Tema 7

Bitwise
Dinamička alokacija memorije
Lista

Operacije nad bitovima

Programski jezik C ima jedan broj operatora čije je delovanje definisano na bitovima.
Takvi se operatori mogu primeniti na celobrojne tipove podataka `char, short, int, long, unsigned`.

To su sledeći operatori:

Operator	Značenje
&	logičko I bit-po-bit
|	logičko ILI bit-po-bit
^	ekskluzivno logičko ILI bit-po-bit
<<	levi pomak (left-shift)
>>	desni pomak (right-shift)
~	1-komplement

`&= |= ^= <<= >>=`

Tabela hexa i bin vrednosti

Dec.	0	1	2	3	4	5	6	7
Hexa.	0	1	2	3	4	5	6	7
Bin.	0000	0001	0010	0011	0100	0101	0110	0111

Dec.	8	9	10	11	12	13	14	15
Hexa.	8	9	A	B	C	D	E	F
Bin.	1000	1001	1010	1011	1100	1101	1110	1111

Zadatak 1

Konvertovati iz binarnog u hexadecimalni zapis:

0100 0111 0101 0111

1101 0100 1010 1001

Konvertovati iz hexadecimalnog u binarni zapis:

0x8000

0x1234

0x9B6C

Primer 1- Za uneseni pozitivan broj ispisuje koliko on ima jedinica u binarnom zapisu

```
#include<stdio.h>

int main()
{
        int x, i, b=0, br_cifra=0;

        printf("Unesite broj: ");
        scanf("%d", &x);

       /* U narednoj while petlji proveravamo bit po bit od poslednjeg bita ka prvom.*/
        while(x!=0)
        {
                /* x & 1 proverava da li je poslednja binarna cifra broja x jednaka 1 */
                if (x & 1)
                    b++;

                x>>=1;
        }

   printf("\n\nBroj jedinica u binarnom zapisu: %d\n", b);

   return 0;
}
```

Primer 2- Ispisivanje u binarnom brojnom sistemu

```
#include <stdio.h>

int main ()
{
  // u petlji ponavljamo sledece dve radnje: unos broja i ispis tog broja u binarnom obliku
  while (1) {
    short dec;
    int i;

    printf ("Decimalni broj? ");
    scanf ("%hd", &dec);

    if (dec == 9999) break; //kada se unese 9999 izlazimo iz programa

    printf ("Binarni broj:   ");
    for (i=1; i<=16; i++) {
      printf ("%d", (dec & 0x8000) != 0);
      dec <<= 1;
      if (i % 4 == 0) printf (" ");
    }
    printf ("\n");
  }

  return 0;
}
```

Zadatak 2

- Učitati dva pozitivna cela broja. Prikazati ih u binarnom zapisu, kao i njihove komplemente i rezultate logičkog I, logičkog ILI i ekskluzivnog logičkog ILI. Realizovati funkciju koja celobrojnu vrednost prikazuje kao binarnu.

Zadatak 3

- Napisati funkciju koja omogućava očitavanje n-tog bita registra u procesoru. - getBit
- Napisati funkciju koja omogućava “setovanje” n-tog bita registra u procesoru na jedinicu. – setBit
- Napisati funkciju koja omogućava postavljanje n-tog bita registra u procesoru na 0. – clearBit
- Napisati funkciju koja omogućava promenu (1->0 i 0->1) n-tog bita registra u procesoru. - flipBit
- Napisati test program koji ilustruje upotrebu ovih funkcija.

Bit pariteta - primer

7 bits of data (number of 1s)	8 bits including parity	
	even – parni paritet	odd – neparni paritet
0000000 (0)	**0**0000000	**1**0000000
1010001 (3)	**1**1010001	**0**1010001
1101001 (4)	**0**1101001	**1**1101001
1111111 (7)	**1**1111111	**0**1111111

Zadatak 4

- Napisati funkciju za zaštitu podataka neparnim paritetom. Podaci su dužine 7 bita. Napisati test program koji ilustruje rad prethodne funkcije nad nizom podataka.
- Napisati funkciju za zaštitu podataka parnim paritetom. Podaci su dužine 7 bita. Napisati test program koji ilustruje rad prethodne funkcije nad nizom podataka.

