

# Računanje sume članova polja

- Zadatak: napisati funkciju koja računa sumu članova polja.
- Najjednostavnije rješenje:

```
int suma1 (int polje[], int n) {                                složenost O(n)
    int i, suma = 0;
    for (i = 0; i < n; i++) {
        suma += polje[i]; }
    return suma;
}
```

- Rekurzivni postupak: sumiranje prvog člana s ostatkom polja:

```
int suma2 (int polje[], int n) {                                složenost O(n), prosječno asimptotsko
    if (n <= 0) return 0;                                       vrijeme izvođenja je 2n
    return polje[n - 1] + suma2(polje, n - 1);
}
```

- Podijeli-pa-vladaj algoritam: polje se dijeli u polovice koje se sumiraju

```
int suma3 (int polje[], int l, int d) { // l (d) – indeks lijevog (desnog) ruba polja
    int p;
    if (d < l) return 0;
    if (d == l) return polje[d];
    p = (d + l) / 2;
    return suma3 (polje, l, p) + suma3 (polje, p + 1, d);
}
```

- Složenost ovog algoritma je također  $O(n)$

- Glavna procedura:

```
void main() {
    int N,i,*a,s1,s2,s3;

    printf("Unesi duljinu polja:\n");
    scanf("%d",&N);
    if ((a = (int *)malloc(N*sizeof(int))) == NULL)
        {printf("Greska u rezerviranju memorije !");
        exit(1); }
```

```
srand(time(NULL));  
printf(" Elementi polja su:\n");  
for(i=0; i < N; i++) {  
    a[i]= 100 * ((float) rand()/(RAND_MAX + 1));  
    printf("%d ",a[i]); }  
printf("\n\n");
```

```
s1 = suma1(a,N);  
printf(" Rezultat funkcije suma1:%d\n",s1);  
s2 = suma2(a,N);  
printf(" Rezultat funkcije suma2:%d\n",s2);  
s3 = suma3(a,0,N-1);  
printf(" Rezultat funkcije suma3:%d\n",s3);  
system("PAUSE");
```

```
}
```

## Računanje binomnih koeficijenata – vrijeme izvršavanja

- Po definiciji binomni koeficijenti su:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}$$

- Rekurzija za računanje binomnih koeficijenata:

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

Ovakva rekurzija mnogo puta računa iste vrijednosti, pa je neefikasna

- Preurediti račun tako da se jednom izračunata vrijednost pospremi i sačuva za daljnja računanja – računanje Pascalovog trokuta
- Algoritam:
  - u prvom redu se upiše samo broj 1
  - za računanje elemenata u sljedećim redovima se uzimaju brojevi iz prethodnog reda koji su lijevo i desno od novog broja, ako broj lijevo ili desno ne postoji uzima se 0

```

          1
        1 1
       1 2 1
      1 3 3 1
     1 4 6 4 1
    1 5 10 10 5 1
   1 6 15 20 15 6 1
  1 7 21 35 35 21 7 1
 1 8 28 56 70 56 28 8 1
```

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys\timeb.h>
#define MAXRED 100

// vraca niz znakova c u zadanoj duljini n
char* nc (int c, int n) {
    static char s[80+1];
    s[n] = '\0'; // prirubi
    while (--n >= 0) s[n] = c; // popuni
    return s;}

// vraca faktorijela (n), broj iteracija, zastavicu pogreske, radi do n=20
long long FAKT (int n, long *freq, int *errorflag) {
    int i;
    long long p;
    p = 1;
    for (i = 2; i <= n; i++) {
        p *= i;
        if (p <= 0) *errorflag = 1;
    }
    *freq += 1; }
return p;}

```

```
// binomni koeficijenti s pomocu faktorijela
long BINOM (int n, int m, long *freq, int *errorflag) {
    long long p;
    *freq += 1;
    p = FAKT (n, freq, errorflag);
    p /= FAKT (m, freq, errorflag);
    p /= FAKT (n - m, freq, errorflag);
    return (long) p;
}
```

```
// binomni koeficijenti rekurzivno
long BINOMR (int n, int m, long *freq) {
    *freq += 1;
    if ((m == 0) || (m == n)) return 1;
    return BINOMR (n-1, m, freq) + BINOMR (n - 1, m - 1, freq);
}
```

