Najjednostavnija izvedba stoga

- Implementacija stoga u C-u pomoću polja vrlo je jednostavna: potrebno nam je jedno polje, stack, u koje ćemo ubacivati i iz kojega ćemo izbacivati elemente, te dvije funkcije, push i pop koje ubacuju odnosno izbacuju element s vrha stoga. Slijedi primjer u kojem se stog puni cijelim brojevima, ispisuje se i zatim se skida element po element dok se stog ne isprazni.

```
#include <stdio.h>
#include <stdlib.h>
#define N 20
int stack[N];
int top = 0;      /*vrh stoga u početku postavljen na dno stacka */
void push(int el){
     stack[top++] = el;
     }
int pop(void){
     return stack[--top];
}

int main(){
int i;
   for(i = 0; i < 10; i++)
              push(i + 1);
   for (i = 0; i < N; i++) {
       printf("stack[%i]: %i\n ",i,stack[i]);
      }
   for(i = 0; i < 10; i++)
              printf("%i", pop());
   system("PAUSE");
   return 0;
}
```

Računanje algebarskih izraza zapisanih u obrnutoj poljskoj notaciji

- Uobičajeni način zapisivanja algebarskih izraza je onaj kod kojega simboli operatora dolaze *između* operanada. Npr. u izrazu 2 + 3 * 5 operatori + i * dolaze između brojeva 2 i 3, odnosno 3 i 5. Redoslijed računanja određen je prioritetom računskih operacija (osim ako nije promijenjen upotrebom zagrada). Tako je 2 + 3 * 5 = 17 jer se prvo obavlja množenje a tek onda zbrajanje.
- Međutim, postoji i drugačiji način zapisivanja izraza, koji se zove **Obrnuta poljska notacija** (engl. **RPN**, **R**everse **P**olish **N**otation). Tu notaciju je prvi uveo poljski matematičar Jan Lukasiewicz. Kod ovog načina, simboli operacija dolaze **nakon** operanada, a računanje se izvodi upravo onim redom kojim su napisane operacije. Zagrade se ne koriste.
- Obrnutu poljsku notaciju koriste gotovo svi HP-ovi kalkulatori, a značajna je i zato jer računala interno upravo tim algoritmom izračunavaju algebarske izraze.

- Evo nekoliko primjera upotrebe RPN-a pri zapisivanju izraza:

- Primjer 1: Izraz 2 + 3 u RPN-u bismo zapisali kao 2 3 +
- Primjer 2: Izraz 2 + 3 * 5 u RPN-u ima oblik 2 3 5 * +
- Primjer 3: Izraz (2 + 3) * 5 u RPN-u ima oblik 2 3 + 5 *

- Želimo sastaviti algoritam kojemu bismo na ulazu dali proizvoljni (korektno zapisani) RPN izraz, a algoritam kao rezultat svog rada treba izračunati vrijednost ulaznog izraza. Rad algoritma je prilično jednostavan - čitamo izraz slijeva nadesno i zatim:
- svaki put kad pročitamo broj, stavimo ga na stog (operacija push).
- svaki put kad pročitamo znak neke operacije, tada skinemo (operacija pop) dva broja sa stoga, nad njima izvršimo traženu operaciju i zatim rezultat te operacije stavimo na stog (operacija push).

- Donji program kao ulaz prima RPN-izraz čiju vrijednost zatim izračunava i ispisuje rezultat. Kao oznaku kraja izraza koristi se znak ";". Stoga prilikom ukucavanja izraze treba unositi ovako

  2 3 5 * + ;

```
#include <stdio.h>
#include <stdlib.h>
#define N 1000

float stack[N];
int top = 0;
void push(float el){
    stack[top++] = el;
}
float pop(){
    return stack[--top];
}

int main(){
 char unos[200];
 float a, b, rez;
 printf("Unesi izraz u RPN obliku: ");
 do {
            scanf("%s", unos);
 switch(unos[0]) {
    case '+':
            b = pop();
            a = pop();
            push(a + b);
            break;
```

```
    case '-':
            b = pop();
            a = pop();
            push(a - b);
            break;
    case '*':
            b = pop();
            a = pop();
            push(a * b);
            break;
    case '/':
            b = pop();
            a = pop();
            push(a / b);
            break;
    case ';':
            rez = pop();
            break;
    default:
            push(atof(unos));
            break;
            }
    } while(unos[0] != ';');
    printf("Rezultat je: %f\n", rez);
    system("PAUSE");
    return 0;
}
```

