# Zbirka zadataka iz C-a
# ©2004 Željko Vrba

Ovo je zbirka raznoraznih zadataka iz C-a. Zadaci su različite težine i *nisu* poredani po težini, a grupirani su ugrubo po sadržaju. Jedini zadaci koji su riješeni su u odjeljku "egzotika". abstract



## 1 Matematički problemi

ZADATAK 1.1 Napisati funkcije koje računaju Taylorov red nekih funkcija prema sljedećim formulama:

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! + \dots$$
$$\cos x = 1 - x^2/2! + x^4/4! - x^6/6! + \dots$$

U glavnom programu učitati $x$ u stupnjevima i točnost $n$ (broj decimala; najviše 5) te ispisati:

- Vrijednost gornje dvije funkcije izračunatih prema formuli.
- Broj članova u izrazu potrebnih da se dostigne tražena točnost.
- Rezultate dobivene kada se računaju ugrađenim funkcijama `sin` i `cos` u `<math.h>`.

*Uputa:* sin i cos uzimaju svoj argument $x$ u *radijanima*. Formula za pretvorbu stupnjeva $d$ u radijane $r$ je $r = d\pi/180$.

Računanje pribrojnika prekidamo kada dostignemo traženu točnost. Neka je $y_k$ vrijednost Taylorovog reda izračunatog sa $k$ pribrojnika, te neka je $\epsilon = 10^{-n}$. Tražena točnost je dostignuta kada je $|y_k - y_{k-1}| < \epsilon$.

ZADATAK 1.2 Napisati funkciju koja će računati sumu prvih $n$ članova nekog reda na zadanu točnost $\epsilon$. Funkcija neka ima prototip:

```
double series(double (*smnd)(int k), double eps);
```

`smnd` je pointer na funkciju koja računa $k$-ti član reda, a `eps` je željena točnost. Napisati funkciju koja će Taylorovim redom izračunati $\sin(x)$ na 3 decimale točnosti. U glavnom programu učitati $x$ u stupnjevima te ispisati rezultat dobiven sumom reda i rezultat dobiven ugrađenom C funkciojm `sin`.

Za dodatna objašnjenja pogledati **zadatak 1.1**.

ZADATAK 1.3 Izračunati najveći zajednički djelitelj dva broja na sljedeće načine (za svaki način napisati posebnu funkciju):

- Uzastopnim oduzimanjem.
- Uzastopnim dijeljenjem.
- Rekurzivno.

U glavnom programu učitati brojnik i nazivnik razlomka te ispisati maksimalno skraćeni razlomak.

ZADATAK 1.4 Svaki prirodni broj $a$ se može napisati kao produkt prostih faktora u obliku $a = p_1^{n_1} \cdot p_2^{n_2} \cdot \ldots \cdot p_k^{n_k}$; npr. $72 = 2^3 \cdot 3^2$.

Napisati funkciju sa prototipom

```
int factorize(unsigned long a, unsigned long *f, unsigned long *p);
```

koja će u polje `f` spremiti proste faktore, a u polje `p` pripadajuće potencije ($p_k$ i $n_k$ u prethodnoj formuli). Funkcija treba vratiti broj različitih prostih faktora (a polja trebaju biti unaprijed alocirana).

U glavnom programu učitati broj $a$ te ispisati njegovu faktorizaciju. Primjer ispisa:

```
unesi broj: 72
72=2^3*3^2
```

*Uputa:* Ukoliko pretpostavimo 32-bitni tip `unsigned long`, tada polja `f` i `p` ne trebaju biti veća od 10 elemenata jer je produkt prvih 10 prostih brojeva (2, 3, 5, 7, 11, 13, 17, 19, 23, 29) veći od $2^{32}$. Znači, nijedan broj $\leq 2^{32}$ ne može imati više od 10 različitih prostih faktora.

ZADATAK 1.5 Izračunati najveći zajednički djelitelj (ponekad se govori i najveća zajednička mjera) NZM i najmanji zajednički višekratnik NZV dva prirodna broja $a$ i $b$ faktorizacijom. U glavnom programu učitati $a$ i $b$ te ispisati NZM, NZV i provjeru računa, tj. produkt za koji mora vrijediti $\text{nzm}(a,b) \cdot \text{nzv}(a,b) = ab$.