```

// Pascalov trokut
void Blaise (int n) {
    int i, j;
    long stari[MAXRED], novi[MAXRED];

    if (n >= MAXRED) return;
    printf("\nIzracunavanje Pascalovog trokuta\n");
    novi[0] = 1;
    for (i = 0; i < n; i++) {
        novi[i+1] = 1;
        for (j = 1; j <= i; j++)
            novi[j] = stari[j-1] + stari[j];
        printf("%s", nc(' ', 2*(n-i)));
        for (j = 0; j <= i+1; j++) {
            printf ("%3d ", novi[j]);
            if (novi[j] < 0) {
                printf ("\n za i=%d i j=%d broj postane prevelik\n", i, j);
                exit (1); }
            stari[j] = novi[j]; }
        printf ("\n");
    }
}

```



```

void main (void) {
    int n, m, i, j;
    int broj;
    long k;
    int errorflag;
    float f[2][2];      // trajanje i broj iteracija
    long trajanje, freq;
    struct timeb vrijeme1, vrijeme2;

    while (1) {
        // citanje parametara
        printf ("Upisite broj obavljanja programa >");
        scanf ("%d",&broj); // npr: 1, 10000
        if (broj <= 0) {
            printf("Gotovo!\n");
            break;  }
        do {
            printf ("Upisite n, m >");
            scanf ("%d %d", &n, &m);
        } while ((n < m) || (n < 0) || (m < 0) ||
                ((m == 0) && (n == 0)));

        // inicijalizacija
        for (i = 0; i < 2; i++)
            for (j = 0; j < 2; j++)
                f[i][j] = 0;
    }
}

```

```

printf ("Program ce se ponoviti %d puta\n", broj);
    errorflag = 0;
    // koristenjem faktorijela
    freq = 0;
ftime (&vrijeme1);
for (i = 1; i <= broj; i++)
    k = BINOM (n, m, &freq, &errorflag);
ftime (&vrijeme2);
trajanje=1000*(vrijeme2.time - vrijeme1.time) + vrijeme2.millitm - vrijeme1.millitm;
f[0][0] += trajanje;
f[1][0] += freq;
printf (" BINOM : %d povrh %d = %ld %s\n",n, m, k, errorflag ? "(pogresno)" : "");
    // rekurzivno
    freq = 0;
ftime (&vrijeme1);
for (i = 1; i <= broj; i++)
    k = BINOMR (n,m,&freq);
ftime (&vrijeme2);
trajanje=1000*(vrijeme2.time - vrijeme1.time) + vrijeme2.millitm- vrijeme1.millitm;
f[0][1] += trajanje;
f[1][1] += freq;
printf ( " BINOMR: %d povrh %d = %ld\n", n, m, k);

```

```

// racun prosjecnih vremena i ispis rezultata
for (i = 0; i < 2; i++) {
    f[0][i] = f[0][i] / (float) broj;
    f[1][i] = f[1][i] / (float) broj;
}
printf ("\nProsjecno vrijeme za %d izvodjenja:\n BINOM: %f\nBINOMR: %f\n",
        broj, f[0][0], f[0][1]);
printf ("\nBroj iteracija:\n BINOM: %ld BINOMR: %ld\n",
        (long) f[1][0], (long) f[1][1]);
}

// Pascalov trokut
while (1) {
    printf ("Unesite broj redaka Pascalovog trokuta >");
    scanf ("%d", &n); // npr: 10
    if (n <= 0 || n >= MAXRED) break;
    Blaise (n);
}
system("PAUSE");
exit (0);
}

```

## Analiza algoritma – a priori i a posteriori

- Zadatak: izračunati mod sortiranog cjelobrojnog polja, tj. odrediti član polja koji se najčešće pojavljuje i prebrojati njegovu učestalost.
  - `mode0` izravno rješavanje
  - `rmode0` rekurzivni postupak
    - Zamislimo da je u polju od  $n$  članova  $a[0:n-1]$  izračunat mod  $m$  i učestalost  $f$  za prvih  $n-1$  članova polja. Pod kojim uvjetima zadnji član polja može promijeniti mod? Ako je  $a[n-1] \neq a[n-2]$  niti mod niti učestalost se ne mijenjaju. Ako jest jednak, kako razlikovati između 3 moguća slučaja:
      - a) nađen je novi mod
      - b) mod je isti, ali se povećava učestalost  $f$
      - c) nema promjene ni moda niti učestalosti
    - Odgovor ovisi o tome da li je  $a[n-1] == a[n-1-f]$ . Ako jest, onda ima  $n-1 - (n-1-f) + 1 = f + 1$  pojavljivanja vrijednosti koje je u  $a[n-1]$ . To znači da je ta vrijednost sigurno ili novi mod ili stari mod s uvećanom učestalošću  $f$ .
  - `rmode1` rekurzivni postupak transformiran u iterativni
  - sva tri postupka imaju vrijeme izvođenja  $\Theta(n)$ . Koji je najbolji?

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys\timeb.h>
#define MAXA 1000

// izravno pronalazi mod i ucestalost u a[n]
int mode0 (int a[], int n, int *f) {
    int mode, i, temp; // mod, trenutni, privremeni

    mode = a[0]; *f = 1; temp = 1;
    for (i = 1; i < n; i++) {
        if (a[i] != a[i - 1]) {
            temp = 1;
        } else {
            temp++;
            if (temp > *f) {
                *f = temp; mode = a[i];
            }
        }
    }
    return mode; // vrati mod, frekvencija se vraca kroz *f
}

