Programiranje 2

Rekurzija

Milena Vujošević Janičić
Jelena Graovac

www.matf.bg.ac.rs/~milena
www.matf.bg.ac.rs/~jgraovac

Beograd, 20. februar, 2020.

Pregled

Pregled

Definicija rekurzije

Rekurzija je ...

... pristup u kojem se neki pojam, objekat ili funkcija definiše na osnovu jednog ili više baznih slučajeva i na osnovu pravila koja složene slučajeve svode na jednostavnije.

Primer:

bazni slučaj:

Roditelj osobe je predak te osobe

rekurzivni korak:

Roditelj bilo kog pretka neke osobe je takođe predak te osobe

Napomena: za ispravnu rekurzivnu definiciju, neophodni su i bazni slučaj i rekurzivni korak.

Matematička osnova rekurzije

- Rekurzija je tesno povezana sa matematičkom indukcijom
  - Dokaz zasnovan na matematičkoj indukciji čine dokazi baznog slučaja (na primer, za $n = 0$) i dokazi induktivnog koraka: pod pretpostavkom da tvrđenje važi za n dokazuje se da tvrđenje važi za $n + 1$
  - Bazni slučaj rekurzivne deifnicije je slučaj koji može biti rešen bez rekurzivnog poziva, dok u rekurzivnom koraku za vrednost n pretpostavljamo da je definicija raspoloživa za vrednost $n - 1$

Rekurzija u računarstvu

- U programiranju, rekurzija je tehnika u kojoj funkcija poziva samu sebe, direktno ili indirektno.
- Rekurzivne funkcije su pogodne za širok spektar informatičkih problema, ali pored svojih dobrih strana imaju i loše.
- Struktura definicije rekurzivne funkcije uključuje dve komponente:
  - uslov pri kome se funkcija rekurzivno poziva i rekurzivni poziv,
  - uslov završetka izvršavanja funkcije i sam završetak (izlaz iz rekurzije).

Suma brojeva

- Suma recipročnih vrednosti pozitivnih prirodnih brojeva može se definisati na sledeći način:

  bazni slučaj:

  za $n = 0$ važi: $\sum_{i=1}^{0}(1/i) = 0$

  rekurzivni korak:

  za $n > 0$ važi: $\sum_{i=1}^{n}(1/i) = \Sigma_{i=1}^{n-1}(1/i) + (1/n)$

- Pethodnoj rekurzivnoj definiciji odgovara rekurzivni kod:

```
double reciprocna_suma(unsigned n) {
    if (n == 0)
        return 0;
    else
        return reciprocna_suma(n-1) + (1.0/n);
}
```

- Nacrtati stek okvire za izvršavanje poziva za n=3.

Suma brojeva

- Suma recipročnih vrednosti pozitivnih prirodnih brojeva može se definisati na sledeći način:

  bazni slučaj:

  za $n = 0$ važi: $\sum_{i=1}^{0}(1/i) = 0$

  rekurzivni korak:

  za $n > 0$ važi: $\sum_{i=1}^{n}(1/i) = \Sigma_{i=1}^{n-1}(1/i) + (1/n)$
- Pethodnoj rekurzivnoj definiciji odgovara rekurzivni kod:

```
double reciprocna_suma(unsigned n) {
    if (n == 0)
        return 0;
    else
        return reciprocna_suma(n-1) + (1.0/n);
}
```

- Nacrtati stek okvire za izvršavanje poziva za n=3.

Nizovi

- I druge tipove podataka moguće je predstaviti induktivno što ih onda čini pogodnim za primenu rekurzije
- Niz je moguće definisati na sledeći način

  bazni slučaj:

  prazan niz predstavlja niz

  rekurzivni korak:

  niz dobijen dodavanjem elementa na kraj nekog niza predstavlja niz

- Ako na ovaj način shvatimo niz, primitivnom rekurzijom je moguće definisati funkcije tako da pri izlasku iz rekurzije obrađuju slučaj praznog niza, dok slučaj nepraznog niza dužine `n` rešavaju tako što rekurzivno razreše njegov prefiks dužine `n-1` i onda rezultat iskombinuju sa poslednjim elementom niza.

