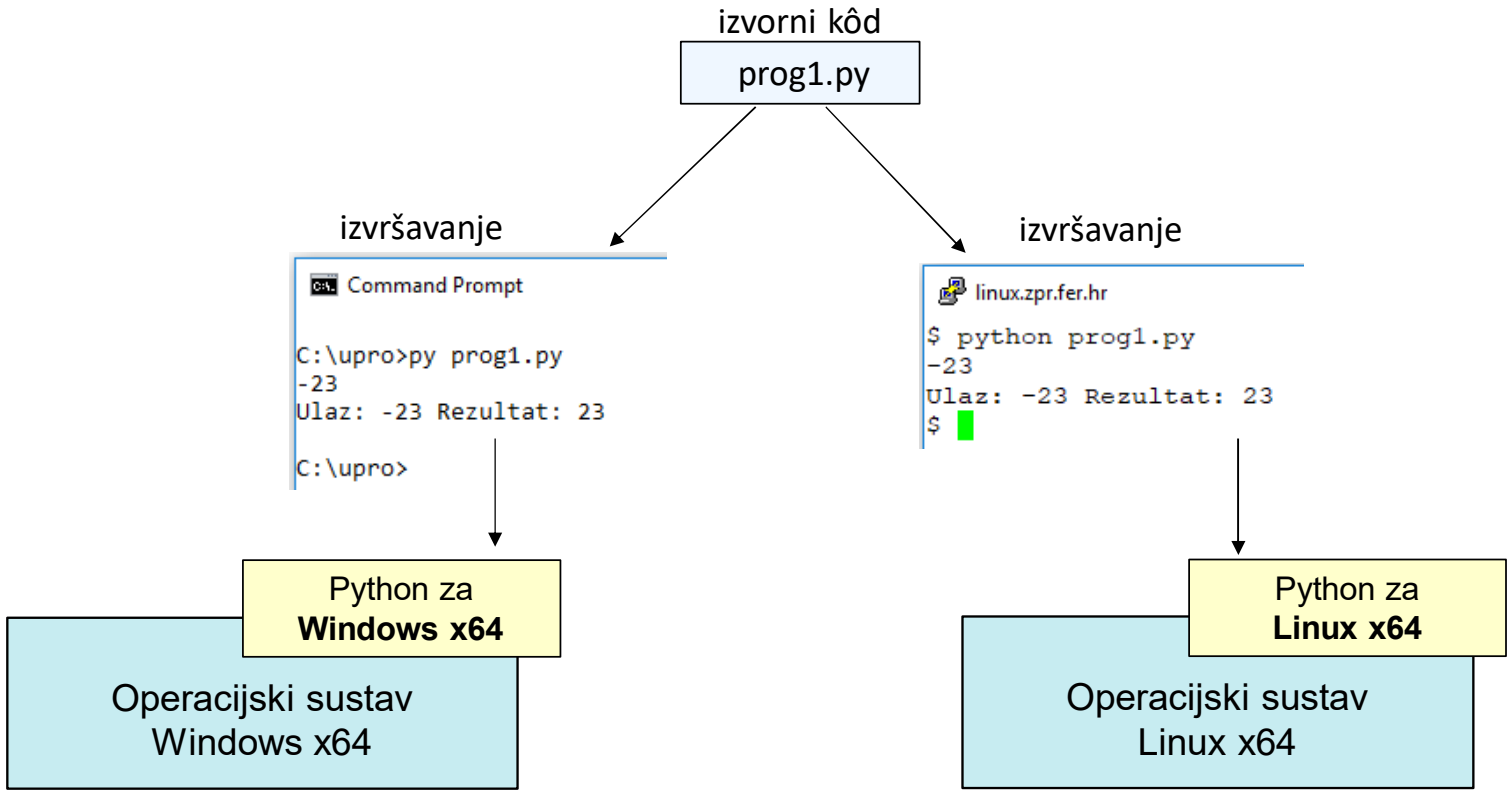


Primjer: Java, prevođenje i izvršavanje



Prevođenje se moglo obaviti i na Linux operacijskom sustavu, pomoću Java prevodioca za Linux. Dobio bi se isti *byte code.*

Primjer: Python, izvršavanje



Prije sljedećeg predavanja

- Edgar:
  - Tutorial: **01. Prije drugog predavanja**
  - (Zadatci za vježbu će početi „tek” od sljedećeg predavanja)