Uvod u programiranje

- predavanja -

listopad 2020.

Osnove programskog jezika C

- 2. dio -

Temeljni elementi jezika C

Relacijski operatori i izrazi

Relacijski operatori i izrazi

- relacijskim operatorima testira se odnos među operandima
  - relacijski operator i pridruženi operandi čine relacijski izraz
  - operandi u relacijskim izrazima mogu biti i realni i cjelobrojni (također i drugih tipova). Operandi mogu biti varijable, konstante i složeniji aritmetički izrazi
  - relacijski izraz je jednostavan logički izraz, atomni sud, koji se evaluira kao istinit ili lažan. Npr.

```
...
if (n < 0)
...
```

  - relacijski izraz n < 0 evaluirat će se kao istinit ako je vrijednost varijable n manja od nule, inače, izraz se evaluira kao lažan

Relacijski operatori

- navedeni su svi relacijski operatori

Operator	Značenje
>	veće
<	manje
>=	veće ili jednako
<=	manje ili jednako
==	jednako
!=	različito

- naročito paziti na sljedeće: operatori = i == imaju posve različito značenje. Uporabu neispravne vrste operacije prevodilac neće moći utvrditi: program će se prevesti i "raditi", ali neispravno.

Logički operatori i izrazi

- logičkim operatorima grade se složeni sudovi
  - logički operator i pridruženi operandi čine logički izraz
  - kao operandi u logičkim izrazima mogu se koristiti relacijski izrazi i složeni logički izrazi. Npr.

```
...
if (n >= 10 && n <= 20)
...
```

  - logički izraz `n >= 10 && n <= 20` evaluirat će se kao istinit ako i samo ako je rezultat relacijskog izraza `n >= 10` istinit i rezultat relacijskog izraza `n <= 20` je istinit

Logički operatori

- navedeni su svi logički operatori

Operator	Značenje
!	logičko NE
&&	logički I
||	logički ILI

Prioritet operatora

- obratiti pažnju na prioritet aritmetičkih, relacijskih, logičkih operatora i operatora pridruživanja. Koristiti zagrade gdje je potrebno

Prioritet operatora
`!`
`*  /  %`
`+  -`
`<  <=  >  >=`
`==  !=`
`&&`
`||`
`=`

- Primjer:
  - `!x > 20` će se evaluirati kao `(!x) > 20`
  - ispravno je `!(x > 20)`

Primjeri logičkih izraza

- Rezultat logičkog izraza je istina
  - ako je vrijednost realne varijable x unutar intervala [3, 5] ili unutar intervala [7, 9]

```
(x >= 3.f && x <= 5.f) || (x >= 7.f && x <= 9.f)
```

  - ako je vrijednost realne varijable x unutar intervala [3, 5] i vrijednost varijable y nije unutar intervala [7, 9]

```
x >= 3.f && x <= 5.f && !(y >= 7.f && y <= 9.f)  ili
x >= 3.f && x <= 5.f && (y < 7.f || y > 9.f)
```

Transformacija prvog u drugi izraz i obrnuto: De Morganova pravila

  - ako je vrijednost realne varijable x pozitivna ili barem za 10 veća i od vrijednosti varijable y i od vrijednosti varijable z

```
x > 0.f || x >= y + 10.f && x >= z + 10.f
```

  - za vježbu: provjerite nedostaju li negdje zagrade ili se još neke zagrade smiju ukloniti

Temeljni elementi jezika C

Naredbe za promjenu programskog slijeda

Jednostavni oblik selekcije

Jednostavni oblik selekcije

- jedna od naredbi za kontrolu toka programa

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

izvršavanje programa nastavlja se ovdje, bez obzira na rezultat logičkog izraza

- ako se logički izraz (u zagradama iza ključne riječi if) izračuna kao istina, obavljaju se sve naredbe unutar prvog para vitičastih zagrada
- inače se obavljaju sve naredbe unutar drugog para vitičastih zagrada (iza ključne riječi else)
- dio naredbe koji započinje ključnom riječi else ne mora se uvijek navesti

