4 Math и GSL

1. Да се прикаже собирање, одземање, множење и делење на комплексни броеви, како и наоѓање комплексно конјугиран број со користење на стандардната math библиотека.

```
#include <stdio.h>
#include <complex.h>
int main()
{
        float complex a = 1.0 + 3.0*I;
        float complex b = 1.0 - 4.0*I;
        printf("\nRabota so kompleksn broevi:\n");
        printf("\nPocetni vrednosti: a = %.2f%+.2fi\tb = %.2f%+.2fi", creal(a), cimag(a), creal(b),
cimag(b));
        float complex    sum = a+b;
        printf("\n\nSuma a + b = %.2f%+.2fi", creal(sum),cimag(sum));
        float complex    difference = a-b;
        printf("\n\nRazlika a - b = %.2f%+.2fi", creal(difference),cimag(difference));
        float complex product = a*b;
        printf("\n\nProizvod a * b = %.2f%+.2fi",creal(product),cimag(product));
        float complex quotient = a/b;
        printf("\n\nKolicnik a / b = %.2f%+.2fi", creal(quotient),cimag(quotient));
        float complex conjugate = conj(a);
        printf("\n\nKonjugirana vrednost od a = %.2f%+.2fi\n", creal(conjugate)
,cimag(conjugate));
        return 0;
}
```

Програмата може да се искомпајлира со наредбата:

gcc zadaca1.c

При компајлирање, библиотеката може да се вклучи во извршната верзија доколку се употреби статичко поврзување:

gcc -static zadaca1.c

2. Да се одреди проекцијата на комплексен број врз риманова сфера со користење на стандардната math библиотека.

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

int main()
{
        float complex a = 1.0 + 3.0*I;
        float complex argument = cargf(a);
        float complex proekcija = cprojf(a);
        printf("\nRabota so kompleksn broevi:\n");
printf("\nArgument od %.2f%+.2fi e ", creal(a), cimag(a),creal(argument), cimag(argument));

f (proekcija == INFINITY + I * copysign (0.0, cimag (a)))
printf("\nProekcijata na %.2f%+.2fi vrz Rimanova sfera e beskonecnost.\n",creal(a), cimag(a));

        else
printf("\nProekcijata na %.2f%+.2fi vrz Rimanova sfera e %.2f%+.2fi\n",  creal(a), cimag(a),
creal(proekcija), cimag(proekcija));
        return 0;
}
```

3. Да се пронајде вкупната импеданса на електрично коло доколку импедансите кои се содржат во колото се задаваат од командна линија. Да се предефинира имагинарната единица како J, за да се разликува од ознаката за струја (I).

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
#undef I
#define J _Complex_I

int main()
{
        float complex impedansa1 = 2.0 + 8.0*J, impedansa2 = 4.0 - 6.0*J;
        printf("\nRabota so kompleksn broevi:\n");
        printf("\nVkupnata impedansa na kolo koe gi sodrzi impedansite %.2f%+.2fj i %.2f%+.2fj
e %.2f%+.2fj", creal(impedansa1), cimag(impedansa1), creal(impedansa2), cimag(impedansa2),
creal(impedansa1+impedansa2), cimag(impedansa1+impedansa2));

return 0;
}
```

4. Да се напише функција која за коло со зададени напон и импеданса ја враќа струјата.

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
#undef I
#define J _Complex_I
// E = I * Z
int main()
{
        float complex napon = 45.0 + 10.0*J, impedansa = 3.0 + 4.0*J;
        float complex struja = napon/impedansa;
        printf("\nRabota so kompleksn broevi:\n");
        printf("\nStrujata sto tece niz kolo so napon %.2f%+.2fj volti i impedansa %.2f%+.2fj omi
e %.2f%+.2fj amperi", creal(napon), cimag(napon), creal(impedansa), cimag(impedansa),
creal(struja), cimag(struja));
        return 0;
}
```

5. Да се напише функција која како аргументи добив низа од комплексни броеви Xk , должина на низата N и реден број n. Функцијата треба да го пресмета и врати излезниот член Xn пресметан според следнава формула:

$$X_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\left(\frac{2\pi kn}{N}\right)}$$

Потоа да се напише главна програма во која ќе се дозволи внесување на N комплексни броеви од тастатура и ќе ги најде и отпечати комплексните броеви Xn, Xn-1 и Xn+1. Потоа, главната програма треба да провери дали $\frac{X_{n+1}+X_{n-1}}{2}$ е ист со Xn.

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
float complex f(float complex *Xk, int N, int n){
float complex Xn=0.0 + 0.0*I;
double eks;
float complex pom;
float complex pom2 = N + 0.0*I;
int k;
for (k=0;k<N;k++){
eks = exp((2*M_PI*k*n)/N);
pom = eks + 0.0*I;
Xn = Xn + (Xk[k]*pom);
}
Xn = Xn/pom2;
return Xn;
}
int main(){
float img,re;
int i,n,N;
float complex niza[100];
float complex Xn,Xn1,Xn2,pom,pom2;
printf("Vnesete go parametarot N\n");
```

```
scanf("%d",&N);
printf("Vnesete go redniot broj na clenot...n\n");
scanf("%d",&n);
for(i=0;i<N;i++){
scanf("%f%f",&re,&img);
niza[i]=re+img*I;
}
for(i=0;i<N;i++){
printf("%.2f%+.2f\n",creal(niza[i]),cimag(niza[i]));
}
Xn=f(niza,N,n);
Xn1=f(niza,N,n+1);
Xn2=f(niza,N,n-1);
pom2=2.0+0.0*I;
pom = (Xn1+Xn2)/pom2;
if(Xn==pom) printf("Isti se\n");
printf("Xn=%.2f%+.2f\n",creal(Xn),cimag(Xn));
printf("Xn1=%.2f%+.2f\n",creal(Xn1),cimag(Xn1));
printf("Xn2=%.2f%+.2f\n",creal(Xn2),cimag(Xn2));
printf("Pom=%.2f%+.2f\n",creal(pom),cimag(pom));
return 0;
}
```

