Primeri kratkih programov v C-ju

- Program, ki bere znake s standardnega vhoda in jih izpisuje na standardni izhod.

```
#include <stdio.h>

main()
{
  int c;

  c=getchar();
  while( c != EOF ) {
    putchar(c);
    c=getchar();
  }
}
```

  Znak EOF na tipkovnici je Ctrl+d.

- Program, ki bere znake iz standardnega vhoda, jih izpisuje na standardni izhod, pri tem pa vse velike črke spremeni v male.

```
#include <stdio.h>

main()
{
  int c;

  c=getchar();
  while( c != EOF ) {
    if( (c>='A') && (c<='Z') )
      putchar(c - 'A' + 'a');
    else
      putchar(c);
    c=getchar();
  }
}
```

- Program, ki bere znake iz standardnega vhoda, jih izpisuje na standardni izhod, pri tem pa vse velike črke spremeni v male.
  Program naj na koncu izpiše, koliko prebranih črk je bilo velikih in koliko malih.

```
#include <stdio.h>

main()
{
  int c;
  int stm, stv;

  stm = stv = 0;
  while( (c=getchar()) != EOF ) {
    if( (c>='A') && (c<='Z') ) {
      putchar(c - 'A' + 'a');
      stv++;
    }
    else {
      if ( (c>='a') && (c<='z') )
        stm++;
```

```
      putchar(c);
    }
  }
  printf("Malih crk je bilo %d, velikih pa %d.\n", stm, stv);
}
```

- Program, ki bere dve celi števili in izpiše njuno vsoto.

```
main()
{
  int prvo, drugo, vsota;

  printf("Vpisite prvo stevilo:\n");
  scanf("%d",&prvo);
  printf("Vpisite drugo stevilo:\n");
  scanf("%d",&drugo);
  vsota = prvo + drugo;
  printf("%d+%d=%d\n", prvo, drugo, vsota);
}
```

- Program, ki bere dve celi števili (x in y) in izpiše x^y.

```
#include <math.h>

main()
{
  int prvo, drugo;
  double potenca;

  printf("Vpisite prvo stevilo:\n");
  scanf("%d",&prvo);
  printf("Vpisite drugo stevilo:\n");
  scanf("%d",&drugo);
  potenca = pow(prvo,drugo);
  printf("%d na potenco %d je %f\n", prvo, drugo, potenca);
}
```

Program prevajate z opcijo `-lm`, ki poveže tudi matematično knjižnico:

```
gcc -o primer -lm primer.c
```

- Program, ki na zaslon izpiše svoje ime in dobljene argumente, vsakega v novo vrstico.

```
main(argc,argv)
int argc;
char *argv[];
{
  int i;

  printf("ime:\n%s\n", argv[0]);
  printf("argumenti:\n");
  for(i=1; i<argc; i++)
    printf("%s\n", argv[i]);
}
```

Program lahko zapišemo tudi krajše:

```
main(argc,argv)
int argc;
char *argv[];
{
  printf(" ime:\n%s\n argumenti:\n", argv[0]);
```

```
  while(--argc > 0)
    printf("%s\n", *++argv);
}
```

- Program, ki na zaslon izpiše vsebino datoteke, ki je podana kot argument ukazne vrstice.

```
#include <stdio.h>

main(argc,argv)
int argc;
char *argv[];
{
  FILE *fp;
  int c;

  if( (fp=fopen(argv[1],"r")) == NULL )
    exit(1);
  while( (c=getc(fp)) != EOF )
    putc(c, stdout);
  fclose(fp);
  exit(0);
}
```

- Program, ki prekopira eno datoteko v drugo. Imeni obeh datotek sta podani kot argumenta ukazne vrstice.

```
#include <stdio.h>

main(argc,argv)
int argc;
char *argv[];
{
  FILE *fp1, *fp2;
  int c;

  if( (fp1=fopen(argv[1],"r")) == NULL )
    exit(1);
  if( (fp2=fopen(argv[2],"w")) == NULL )
    exit(1);
  printf("kopiranje datoteke %s v datoteko %s ... \n", argv[1],
argv[2]);
  while( (c=getc(fp1)) != EOF )
    putc(c, fp2);
  fclose(fp1);
  fclose(fp2);
  exit(0);
}
```

- Program, ki pregleda vhodne argumente in izpiše, katere od opcij (a, b ali c) smo uporabili.

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i;

  for (i=1; i<argc; i++) {
    if (*argv[i]=='-')
      switch (*(argv[i]+1)) {
        case 'a': fprintf(stderr,"opcija a\n");
                  break;
        case 'b': fprintf(stderr,"opcija b\n");
```

```
                    break;
            case 'c': fprintf(stderr,"opcija c\n");
                    break;
            default: fprintf(stderr,"Neznana opcija, KONEC!\n");
                    exit (-1);
          }
      }
      return (0);
}
```

Primer klica programa:
`parse -a -b -c` (opcije a, b in c)
`parse -c -b -a -b` (opcije c, b, a in b)
`parse -a b -c` (nepoznana opcija)

- Program, ki pregleda vhodne argumente in izpiše, katere od opcij (a, b ali c) smo uporabili. Opcije lahko pišemo tudi skupaj (npr. -abc ali -ac)

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i,j;

  for (i=1; i<argc; i++) {
    if (*argv[i]=='-') {
      for (j=1; j<strlen(argv[i]); j++)
        switch (*(argv[i]+j)) {
          case 'a': fprintf(stderr,"opcija a\n");
                    break;
          case 'b': fprintf(stderr,"opcija b\n");
                    break;
          case 'c': fprintf(stderr,"opcija c\n");
                    break;
          default: fprintf(stderr,"Neznana opcija, KONEC!\n");
                   exit (-1);
        }
    }
  }
  return (0);
}
```

Primeri klica programa:
`parse -abc` (opcije a, b in c)
`parse -c -ba` (opcije c, b in a)
`parse -adc` (nepoznana opcija)

- Program, ki pregleda vhodne argumente in izpiše uporabljene opcije (a, b ali c). Opciji `a` mora slediti realno število, opciji `b` pa niz znakov.