Implementacija stoga pomoću polja

- Na predavanjima je uveden apstraktni tip podataka stog i napravljen C pseudokod za jednu njegovu varijantu. Napisati funkcije koje obavljaju zadane operacije na stogu upotrebom polja za primjer s predavanja.

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLENGTH 4

typedef int elementtype;
typedef struct stack * stackptr;
typedef struct stack STACK;

void MakeNull(stackptr s);
int Empty(stackptr s);
void Push(elementtype x, stackptr s);
void Pop(stackptr s);
elementtype Top(stackptr s);
void PrintErrorAndTerminate(char errormsg[]);
```

```
struct stack {
     int top;
     elementtype elements[MAXLENGTH];
};

void MakeNull(stackptr s) {
   s->top = MAXLENGTH;
}

int Empty(stackptr s) {
   if (s->top == MAXLENGTH)
        return 1;
   else
        return 0;
}

void Push(elementtype x, stackptr s) {
     if (s->top == 0)
              PrintErrorAndTerminate("Push: Stack je pun!");
     else
              s->elements[--s->top] = x;
}
```

```
void Pop(stackptr s){
    if (Empty(s))
        PrintErrorAndTerminate("Pop: Stack je prazan!");
    else
        s->top++;
}

elementtype Top(stackptr s)
{
    if (Empty(s))
        PrintErrorAndTerminate("Top: Stack je prazan!");

    return(s->elements[s->top]);
}

void PrintErrorAndTerminate(char errormsg[]){
    printf("%s\n", errormsg);
    exit(-1);
}
```

```
int main(){
 STACK s;

 MakeNull(&s);
Push(1, &s);
Push(2, &s);
Push(3, &s);
Push(4, &s);
 printf("%d\n", Top(&s));
Pop(&s);
 printf("%d\n", Top(&s));
Push(5, &s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
printf("%d\n", Top(&s));
Pop(&s);
system("PAUSE");
return 0;
}
```

Još jedan primjer stoga ostvarenog poljem

- Stog duljine 6 elemenata se puni generiranim slučajnim brojem ako je on neparan, ukoliko je generiran parni broj, vrh stoga se skine. Ispisati sadržaj stoga nakon svakog koraka te upozoriti ukoliko je stog prazan ili pun.

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

#define MAXSTOG 6

int dodaj (int stavka, int stog[], int n, int *vrh) {
  if (*vrh >= n-1) return 0;
  (*vrh)++;                          // ne pisati  *vrh++; !
  stog[*vrh] = stavka;
  return 1;
}
```

```
int skini (int *stavka, int Stog[], int *vrh) {
      if (*vrh < 0) return 0;
      *stavka = Stog[*vrh];
      (*vrh)--;
      return 1;
}

void main (void) {
  int novi, stari, stog [MAXSTOG];
  int vrh,i,ch;

  vrh = -1; // prazan stog

      printf (“Generiraju se slucajni nenegativni cijeli brojevi.\n");
      printf ("Neparni brojevi upisuju se na stog\n");
      printf ("Parni broj znaci skidanje sa stoga\n");
      printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n\n");

      srand ((unsigned) time (NULL));

      while (isspace(getchar())) {
            if (vrh == -1) {
                  printf("prazan stog");
            }
```

```
            else {
                        printf ("Stog:");
                        for (i=0; i <= vrh; ++i) printf (" %d", stog[i]);
            }
            putchar ('\n');
            novi = rand ();
            if (novi%2) {                       // Neparni se upisuju na stog
                        printf ("Dodaj %d\n", novi);
                        if (!dodaj (novi, stog, MAXSTOG, &vrh)) printf("Stog je pun!\n");
            } else {                            // Parni broj znaci skidanje sa stoga

                        printf ("Skini...");
                        if (skini (&stari, stog, &vrh))    {
                                    printf ("Skinut %d\n", stari);
                        } else {
                                    printf("Stog je prazan!\n");
                        }
    }
  }
     system("PAUSE");
     exit(0);
}
```

Implementacija stoga pomoću pokazivača

- Napisati program za realizaciju stoga pomoću pokazivača: iz ulazne datoteke se čitaju podaci i upisuju u stog. Ispisati vrijednosti elemenata u stogu i lokacije na kojima se nalaze. Nakon toga skidati podatke iz stoga sve dok ne ostane prazan.