Zadatak 4 – primer main-a

```
int main() {

    char data[]={'0', 'Q', '(', 'f', ','};
    int n=5;

    // funkcija printData ispisuju podatke u binarnom obliku
    // funkcija neparniParitet postavlja bite pariteta
    printData(data, n);
    neparniParitet(data, n);
    printData(data, n);

    return 0;
}
```

Zadatak 5

- Napisati funkciju za proveru ispravnosti primljenih podataka Podaci su zaštićeni neparnim paritetom. Podaci su dužine 7 bita + bit parnosti. Napisati test program koji ilustruje rad prethodne funkcije nad nizom podataka.

cross-parity check

Letter	ASCII Code (7-bit)	Parity bit (LRC)
H	1001000	0
E	1000101	1
L	1001100	1
L	1001100	1
0	1001111	1
VRC	1000010	0

Zadatak 6

- Napisati funkciju za zaštitu podataka neparnim paritetom, ali tako da je moguće i ispraviti grešku. Podaci su dužine 7 bita. Napisati test program koji ilustruje rad prethodne funkcije nad nizom podataka.

Dinamička alokacija memorije

Primer 1

```
#include <stdio.h>
#include <stdlib.h> /* u stdlib.h definisane su funkcije malloc i free */

int main () {
     int *br;
     br = (int *) malloc (sizeof(int));
/* malloc-u se prosledjuje kolicina potrebne memorije u bajtovima, a on vraca
     genericki pokazivac (void*) koji pre koriscenja treba konvertovati u
     odgovarajuci pokazivacki tip.*/

   printf("Unesite celi broj: ");
   scanf("%d", br);

   printf("Ucitan broj je: %d\n\n", *br);

     free(br); /* oslobadjanje (dealociranje) */

     return 0;
}
```

Funkcije za dinamičko upravljanje memorijom

```
void *malloc (size_t size);
void *calloc(int n, size_t size);
void *realloc(void *ptr,size_t size);
void free (void * ptr);
```

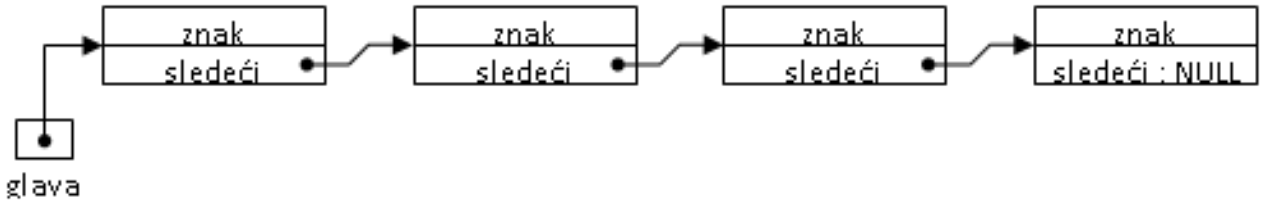
```c
/*Unos niza proizvoljne dimenzije i pronalaženje najveceg elementa                */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n;// Dimenzija niza
    int* a;
    int i, max;

    printf("Unesi dimenziju niza : ");
    scanf("%d", &n);

    /*  Sada znamo koliko je memorije potrebno i pozivamo funkciju malloc za dinamicku alokaciju memorije*/
    a = (int*) malloc(n*sizeof(int));   // U slucaju da nema dovoljno memorije malloc vraca NULL
    if (a == NULL)
    {
        printf("Greska : Nema dovoljno memorije!\n");
        return 1;
    }
    /* Nadalje a koristimo indeksnom notacijom                    */
    for (i = 0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
    }
    /* Nalazimo maksimum                          */
    max = a[0];
    for (i = 1; i<n; i++)
        if (a[i] > max)
            max = a[i];
    printf("Najveci element je %d\n", max);
    free(a);
    return 0;
}
```

U nastavku će biti objašnjena dinamička struktura podataka u obliku jednostruko spregnute liste. Element liste sastoji se od dva polja:

- **podatak** (u ovom slučaju proizvoljan karakter)
- **pokazivač** na sledeći element liste (sadrži adresu sledećeg elementa liste)

Da bismo mogli pristupati elementima liste potreban nam je pokazivač na prvi elemenat liste, koji se naziva **glava** liste.