*Uputa:* Prirodne brojeve $a$ i $b$ napišimo kao produkt potencija prostih brojeva:

$$a = a_1^{p_1} \cdot a_2^{p_2} \cdot \ldots \cdot a_k^{p_k}$$
$$b = a_1^{q_1} \cdot a_2^{q_2} \cdot \ldots \cdot a_k^{q_k}$$

gdje su $a_i$ i $b_i$ prosti brojevi. Faktorizacija dva broja se uvijek može napisati tako da svaki broj ima *iste* proste faktore: ako se neki prosti faktor ne pojavljuje u nekom broju uzimamo ga s potencijom 0.

Zatim računamo produkt:

$$x = a_1^{r_1} \cdot a_2^{r_2} \cdot \ldots \cdot a_k^{r_k}$$

Uvrsti li se u prethodnu formulu $r_i = \min(p_i, q_i)$ dobit će se za $x$ NZM, a uvrsti li se $r_i = \max(p_i, q_i)$ dobit će se NZV.

U rješenju se smije koristiti funkciju `factorize` iz **zadatka 1.4**.

*Primjer:* Neka je $a = 6615 = 3^3 \cdot 5^1 \cdot 7^2$ te $b = 13200 = 2^4 \cdot 3^1 \cdot 5^2 \cdot 11$. Prilagodimo faktorizacije tako da oba broja imaju iste proste brojeve: $a = 2^0 \cdot 3^3 \cdot 5^1 \cdot 7^2 \cdot 11^0$ i $b = 2^4 \cdot 3^1 \cdot 5^2 \cdot 7^0 \cdot 11^1$. Tada je $\text{nzm}(6615, 13200) = 2^0 \cdot 3^1 \cdot 5^1 \cdot 7^0 \cdot 11^0 = 15$, $\text{nzv}(6615, 13200) = 2^4 \cdot 3^3 \cdot 5^2 \cdot 7^2 \cdot 11^1 = 5821200$. Također vrijedi $15 \cdot 5821200 = 6615 \cdot 13200 = 8731800$.

ZADATAK 1.6 Napisati funkciju koja će za prirodni broj $n$ uzastopnim oduzimanjem izračunati $\lfloor \sqrt{n} \rfloor$; npr. $\lfloor \sqrt{23} \rfloor = 4$. U glavnom programu učitavati cijele brojeve i ispisivati cjelobrojni kvadratni korijen. Program prekinuti kada se unese negativan broj.

*Uputa:* Algoritam se temelji na činjenici da je zbroj prvih $n$ uzastopnih neparnih brojeva jednak $n^2$. Npr. za $n = 8$ imamo $1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 = 64 = 8^2$.

ZADATAK 1.7 Napisati funkciju

```
double deriv(double (*f)(double), double x, double eps);
```

koja će izračunati derivaciju funckije $f$ u točki $x$ za zadanom točnoću $\epsilon$. U glavnom programu učitati broj decimala (ne više od 5) te ispisati derivaciju funkcije `sin` u točkama 0, $\pi/4 \approx 0.7854$, $\pi/2 \approx 1.5708$. Zadatak riješiti bez upotrebe funkcije `pow`.

*Uputa:* Derivacija funkcije (piše se $f'(x)$) definirana je kao:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

U programu treba računati kvocijente $y_k = (f(x+h_k) - f(x))/h_k$; u $k$-tom koraku neka je $h_k = 2^{-k}$. Za vezu broja decimala i točnosti $\epsilon$ pogledati **zadatak 1.1**.

ZADATAK 1.8 U glavnom programu učitati dva cijela broja te željeni broj decimala. Korištenjem algoritma pismenog dijeljenja, ispisati njihov kvocijent na zadani broj decimala.

ZADATAK 1.9 Napisati program koji će pronaći i ispisati sve proste brojeve $< n$ metodom *Eratostenovog sita*. $n$ neka se uzima sa komandne linije.