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
typedef int tip;
struct cv {
  tip element;
  struct cv *sljed;
};
typedef struct cv cvor;

cvor *dodaj (cvor *vrh, tip element) {
  cvor *novi;
  if ((novi = (cvor *) malloc(sizeof(cvor))) != NULL) {
    novi->element = element;
    novi->sljed = vrh;
    printf("Na adresu %p dodao sam %d, a sljedeci je %p\n",novi, element, vrh);
  }
  return novi;          // vrati pokazivac na novi cvor
}
```

```
cvor *skini (cvor *vrh, int *element) {
     cvor *pom;

  *element = vrh->element;
     printf ("S adrese %p ", vrh);
     pom = vrh->sljed;
  free (vrh);                        // oslobodi vrh
  return pom;                        // vrati novi vrh
}

void main (void) {
  FILE *fi;
  int j;
  int element;          // element stoga
  cvor *vrh, *p;// pokazivac na vrh i pomocni pokazivac

  fi = fopen ("UlazZaStog.txt", "r");
  if (fi) {
             vrh = NULL;
             j = 0;
    while (fscanf (fi, "%d", &element) != EOF) {
             j++;
     if ((p = dodaj (vrh, element)) != NULL) {
       vrh = p;
             printf ("%d. ulazni podatak je %d\n", j, vrh->element);
 }
```

```
        else {
       printf("Nema vise mjesta za stog\n");
          break;
      }
    }
    fclose (fi);
    p = vrh;
              // Skidanje elemenata sa stoga
    while (vrh) {
      vrh = skini (vrh, &element);
      printf ("skinuo sam element %d\n", element);
    }
  }
   else {
    printf ("Nema ulazne datoteke\n");
              exit (1);
  }
  system("PAUSE");
 exit(0);
}
```

Izvedba reda pomoću polja

- Napisati program u kojem se red realizira upotrebom polja. Efikasan način realizacije reda statičkom strukturom je jednodimenzionalno polje zadane podatkovne strukture koje se koristi cirkularno. Koriste se dva indeksa (ulaz i izlaz), a cirkularnost se ostvaruje uporabom operatora modulo (%) (vidjeti predavanja). Ulazni podaci čitaju se iz datoteke dok se red ne popuni. Ukoliko ima još elemenata u ulaznoj datoteci, izbrišu se postojeći elementi iz reda, te se upisuju novi dokle god ima podataka u ulaznoj datoteci. Prije završetka izvršavanja programa izbrisati sve elemente reda. Ispisati broj elemenata u redu i element na kojem se obavlja operacija nakon svake operacije.

```
#include <stdlib.h>
#include <stdio.h>
#define MAXRED 10
typedef int tip;

// dodaje element u polje red od max n clanova, mijenja ulaz, tj straznji kraj
// vraca 1 ako ima mjesta u redu, inace 0
int DodajURed (tip element, tip red[], int n, int izlaz, int *ulaz) {
  if (((*ulaz+1) % n) == izlaz) return 0;
  (*ulaz)++;
  *ulaz %= n;
  red [*ulaz] = element;
  return 1;
}

// logicki uklanja element iz polja red od max n clanova, mijenja izlaz, tj prednji kraj
// vraca 1 ako ima clanova u redu, inace 0
int SkiniIzReda (tip *element, tip red[], int n, int *izlaz, int ulaz) {
  if (ulaz == *izlaz) return 0;
  (*izlaz) ++;
  *izlaz %= n;
  *element = red[*izlaz];
  return 1;
}
```

