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

- Web stranica kolegija: http://lnrpc2.irb.hr/soya/nastava/index.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.Im = 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>
#include <stdlib.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);
system("PAUSE");
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>
#include <stdlib.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);
system("PAUSE");
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>
#include <stdlib.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");
 system("PAUSE");
  return 0;
  }
  void func(int a[])   {
          a[0] = a[0] + 100;
   }
```

Primjer s pokazivačima

```
#include <stdio.h>
#include <stdlib.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);
        system("PAUSE");
        return 0;
  }
```

Pogađanje brojeva

- Napišite program u kojem pogađate generiran slučajan broj. Upisivanjem broja dobivate obavijest da li je on veći ili manji od traženog broja.
  Funkcija rand() vraća slučajan broj između 0 i RAND_MAX. (Obično je RAND_MAX jednak 32767).
  Ako želimo dobiti slučajan prirodan broj između m i n (m < n) uključujući i granice m i n, tada treba napraviti ovo (vrijedi samo za prirodne brojeve !):
  - za lcc na MS Windows

  ```
  int x = m + (n - m + 1) * ((float)rand() / (RAND_MAX+1));
  ```

  - za gcc na Linux

  ```
  int x = m + (n - m + 1) * ((float)rand() / RAND_MAX);
  ```

- to su pseudoslučajni brojevi, nizovi upisanih brojeva
- potreban različit začetak za različite nizove slučajnih brojeva, što se radi s:
  `srand(time(NULL));`

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    int br, x, brpok;
    int pogodak = 0;
    srand(time(NULL)*10);
    x =  1 + 100 * ((float)rand() / RAND_MAX);
    printf("Zamislio sam broj izmedju 1 i 100. Pogodi ga!\n");
    brpok = 0;
    while(pogodak == 0)      {
        printf("Unesi broj: ");
        scanf("%d", &br);
        brpok++;
        if(br == x) {
            printf("Bravo! Pogodio si u pokusaju br. %d\n", brpok);
            pogodak = 1;
        } else     {
            if (x > br)
                printf("Trazeni broj je veci!\n");
            else
                printf("Trazeni broj je manji!\n!!");
        }          }
    system("PAUSE");
    return 0;
}
```

Empirijska provjera najčešćeg ishoda slučajnog pokusa bacanja dviju kocaka

- Lako se može izračunati (elementarna teorija vjerojatnosti) da je pri bacanju dviju kocaka najvjerojatnije da se kao zbroj točkica na kockama pojavi zbroj 7. Lako je vidjeti da vjerojatnost tog događaja iznosi p = 6/36 = 0.166666...
  Napisati program koji će i empirijski provjeriti ovaj teorijski rezultat. Program treba veliki broj puta (npr. milijun ili više puta) simulirati bacanje dviju kocaka, na prikladan način spremati ishode tih bacanja i na kraju prebrojiti i ispisati kojih ishoda je bilo najviše, te izračunati i ispisati relativnu frekvenciju tog (najčešćeg) ishoda.
  Dakle, moguće situacije su:

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define BR_PONAVLJANJA 1000000
int IndeksNajveceg(int a[], int length);
int main(){
	int i,inajv, frekv[13] = { 0 };
	float relfrekv;
	srand(time(NULL));  printf("Bacam kocke...\n");
	for(i = 0; i < BR_PONAVLJANJA; i++){
		int kocka1 = 1 + 6 * ( (float)rand() /RAND_MAX );
		int kocka2 = 1 + 6 * ( (float)rand() /RAND_MAX );
		frekv[kocka1 + kocka2]++;
	}
	inajv = IndeksNajveceg(frekv, 13);
	relfrekv = (float)frekv[inajv] / BR_PONAVLJANJA;
	printf("Najcesci zbroj pri bacanju dviju kocki je %d\n", inajv);
	printf("Rel. frekvencija pojave zbroja %d je %f\n", inajv, relfrekv);
	system("PAUSE"); return 0;
}
int IndeksNajveceg(int a[], int length){
	int i, indmax = 0;
	for(i = 1; i < length; i++)
		if (a[i] > a[indmax]) indmax = i;
	return indmax;
}
```

Računanje volumena kugle generiranjem slučajnih brojeva – Monte Carlo metoda

- Napisati program koji će upotrebom velikog broja generiranih trojki slučajnih brojeva izračunati volumen kugle radijusa r s centrom u središtu koordinatnog sustava (0,0,0). Usporediti dobiveni rezultat s izračunatom pravom vrijednošću volumena kugle.

- Postupak rješavanja: kuglu radijusa r smjestiti unutar kocke stranica duljine 2r. Provjeravati za svaku generiranu točku (x,y,z) koja se mora nalaziti unutar kocke da li se nalazi i unutar kugle (volumen kugle određen je izrazom $x^2+y^2+z^2 <= r^2$). Volumen kugle tada odgovara omjeru broja točaka unutar kugle prema ukupnom broju točaka pomnoženim s volumenom kocke.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define PI 3.1415926535897793
int main (){
     int N=0, n=3, i=0, brojilo=0, pogodak=0;
     float A[n], R=0, d=0, volumen, volf;
     srand(time(NULL));
     printf("Unesite radius kugle: R=");
       scanf("%f", &R);
     printf("Unesite zeljeni broj tocaka da odredite preciznost: N=");
       scanf("%d", &N);
     for(brojilo=0;brojilo<N;brojilo++){
            for(i=0;i<n;i++){
                    A[i]=-R+(2*R)*((float)rand()/(RAND_MAX+1));
            }
     d=sqrt(A[0]*A[0]+A[1]*A[1]+A[2]*A[2]);
     if(d<=R) pogodak++;
     }
     volumen=((float)pogodak/N)*(8*R*R*R);
     volf=4./3.*PI*R*R*R;
     printf("Volumen kugle je: %f, formula daje rezultat %f:\n", volumen, volf);
     system("PAUSE");
     return 0;
}
```

Maksimalna granična vrijednost: primjer s kompleksnim brojevima

- Međurezultati množenja u nekim izrazima mogu biti veći od maksimalne granične vrijednosti za neki tip podataka pa dolazi do prelijevanja (overflow) znamenki, iako je konačni rezultat unutar granica koje se mogu predstaviti tim tipom podataka
- primjer množenja kompleksnih brojeva: do prelijevanja će doći samo kada je i krajnji rezultat blizu maksimalne granične vrijednosti
- Dobar primjer je algoritam za određivanje modula kompleksnog broja dan izrazom:

$$|a + ib| = \sqrt{a^2 + b^2}$$

U rješavanju gornjeg izraza često će međurezultat (kvadrat) biti veći od graničnog.

Ispravan način na koji ovaj račun ostaje unutar granica je:

$$|a + ib| = \begin{cases} |a|\sqrt{1 + (b/a)^2} & |a| \geq |b| \\ |b|\sqrt{1 + (a/b)^2} & |a| < |b| \end{cases}$$

Isti problem se javlja i kod dijeljenja kompleksnih brojeva:

$$\frac{a+ib}{c+id} = \begin{cases} \dfrac{[a+b(d/c)]+i[b-a(d/c)]}{c+d(d/c)} & |c| \geq |d| \\ \dfrac{[a(c/d)+b]+i[b(c/d)-a]}{c(c/d)+d} & |c| < |d| \end{cases}$$

Ispravan postupak računanja je:

$$w \equiv \begin{cases} 0 & c = d = 0 \\ \sqrt{|c|}\sqrt{\dfrac{1+\sqrt{1+(d/c)^2}}{2}} & |c| \geq |d| \\ \sqrt{|d|}\sqrt{\dfrac{|c/d|+\sqrt{1+(c/d)^2}}{2}} & |c| < |d| \end{cases}$$

$$\sqrt{c+id} = \begin{cases} 0 & w = 0 \\ w + i\left(\dfrac{d}{2w}\right) & w \neq 0,\ c \geq 0 \\ \dfrac{|d|}{2w} + iw & w \neq 0,\ c < 0,\ d \geq 0 \\ \dfrac{|d|}{2w} - iw & w \neq 0,\ c < 0,\ d < 0 \end{cases}$$