*Eratostenovo sito:* Napišu se svi brojevi od 2 do $n$. Zaokruži se 2 (to je prvi prosti broj) i prekriže se svi višekratnici od 2 (4, 6, 8, ...). Zatim se zaokruži od početka prvi idući neprecrtani broj (3) i prekriže se svi njegovi višekratnici (6, 9, 12, ...). Na kraju postupka (kada u listi nema više nezaokruženih i neprecrtanih brojeva) zaokruženi brojevi su prosti brojevi.

ZADATAK 1.10 Učitati realni broj $a$ i broj decimala $d$.

- Ispisati $a$ zaokružen na $d$ decimala.
- Ispisati $a$ zaokružen na najbliži cijeli broj.

ZADATAK 1.11 Napisati funkciju koja vraća $n$-ti član sljedećeg niza: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5... (niz je nepadajući i broj $k$ se pojavljuje točno $k$ puta).

ZADATAK 1.12 Zadan je niz poluotvorenih intervala $[x_1, y_1), [x_2, y_2), \ldots, [x_n, y_n)$. Niz je sortiran tako da vrijedi $x_1 \leq x_2 \leq \ldots \leq x_n$. Napisati funkciju koja će spajanjem intervala vratiti minimalan niz intervala koji pokrivaju iste brojeve kao i ulazni niz.

Dva intervala $[a, b)$ i $[c, d)$ se mogu spojiti ako vrijedi $a \leq c \leq b$ i tada je rezultantni interval $[a, \max(b, d))$.

ZADATAK 1.13 Zadan je sortirani niz zatvorenih intervala $[x_1, y_1], [x_2, y_2], \ldots, [x_n, y_n]$ tako da vrijedi $x_1 \leq x_2 \leq \ldots \leq x_n$. Napisati funkciju koja će izračunati najveći broj intervala koji se istovremeno preklapaju.

Dva intervala $[a, b]$ i $[c, d]$ se preklapaju ako vrijedi $a \leq c \leq b$.

# 2 Računalna aritemtika i bitovni operatori

ZADATAK 2.1 Napisati funkcije koje će provjeriti da li će zbroj, odn. razlika dva unsigned int broja $a$ i $b$ izaći izvan opsega (tzv. *overflow*).

U glavnom programu učitavati $a$ i $b$ dok se ne unese $a = b = 0$ te ispisivati zbroj i razliku, kao i da li je rezultat izašao izvan opsega.

ZADATAK 2.2 Napisati funkcije koje će provjeriti da li će zbroj, odn. razlika dva signed int broja $a$ i $b$ izaći izvan opsega (tzv. *overflow*).

U glavnom programu učitavati $a$ i $b$ dok se ne unese $a = b = 0$ te ispisivati zbroj i razliku, kao i da li je rezultat izašao izvan opsega.

ZADATAK 2.3 Pomoću shift i logičkih operatora napraviti funkcije za rotiranje ulijevo i udesno `unsigned` broja $x$ za $n$ mjesta. Funkcije neka imaju prototip:

```
unsigned long rotl(unsigned long x, int n);
unsigned long rotr(unsigned long x, int n);
```

*Uputa:* Primjer za bajt od 8 bitova označenih sa `abcdefgh`. Ovaj bajt rotiran ulijevo za $n = 2$ će imati redoslijed bitova `cdefghab`, a udesno za $n = 2$ će imati redoslijed bitova `ghabcdef`.

ZADATAK 2.4 Napisati funkciju

```
int has_1(unsigned int x);
```

koja provjerava da li broj $x$ ima susjednih jedinica u binarnom zapisu (npr. za binarni broj $10001010_2$ vraća 0, a za broj $10011001_2$ vraća 1). U glavnom programu učitati $n$ i ispisati sve brojeve $0 \leq x \leq n$ koji *nemaju* susjednih jedinica u binarnom zapisu.

*Uputa:* Broj $x$ nema susjednih jedinica ako je `(x >> 1) & x == 0`.

ZADATAK 2.5 Napisati funkciju

```
unsigned int bit1swap(unsigned int x);
```

koja zamjenjuje parove susjednih bitova u broju $x$ i vraća rezultat. Npr. za 8-bitni broj $01001110_2$ vraća $10001101_2$.

*Uputa:* Za slučaj 8-bitnih brojeva korisne su konstante $10101010_2 = AA_{16}$ i $01010101_2 = 55_{16}$. Funkcija mora raditi za 32-bitne brojeve.