```
// vraca broj elemenata u redu
int prebroji (int n, int izlaz, int ulaz) {
  if (ulaz >= izlaz) {
            return (ulaz - izlaz);              // standardno
     } else {
       return (ulaz - izlaz + n);            // cirkularnost
     }
}

void main (void) {
  int red[MAXRED];
  int element, ulaz, izlaz;
  FILE *fi;

  ulaz = 0; izlaz = 0;
     fi = fopen ("UlazZaRed.txt", "r");
     if (fi) {
            while (fscanf (fi, "%d", &element) != EOF) {
      if ((DodajURed (element, red, MAXRED, izlaz, &ulaz))) {
                        printf ("U red dodan element %d\n", element);
                        printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
                        } else {
                                    printf ("Nema vise mjesta u redu\n\n\n");
```

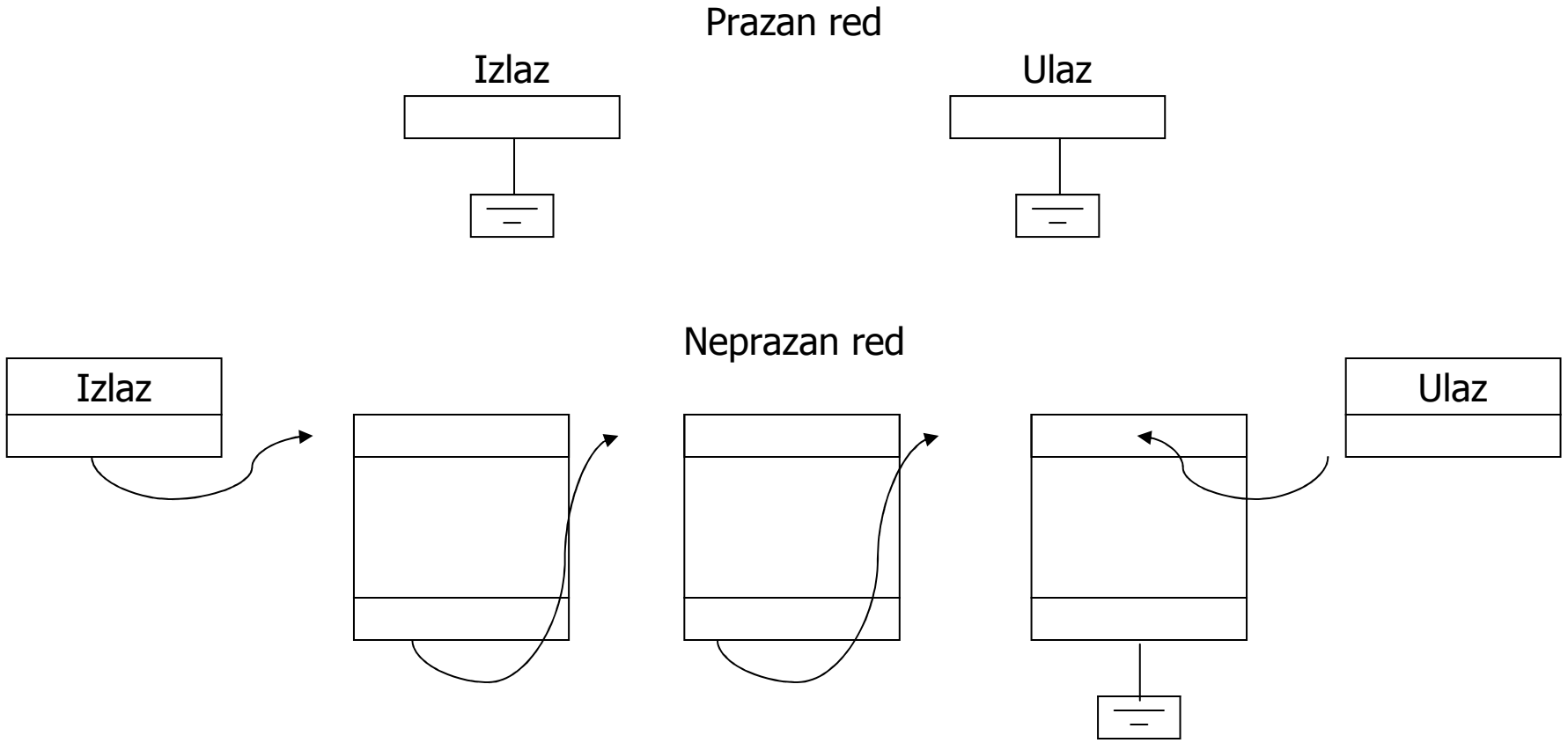
```
        // uklanjanje iz reda
            while (SkiniIzReda (&element, red, MAXRED, &izlaz, ulaz)) {
                        printf ("Iz reda skinut element %d\n", element);
                        printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
                                    }
                                    printf("\n");
     }
    }
    fclose (fi);