Grafička predstava jednostruko spregnute liste karaktera je:



Operacije sa listom:

1) inicijalizacija liste,
2) unos novog elementa na pocetak liste,
3) listanje liste (prikaz svih elemenata iz liste),
4) brisanje elementa iz liste i
5) brisanje liste (oslobađanje memorije).

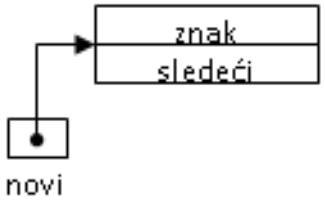
1) Inicijalizacija znači da kreiramo praznu listu bez elemenata. To postižemo tako što promenljivoj glava dodelimo vrednost NULL. Promenljiva glava treba da pokazuje na prvi element liste, pa ako ovoj promenljivoj dodelimo vrednost NULL to znači da ona ne pokazuje ni na šta, tj da nema ni jedan element u listi.

```
glava=NULL;
```

2) Unos novog elementa u listu može se obaviti dodavanjem elementa na početak liste ili dodavanjem elementa na kraj liste. Dodajemo na kraj.
Sa:

```
novi=(Tcvor *)malloc(sizeof(Tcvor));
```

formira se slog koji je tipa 'Tcvor' i pokazivač 'novi' pokazuje na novoformirani slog.

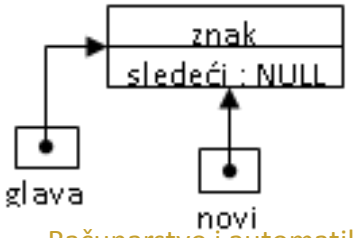


Polja sloga popunjavaju na osnovu sledećih naredbi:

```
c=getc();
novi->znak=c;
novi->sledeci=NULL;
```

- Ako je lista prazna tada se novi elemenat ubacuje kao prvi element liste:

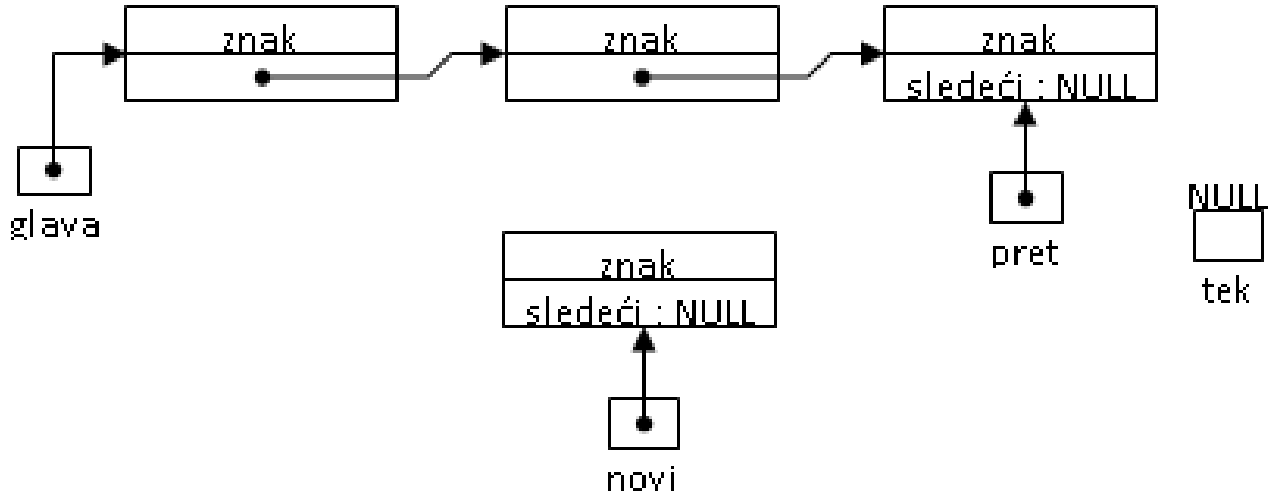
```
if(glava==NULL)
{
        glava=novi;
}
```