Temeljni elementi jezika C

Funkcije za čitanje iz standardnog ulaza i
pisanje na standardni izlaz

Učitavanje vrijednosti iz standardnog ulaza

```
scanf("%d", &n);
```

- poziv funkcije za učitavanje vrijednosti s tipkovnice
  - prvi argument je format - niz znakova unutar dvostrukih navodnika koji sadrži jednu ili više tzv. *konverzijskih specifikacija*
    - `"%d"` je format koji omogućuje čitanje jedne cjelobrojne vrijednosti
    - `"%f"` je format koji omogućuje čitanje jedne realne vrijednosti
    - `"%d %d %f %d"` je format koji omogućuje čitanje (redom) dvije cjelobrojne, jedne realne i još jedne cjelobrojne vrijednosti
  - jedan ili više sljedećih argumenata predstavljaju adrese varijabli u koje funkcija treba upisati vrijednosti pročitane s tipkovnice. Adresa varijable dobije se tako da se ispred imena varijable napiše znak &. Tipovi varijabli moraju po broju i tipu odgovarati konverzijskim specifikacijama navedenim u prvom argumentu poziva funkcije.

Primjer

```
int i, j, k;
float x;
scanf("%d %d %f %d", &i, &j, &x, &k);
```

- ako se tipkovnicom upišu vrijednosti: `37 5 3.14 2↵`
- nakon izvršavanja funkcije scanf, varijable `i`, `j`, `x`, `k` će redom sadržavati vrijednosti `37`, `5`, `3.14` i `2`

oznakom ↵ se naglašava da se je na tom mjestu na zaslon "ispisan skok u novi red" ili da je tijekom unosa podataka preko tipkovnice pritisnuta tipka Enter, odnosno Return

Ispis vrijednosti na standardni izlaz

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

- poziv funkcije za ispis vrijednosti na zaslon
  - prvi argument je format - niz znakova unutar dvostrukih navodnika koji sadrži kombinaciju znakova koji će se ispisati i/ili konverzijskih specifikacija
    - %d je konverzijska specifikacija za ispis cjelobrojne vrijednosti
    - %f za ispis realne vrijednosti
    - %e za ispis realne vrijednosti u znanstvenoj notaciji
  - daljnji argumenti (ako su navedeni) predstavljaju vrijednosti koje će se ispisati na mjestima na kojima su u formatu navedene konverzijske specifikacije.
    - vrijednosti mogu biti varijable, konstante, ali i složeniji izrazi
    - vrijednosti moraju po broju i tipu odgovarati konverzijskim specifikacijama navedenim u prvom argumentu poziva funkcije.

Prilagodba širine ispisa cijelog broja

- konverzijskom specifikacijom moguće je utjecati na širinu ispisa cijelog broja
  - ako se širina ne navede ili je navedena širina manja od potrebnog broja znamenki, ispisat će se minimalni broj znamenki potreban za ispravan prikaz broja
  - ako je navedena širina veća od minimalno potrebnog broja znamenki, ispred znamenki će se ispisati odgovarajući broj praznina

```
printf("%d,%5d,%2d", 128, -12, 256);
```

```
128,  -12,256
```

Prilagodba širine i preciznosti ispisa realnog broja

- kod ispisa realnog broja moguće je prilagoditi ukupnu širinu ispisa i broj ispisanih znamenki (preciznost) iza decimalne točke

```
   printf("%f,%f,%f\n", 4.5f, 1280.0f, -3.4555555f);
```

```
4.500000,1280.000000,-3.455556
```

`%f` ukupna širina prema potrebi, 6 znamenki iza dec. točke, zaokružiti prema potrebi

```
   printf("%5.2f,%12.4f,%.4f", 5.128f, -256.128f, 12.4555555f);
```

```
 5.13,   -256.1280,12.4556
```