```

```

// rekurzivno pronalazi mod i ucestalost u a[0:i]
int rmode0 (int a[], int i, int *f) {
    int mode;
    if (i == 0) {                // osnovni slucaj
        mode = a[0]; *f = 1;
    } else {
        mode = rmode0 (a, i - 1, f);    // rekurzivni mod svih prethodnika
        if (a[i] == a[i - *f]) {        // novi mod ili stari mod s uvecanom ucestaloscju
            mode = a[i]; (*f)++;    }
    }
    return mode;    // vrati mod
}

```

```

// rekurzivni postupak transformiran u iterativni
int rmode1(int a[], int n, int *f) {
    int mode, i;
    mode = a[0]; *f = 1;
    for (i = 1; i < n; i++) {    // povratak iz rekurzije udesno
        if (a[i] == a[i - *f]) {
            mode = a[i]; (*f)++;    }
    }
    return mode;    }

```

```

// U sortiranom polju a pronalazi se mod i ucestalost.
void main (void) {
    int a[MAXA], n, m; // polje, broj clanova, najveci clan
    int i, j, pom; // indeksi petlji, pomocna za sort
    int broj, freq, p; // broj ponavljanja, ucestalost, nadjeni mod
    struct timeb vrijeme1, vrijeme2; // poc. i zav. vrijeme
    long trajanje [3]; // vremena izvodjenja u ms

    do {
        printf ("Upisite broj clanova polja i maks. clan >");
        scanf ("%d %d", &n, &m);
    } while (n > MAXA);
    printf ("Upisite broj obavljanja programa >");
    scanf ("%d",&broj);
    printf("Izracuni ce se ponoviti %d puta\n", broj);
    srand ((unsigned) time (NULL));
    for (i = 0; i < n; i++) {
        a[i] = rand () % (m+1); }
    // sortiranje polja
    for (i = 0; i < n - 1; i++) {
        for (j = i; j < n; j++) {
            if (a[i] > a[j]) {
                pom = a[i]; a[i] = a[j]; a[j] = pom; } } }

```

```

for (i = 0; i < n; i++) printf ("%4d", a[i]);
    // izravno
ftime (&vrijeme1);
for (i = 1; i <= broj; i++)    p = mode0 (a, n, &freq);
ftime (&vrijeme2);
trajanje[0]=1000*(vrijeme2.time - vrijeme1.time) +vrijeme2.millitm - vrijeme1.millitm;
printf("\n mode0: Mod = %d, ucestalost = %3d\n",p,freq);
    // rekurzivno
ftime (&vrijeme1);
for (i = 1; i <= broj; i++)    p = rmode0 (a, n-1, &freq);
ftime (&vrijeme2);
trajanje[1]=1000*(vrijeme2.time - vrijeme1.time) + vrijeme2.millitm - vrijeme1.millitm;
printf("\n rmode0: Mod = %d, ucestalost = %3d\n",p,freq);
    // iterativna transformacija rekurzivnog
ftime (&vrijeme1);
for (i = 1; i <= broj; i++)    p = rmode1 (a, n, &freq);
ftime (&vrijeme2);
trajanje[2]=1000*(vrijeme2.time - vrijeme1.time) +vrijeme2.millitm - vrijeme1.millitm;
printf ("\n rmode1: Mod = %d, ucestalost = %3d\n", p, freq);
printf ("\nBroj milisekundi za %d izvodjenja:\n mode0: %d\nrmode0: %d\nrmode1:
    %d\n",broj, trajanje [0], trajanje [1], trajanje [2]);
system("PAUSE");
exit (0);
}

```