

1. S tipkovnice učitati 10 cijelih brojeva i zatim ih ispisati u obrnutom poretku u odnosu na poredak kojim su učitani, odvojene zarezom i prazninom. **Uputa:** učitavati brojeve i pohranjivati ih u polje, zatim ih ispisati pristupajući članovima polja od 10. člana prema prvom.

Primjer izvršavanja programa

```
Upisite 10 cijelih brojeva > 1 7 3 -11 15 4 2 2 1 -4↵
-4, 1, 2, 2, 4, 15, -11, 3, 7, 1
```

2. S tipkovnice učitati 10 cijelih brojeva u polje, poredak članova obrnuti **unutar polja**. Nakon što se poredak članova u polju promijeni, ispisati novi sadržaj polja, pri čemu vrijednosti trebaju biti razdvojene zarezom i prazninom. **Uočiti:** zadatak nije jednak prethodnom.

Primjer izvršavanja programa

```
Upisite 10 cijelih brojeva > 1 7 3 -11 15 4 2 2 1 -4↵
-4, 1, 2, 2, 4, 15, -11, 3, 7, 1
```

3. Učitati pozitivni cijeli broj. Nije potrebno provjeravati je li upisan broj ispravan. Uzastopnim dijeljenjem s 2 odrediti znamenke ekvivalentnog binarnog broja. Ispisati ekvivalentni binarni broj i to tako da se znamenke binarnog broja ispišu ispravnim redoslijedom. **Uputa:** znamenke dobivene uzastopnim dijeljenjem ubacivati u polje. Ispisati članove polja obrnutim redoslijedom. Broj znamenki ekvivalentnog binarnog broja sigurno neće biti veći od 31.

Primjeri izvršavanja programa

```
Upisite cijeli broj > 25↵
11001
```

```
Upisite cijeli broj > 1↵
1
```

```
Upisite cijeli broj > 3521↵
110111000001
```

4. Cjelobrojno polje je potrebno napuniti s prvih 40 članova Fibonaccijevog niza, a zatim ih ispisati. **Uputa:** u prva dva člana polja upisati prva dva člana niza, 1 i 1. Svaki sljedeći član izračunati na temelju prethodna dva. Uočiti da se prethodna dva člana uvijek lako dohvate iz polja, stoga nije potrebno koristiti varijable `a_i_minus2` i `a_i_minus1` kao u do sada viđenim primjerima izračunavanja članova Fibonaccijevog niza.

Primjer izvršavanja programa (prikazano je prvih 5 i zadnja 3 člana)

```
1↵
1↵
2↵
3↵
5↵
...
39088169↵
63245986↵
102334155↵
```

5. Cjelobrojno polje napuniti s prvih 500 prim brojeva. Koristiti činjenicu da je neki broj  $n$  prim broj ako nije djeljiv s niti jednim prim brojem manjim od  $n$ . Nakon punjenja polja ispisati sadržaj polja. **Uputa:** u polje kao prvi član ubaciti prvi prim broj, broj 2. Zatim redom za svaki neparni broj veći od 2 ("broj kandidat") ispitivati je li djeljiv s bilo kojim prim brojem koji se do tada već nalazi u polju prim brojeva (pri tome uvijek preskočiti prvi član, tj. 2). U vanjskoj petlji povećava se "broj kandidat" sve dok se ne popuni 500 prim brojeva, a unutarnjom petljom se testira djeljivost "broja-kandidata" sa svim prethodno upisanim prim brojevima osim 1. člana. Kada se za neki broj utvrdi da jest prim broj, dodaje ga se u polje. Na kraju ispisati sve prim brojeve, zajedno s rednim brojevima.

Primjer izvršavanja programa

```
1.....2.␣
2.....3.␣
3.....5.␣
4.....7.␣
...
498...3557.␣
499...3559.␣
500...3571.␣
```

6. S tipkovnice učitati cijeli broj  $n$  koji mora biti iz intervala  $[3, 20]$ . Učitavanje broja  $n$  ponavljati dok god ne bude ispravno upisan. Nakon toga s tipkovnice učitati  $n$  cijelih brojeva i pohraniti ih u polje ulaz. U polje parni redom prepisati sve parne vrijednosti, a u polje neparni sve neparne vrijednosti iz polja ulaz. Na kraju ispisati članove polja ulaz, parni i neparni, svako polje u svom retku, s članovima međusobno odvojenim prazninom. **Napomena:** sva polja definirati tako da se za njihovu pohranu troši minimalni potreban prostor - primijenjeno na priloženi primjer: polje ulaz treba biti definirano sa 7 članova, polje parni s 4 člana, a polje neparni s 3 člana.