`%5.2f` min. ukupna širina 5 znakova, 2 znamenke iza dec. točke, zaokružiti prema potrebi

`%12.4f` min. ukupna širina 12 znakova, 4 znamenke iza dec. točke, zaokružiti prema potrebi

`%.4f` 4 znamenke iza decimalne točke, a ispred minimalno koliko je potrebno

Ispis realnog broja u znanstvenoj notaciji

- specifikacija %e vrlo je slična specifikaciji %f. Jedina razlika je u tome što se broj ispisuje u obliku koji sadrži eksponent broja 10

```
printf("%e,%e,%e\n", 4.5f, 1280.0f, -3.4555555f);
```

```
4.500000e+000,1.280000e+003,-3.455556e+000
```

%e ukupna širina prema potrebi, 6 znamenki iza dec. točke, zaokružiti prema potrebi

```
printf("%12.2e,%15.4e,%.4e", 5.128f, -256.128f, 12.4555555f);
```

```
5.13e+000,   -2.5613e+002,1.2456e+001
```

%12.2e min. ukupna širina 12 znakova, 2 znamenke iza dec. točke, zaokružiti prema potrebi

%15.4e min. ukupna širina 15 znakova, 4 znamenke iza dec. točke, zaokružiti prema potrebi

%.4e 4 znamenke iza decimalne točke, a ispred minimalno koliko je potrebno

Raspon i preciznost

Cjelobrojni i realni tip

Raspon cjelobrojnih varijabli i konstanti

- zbog načina pohrane cijelih brojeva u računalu, moguće je prikazati tek ograničeni skup cijelih brojeva
- detaljnije će se pohrana cijelih brojeva razmatrati kasnije, za sada je dovoljno znati sljedeće:
  - cjelobrojni tip podatka (varijabla i konstanta tipa int) može se koristiti za prikaz cijelih brojeva u intervalu [-2 147 483 648, 2 147 483 647]. Pokušaj korištenja cijelih brojeva izvan tog intervala može dovesti do neočekivanih rezultata, npr.

```
int min = -2147483648, max = 2147483647, rez1, rez2;
rez1 = min - 1;
rez2 = max + 1;
printf("-2147483648 - 1 > %d\n", rez1);
printf("2147483647 + 1 > %d", rez2);
```

```
-2147483648 - 1 > 2147483647
2147483647 + 1 > -2147483648
```

Raspon realnih varijabli i konstanti

- zbog načina pohrane realnih brojeva u računalu, moguće je prikazati samo sljedeće realne brojeve
  - brojeve iz intervala $[-3.402823 \cdot 10^{38}, -1.401298 \cdot 10^{-45}]$ ili
  - brojeve iz intervala $[1.401298 \cdot 10^{-45}, 3.402823 \cdot 10^{38}]$ ili
  - realni broj 0.0
- detaljnije će se pohrana realnih brojeva razmatrati kasnije
- pokušaj korištenja realnih brojeva izvan tog intervala može dovesti do neočekivanih rezultata

```
float x = 3.4e38f * 1.1f;
float y = 1.401298e-45f / 2.f;
printf("%f\n%e\n", x, y);
```

```
inf
0.000000e+000
```

Preciznost realnih varijabli i konstanti

- zbog načina pohrane realnih brojeva, nije moguće potpuno točno pohraniti sve realne brojeve iz prethodno navedenih intervala
  - zapravo, beskonačno mnogo brojeva iz navedenih intervala nije moguće prikazati potpuno točno
  - detaljnije će se preciznost pohrane realnih brojeva razmatrati kasnije

```
float x = 0.0625f;
float y = 0.0624f;
printf("%20.18f\n%20.18f", x, y);
```

```
0.062500000000000000
0.062399998307228088
```