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

int main(int argc, char *argv[])
{
  int i;
  double f;
  char string[100];

  for (i=1; i<argc; i++) {
    if (*argv[i]=='-')
      switch (*(argv[i]+1)) {
        case 'a': if (++i>=argc) {
                    fprintf(stderr,"Premalo parametrov za opcijo
```

```
-a, KONEC!\n");
                        exit (-1);
                      }
                      f=atof(argv[i]); //pretvori niz znakov v
realno stevilo
                      fprintf(stderr,"Za opcijo a si vpisal
parameter %lf\n",f);
                      break;
            case 'b': if (++i>=argc) {
                        fprintf(stderr,"Premalo parametrov za opcijo
-b, KONEC!\n");
                        exit (-1);
                      }
                      strcpy(string,argv[i]);
                      fprintf(stderr,"Za opcijo b si vpisal
parameter %s\n",string);
                      break;
            case 'c': fprintf(stderr,"opcija c\n");
                      break;
            default: fprintf(stderr,"Neznana opcija, KONEC!\n");
                     exit (-1);
          }
      }
      return (0);
    }
```

Primer klica programa:
`parse -a 11.5 -b "niz znakov" -c`

- Program, ki demonstrira delo z enosmernimi neurejenimi linearnimi kazalčnimi seznami.

```
    #include <stdio.h>

    struct element
    {
      int vrednost;                /*vrednost elementa*/
      struct element *naslednji;   /*kazalec na naslednji element
seznama*/
    };

    // dodajanje novega elementa na konec seznama
    // prvi argument je kazalec na kazalec na prvi element seznama,
    // ker se kazalec na zacetek seznama v funkcuji spremeni,
    // drugi argument je vrednost elementa, ki ga dodajamo

    void dodaj(struct element **p, int e) {
      struct element *q, *r;

      q = (struct element*) malloc(sizeof(struct element)); //
rezerviramo prostor za nov element
      q->vrednost = e; // vpisemo vrednost novega elementa
      q->naslednji = NULL; // kazalec na naslednji element je pri
novem elementu prazen (NULL)
      if (*p == NULL)  // seznam, v katerega dodajamo, je prazen,
zato dodaj na zacetek
        *p = q;
      else {  // dodaj na konec
        r = *p;  // pomozni kazalec r postavimo na zacetek seznama
        while ( r->naslednji != NULL ) // s pomoznim kazalcem r se
sprehodimo do zadnjega elementa seznama
          r = r->naslednji;
        r->naslednji = q;  // zadnji element seznama povezemo z
novim elementom
      }
```

```
    }


    void brisi(struct element **p, int e) {
      struct element *q, *r;

      while( (*p != NULL) && ((*p)->vrednost == e) ) {  // brisemo
vse elemente z vrednostjo e, ki so na zacetku seznama
          q = *p;
          *p = (*p)->naslednji;
          free(q);
      }
      q = *p;
      if (q == NULL )
        return;
      while ( q->naslednji != NULL ) {  // pregledamo se preostanek
seznama
        if (q->naslednji->vrednost == e) {  // element ima vresnost
enako e
          r = q->naslednji;
          q->naslednji = q->naslednji->naslednji;  // prevezemo
          free(r);  // sprostimo pomnilnik
        }
        else
          q = q->naslednji;
      }
    }


    void izpisi(struct element *p) {
      struct element *q;

      q = p;
      while ( q != NULL ) {  // sprehodimo se po celotnem seznamu
        printf("%d  ", q->vrednost);  // izpisujemo vrednost
elementa
        q = q->naslednji;
      }
    }


    // glavni program, ki klice funkcije za delo s seznami,
    // tako preizkusimo delovanje napisanih funkcij
    main() {
      struct element *zacetek;
      int i;

      zacetek = NULL;  // inicializacija seznama; seznam je na
zacetku prazen
      for(i=1;i<11;i++){  // v zanki dodajamo elemente 1..10 in
sproti izpisujemo cel seznam
         dodaj(&zacetek,i);
         izpisi(zacetek);
         printf("\n\n");
      }
      dodaj(&zacetek,0);  // dodamo se element 0 (seznam ni urejen,
dodajamo na konec)
      izpisi(zacetek);
      printf("\n\n");
      brisi(&zacetek,5);  // brisemo element iz sredine seznama
      izpisi(zacetek);
      printf("\n\n");
      brisi(&zacetek,5);  // brisemo element, ki ga ni v seznamu -
seznam se ne spremeni
      izpisi(zacetek);
      printf("\n\n");
      brisi(&zacetek,1); // brisemo prvi element seznama
      izpisi(zacetek);
      printf("\n\n");
```

```
    brisi(&zacetek,0);  // brisemo zadnji element seznama
    izpisi(zacetek);
    printf("\n\n");
}
```

Datoteka seznam.c z izvorno kodo programa.
