

# STRUKTURE PODATAKA I ALGORITMI

## REŠENJA ZADATAKA SA I KOLOKVIJUMA ODRŽANOG 31.03.2007.

### 1. Zadatak

```
// Napisati metodu „Ubaci“ koja implementira algoritam za
// ubacivanje novog elementa u stak koji je implementiran
// kao jednostruko spregnuta lista.
void Ubaci(int Element)
{
    // kreira se novi čvor liste, kome je sledeći element onaj
    // koji je do tada bio prvi i pokazivač na početak liste
    // se postavlja da pokazuje na novokreirani
    Head = new CvorListe(Element, Head);
}

// Napisati metodu „Izbaci“ koja implementira algoritam za
// izbacivanje elementa iz stak koji je implementiran
// kao jednostruko spregnuta lista.
int Izbaci()
{
    // ako je lista prazna, onda nemamo šta da izbacimo
    if (Head == null)
        return Integer.MIN_VALUE;
    // prihvati podatak iz elementa koji se izbacuje
    int a = Head.Podatak;
    // premesti pokazivač početka liste na sledeći čvor i time
    // izbaci prvi element
    Head = Head.Sledeci;
    // vrati broj koji se nalazio u izbačenom elementu
    return a;
}

// Napisati metodu „Ubaci“ koja implementira algoritam za
// ubacivanje novog elementa u reda koji je implementiran preko niza.
void Ubaci(int Element)
{
    // Promenljiva "pun" se ažurira tako da označava da li je niz pun
    if (pun)
        return; // nema mesta, te se ubacivanje ne vrši.
    // promenljiva "poc" označava mesto na koje se ubacuje novi element
    niz[poc] = Element;

    // nakon ubacivanja treba ažurirati "poc" tako da pokazuje
    // na novo prazno mesto. Prvo se poc poveća za 1, a zatim se
    // uzima ostatak od deljenja sa dimenzijom niza. Na taj način
    // se poc vraća na 0 ako je došao do kraja niza.
    poc = ++poc % niz.length;
    // ili: (poc == niz.length) ? poc = 0 : poc++;

    // ako se poc i kraj poklope to znači da je niz pun
    pun = (poc == kraj);
}

// Napisati metodu „Izbaci“ koja implementira algoritam za
// izbacivanje elementa iz reda koji je implementiran preko niza.
int Izbaci()
```

```

{
    // ako je niz prazan, onda nemamo šta da izbacimo
    if (! pun && poc == kraj)
        return Integer.MIN_VALUE;

    // promenljiva "kraj" označava mesto sa koga se izbacuje element
    int a = niz[kraj];

    // nakon izbacivanja treba ažurirati "kraj" tako da pokazuje na mesto
    // sledeceg elementa koji se izbacuje. Prvo se kraj poveća za 1,
    // a zatim se uzima ostatak od deljenja sa dimenzijom niza.
    // Na taj način se kraj vraća na 0 ako je došao do kraja niza.
    kraj = ++kraj % niz.length;
    // ili: (kraj == niz.length) ? kraj = 0 : kraj++;

    // nakon izbacivanja niz sigurno nije više pun
    pun = false;

    // vrati izbačeni element
    return a;
}

```

## 2. Zadatak

```

// Dat je niz celih brojeva sortiran u rastućem redosledu.
// Napisati operaciju koja ubacuje nov element u niz tako da
// niz ostane sortiran. Pretpostaviti da ima mesta da se nov element ubaci.
void Ubaci_U_Sort(int[] niz, int Element)
{
    // počinje se od kraja niza
    int i = niz.length - 1;
    // prolazi se od kraja niza ka početku i svi elementi koji su veći od
    // onog koji ubacujemo se pomeraju za 1 mesto unapred,
    // oslobađajući tako mesta za novi element
    while(i > 0 && niz[i-1] > Element)
    {
        niz[i] = niz[i-1];
        i--;
    }
    // u oslobođeno mesto ubacujemo novi element
    niz[i] = element;
}

```

## 3. Zadatak

```

// Dat je pokazivač na kraj dvostruko spregnute liste celih brojeva.
// Napisati funkciju koja će prvi čvor prebaciti na poslednje mesto.
void PrviNaPoslednje(CvorDListe Kraj)
{
    // ako je lista prazna ili ako ima samo 1 element,
    // onda se ne radi nista
    if (Kraj == null || Kraj.Prethodni == null)
        return;

    CvorDListe Pom = Kraj;
    // premesti pomoćni pokazivač na početak liste
    while(Pom.Prethodni != null)
        Pom = Pom.Prethodni;
    // pošto se prebacuje prvi element, onda drugi postaje prvi, te mu
    // treba podesiti da nema prethodnika
}

```

```

Pom.Sledeci.Prethodni = null;
// prvi ide na poslednje mesto, te on nema sledbenika
Pom.Sledeci = null;

// da bi Pom postao poslednji onda treba da ga postavimo
// nakon trenutno poslednjeg elementa
Pom.Prethodni = Kraj;
Kraj.Sledeci = Pom;
}

// Dat je pokazivač na početak dvostruko spregnute liste celih brojeva.
// Napisati funkciju koja će poslednji čvor prebaciti na prvo mesto.
void PoslednjiNaPrvo(CvorDListe Prvi)
{
    // ako je lista prazna ili ako ima samo 1 element,
    // onda se ne radi nista
    if (Prvi == null || Prvi.Sledeci == null)
        return;

    CvorDListe Pom = Prvi;
    // premesti pomoćni pokazivač na kraj liste
    while(Pom.Sledeci != null)
        Pom = Pom.Sledeci;
    // pošto se prebacuje poslednji element, onda pretposlednji postaje
    // poslednji, te mu treba podesiti da nema sledbenika
    Pom.Prethodni.Sledeci = null;
    // poslednji ide na prvo mesto, te on nema prethodnika
    Pom.Prethodni = null;

    // da bi Pom postao prvi onda treba da ga postavimo
    // pre trenutno prvog elementa
    Pom.Sledeci = Prvi;
    Prvi.Prethodni = Pom;
}

```

## 4. Zadatak

```

// Napisati metodu Kloniraj(Stack Izvor, Stack Klon) koja klonira
// (identično kopira) stak celih brojeva. Dozvoljena je samo upotreba
// operacija nad stakom. Na kraju operacije početni stak treba da
// ostane nepromenjen.
void Kloniraj(Stack Izvor, Stack Klon)
{
    // kada se dođe do kraja staka, algoritam se prekida
    if (Izvor.Peek() != Integer.MIN_VALUE)
    {
        // uzmi element sa vrha staka
        int a = Izvor.Pop();
        // da se ne bi desilo da se elementi u novi stak ubace u
        // obrnutom redosledu, prvo pustimo da se naredni elementi ubace
        Kloniraj(Izvor, Klon);
        // a zatim ubacujemo uzeti element i u klon i u izvor kako bi na
        // kraju oni bili identični
        Klon.Push(a);
        Izvor.Push(a);
    }
}

```