            // uklanjanje preostalih elemenata
            printf("\n\n");
            while (SkiniIzReda (&element, red, MAXRED, &izlaz, ulaz)) {
                        printf ("Iz reda skinut element %d\n", element);
                        printf ("Broj elemenata u redu je %d\n",prebroji (MAXRED, izlaz, ulaz));
            }

  } else {
    printf ("Nema ulazne datoteke\n");
            exit (1);
  }
     system("PAUSE");
     exit (0);
}
```

Izvedba reda pomoću pokazivača

- Napisati program za izvedbu reda pomoću pokazivača. Upotrijebiti generator slučajnih brojeva za punjenje reda: neparni broj se upisuje na kraj reda (ulaz), a ako je generiran parni broj briše se iz reda prvi broj na početku reda (izlaz).



```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <ctype.h>
struct cv {
    int element;
    struct cv *sljed;
};
typedef struct cv cvor;

// dodaje element u red, vraca 1 ako uspije, inace 0
int DodajURed (int element, cvor **ulaz, cvor **izlaz) {
    cvor *novi;
    if (novi = malloc (sizeof (cvor))) {
        novi->element = element;
        novi->sljed = NULL;
        if (*izlaz == NULL) {
            *izlaz = novi;          // ako je red bio prazan
        } else {
            (*ulaz)->sljed = novi;          // inace, stavi na kraj
        }
        *ulaz = novi;       //zapamti zadnjeg
        return 1;
    }
    return 0;
}
```

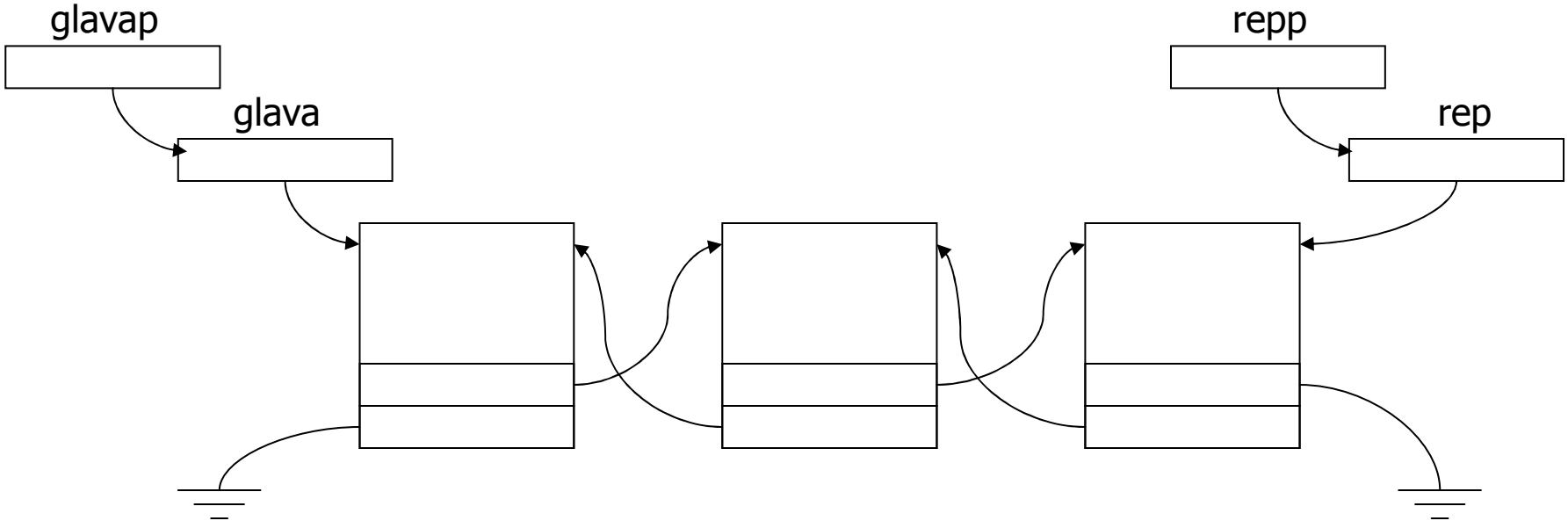
```
// uklanja element iz reda, vraca 1 ako uspije, inace 0
int SkiniIzReda (int *element, cvor **ulaz, cvor **izlaz) {
     cvor *stari;
     if (*izlaz) {                    // ako red nije prazan
          *element = (*izlaz)->element;          // element koji se skida
          stari = *izlaz;                    // zapamti trenutni izlaz
          *izlaz = (*izlaz)->sljed;                              // novi izlaz
          free (stari);                                          // oslobodi memoriju skinutog
          if (*izlaz == NULL) *ulaz = NULL; // prazan red
          return 1;
  }
     return 0;
}