ZADATAK 2.6 Napisati funkciju

```
int ispow2(unsigned int n);
```

koja provjerava da li je $n$ potencija broja 2. Ne smije se koristiti floating-point aritmetika.

*Uputa:* broj je potencija broja 2 ako je $n\&(n-1) = 0$.

ZADATAK 2.7 Generalizirati funkciju iz **zadatka 2.5**.

```
unsigned int bitNswap(unsigned int x, unsigned int n);
```

koja zamjenjuje parove susjednih $n$ bitova. $n$ mora biti potencija broja 2, a za slučaj $n = 1$ funkcija radi isto što i funkcija iz **zadatka 2.5**. Za $n = 0$ funkcija neka vraća $x$.

*Uputa:* Za 32-bitni ulaz potrebno je izračunati konstante za $n \in \{0, 1, 2, 4, 8, 16\}$

ZADATAK 2.8 Napisati funkciju

```
unsigned int bitrev(unsigned int x);
```

koja obrće redoslijed bitova u broju $x$ i vraća rezultat. Npr. za 8-bitni broj $10011110_2$ vraća $01111001_2$. Funkcija mora raditi za 32-bitne brojeve.

ZADATAK 2.9 Napisati funkciju

```
unsigned int bitperm(unsigned int x, unsigned char *p);
```

koja permutira bitove prema nizu $p$ u broju $x$ i vraća rezultat. Označimo sa $x_i$ $i$-ti bit ulaza, a sa $r_i$ $i$-ti bit rezultata. Tada je $r_i = x_{p[i]}$, $0 \leq i < 32$ (funkcija mora raditi za 32-bitne brojeve).

ZADATAK 2.10 Napisati funkcije

```
int bsf(unsigned int x);
int bsr(unsigned int x);
```

koje vraćaju poziciju prvog 1-bita brojeći od:

- bita najmanje težine (bit 0) za `bsf`
- bita najvece težine (bit 31) za `bsr`

Vratiti -1 ako je $x = 0$.

Na primjer, za 32-bitni `x == 0x74 == 1110100_2`, vrijedi `bsf(x) == 2` (pozicija najdesnije jedinice) i `bsr(x) == 6` (pozicija najljevije jedinice).

ZADATAK 2.11 Koristeći funkciju `bsr` iz **zadatka 2.10**, izračunati cjelobrojni logaritam pozitivnog broja $x$ po bazi 2 (funkcija `ilog2`). Vratiti -1 ako je $x = 0$.

Na primjer, `ilog2(65) == 6`.

ZADATAK 2.12 Napisati funkciju

```
unsigned int shuf16(unsigned short a, unsigned short b);
```

Brojevi $a$ i $b$ su 16-bitni i mogu se rastaviti u dvije komponente, svaka od 8 bitova. Tako se $a$ može zapisati kao $a = a_1|a_0$ ($a_1$ su bitovi 8-15, $a_0$ su bitovi 0-7) i $b = b_1|b_0$. Funkcija treba vratiti 32-bitni broj $x$ sastavljen na sljedeći način: $x = a_1|b_0|b_1|a_0$.

ZADATAK 2.13 Zadani su struktura i prototip funkcije:

```
struct pair {
        unsigned short a, b;
};

struct pair unshuf16(unsigned int x);
```

Implementirati funkciju `unshuf16` koja radi inverznu operaciju od one iz **zadatka 2.12**. $x$ je 32-bitni broj sastavljen od 4 8-bitne komponente: $x = x_3|x_2|x_1|x_0$ te treba vratiti strukturu u kojoj će biti $a = x_3|x_0$, $b = x_1|x_2$.

ZADATAK 2.14 Napisati funkcije

```
unsigned int compress_left(unsigned int x);
unsigned int compress_right(unsigned int x);
```

koje za ulazni 32-bitni broj $x$ "komprimiraju" sve jedinice lijevo ili desno.

Na primjer, za `x == 0x74 == 1110100_2`, vrijedi: `compress_left(x) == 0xF0000000`, `compress_right(x) == 0x0000000F`.