Preciznost realnih varijabli i konstanti

- ponekad se samo zbog zaokruživanja pri ispisu (ako se navede dovoljno malen broj znamenki koje treba ispisati iza decimalne točke) čini da je broj pohranjen posve točno

```
float x = 0.01f;
printf("%f\n", x);
printf("a zapravo je pohranjeno:\n", x);
printf("%20.18f", x);
```

```
0.010000
a zapravo je pohranjeno:
0.00999999776482582
```

Korisne matematičke funkcije

Korisne matematičke funkcije

- navedene funkcije se mogu upotrijebiti na mjestu operanda u aritmetičkom ili relacijskom izrazu
- argumenti ovih funkcija mogu biti varijable, konstante ili aritmetički izrazi
- argumenti i rezultati sljedećih funkcija su realni brojevi *dvostruke preciznosti*, međutim za sada se to može zanemariti.

Funkcija	Značenje
`sqrt(x)`	$\sqrt{x}$
`pow(x, y)`	x^y
`sin(x)`	sinus (rad)
`cos(x)`	kosinus (rad)
`tan(x)`	tangens (rad)
`log(x)`	ln x
`log10(x)`	$\log_{10} x$

Primjer

```
#include <stdio.h>
#include <math.h>     // Uočiti ovaj include!

int main(void) {
   printf("sqrt(20.25) = %f\n", sqrt(20.25f));
   printf("pow(6.25, -0.5) = %f\n", pow(6.25f, -0.5f));
   printf("sin(3.1415926/2) = %f\n", sin(3.1415926f/2));
   printf("ln(2.7182818) = %f\n", log(2.7182818f));
   printf("log(1000.0) = %f\n", log10(1000.f));

   return 0;
}
```

```
sqrt(20.25) = 4.500000
pow(6.25, -0.5) = 0.400000
sin(3.1415926/2) = 1.000000
ln(2.7182818) = 1.000000
log(1000.0) = 3.000000
```

Primjer

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

int main(void) {
   float a, b, c;
   printf("Upisite katete > ");
   scanf("%f %f", &a, &b);