// vraca broj elemenata u redu
int Prebroji (cvor *izlaz) {
     int n;
     for (n = 0; izlaz; izlaz = izlaz->sljed) {
          printf ("%d -> ", izlaz->element);
          n++ ;
     }
     printf ("NULL\n");
     return n;
}
```

```
void main (void) {
  int broj;
  cvor *ulaz = NULL;              // krajevi
  cvor *izlaz = NULL;
     printf (“Generiraju se slucajni nenegativni cijeli brojevi.\n");
     printf ("Neparni brojevi upisuju se u red, a parni broj znaci skidanje iz reda\n");
     printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n");
     srand ((unsigned) time (NULL));
     while (isspace(getchar())) {
            broj = rand ();
            if (broj%2) {                       // Neparne upisujemo u red
                        printf ("U red se upisuje broj %d\n", broj);
                        if (!DodajURed (broj, &ulaz, &izlaz))
                                    printf("Nema vise memorije\n");
    } else {                                    // Parni broj znaci skidanje iz reda
                        if (SkiniIzReda (&broj, &ulaz, &izlaz)) {
                                    printf ("Iz reda je skinut podatak %d\n", broj);
                        } else {
                                    printf("Red je prazan\n");
                        }
    }
            printf ("Broj elemenata u redu: %d\n", Prebroji (izlaz));
  }
  system(“PAUSE”);
}
```

Izvedba reda kao dvostruko povezane liste

- Radi bržeg traženja u oba smjera kretanja po listi, ona može biti dvostruko povezana. Svaki čvor osim elementa s podacima, sadrži pokazivač na sljedeći čvor i pokazivač na prethodni čvor.
- Lista ima *glavu* i *rep*, što je prikladno za izvedbu reda.
- funkcije za dodavanje i skidanje rukuju s pokazivačima na glavu (*glavap*) i rep (*repp*)



- Napisati program za realizaciju reda kao dvostruko povezane općenite liste s funkcijama koje upisuju novi element na kraj reda i skidaju prvi element na čelu reda. Također napisati funkciju za dvostruko povezanu listu koja skida proizvoljan element iz liste tražeći vrijednost elementa koji skida. Punjenje i pražnjenje reda ostvariti upotrebom generatora slučajnih brojeva gdje se neparni broj upisuje na kraj reda, a parni broj znači skidanje prvog člana u redu. Ispisati red nakon svake operacije. Na kraju obrisati član reda čija se vrijednost učita s tastature (to nije funkcija za red nego za listu).

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

struct cv2 {
    int element;
    struct cv2 *sljed;
    struct cv2 *preth;
};
typedef struct cv2 cvor2;
```

```
// dodavanje u red realizirano dvostruko povezanom listom
// funkcija vraca 1 ako uspije, inace 0
int DodajURed (int element, cvor2 **glavap, cvor2 **repp) {
    cvor2 *novi;

    if (novi = malloc (sizeof (cvor2))) {
        novi->element = element;
        novi->sljed = NULL;
        novi->preth = NULL;

        if (*glavap == NULL) {                    // Ako je red bio prazan
            *glavap = novi; *repp = novi;
        }
     else {                                       // inace, stavi na kraj
            (*repp)->sljed = novi;
            novi->preth = *repp;
            *repp = novi;
        }
        return 1;
    }
    return 0;
}
```

```
// skidanje iz reda, funkcija vraca 1 ako uspije, inace 0
int SkiniIzReda (int *element, cvor2 **glavap, cvor2 **repp) {