ZADATAK 2.15 Napisati funkciju

```
unsigned int mask_combine(unsigned int a, unsigned int b,
  unsigned int m);
```

Na mjestu bita gdje je vrijednost maske 0, u rezultatu treba biti odgovarajući bit iz $a$, a gdje je vrijednost maske $m$ 1, u rezultatu treba biti odgovarajući bit iz b.

Na primjer, za $a = 00110110_2$, $b = 11010011_2$, $m = 01100011_2$, rezultat je $z = 01010111_2$.

ZADATAK 2.16 Zadani su struktura i prototip funkcije:

```
struct run {
        unsigned int pos, len;
};

struct run bit_run1(unsigned int x);
```

Implementirati funkciju `bit_run1` koja će vratiti poziciju (u članu `pos`) i duljinu (u članu `len`) najdužeg niza uzastopnih jedinica u broju $x$.

ZADATAK 2.17 Napisati funkciju

```
struct run bit_run0(unsigned int x);
```

koja će vratiti poziciju i duljinu najdužeg niza uzastopnih nula u broju $x$. Obavezno koristiti funkciju `bit_run1` iz **zadatka 2.16**!

ZADATAK 2.18 Napisati funkciju

```
unsigned int bit_runs1(unsigned int x);
```

koja vraća broj podnizova uzastopnih jedinica (između kojih mora biti bar jedna nula) u broju $x$. Za $x = 0$ funkcija neka vrati 0. Npr. za $x = 1101110100_2$ funkcija vraća 3.

# 3 Pointeri i stringovi

ZADATAK 3.1 Napisati funkcije koje će u stringu:

- Pretvoriti sva mala slova a-z u velika.
- Pretvoriti sva velika slova A-Z u mala.
- Sva mala slova pretvoriti u velika i obratno.

U glavnom programu učitati string i ispisati ga nakon primjene svake funkcije.

ZADATAK 3.2 U ovoj vježbi ćemo za *riječ* smatrati niz koji se sastoji isključivo od znakova A-Z i a-z. Velika slova A-Z se smiju pojaviti samo na početku riječi (može biti više od jednog velikog slova na početku riječi). Napisati funkciju prototipa

```
char *isword(char *s);
```

koja treba vratiti pointer na prvi znak u nizu `s` koji ne čini riječ.

U glavnom programu učitati niz znakova i ispisati da li on čini riječ. Zadatak riješiti na dva načina: samostalno i upotrebom funkcije `strspn`.

ZADATAK 3.3 Napisati funkciju

```
char *spccmp(char *str);
```

koja će u stringu `str` zamijeniti više uzastopnih razmaka sa jednim. Funkcija vraća ulazni string (dakle, `str`) i ne smije alocirati dodatnu memoriju. U glavnom programu učitati string i ispisati ga nakon primjene funkcije.

ZADATAK 3.4 Napisati funkciju koja će zamijeniti sadržaj dva bloka memorije zadane duljine. Funkcija neka ima prototip:

```
void mswap(void *a, void *b, size_t len);
```

Predložiti i u glavnom programu implementirati provjeru rada funkcije.

ZADATAK 3.5 Napisati program koji učitava rečenicu te ispisuje pojedinačne riječi iz rečenice. Zadatak riješiti na dva načina:

- Upotrebom funkcije iz **zadatka 3.2**.
- Upotrebom funkcije `strtok`. Za separatore riječi uzeti razmak i TAB (`" \t"`).

ZADATAK 3.6 Napisati funkciju

```
int is_anagram(const char *w1, const char *w2)
```

koja će ispitati da li su riječi `w1` i `w2` anagrami. U glavnom programu učitati dvije riječi i ispitati da li je jedna anagram druge.

*Uputa:* dvije riječi su anagrami ako se permutiranjem slova jedne riječi može dobiti druga riječ (npr. riječi "anka" i "kana"). Riješite problem tako da prebrojite koliko svakog slova ima u svakoj riječi (npr. u riječi "radijator" ima dva "r", dva "a", te po jedno "d", "i", "j", "t" i "o"). Ukoliko su u obje riječi ista slova s jednakim brojem pojavljivanja, tada su riječi anagrami.