   c = sqrt(pow(a, 2.f) + pow(b, 2.f));
   printf("Hipotenuza = %f\n", c);
   return 0;
}
```

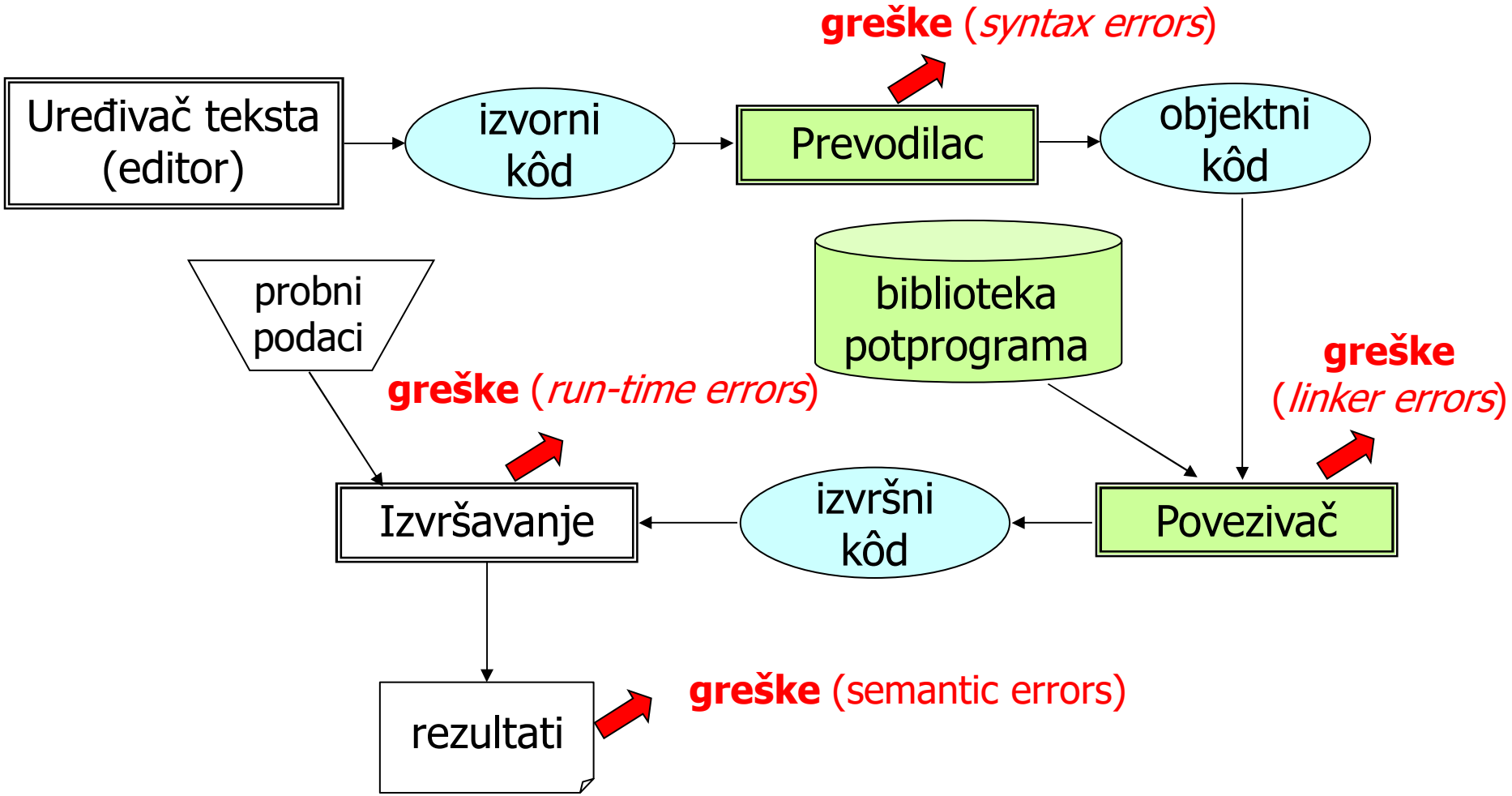
```
Upisite katete > 6 8↵
Hipotenuza = 10.000000↵
```

Programiranje, prevodilac, vrste grešaka

Postupci izrade manjih programa

1. Razvoj algoritma (npr. pomoću pseudo-koda)
2. Implementacija algoritma u programskom jeziku (kodiranje)
3. Prevođenje programa u objektni kôd (kompilacija)
   - prevodilac dojavljuje formalne greške
4. Ispravljanje formalnih grešaka
5. Povezivanje objektnog kôda (*linking*)
   - povezivač (*linker*) dojavljuje greške povezivanja
6. Ispravljanje grešaka povezivanja
7. Izvršavanje programa uz primjenu testnih podataka
   - uočavaju se greške izvršavanja i semantičke greške
8. Ispravljanje grešaka izvršavanja i semantičkih grešaka

Postupci izrade manjih programa



Postupci izrade manjih programa

- Implementacija algoritma u programskom jeziku (kodiranje)
  - obavlja se upisivanjem izvornog kôda u datoteku pomoću uređivača teksta (*editor*). Npr. notepad, vi, ...
  - ili pomoću uređivača teksta ugrađenog u radnu okolinu programera, npr. VSCode, Eclipse, MS Visual Studio, ...

- Prevođenje izvornog programskog koda u objektni program
  - obavlja se prevodiocem (*compiler*)
    - prevodilac (i pretprocesor) otkrivaju i dojavljuju sintaktičke (pravopisne, formalne) greške
    - programer ispravlja izvorni kôd i ponovo pokreće prevođenje
    - postupak se ponavlja dok god prevodilac dojavljuju formalne greške

Postupci izrade manjih programa

- Povezivanje (*linking*) prevedenog objektnog kôda u izvršni (apsolutni) programski kôd
  - obavlja se povezivačem (*linker*)
    - povezuje se objektni kôd iz jednog ili više prevedenih modula i kôd iz programskih biblioteka
    - povezivač otkriva i dojavljuje greške povezivanja
    - programer ispravlja izvorni kôd i ponovo pokreće prevođenje i povezivanje
    - ponavlja se dok god povezivač dojavljuje greške povezivanja

Postupci izrade manjih programa

- Testiranje programa
  - obavlja se definiranjem skupova ulaznih podataka i očekivanih rezultata, a zatim izvršavanjem programa, npr. pokretanjem programa iz ljuske operacijskog sustava
  - pri tome se uočavaju
    - greške koje uzrokuju abnormalni prekid programa (*run-time errors*)

      i
    - neispravni rezultati koji upućuju na postojanje semantičkih grešaka (*semantic errors*)
    - programer ispravlja izvorni kôd i ponovo pokreće prevođenje, povezivanje i testiranje
    - ponavlja se dok god se tijekom izvršavanja programa događaju abnormalni prekidi rada programa ili se dobivaju neispravni rezultati

Primjer - razvoj malog programa