Primjer izvršavanja programa

```
Upisite broj iz intervala [3, 20] > 2.␣
Upisite broj iz intervala [3, 20] > 22.␣
Upisite broj iz intervala [3, 20] > 7.␣
Upisite cijele brojeve (7) > -12 2 9 0 -15 2 15.␣
Ulaz: -12 2 9 0 -15 2 15 ␣
Parni: -12 2 0 2 ␣
Neparni: 9 -15 15
```

7. Učitavati cijele brojeve iz zatvorenog intervala [1, 20000] dok god se ne upiše broj izvan tog intervala. Prebrojati koliko je upisano brojeva iz intervala [1, 100], koliko iz intervala [101, 200], koliko iz intervala [201, 300], ..., koliko iz intervala [19901, 20000]. Pronaći i ispisati interval u kojem ima najviše upisanih brojeva. Ako ima više intervala u kojima ima jednak broj (najveći) upisanih brojeva, ispisati sve takve intervale.

Primjer izvršavanja programa

```
Upisite brojeve [1, 20000] > 1501 107 115 1521 55 11 19054 1545 199 0↵
[101, 200]: 3↵
[1501, 1600]: 3↵
```

8. S tipkovnice učitati 10 pozitivnih cijelih brojeva. Nije potrebno provjeravati jesu li ispravno upisani. Brojeve sortirati prema vrijednostima njihove posljednje znamenke, tako da se u poretku prvo pojavljuju brojevi s manjom posljednjom znamenkom. Ispisati poredane brojeve, međusobno odvojene prazninom.

Primjer izvršavanja programa

```
Upisite 10 pozitivnih brojeva > 56 12 98 10 12 710 1001 4 1 3↵
10 710 1001 1 12 12 3 4 56 98
```

9. S tipkovnice učitati cijeli broj n iz intervala [5, 20]. Nije potrebno provjeravati je li upisan ispravan broj. Nakon toga s tipkovnice učitati n cijelih brojeva i pohraniti ih u polje. Promijeniti sadržaj polja na sljedeći način: između svaka dva člana originalnog polja umetnuti dodatni član čija se vrijednost dobije zbrajanjem susjednih članova polja. Ispisati novi sadržaj polja. **Napomena:** polje definirati tako da se za njegovu pohranu troši minimalni potreban prostor - primijenjeno na priloženi primjer: polje treba biti definirano s 11 članova jer je korisnik odlučio upisati 6 brojeva, a dodatnih 5 će biti potrebno za "umetanje" zbrojeva susjednih članova. **Uputa:** nove vrijednosti polja postavljati od kraja polja prema početku; na papir nacrtajte stari i novi sadržaj polja jedan ispod drugog i uočite na koje pozicije treba seliti stare vrijednosti članova.

Primjer izvršavanja programa

```
Upisite cijeli broj [5, 20] > 6↵
Upisite cijele brojeve (6): 3 -2 15 11 2 7↵
3 1 -2 13 15 26 11 13 2 9 7
```

10. S tipkovnice učitati cijeli broj n iz intervala [5, 20]. Nije potrebno provjeravati je li upisan ispravan broj. Nakon toga s tipkovnice učitati n cijelih brojeva i pohraniti ih u polje. Iz polja izbaciti sve elemente koji su smješteni na parnim indeksima (dakle, članove s indeksima 0, 2, itd.). Ispisati novi sadržaj polja. **Napomena:** polje definirati tako da se za njegovu pohranu troši minimalni potreban prostor - primijenjeno na priloženi primjer: polje treba biti definirano sa 7 članova jer je korisnik odlučio upisati 7 brojeva.

Primjer izvršavanja programa

```
Upisite cijeli broj [5, 20] > 7↵
Upisite cijele brojeve (7): 3 -2 15 11 2 7 19↵
-2 11 7
```

11. S tipkovnice učitati prirodni broj  $n_{\max}$  iz intervala  $[1, 10\ 000]$ . Nije potrebno provjeravati je li upisan ispravan broj. Nakon toga ispisati sve brojeve iz intervala  $[1, n_{\max}]$  čiji pripadni čarobni nizovi imaju najveći *total stopping time*. **Napomena:** što su to čarobni niz i *total stopping time* čarobnog niza, opisano je u posljednjem zadatku 6. vježbi uz predavanja.