ZADATAK 3.7 Zadan je niz pozitivnih cijelih brojeva *neparne duljine* u kojemu se svaki broj, *osim jednoga*, pojavljuje točno dvaput. Napisati funkciju

```
unsigned int find_single(unsigned int *v, size_t len);
```

koja u nizu `v` duljine `len` traži broj koji se pojavljuje jedanput i vraća ga kao rezultat.

"Očito" (i najlakše) rješenje ima složenost $O(n^2)$. Moguća su algoritamski brža rješenja sa složenostima $O(n \log n)$ i $O(n)$. Pokušajte riješiti problem tako da ima što manju složenost.

*Uputa:* Brža rješenja uključuju korištenje sortiranja ili XOR bitovnog operatora.

# 4 Datoteke

ZADATAK 4.1 Sa komandne linije uzeti ime ulazne i izlazne datoteke. Iz ulazne datoteke učitavati neki tekst. Treba promijeniti sva slova a-z na početku rečenice u velika, te ispisati ispravke i koliko

ih ima. Ispravljeni tekst spremati u izlaznu datoteku. Smatra se da točka (`.`) završava rečenicu (ne uzimaju se u obzir kratice).

ZADATAK 4.2 Sa komandne linije uzeti ime dvije datoteke. Prva je *ulazna*, a druga je *rječnik* riječi (prema definiciji riječi u **zadatku 3.2**). Ispisati:

- Broj nizova znakova u ulaznoj datoteci koji se nalaze u rječniku.
- Broj nizova znakova u ulaznoj datoteci koji se *ne* nalaze u rječniku.
- Broj nizova znakova koji *nisu riječi*.

"Niz znakova" u ovoj vježbi čini svaki neprekinuti niz znakova koji u sebi *ne sadrži razmak*.

ZADATAK 4.3 Zadana je datoteka koja sadrži rječnik (jedna riječ u svakom redu). Slučajnim izborom riječi iz rječnika treba generirati datoteku koja sadrži unaprijed zadan broj riječi i redova. Riječi rasporediti u redove tako da svaki red ima podjednak broj riječi.

Svi parametri neka se uzimaju sa komandne linije i to sljedećim redoslijedom: ime rječnika, ime izlazne datoteke, broj riječi, broj redova.

ZADATAK 4.4 Napisati funkciju koja izračuna duljinu datoteke u bajtovima. Funkcija neka ima prototip:

```
long fsize(const char *fname);
```

U glavnom programu sa komandne linije uzeti ime datoteke te ispisati njenu duljinu.

ZADATAK 4.5 Napisati funkciju prototipa

```
long cmpfile(FILE *a, FILE *b);
```

koja uspoređuje (bajt po bajt) sadržaj dvije datoteke. Funkcija treba vratiti offset prvog bajta u kojemu se datoteke razlikuju. Ukoliko je jedna datoteka kraća od druge, a ne razlikuju se do zadnjeg znaka kraće datoteke, treba vratiti $l + 1$ gdje je $l$ duljina kraće datoteke. Funkcija neka vraća -1 ukoliko su datoteke identične.

# 5 Egzotika

## 5.1 Rekurzija 1

Sljedeći problem predložio je kolega prilikom smišljanja zadataka za vježbe, a koji će demonstrirati funkcije za dinamičku alokaciju memorije (`malloc`, `realloc` i `free`). Trebao je zadatak u kojem se *moraju* koristiti te funkcije:

ZADATAK 5.1 Učitavati brojeve (unaprijed nepoznat broj) i ispisati ih sortirane.

Međutim, nakon kraćeg razmišljanja zaključili smo da čak niti za ovaj problem ne treba koristiti funkcije za dinamičku alokaciju memorije. Tako je nastao sljedeći program:

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

```
struct node {
  int num;
  struct node *next;
};

void sort(struct node *first)
{
  struct node elem;

  if(scanf("%d", &elem.num) == 1) {
    struct node *x;

    elem.next = first;
    for(x = &elem; x->next; x = x->next) {
      if(x->num > x->next->num) {
        int t = x->num;
        x->num = x->next->num;
        x->next->num = t;
      }
    }

    sort(first = &elem);
  } else {
    printf("SORTED:\n");
    return;
  }
  printf("%d\n", first->num);
}

int main(void)
{
  printf("Enter numbers, press ^D to end\n");
  sort(NULL);

  return 0;
}
```

Ideja je rekurzivno učitavati brojeve, a već učitane brojeve držati u vezanoj listi na stacku. Kada se unese novi broj, on se umetne na ispravno mjesto u već postojećoj listi. U vezanoj listi na stacku ne mogu se mijenjati pointeri koji povezuju elemente liste jer se lista ispisuje obrnutim redoslijedom od stvaranja čvorova prilikom izlaska iz rekurzije (a NE praćenjem pointera). Zato se zamjenjuju elementi unutar liste.

