Programiranje 2
Debager. Profajler. Složenost. Vežbanje.

1 DEBAGER

Osnovne komande:

- Pri prevodjenju programa je potrebno dodati opciju -g. Na primer:

  gcc program.c -Wall -Wextra -g

- Debager pokrećemo komandom gdb ime_izvršnog_fajla. Na primer:

  gdb ./a.out

- Ako postoje argumenti komandne linije (npr. 2), debager pokrećemo sa:

  gdb ./a.out argument1 argument2

- Komandom run pokrećemo izvršavanje našeg programa

  run

- Izlazak iz debagera

  quit

Tačke prekida (eng. breakpoint): Predstavljaju mesto gde ce nas program stati sa izvršavanjem i čekati da mu damo instrukciju šta dalje treba da radi. Tačke prekida se koriste u situacijama kada hoćemo da pratimo izvršavanje našeg programa korak po korak, kada hoćemo da proverimo vrednosti nekih promenljivih i slično. Tačke prekida se zadaju nakon pokretanja debagera, a pre nego sto se pokrene izvršavanje programa sa komandom run. Mogu zadati na vise načina:

1. b broj_linije
2. b ime_funkcije
3. b ime_fajla:broj_linije
4. b [nesto od predthodna tri navedena] if uslov

Nove tačke prekida se mogu dodati i u trenutku kada se nalazimo na nekoj od prethodno postavljenih tačaka prekida.

Posmatrajmo sledeći primer (neispravnog programa čije je ime program.c) :

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.
4.  int  f(x)
5.  {
6.   return 1 + f(x/10);
7.  }
8.
9.  int main()
10. {
11.
12.  printf("%d\n", f(12345));
13.  return 0;
14. }
15.
```

Tačku prekida na liniji 6 možemo postaviti na neki od sledećih načina:

- b 6
- b program.c:6
- b 6 if x==0 – ovom komandom ce se na tacku prekida stati samo ukoliko promenljiva x ima vrednost 0

Kada se jednom zaustavimo na tački prekida, treba da zadamo debageru instrukciju kojom ćemo odrediti šta dalje želimo da radimo. Neke od opcija su:

- next – izvrši prvu sledeću instrukciju (i zaustavi se cim je izvršiš)
- step – isto kao next, izuzev što ukoliko se u instrukciji koja izvršava nalazi poziv neke funkcije, step ce ući na prvu liniju te funkcije koja se poziva, a next neće. Next celu liniju smatra jednom instrukcijom i prelazi na sledeću.
- continue – nastavi izvršavanje (ili do sledeće tačke prekida - ako je ima ili dok se program ne završi)
- print ime_promenljive – štampaj vrednost zadate promenljive
- display ime_promenljive – prikazuj vrednost zadate promenljive svaki put kada se zaustaviš na nekoj tački prekida
- d redni_broj – brise tačku prekida sa zadatim rednim brojem. Redni broj tačke prekida se prikazuje pri pozivu komande b.
- undisplay ime_promenljive – poništava komandu display za zadatu promenljivu

Komanda info

- info – izlistava sve moguce opcije za komandu info
- info args – ispisuje sve argumente koji su prosledjeni tekućoj funkciji
- info locals – ispisuje vrednosti svih lokalnih promenljivih za tekuću funkciju
- info stack – ispisuje sadržaj steka
- info source – ispisuje podatke o izvornom fajlu
- info display – ispisuje sve promenljive koje posmatramo kada se program zaustavi
- info breakpoints – ispisuje sve tačke prekida

2 DEBAGER - VEŽBANJE

Zadatak 1 *Zadatak sa razlomcima*

- *Sa sajta preuzeti zadatak 2.c*
- *Zadatak prevesti sa opcijom -g*
- *Pokrenuti debager*
- *Postaviti tacku prekida na liniju 36*
- *postaviti tacku prekida na liniju 45*
- *pokrenuti program*
- *u toku rada programa regularno zadavati ulaz (prvo n, a zatim n razlomaka)*
- *kada se zaustavite na prvoj tački prekida, izlistati informacije za sve lokalne promenljive, a zatim postaviti da se pri svakom zaustavljanju prikazuje vrednost niz[i].*
- *kada se prvi put zaustavite na drugoj tački prekida, postaviti da se niz[i] vise ne prikazuje, ali da se prikazuje vrednost (float)niz[i].br/niz[i].im, kao i vrednost proseka.*

3 SLOŽENOST

Zadatak 2 *Dokazati da je* $4n + 2 \in \Theta(n)$

Zadatak 3 *Odrediti broj koraka narednog programskog koda i rezultat prikazati u O noraciji:*

```
for(i = 0; i < n; i++)
{
  c = c + i;
}
```

Zadatak 4 *Naći opšte rešenje sledeće rekurentne jednačine:* $T(n) = 2T(n-1)+1$, *za početni uslov* $T(0) = 1$.

Zadatak 5 *Odrediti vremensku slozenost naredne dve funkcije i diskutovati o dobijenim rezultatima:*

```
int fun1(int n)
{
  int cifra, max;

  if(n < 10)
    return n;

  max = fun1(n/10);
  cifra = n%10;

  if (cifra > max)
   return cifra;
  else
   return max;
}
```

```
int fun2(int n)
{
  int cifra;

  if(n < 10)
    return n;

  cifra = n%10;

  if (cifra > fun1(n/10))
   return cifra;
  else
   return fun1(n/10);
}
```

Zadatak 6 *Odrediti vremensku složenost narednog programskog koda i rezultat prikazati u O notaciji:*

```
int faktorijel(int n)
{
  if(n == 0)
    return 1;

  else
    return n * faktorijel(n-1);
}
```

Zadatak 7 *Prikazati rezultat sledeće rekurentne jednačine u O notaciji:* $T(n) = 2T(n/2) + 2n$

Zadatak 8 *Napisati rekurzivnu funkciju za izračunavanje n-tog člana fibonačijevog niza. Diskutovati o vremenskoj složenosti.*