Ako lista već postoji preskočiće se gornja if naredba i unos će se obaviti sledećim delom koda:

```
tek=glava;
pret=glava;
while(tek!=NULL)
{
        pret=tek;
        tek=tek->sledeci;
}
pret->sledeci=novi;
```

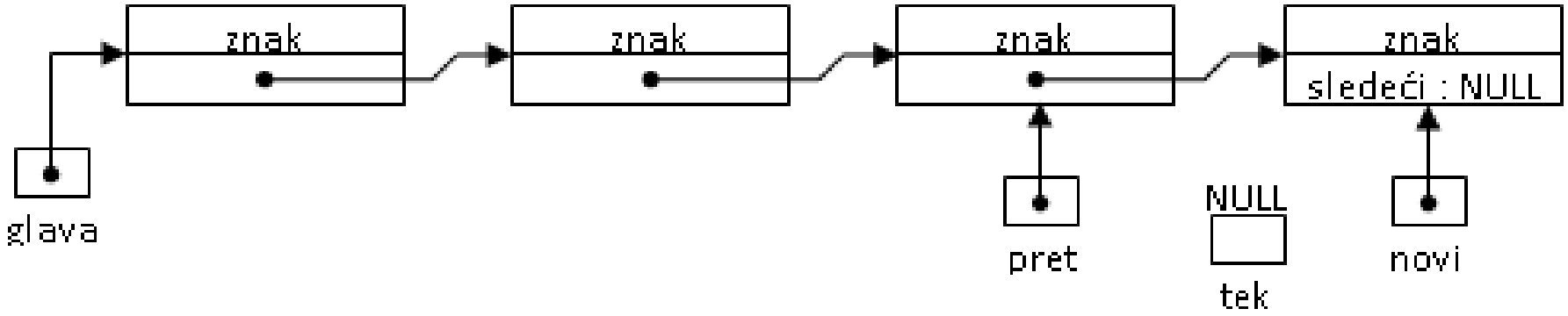
- Nakon prolaska kroz while ciklus imaćemo sledeću situaciju:
  '**tek**' ne pokazuje ni na šta, a '**pret**' koji ga je "pratio" pokazaće na poslednji slog iz liste.



Nakon:

```
pret->sledeci=novi;
```

novi slog će biti povezan kao poslednji elemenat u listi.



3) Listanje liste predstavlja prikazivanje svih elemenata iz liste

- Ako je glava==NULL znači da je lista prazna i nemamo šta prikazati.
- Ako lista nije prazna, novim pokazivačem polazi se od prvog sloga liste (tek=glava) i dok se ne dođe do kraja liste (tek==NULL) prikazuju se slogovi liste.

```
tek=glava;
while(tek!=NULL)
{
    printf("   %c",tek->znak);     /* Sa strelicom (->) se pristupa
                                   elementu sloga (strukture). */
    tek=tek->sledeci;              /* Prelazimo na sledeći elemenat iz liste. */
}
```

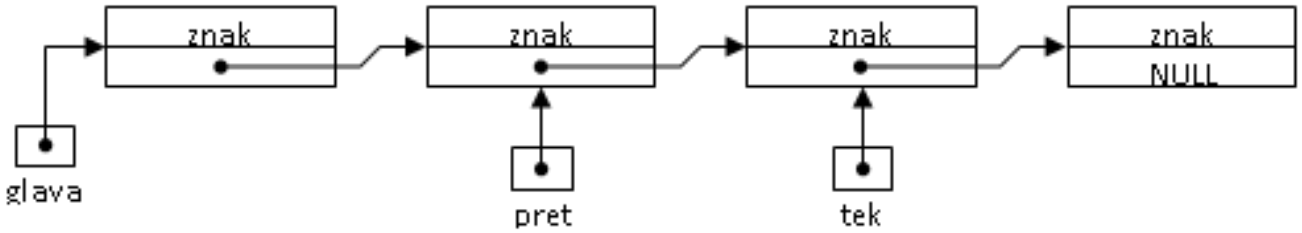
4) Brisanje elementa iz liste realizuje se tako što se mora element prvo pronaći (vrši se traženje elementa u listi). To se može realizovati sledećim delom koda:

```
tek=glava;
pret=glava;
while(tek!=NULL && (tek->znak != c))
{
        pret=tek;
        tek=tek->sledeci;
}
```