    cvor2 *stari;

    if (*repp) {       // provjera da li je red prazan
        *element = (*glavap)->element; // vrati element
        if (*glavap == *repp) {         // Ako je samo jedan clan
            stari = *glavap;
            *glavap = NULL; *repp = NULL;
        } else {                        //inace, povezi ih
            (*glavap)->sljed->preth = NULL; // prvi u redu nema prethodnika
            stari = *glavap;
            *glavap = stari->sljed;    // nova glava je sljedbenik stare
        }  free (stari);
        return 1;
  }
    return 0;
}

// ispis reda
void IspisiRed (cvor2 *glava) {
    for (; glava; glava = glava->sljed)
        printf ("%d ", glava->element);
    printf ("\n");
}
```

```
// brisanje iz reda clana sa zadanim kljucem
int BrisiIzReda (cvor2 **glavap, cvor2 **repp, int element) {
    cvor2 *pom;

    if (*glavap) {     // neprazan red
        for (pom = *glavap; pom && (pom->element != element); pom = pom->sljed)
            ;
        if (pom) {// Ako je nadjen,
            if (pom == *glavap) {          // ako je prvi
                *glavap = pom->sljed;
                if (pom->sljed)      {          // ako nije jedini
                    pom->sljed->preth = NULL;
                } else {                        //ako jest jedini
                    *glavap = NULL; *repp = NULL;
                }
            } else if (pom == *repp) {     // ako je zadnji, ali ne i jedini
                (*repp)->preth->sljed = NULL; // predzadnji postaje zadnji
                *repp = (*repp)->preth;
            } else {    // nije ni prvi ni zadnji
                pom->preth->sljed = pom->sljed;
                pom->sljed->preth = pom->preth;
            }  free (pom);
            return 1;
        }
    }       return 0;              // Nije nadjen ili lista prazna
}
```

```
void main (void) {
    cvor2 *glava = NULL;       // glava reda
    cvor2 *rep = NULL;         // rep reda
  int broj;

    printf ("Generiraju se slucajni nenegativni cijeli brojevi.\n");
    printf ("Neparni brojevi upisuju se u red, a parni broj simulira skidanje iz reda\n");
    printf ("Za obavljanje jednog koraka pritisnuti ENTER, za kraj bilo koji znak\n");

    srand ((unsigned) time (NULL));
    while (isspace(getchar())) {
            broj = rand ();
            if (broj%2) { // Neparne upisujemo u red
                    printf ("U red se upisuje broj %d\n", broj);
                    if (!DodajURed (broj, &glava, &rep))
                            printf("Nema vise memorije\n");
   } else {                                 // Parni broj znaci skidanje iz reda
                    if (SkiniIzReda (&broj, &glava, &rep)) {
                            printf ("Iz reda je skinut podatak %d\n", broj);
                    } else {
                            printf("Red je prazan\n");
                    }
   }
            IspisiRed (glava);
  }
```

```
    // brisanje iz reda bilo kojeg clana
    printf("\nSad cemo red tretirati kao listu, brise se bilo koji clan\n");
    while (1) {
            IspisiRed (glava);
            printf ("Upisite podatak koji se brise iz reda >");
            scanf ("%d", &broj);
            if (!BrisiIzReda (&glava, &rep, broj))
                    break;
    }
    system("PAUSE");
    exit (1);
}
```