Suma niza

- Sumiranje elemenata niza se može definisati na sledeći način
  bazni slučaj:
  za $n = 0$ važi: $\sum_{i=1}^{0} a_i = 0$
  induktivni korak:
  za $n > 0$ važi: $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n-1} a_i + a_n$.
- Pethodnoj rekurzivnoj definiciji odgovara rekurzivni kod:

```
float sum(float a[], unsigned n) {
  if (n == 0)
     return 0.0f;
  else
     return sum(a, n-1) + a[n-1];
}
```

- Nacrtati stek okvire za izvršavanje funkcije sa dimenzijom niza n=3 i članovima niza a[3]={1.0, 2.0, 3.0}.

Suma niza

- Sumiranje elemenata niza se može definisati na sledeći način
  bazni slučaj:
  $$\text{za } n = 0 \text{ važi: } \sum_{i=1}^{0} a_i = 0$$
  induktivni korak:
  $$\text{za } n > 0 \text{ važi: } \sum_{i=1}^{n} a_i = \sum_{i=1}^{n-1} a_i + a_n.$$
- Pethodnoj rekurzivnoj definiciji odgovara rekurzivni kod:

```
float sum(float a[], unsigned n) {
  if (n == 0)
     return 0.0f;
  else
     return sum(a, n-1) + a[n-1];
}
```

- Nacrtati stek okvire za izvršavanje funkcije sa dimenzijom niza n=3 i članovima niza a[3]={1.0, 2.0, 3.0}.

Totalna rekurzija

- U nekim slučajevima, potrebno je koristiti i naprednije oblike indukcije, kakva je, na primer, *totalna indukcija*.
- Nakon dokazivanja induktivne baze, u okviru induktivnog koraka moguće je pretpostaviti tvrđenje za sve brojeve manje od n i iz te pretpostavke dokazati tvrđenje za broj n.
- Slično, umesto primitivne rekurzije, dozvoljeno je pisati funkcije koje su *totalno rekurzivne*.
- Na primer, prilikom razmatranja nekog broja `n`, dozvoljeno je izvršiti rekurzivni poziv za bilo koji broj manji od njega (pa čak i više rekurzivnih poziva za različite prirodne brojeve manje od njega).

Fibonačijevi brojevi

- Fibonačijev niz {0,1,1,2,3,5,8,13,...} može se definisati u vidu totalne rekurzivne funkcije F:

  bazni slučaj:

  za $n = 0$ i $n = 1$ važi: $F(0) = 0$ i $F(1) = 1$

  rekurzivni korak:

  za $n > 1$ važi: $F(n) = F(n-1) + F(n-2)$

- Funkcija za izračunavanje n-tog elementa Fibonačijevog niza:

```
unsigned fib(unsigned n) {
  if(n == 0 || n == 1)
    return n;
  else
    return fib(n-1) + fib(n-2);
}
```

- Nacrtati stek okvire za $n = 5$.

Fibonačijevi brojevi

- Fibonačijev niz {0,1,1,2,3,5,8,13,...} može se definisati u vidu totalne rekurzivne funkcije F:

  bazni slučaj:

  za $n = 0$ i $n = 1$ važi: $F(0) = 0$ i $F(1) = 1$

  rekurzivni korak:

  za $n > 1$ važi: $F(n) = F(n-1) + F(n-2)$

- Funkcija za izračunavanje n-tog elementa Fibonačijevog niza:

```
unsigned fib(unsigned n) {
  if(n == 0 || n == 1)
    return n;
  else
    return fib(n-1) + fib(n-2);
}
```

- Nacrtati stek okvire za $n = 5$.

