C kompajleri

- GCC, the GNU Compiler Collection: http://gcc.gnu.org/
- GNU C biblioteka
- lcc-win32: A Compiler system for windows: http://www.cs.virginia.edu/~lcc-win32/
- Knjiga "Programming with lcc-win32", priručnik uz lcc-win32
- Knjiga "Advanced Programming with lcc-win32", priručnik uz lcc-win32
- Knjiga "C Programming" autor Steven Holmes: http://oopweb.com/CPP/Documents/CProgramming/VolumeFrames.html

Definiranje novih tipova podataka u C pomoću ključne riječi struct

- definiranje varijabli u C

```
int i;
float f;
char c;
int p[10];
```

- Tipove int, float i char zovemo osnovni ili *ugrađeni tipovi* (ugrađeni u svaki standardni C kompajler)

- Jezik C omogućuje, koristeći se osnovnim tipovima kao temeljnim gradivim jedinicama, definiranje složenijih tipove koji se koriste ravnopravno ugrađenim tipovima. To se radi pomoću ključne riječi struct:

```
struct complex {
        float Re;
        float Im;   };
```

- definiran novi, složeni tip podataka, čije je ime struct complex i koji je sastavljen od dva člana: realnog broja Re i realnog broja Im. Upotreba novodefiniranog tipa:

```
struct complex z1;
struct complex z2;
```

- definirane 2 varijable, z1 i z2, koje su tipa struct complex. Kraće pisanje: možemo se poslužiti ključnom riječi typedef

```
typedef struct complex Complex;
```

- Ključna riječ typedef služi za preimenovanje postojećih tipova. Upotreba
  typedef stari_tip novo_ime_za_stari_tip;

- pomoću naredbe typedef struct complex Complex; kompajler se upućuje da ime Complex od sada pa nadalje označava novo ime za stari tip struct complex. Deklariranje kompleksnog broja z:
  Ovako: struct complex z;
  Ili ovako: Complex z;

- deklariranje pokazivača na kompleksni tip: pokazivač cpok na tip Complex
  Complex * cpok;

- Varijable z1, z2 i z su nakon gornjih deklaracija spremne za uporabu:
  z1.Re = 1;
  z1.Re = 3;
  z2.Re = 2;
  z2.Im = 5;
  cpok = &z1;
  cpok->Re = 7;

- upotreba operatora . na mjestima na kojima pristupamo članovima Re i Im od kojih su sastavljene varijable z1 i z2 tipa Complex
- upotreba operatora -> pomoću kojeg pristupamo članovima Re i Im preko pokazivača cpok.
- kada imamo varijablu tipa Complex, njenim članovima pristupamo pomoću operatora točka (.). No, kada imamo *pokazivač* koji je tipa Complex, tada članovima pristupamo pomoću operatora ->.

- Pogledajmo sada potpuni primjer upotrebe novog tipa struct complex :

```
#include <stdio.h>
 struct complex {
float Re;
float Im;
};
typedef struct complex Complex;

Complex Add(Complex u, Complex v) {
 Complex rez;
 rez.Re = u.Re + v.Re;
 rez.Im = u.Im + v.Im;
 return rez;
}

int main() {
 Complex z1 = {1, 3};                    /* z1 = 1 + 3i */
 Complex z2;
 Complex r;
 Complex * rpok;
 z2.Re = 2;
 z2.Im = 5;                              /* z2 = 2 + 5i */
printf("Prvi kompleksni broj: %f + %fi\n",z1.Re, z1.Im);
printf("Drugi kompleksni broj: %f + %fi\n",z2.Re, z2.Im);
```

```
 r = Add(z1, z2);                                   /* r  = 3 + 8i */
  rpok = &r;                                        /* sada rpok pokazuje na r */
 printf("Rezultat zbrajanja: %f + %fi\n", r.Re, r.Im);

 rpok->Re = rpok->Re + 1;              /* indirektno mijenjamo r, preko pointera */
 rpok->Im = rpok->Im * 2;
 printf("Nakon promjene : %f + %fi\n", r.Re, r.Im);
return 0;
  }
```

Rješenje kvadratne jednadžbe: $ax^2+bx+c=0$

```
#include <stdio.h>
#include <math.h>
int main(){
 float a, b, c;
 float D;
 float x1, x2;
 printf("Unesi a: ");          scanf("%f", &a);
 printf("Unesi b: ");          scanf("%f", &b);
 printf("Unesi c: ");          scanf("%f", &c);
 D = b * b - 4 * a * c;
  if(D >= 0) {
        x1 = (-b - sqrt(D)) / (2 * a);
        x2 = (-b + sqrt(D)) / (2 * a);
        printf("x1 = %f\n", x1);
        printf("x2 = %f\n", x2);
  } else
        printf("Rjesenja su kompleksni brojevi!");
  return 0;
  }
```

- Test:

  a=1, b=5, c=4, rješenja: -1, -4

  a=5, b=6, c=1, rješenja: -1, -0.2

  a=1, b=1, c=-2, rješenja: 1, -2

Promjena ulazne varijable u funkciji

Funkcija ne mijenja svoju ulaznu varijablu

```
#include <stdio.h>

void func(int);

int main() {
int a = 1;
printf("Prije poziva funkcije func: a = %d\n", a);
func(a);
printf("Nakon poziva funkcije func: a = %d\n", a);
return 0;
}

void func(int a) {
a = a + 100;
}
```

Funkcija mijenja svoju ulaznu varijablu

```
#include <stdio.h>
#include <stdlib.h>
void func(int*);

int main(){
int *a;
a=malloc(sizeof(int));
*a = 1;
printf("Prije poziva funkcije func: a = %d\n", *a);
func(a);
printf("Nakon poziva funkcije func: a = %d\n",*a);
return 0;
}

void func(int *a) {
*a = *a + 100;
printf("Unutar funkcije func: a= %d\n",*a);
}
```

Funkcija mijenja elemente polja

```
#include <stdio.h>
void func(int[]);

 int main() {
 int i;
 int a[3] = {0, 1, 2};
 printf("Prije poziva funkcije func:\n");
 for(i = 0; i < 3; i++)
        printf("%d ", a[i]);
 printf("\n\n");
 func(a);
 printf("Nakon poziva funkcije func:\n");
 for(i = 0; i < 3; i++)
        printf("%d ", a[i]);
 printf("\n");
 return 0;
 }

 void func(int a[])   {
        a[0] = a[0] + 100;
  }
```

Primjer s pokazivačima

```
#include <stdio.h>
 int main() {
  int a,b,*p;

        a=10;
        b=5;
        p=&a;

        printf("a= %d\n",a);
        printf("a= %d\n",*p);
        a++;
        printf("a= %d\n",a);
        printf("a= %d\n",*p);
        *p=*p+b;
        printf("a= %d\n",a);
        printf("a= %d\n",*p);
        return 0;
  }
```