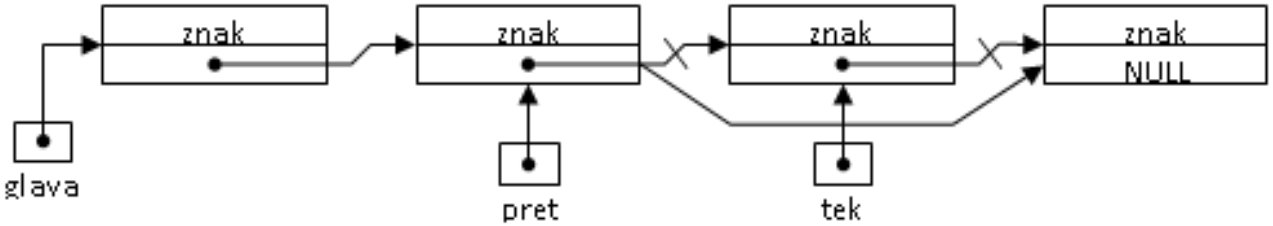
Pri čemu je 'c' promenljiva čija vrednost se traži. Nakon završetka while ciklusa ako pokazivač 'tek' ima vrednost NULL znači da elementa nema u listi. U slučaju da je tek!=NULL elemenat se nalazi u listi, a pokazivač 'pret' pokazuje na njegovog prethodnika iz liste.
Razlikujemo dva slučaja:

a) da slog koji se briše nije prvi elemenat liste,
b) da slog koji se briše jeste prvi element liste (u tom slučaju moramo ažurirati promenljivu glava koja pokazuje na prvi element).

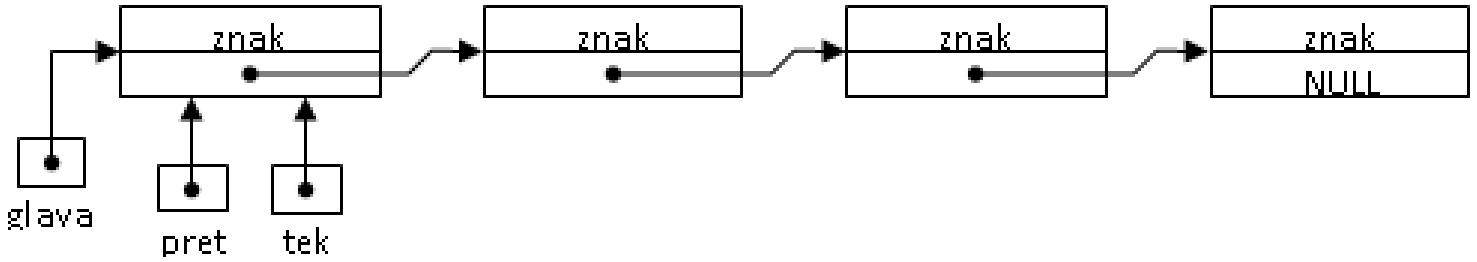
a) Slog koji se briše nije prvi elemenat liste



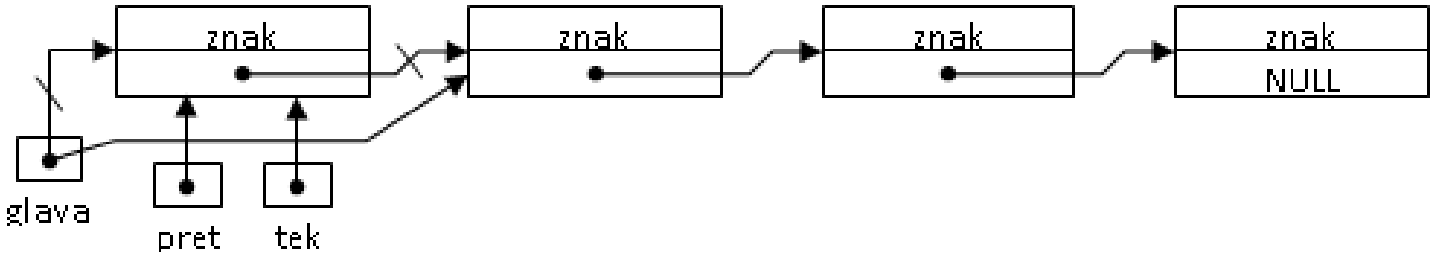

```
pret->sledeci=tek->sledeci;
tek->sledeci=NULL;
free(tek);
```