```
#inklude <stdio.h>

int main(void) {
   int n; rez;
   scan("%d", -n);

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

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

Primjer - formalne greške

```
C:\upro>gcc -std=c11 -Wno-all -o prog1.exe prog1.c
prog1.c:1:2: error: invalid preprocessing directive #inklude
 #inklude <stdio.h>
  ^~~~~~~~
prog1.c: In function 'main':
prog1.c:4:11: error: 'rez' undeclared (first use in this function)
    int n; rez;
           ^~~
prog1.c:4:11: note: each undeclared identifier is reported only once for
each function it appears in
```

- ispraviti greške u izvornom kodu i ponoviti prevođenje

```
#inklude <stdio.h>

int main(void) {
   int n; rez;
   scan("%d", -n);
...
```

Primjer - greške povezivanja

```
C:\upro>gcc -std=c11 -Wno-all -o prog1.exe prog1.c
C:\Users\user\AppData\Local\Temp\cc6jyeMA.o:prog1.c:(.text+0x20):
undefined reference to `scan'
C:\Users\user\AppData\Local\Temp\cc6jyeMA.o:prog1.c:(.text+0x41):
undefined reference to `print'
collect2.exe: error: ld returned 1 exit status
```

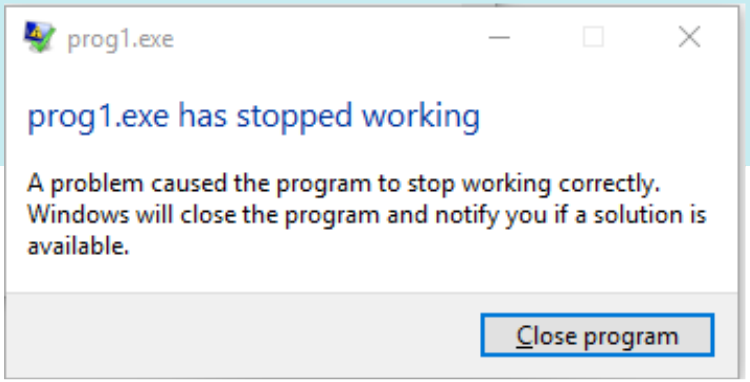
- ispraviti greške u izvornom kodu i ponoviti prevođenje i povezivanje

```
...
   int n = 0, rez;
   scan("%d", -n);
...
   print("Ulaz: %d Rezultat: %d", n, rez);
...
```

Primjer - greške tijekom izvršavanja

```
C:\upro>gcc -std=c11 -Wno-all -o prog1.exe prog1.c
C:\upro>prog1.exe
-11

C:\upro>
```



- kako otkriti greške koje prevodilac i povezivač nisu uspjeli prepoznati?