Ograničenje na broj unijetih brojeva je veličina stacka na platformi na kojoj se program izvršava.

## 5.2 Svojstva floating-point aritmetike

ZADATAK 5.2 Koristeći arimetiku sa `float` brojevima izračunati vrijednost beskonačne sume

$$h = \sum_{k=1}^{\infty} \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \ldots$$

Ovaj zadatak je iz jedne Knuthove TAOCP knjige.

Matematički, ova suma divergira, tj. beskonačna je. Međutim, zbog konačne preciznosti `float` brojeva, u jednom trenutku se *vrijednost sume neće povećavati nakon dodavanja novih pribrojnika:* u tom trenutku treba prekinuti računanje.

```
#include <stdio.h>

/* funkcija usput izbroji i koliko pribrojnika je u sumi */
float hsum(unsigned long *k)
{
  float h0 = 0, h1 = 1;
  *k = 2;

  while (h0 != h1) {
    h0 = h1;
    h0 += 1.0f / *k;
    h1 = h0 + 1.0f / (*k + 1);
    *k += 2;
  }
  return h1;
}

int main(void)
{
  unsigned long k;
  float h = hsum(&k);

  printf("nakon %lu pribrojnika suma iznosi %f\n", k - 2, h);
  return 0;
}
```

## 5.3 "Transpozicija" tekstualne datoteke

```
omi
va
olt
 or
j e
ekc
 ri
pa
rcr
vie
i d
 da
rrk
eu
dgj
aie
k
  n
  a
  j
  d
  u
  z
  i
```

ZADATAK 5.3 Zadana je tekstualna datoteka u kojoj je svaki redak dugačak najviše 80 znakova. U novu datoteku treba ispisati "transponirani" tekst, tj. tako da se zamijene redovi i stupci. Imena ulazne i izlazne datoteke uzimaju se sa komandne linije.

Na primjer, za sljedeću ulaznu datoteku s 3 retka:

```
ovo je prvi redak
malo kraci drugi
i treci redak je najduzi
```

treba generirati datoteku sa tri stupca prikazana lijevo.

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

struct line {
  char *txt;
  int len;
};

void add_line(const char *txt, struct line **ll, int n)
{
  struct line *l = malloc(sizeof(struct line));
  l->len = strlen(txt);
  l->txt = malloc(l->len+1);
  strcpy(l->txt, txt);
  ll[n] = l;
}

int read_file(const char *name, struct line ***ll)
{
  FILE *f = fopen(name, "r");
```

```
  int allocd = 0, numlines = 0;
  char buf[82], *p;

  if(!f) return 0;

  *ll = 0;
  while(fgets(buf, 82, f)) {
    if(numlines >= allocd) {
      allocd = allocd*11/10+25;
      if(!(*ll = realloc(*ll, allocd*sizeof(struct line*)))) return 0;
    }
    if((p = strchr(buf, '\n'))) *p = 0;
    add_line(buf, *ll, numlines++);
  }

  return numlines;
}

int main(void)
{
  FILE *f;
  struct line **lines;
  int numlines = read_file("tekst.txt", &lines);
  int i, j, flag;

  if(!(f = fopen("transp.txt", "w")) || (numlines <= 0)) return 1;
  for(i = 0, flag = 1; flag; i++) {
    flag = 0;
    for(j = 0; j < numlines; j++) {
      if(i < lines[j]->len) {
        fputc(lines[j]->txt[i], f);
        flag = 1;
      } else {
        fputc(' ', f);
      }
    }
    fputc('\n', f);
  }
  fclose(f);

  return 0;
}
```