Primjeri izvršavanja programa.

Upisite broj iz intervala  $[1, 10000]$  > 5↵  
U intervalu  $[1, 5]$  najveći  $tst(=7)$  imaju nizovi za  $n = 3$

Upisite broj iz intervala  $[1, 10000]$  > 20↵  
U intervalu  $[1, 20]$  najveći  $tst(=20)$  imaju nizovi za  $n = 18\ 19$

Upisite broj iz intervala  $[1, 10000]$  > 10000↵  
U intervalu  $[1, 10000]$  najveći  $tst(=261)$  imaju nizovi za  $n = 6171$

Upisite broj iz intervala  $[1, 10000]$  > 1↵  
U intervalu  $[1, 1]$  najveći  $tst(=0)$  imaju nizovi za  $n = 1$

**Rješenja:**

1. #include &lt;stdio.h&gt;

#define MAXCLAN 10

```
int main(void) {
    int i;
    int broj[MAXCLAN];

    printf("Upisite %d cijelih brojeva > ", MAXCLAN);
    for (i = 0; i < MAXCLAN; i = i + 1) {
        scanf("%d", &broj[i]);
    }

    for (i = MAXCLAN - 1; i >= 0; i = i - 1) {
        if (i < MAXCLAN - 1) {
            printf(", ");
        }
        printf("%d", broj[i]);
    }

    return 0;
}
```

2. #include &lt;stdio.h&gt;

#define MAXCLAN 10

```
int main(void) {
    int i;
    int broj[MAXCLAN];

    printf("Upisite %d cijelih brojeva > ", MAXCLAN);
    for (i = 0; i < MAXCLAN; i = i + 1) {
        scanf("%d", &broj[i]);
    }

    int pomocna;
    for (i = 0; i < MAXCLAN / 2; i = i + 1) {
        pomocna = broj[i];
        broj[i] = broj[MAXCLAN - 1 - i];
        broj[MAXCLAN - 1 - i] = pomocna;
    }

    for (i = 0; i < MAXCLAN; i = i + 1) {
        if (i > 0) {
            printf(", ");
        }
        printf("%d", broj[i]);
    }

    return 0;
}
```

## 3. #include &lt;stdio.h&gt;

```
#define MAXZNAM 31

int main(void) {
    int i, brojZnamenki = 0;
    int dekadski;
    int binarni[MAXZNAM];

    printf("Upisite cijeli broj > ");
    scanf("%d", &dekadski);

    while (dekadski > 0) {
        binarni[brojZnamenki] = dekadski % 2;
        dekadski = dekadski / 2;
        brojZnamenki = brojZnamenki + 1;
    }

    for (i = brojZnamenki - 1; i >= 0; i = i - 1) {
        printf("%d", binarni[i]);
    }

    return 0;
}
```

## 4. #include &lt;stdio.h&gt;

```
#define MAXCLAN 40

int main(void) {
    int i;
    int fbroj[MAXCLAN];

    fbroj[0] = fbroj[1] = 1;
    for (i = 2; i < MAXCLAN; i = i + 1) {
        fbroj[i] = fbroj[i - 1] + fbroj[i - 2];
    }

    for (i = 0; i < MAXCLAN; i = i + 1) {
        printf("%d\n", fbroj[i]);
    }

    return 0;
}
```

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

#define MAXPRIM 500

int main(void) {
    int prim[MAXPRIM];
    int slobodanInd = 1, indPrim, kandidat = 3, jestPrim, i;
    prim[0] = 2;

    while (slobodanInd < MAXPRIM) {
        jestPrim = 1;
        indPrim = 1;
        while (indPrim < slobodanInd && jestPrim == 1) {
            if (kandidat % prim[indPrim] == 0) {
                jestPrim = 0;
            }
            indPrim = indPrim + 1;
        }
        if (jestPrim) {
            prim[slobodanInd] = kandidat;
            slobodanInd = slobodanInd + 1;
        }
        kandidat = kandidat + 2;
    }

    for (i = 0; i < MAXPRIM; i = i + 1) {
        printf("%d. %5d\n", i + 1, prim[i]);
    }