b) Slog koji se briše jeste prvi elemenat liste

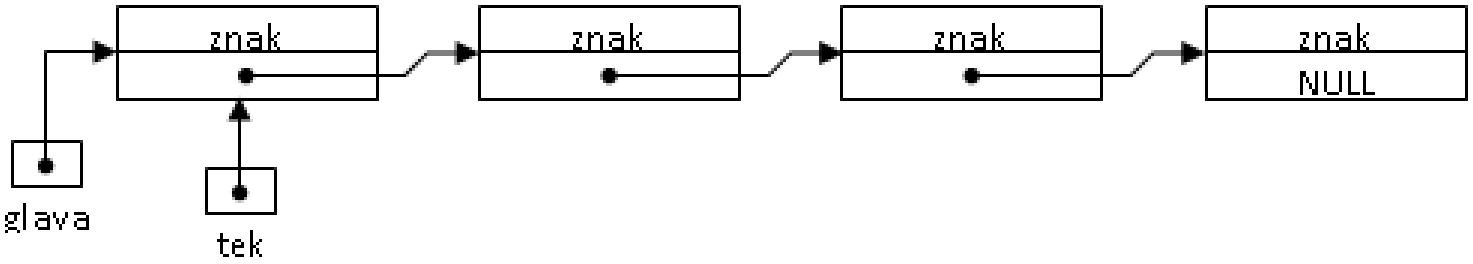


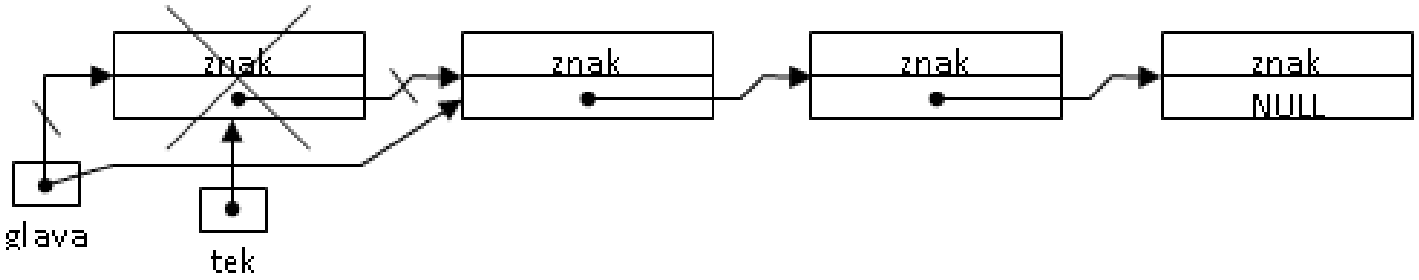

```
glava=tek->sledeci;
tek->sledeci=NULL;
free(tek);
```



5) Brisanje liste

```
while(glava!=NULL)
{
        tek=glava;
        glava=tek->sledeci;
        free(tek);
}
```





Zadatak 1

Napisati C program koji realizuje navedene operacije sa dinamičkom strukturom podataka oblika jednostruko spregnute liste. Element liste sastoji se od dva polja:

- **podatak** (jedan karakter)
- **pokazivač** na sledeći element liste

Operacije koje treba realizovati su:

- **unos novog karaktera,** ako se ne nalazi u listi,
- **modifikacija karaktera** (umesto jednog karaktera treba upisati drugi karakter),
- **traženje karaktera** u listi i
- **brisanje karaktera** iz liste.

Zadatak iz zbirke „Rešeni zadaci iz programskog jezika C“

- Zadatak 7.6
  - Iterativno (ciklusima)
  - Rekurzivno
- Zadatak 7.7