Traženje grešaka

- Opcije za traženje grešaka tijekom izvršavanja i semantičkih grešaka
  - čitati program i razmišljati
  - dodavati pomoćne ispise na strateška mjesta u programu

```
#include <stdio.h>
int main(void) {
   int n; rez;
   scanf("%d", -n);
   printf("Procitan je n=%d\n", n);

   // izracunaj apsolutnu vrijednost
   if (n < 0) {
      printf("n je manji od nule");
      rez = --n;
      printf("izracunat je rez=%d", rez);
   } else {
      printf("n je veci od nule");
      rez = n;
   }
   printf("Ulaz: %d Rezultat: %d", n, rez);
   return 0;
}
```

Traženje grešaka

- Bolje: koristiti specijalizirane programe za traženje grešaka – (*debuggers*)
  - u naredbenoj liniji (ljusci operacijskog sustava)
  - integrirano u programerskoj okolini (VSCode, Eclipse, ...)
  - u nastavku je prikazan samo mali skup mogućnosti programa gdb koji se koristi u kombinaciji s prevodiocem gcc

Debugger - traženje mjesta prekida programa

```
C:\upro>gcc -std=c11 -ggdb -o prog1.exe prog1.c
C:\upro>gdb prog1.exe
(gdb) start
Temporary breakpoint 1 at 0x401497: file prog1.c, line 5.
Starting program: C:\upro/prog1.exe
Temporary breakpoint 1, main () at prog1.c:5
5           scanf("%d", -n);
(gdb) frame                 koja naredba će se izvršiti kao sljedeća?
#0  main () at prog1.c:5
5           scanf("%d", -n);
(gdb) next                  izvrši dotičnu sljedeću naredbu
-11                         upis vrijednosti za n

Program received signal SIGSEGV, Segmentation fault.
0x7593ff0b in ungetwc () from C:\WINDOWS\SysWOW64\msvcrt.dll
```

- očito, program je prekinut nakon što je pozvana funkcija scanf
- sada više nije teško uočiti grešku. Ispraviti i ponoviti testiranje

```
scanf("%d", -n); → scanf("%d", &n);
```

Debugger - traženje semantičkih grešaka

```
C:\upro>gcc -std=c11 -ggdb -o prog1.exe prog1.c
C:\upro>prog1.exe
-11
Ulaz: -11 Rezultat: 4194432
```

- Program nije prekinut, ali rezultat je neispravan. Gdje bi mogla biti (semantička) greška?

```
   scanf("%d", &n);          je li vrijednost varijable n sada ispravno učitana?

   // izracunaj apsolutnu vrijednost
   if (n < 0) {              je li ispravno evaluiran ovaj relacijski izraz? Po čemu ćemo znati?
      rez = --n;             je li u varijablu rez upisan ispravan rezultat?
   } else {
      rez = n;
   }
```

Debugger - traženje semantičkih grešaka

```
C:\upro>gcc -std=c11 -ggdb -o prog1.exe prog1.c
C:\upro>gdb prog1.exe
(gdb) start                 pokreni program i zaustavi se već na prvoj naredbi
...
(gdb) watch rez             nadgledaj varijablu rez - kad god se promijeni, zaustavi program
Hardware watchpoint 2: rez
(gdb) continue              nastavi s izvršavanjem
Continuing.
-11                         upis vrijednosti za n
Hardware watchpoint 2: rez    promijenila se varijabla rez. U nastavku piše gdje i kako:
Old value = 4194432
New value = -12
0x004014c6 in main () at prog1.c:9
9               rez = --n;   Aha! Tu si! Mora biti da nešto nije u redu s rez = --n;
```

Slijedi: testiranje, testiranje, testiranje, ...

- Ispraviti grešku, ponovo prevesti, povezati i testirati s različitim ulaznim podacima

```
C:\upro>gcc -std=c11 -Wall -pedantic-errors -o prog1.exe prog1.c
C:\upro>prog1.exe
-11
Ulaz: -11 Rezultat: 11
C:\upro>prog1.exe
11
Ulaz: 11 Rezultat: 11
C:\upro>prog1.exe
0
Ulaz: 0 Rezultat: 0
```