    return 0;
}
```

Varijabla `slobodanInd` ima vrijednost indeksa prvog slobodnog člana u polju `prim` brojeva. Varijabla `kandidat` sadrži neparni broj kojeg treba testirati i poprima vrijednosti 3, 5, 7, ...

Postupak uvećanja varijable `kandidat`, njenog testiranja i eventualnog ubacivanja u polje ponavlja se dok god se polje ne popuni s 500 prim brojeva, tj. dok je `slobodanInd < MAXPRIM`.

Testiranje broja kandidata se obavlja tako da se ispituje ostatak dijeljenja kandidata sa svakim prim brojem koji se već nalazi u polju (osim `prim[0]`). Kad se (ako se) naiđe na prim broj s kojim je kandidat djeljiv, varijabla `jestPrim` se postavlja na 0, a petlja se prekida. Ako je varijabla `jestPrim` tijekom testiranja ostala na vrijednosti 1, tada se broj kandidat ubacuje u polje kao novi prim broj.

Može li se algoritam unaprijediti tako da se uvjet `indPrim < slobodanInd && jestPrim == 1` dopuni: `indPrim < slobodanInd && jestPrim == 1 && prim[indPrim] <= sqrt(kandidat)`?

Izvršavanjem programa s različitim ulaznim podacima pokažite da ovako unaprijeđeni algoritam funkcionira, te da pri tome obavlja više od 10 puta manje testiranja broja kandidata. **Uputa:** prebrojite koliko puta će se tijelo unutarnje petlje obaviti u originalnom rješenju, a koliko puta u rješenju s dodatnim uvjetom.

6. #include <stdio.h>

```
#define MINCLAN 3
#define MAXCLAN 20
```

```
int main(void) {
    int i, brojUlaznih, brojParnih = 0, brojNeparnih = 0;

    do {
        printf("Upisite broj iz intervala [3, 20] > ");
        scanf("%d", &brojUlaznih);
    } while (brojUlaznih < MINCLAN || brojUlaznih > MAXCLAN);

    int ulaz[brojUlaznih];

    printf("Upisite cijele brojeve (%d) > ", brojUlaznih);
    for (i = 0; i < brojUlaznih; i = i + 1) {
        scanf("%d", &ulaz[i]);
    }

    // prebroji, da bi se mogle odrediti dimenzije polja
    for (i = 0; i < brojUlaznih; i = i + 1) {
        if (ulaz[i] % 2 == 0) {
            brojParnih = brojParnih + 1;
        } else {
            brojNeparnih = brojNeparnih + 1;
        }
    }

    int parni[brojParnih];
    int neparni[brojNeparnih];

    int indParnog = 0;
    int indNeparnog = 0;
    for (i = 0; i < brojUlaznih; i = i + 1) {
        if (ulaz[i] % 2 == 0) {
            parni[indParnog] = ulaz[i];
            indParnog = indParnog + 1;
        } else {
            neparni[indNeparnog] = ulaz[i];
            indNeparnog = indNeparnog + 1;
        }
    }

    printf("Ulaz: ");
    for (i = 0; i < brojUlaznih; i = i + 1) {
        printf("%d ", ulaz[i]);
    }

    printf("\nParni: ");
    for (i = 0; i < brojParnih; i = i + 1) {
        printf("%d ", parni[i]);
    }

    printf("\nNeparni: ");
    for (i = 0; i < brojNeparnih; i = i + 1) {
        printf("%d ", neparni[i]);
    }

    return 0;
}
```



7. #include <stdio.h>

```
#define VEL_INTERV 100
#define BROJ_INTERV 200
#define DONJA_GR 1

int main(void) {
    int brojac[BROJ_INTERV] = {0};
    int broj;

    printf("Upisite brojeve [%d, %d] > ", DONJA_GR, VEL_INTERV * BROJ_INTERV);
    do {
        scanf("%d", &broj);
        if (broj >= DONJA_GR && broj <= VEL_INTERV * BROJ_INTERV) {
            brojac[(broj - 1) / VEL_INTERV] = brojac[(broj - 1) / VEL_INTERV] + 1;
        }
    } while (broj >= DONJA_GR && broj <= VEL_INTERV * BROJ_INTERV);

    int i, najvecFrekvencija = brojac[0];
    for (i = 1; i < BROJ_INTERV; i = i + 1) {
        if (brojac[i] > najvecFrekvencija) {
            najvecFrekvencija = brojac[i];
        }
    }

    for (i = 0; i < BROJ_INTERV; i = i + 1) {
        if (brojac[i] == najvecFrekvencija) {
            printf("[%d, %d]: %d\n", i * VEL_INTERV + 1, (i + 1) * VEL_INTERV,
                brojac[i]);
        }
    }

    return 0;
}
```

8. #include <stdio.h>

#define MAXCLAN 10

```
int main(void) {
    int i;
    int broj[MAXCLAN];