NZD — Euklidov algoritam

- NZD dva broja može se definisati u vidu totalne rekurzivne funkcije *nzd*:

  bazni slučaj:

  $$\text{za } b = 0 \text{ važi } nzd(a, 0) = a$$

  rekurzivni korak:

  $$\text{za } b > 0 \text{ važi } nzd(a, b) = nzd(b, a\%b)$$

- Funkcija *nzd*:

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
     return a;
  else
     return nzd(b, a % b);
}
```

- Nacrtati stek okvire za $a = 18$ i $b = 4$

NZD — Euklidov algoritam

- NZD dva broja može se definisati u vidu totalne rekurzivne funkcije *nzd*:

  bazni slučaj:

  $$\text{za } b = 0 \text{ važi } nzd(a, 0) = a$$

  rekurzivni korak:

  $$\text{za } b > 0 \text{ važi } nzd(a, b) = nzd(b, a\%b)$$

- Funkcija *nzd*:

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
     return a;
  else
     return nzd(b, a % b);
}
```

- Nacrtati stek okvire za $a = 18$ i $b = 4$

Kule Hanoja

Problem kula Hanoja

Data su tri tornja i na prvom od njih n diskova opadajuće veličine; zadatak je prebaciti sve diskove sa prvog na treći toranj (koristeći i drugi) ali tako da nikada nijedan disk ne stoji iznad manjeg.

Kule Hanoja

- Animacija: Tower_of_Hanoi.gif

Kule Hanoja

- Iterativno rešenje ovog problema je veoma kompleksno, a rekurzivno je prilično jednostavno:

  bazni slučaj:
  : ukoliko je $n = 0$, nema diskova koji treba da se prebacuju

  rekurzivni korak:
  : 1) prebaci (rekurzivno) $n - 1$ diskova sa polaznog na pomoćni toranj (korišćenjem dolaznog tornja kao pomoćnog),
  : 2) prebaci najveći disk sa polaznog na dolazni toranj
  : 3) prebaci (rekurzivno) $n - 1$ diskova sa pomoćnog na dolazni disk (korišćenjem polaznog tornja kao pomoćnog).

Kule Hanoja

```
void prebaci(unsigned n, char polazni, char dolazni, char pomocni) {
  if (n > 0) {
     prebaci(n-1,polazni,pomocni,dolazni);
     printf("Prebaci disk sa kule %c na kulu %c\n",polazni,dolazni);
     prebaci(n-1,pomocni,dolazni,polazni);
  }
}
```

- Za vežbu:
  1. Rešiti modifikovani problem kula Hanoja u kojem su dva pomoćna štapa na raspolaganju.
  2. Rešiti modifikovani problem kula Hanoja u kojem su tri pomoćna štapa na raspolaganju.

- U dosadašnjim primerima, rekurzivne funkcije su pozivale same sebe direktno.
- Postoji i mogućnost da se funkcije međusobno pozivaju i tako stvaraju *uzajamnu rekurziju*.

Parnost

- Problem da li je broj paran može se rekurzivno definisati preko pojma neparnosti, pri čemu je bazni slučaj (za $n = 0$) da je broj paran.

```
int paran(int n) {
  if (n==0)
    return 1;
  else
    return neparan(n-1);
}
```

```
int neparan(int n) {
  if (n==0)
    return 0;
  else
    return paran(n-1);
}
```

Nacrtati stek okvire za $n = 3$.

Pregled

1. Rekurzija — osnovni pojmovi
2. Dobre i loše strane rekurzije
   - Dobre strane rekurzije
   - Mane rekurzije
3. Eliminisanje rekurzije
4. Literatura

Dobre strane rekurzije

Dobre strane rekurzije su

- čitljiv i kratak kod,
- kod je jednostavan za
  - razumevanje
  - analizu
  - dokazivanje korektnosti
  - održavanje
- pogodna je za obradu rekurzivno definisanih struktura podataka (stablo, lista...)

Mane rekurzije

Ipak, rekurzivna rešenja imaju i mana.

- Cena poziva — Prilikom svakog rekurzivnog poziva kreira se novi stek okvir i kopiraju se argumenti funkcije. Kada rekurzivnih poziva ima mnogo, ovo može biti veoma memorijski i vremenski zahtevno, te je poželjno rekurzivno rešenje zameniti iterativnim.
- Suvišna izračunavanja — U nekim slučajevima prilikom razlaganja problema na manje potprobleme dolazi do preklapanja potproblema i do višestrukih rekurzivnih poziva za iste potprobleme.