6. Со користење на GSL библиотеката да се напише програма за решавање на квадратна равенка.

```
#include <stdio.h>
#include <gsl/gsl_poly.h>
#include <gsl/gsl_complex.h>
// a x^2 + b x + c = 0
int main() {
double a,b,c,x0,x1;
gsl_complex cx0, cx1;
int brKoreni; //1 ili 2
scanf("%lf %lf %lf", &a, &b, &c);
brKoreni = gsl_poly_solve_quadratic (a, b, c, &x0, &x1);
if ( brKoreni == 1)
printf("Ravenstvoto ima 1 koren:");
else
printf("Ravenstvoto ima 2 korenja:");
//printf("%.2f%+.2fi\n", GSL_REAL(x0), GSL_IMAG(x0));

printf("%f \n",x0);

if ( brKoreni == 2)
printf("%f \n",x1);
//printf(" %.2f%+.2fi\n", GSL_REAL(x1), GSL_IMAG(x1));

gsl_poly_complex_solve_quadratic(a,b,c, &cx0, &cx1);
printf("1,%f%+f\n",GSL_REAL(cx0),GSL_IMAG(cx0));
printf("2,%f%+f\n",GSL_REAL(cx1),GSL_IMAG(cx1));
return 0;
}
```

7. Со користење на библиотеката GSL да се изберат 10 реални броеви по случаен избор во опсегот [0, 1]. Како може да се искомпајлира програмата (со статичко и динамичко поврзување на библиотеката gsl) ?

   Функцијата **gsl_rng_uniform** генерира реални броеви од опсегот [0, 1].

```
#include <stdio.h>
#include <gsl/gsl_rng.h>
int main (void)
{
        gsl_rng * r;
        int i, n = 10;
        gsl_rng_env_setup();  // vospostavuvanje na okolinata za rng
        r = gsl_rng_alloc (gsl_rng_default);  // alociranje predefiniran generator
        for (i = 0; i < n; i++)
         {
           double u = gsl_rng_uniform (r);  // uniformno generiranje na slucajni broevi
           printf ("%.5f\n", u);
         }

        gsl_rng_free (r);
        return 0;
}
```

   Програмата може да се искомпајлира со динамичко поврзување со gsl библиотеката со наредбата:

   gcc matrici.c -lgsl -lgslcblas -lm

   Програмата може да се искомпајлира со статичко поврзување со gsl библиотеката со наредбата:

   gcc -static matrici.c -lgsl -lgslcblas -lm

8. Со користење на функцијата за генерирање псевдо-случајни броеви да се напише програма во која ќе се извлекуваат по 5 карти по случаен избор од шпил без џокери. Потоа картите треба да се испечатат на екран. Да се обезбеди начин, програмата со секое извршување да влече различни карти. Зошто ова не е стандарден случај?

   РЕШЕНИЕ

```
#include <stdio.h>
#include <gsl/gsl_rng.h>
#include <time.h>

int main (void)
{
        gsl_rng * r;
        int i, n = 5;

        gsl_rng_env_setup();
```

```
        r = gsl_rng_alloc(gsl_rng_default);
        gsl_rng_set (r, time(0));

        for (i = 0; i < n; i++)
        {
                double u = gsl_rng_uniform (r);
                int izbor = ((int)(u*100))%13;
                char *znak, *broj ;
                switch (((int)(u*100))%4)
                {
                        case 0: znak = "list\0"; break;
                        case 1: znak = "detelina\0"; break;
                        case 2: znak = "srce\0"; break;
                        case 3 : znak = "baklava\0"; break;
                }


                if (izbor < 10)
                {
                        broj = (char*)malloc(3);
                        sprintf(broj, "%d", izbor+1);
                }
                else
                        switch(izbor)
                        {
                                case 10: broj = "dzhandar\0"; break;
                                case 11: broj = "dama\0"; break;
                                default: broj = "kral\0"; break;
                        }

                printf ("%s %s\n", broj, znak);
        }
        gsl_rng_free (r);
        return 0;
}
```

Функциите за генерирање псевдо-случајни броеви секогаш започнуваат со користење на иста почетна вредност (семе) и при секое извршување на програмата даваат иста секвенца од „случајно“ избрани броеви.

Единствен начин да се обезбеди уникатност во изборот на случајни броеви е при секое извршување да се задава различна (уникатна) вредност за семето. Вредноста треба да биде long int број.

Еден начин да се земе уникатна вредност се тековниот датум и време. За таа цел се користи наредбата time(0) која враќа број на секунди изминати од 01.01.1970 сè до моментот на нејзиното извршување.

Кои сè начини може да се искористат за добивање уникатна вредност?

Дали добар начин за постигнување случајност е да се побара од корисникот да внесе случаен текст и тој текст се искористи за добивање почетна вредност?