    printf("Upisite %d cijelih brojeva > ", MAXCLAN);
    for (i = 0; i < MAXCLAN; i = i + 1) {
        scanf("%d", &broj[i]);
    }

    int j, ind_min;
    for (i = 0; i < MAXCLAN - 1; i = i + 1) {
        ind_min = i + 1;
        for (j = i + 2; j < MAXCLAN; j = j + 1) {
            if (broj[j] % 10 < broj[ind_min] % 10) {
                ind_min = j;
            }
        }

        if (broj[ind_min] % 10 < broj[i] % 10) {
            int pomocna = broj[i];
            broj[i] = broj[ind_min];
            broj[ind_min] = pomocna;
        }
    }

    for (i = 0; i < MAXCLAN; i = i + 1) {
        printf("%d ", broj[i]);
    }

    return 0;
}
```

## 9. #include &lt;stdio.h&gt;

```
int main(void) {
    int i, brojUlaznih;

    printf("Upisite cijeli broj [5, 20] > ");
    scanf("%d", &brojUlaznih);

    int novaVelicinaPolja = 2 * brojUlaznih - 1;
    int polje[novaVelicinaPolja];

    printf("Upisite cijele brojeve (%d) > ", brojUlaznih);
    for (i = 0; i < brojUlaznih; i = i + 1) {
        scanf("%d", &polje[i]);
    }

    for (i = brojUlaznih - 1; i > 0; i = i - 1) {
        polje[i * 2] = polje[i];
        polje[i * 2 - 1] = polje[i] + polje[i - 1];
    }

    for (i = 0; i < novaVelicinaPolja; i = i + 1) {
        printf("%d ", polje[i]);
    }

    return 0;
}
```

## 10. #include &lt;stdio.h&gt;

```
int main(void) {
    int i, brojUlaznih;

    printf("Upisite cijeli broj [5, 20] > ");
    scanf("%d", &brojUlaznih);

    int novaVelicinaPolja = brojUlaznih / 2;
    int polje[brojUlaznih];

    printf("Upisite cijele brojeve (%d) > ", brojUlaznih);
    for (i = 0; i < brojUlaznih; i = i + 1) {
        scanf("%d", &polje[i]);
    }

    for (i = 0; i < novaVelicinaPolja; i = i + 1) {
        polje[i] = polje[2 * (i + 1) - 1];
    }

    for (i = 0; i < novaVelicinaPolja; i = i + 1) {
        printf("%d ", polje[i]);
    }

    return 0;
}
```

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

```
int main(void) {
    int n_max, n, ai, i;
    int najvećiTst;

    printf("Upisite broj iz intervala [1, 10000] > ");
    scanf("%d", &n_max);

    // mora biti VLA
    int tst[n_max];

    for (n = 1; n <= n_max; n = n + 1) {
        // Za trenutacni n izracunaj total stopping time (tst),
        // na nacin da se za svaki sljedeci clan niza, brojac tst[n - 1] uveca za 1.
        // Uociti, zasto tst[n - 1], zasto ne tst[n]?
        i = 0;
        tst[n - 1] = 0;
        do {
            if (i == 0) {
                // izracunaj clan a0
                ai = n;
                // Kod brojanja clanova za tst, prvi clan se ne uzima u obzir.
                // Dakle, ovdje ne treba uvecati tst[n - 1];
            } else {
                // izracunaj sljedeci clan ai na temelju prethodnog clana ai
                if (ai % 2 == 0) {
                    ai = ai / 2;
                } else {
                    ai = ai * 3 + 1;
                }
                // uvecati brojac clanova niza za n (taj se nalazi na indeksu n-1)
                tst[n - 1] = tst[n - 1] + 1;
            }
            i = i + 1;
        } while (ai != 1);
    }

    // pronadji najveću vrijednost medju svim brojacima, tj. najveći tst
    najvećiTst = tst[0];
    for (n = 1; n < n_max; n = n + 1) {
        if (tst[n] > najvećiTst) {
            najvećiTst = tst[n];
        }
    }

    printf("U intervalu [1, %d] najveći tst(=%d) imaju nizovi za n = ",
           n_max, najvećiTst);

    // za sve brojace kojima je vrijednost jednaka najvećem tst, ispisi pripadni n
    for (n = 0; n < n_max; n = n + 1) {
        if (tst[n] == najvećiTst) {
            printf(" %d", n + 1);
        }
    }

    return 0;
}
```