Mane rekruzije

- Testirati Fibonačijeve brojeve za različite vrednosti
- Eliminacija suvišnog izračunavanja:

```
unsigned memo[MAX_FIB];
unsigned fib(unsigned n) {
  if (memo[n]) return memo[n];
  if(n == 0 || n == 1)
    return memo[n] = n;
  else
    return memo[n] = fib(n-1) + fib(n-2);
}
```

Pregled

Eliminisanje rekurzije

- Svaku rekurzivnu funkciju je moguće implementirati na drugi način tako da ne koristi rekurziju.
- Ne postoji jednostavan opšti postupak za generisanje takvih alternativnih implementacija
- Međutim, takav postupak postoji za neke specijalne slučajeve.

Repna rekurzija

- Rekurzivni poziv je *repno rekurzivni* ukoliko je vrednost rekurzivnog poziva upravo i konačan rezultat funkcije, tj. nakon rekurzivnog poziva funkcije ne izvršava se nikakva dodatna naredba.
- U tom slučaju, nakon rekurzivnog poziva nema potrebe vraćati se u kôd pozivne funkcije, te nema potrebe na stek smeštati trenutni kontekst (vrednosti lokalnih promenljivih).

Primer — da li je ovo repna rekurzija?

- Funkcija *nzd*:

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
    return a;
  else
    return nzd(b, a % b);
}
```

- Funkcija *faktorijel*:

```
unsigned faktorijel(unsigned n) {
  if (n == 0)
    return 1;
  else
    return n*faktorijel(n-1);
}
```

Primer — da li je ovo repna rekurzija?

- Funkcija *nzd*: —>DA

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
    return a;
  else
    return nzd(b, a % b);
}
```

- Funkcija *faktorijel*:

```
unsigned faktorijel(unsigned n) {
  if (n == 0)
    return 1;
  else
    return n*faktorijel(n-1);
}
```

Primer — da li je ovo repna rekurzija?

- Funkcija *nzd*: —>DA

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
    return a;
  else
    return nzd(b, a % b);
}
```

- Funkcija *faktorijel*: —>NE

```
unsigned faktorijel(unsigned n) {
  if (n == 0)
    return 1;
  else
    return n*faktorijel(n-1);
}
```

Eliminacija repne rekurzije — primer

```
unsigned nzd(unsigned a, unsigned b) {
  if (b == 0)
    return a;
  else
    return nzd(b, a % b);
}
```

```
unsigned nzd(unsigned a, unsigned b) {
pocetak:
  if (b == 0)
    return a;
  else {
    unsigned tmp = a % b;
    a = b; b = tmp;
    goto pocetak;
  }
}
```

Eliminacija repne rekurzije — primer

```
unsigned nzd(unsigned a, unsigned b) {
pocetak:
  if (b == 0)
    return a;
  else {
    unsigned tmp = a % b;
    a = b; b = tmp;
    goto pocetak;
  }
}
```

```
unsigned nzd(unsigned a, unsigned b) {
  while (b != 0) {
    unsigned tmp = a % b;
    a = b; b = tmp;
  }
  return a;
}
```

Pogledati eliminaciju repne rekurzije u opštem slučaju u knjizi!

Eliminacija repne rekurzije — primer

```
unsigned nzd(unsigned a, unsigned b) {
pocetak:
  if (b == 0)
    return a;
  else {
    unsigned tmp = a % b;
    a = b; b = tmp;
    goto pocetak;
  }
}
```

```
unsigned nzd(unsigned a, unsigned b) {
  while (b != 0) {
    unsigned tmp = a % b;
    a = b; b = tmp;
  }
  return a;
}
```

Pogledati eliminaciju repne rekurzije u opštem slučaju u knjizi!

Pregled

Literatura

- Slajdovi su pripremljeni na osnovu knjige
  Predrag Janičić, Filip Marić: Programiranje 2
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je učiti iz knjige!