Računske vježbe 6

Programiranje I

1. Napisati program koji simulira bacanje kocke i koji provjerava da li je kocka fer. Kocka je fer ukoliko su svi ishodi bacanja kocke (1, 2 ..., 6) približno vjerovatni.

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

int main()
{
    int count[6] = {0}; // inicijalizujemo sve clanove niza na 0
    int i, dice, repeat = 10000;

    srand(time(NULL));

    for (i = 0; i < repeat; i++)
    {
        dice = throw_dice();
        count[dice - 1]++; //throw_dice() daje 1-6, a indeksi su 0-5
    }

    printf("Vjerovatnoce su: ");
    for (i = 0; i < 6; i++)
    {
        printf("%lf ", count[i] / (double)repeat);
    }
    printf("\nOcekivana vjerovatnoca je: %lf", 1.0 / 6);
}

int throw_dice()
{
    return 1 + rand() % 6; // sa 1 opseg 0-5 pomjeramo kako bismo dobili opseg 1-6
}
```

Mada bi nam `Code::Blocks` IDE (integrisano razvojno okruženje (engl. *integrated development environment*)) sve do sada dodavao zaglavlje `stdlib.h` kada napravimo novi projekat mi smo ga brisali jer nam nije bilo potrebno. Ovo je zaglavlje za **St**andar**d Lib**rary biblioteku koja nam pruža razne korisne funkcije i sa nekim od njih ćemo se upoznati u toku kursa. Funkcija `rand()` koju nam pruža standardna biblioteka predstavlja pseudo-slučajni generator brojeva. Sekvenca slučajnih brojeva generisana ovim generatorom nije stvarno slučajna jer je u potpunosti određena u zavisnosti od sjemena (engl. **seed**) koje predstavlja inicijalnu vrijednost. Dakle, ako znamo sjeme nečijeg generatora možemo ponoviti istu sekvencu. Da bi se što više približili nasumičnim brojevima obično kao sjeme koristimo tekuće vrijeme u sekundama. Funkcija `time()` računa razliku od Unix epohe (referentna tačka u računarskom vremenu: 00:00:00 UTC on 1 January 1970) do datog trenutka u sekundama. Kao argument uzima pokazivač u koji smješta rezultat mada ga i vraća. Mi joj prosljeđujemo `NULL` pokazivač jer nam rezultat dalje neće biti potreban. Funkcija `srand()` „sije" generator nasumičnih brojeva. Opseg funkcije `rand()` možemo kontrolisati pomoću operatora `%`:

```c
broj = rand() % duzina;
```

jer znamo da ostatak pri dijeljenju može biti u intervalu od `0` do `duzina-1`.

2. Napisati program koji miješa niz `X`. U tu svrhu, generisati nasumičnu permutaciju indeksa niza `X` (npr. od redosljeda indeksa 0, 1, ..., N-1, pravi redosljed 5, 9, 0, ..., 4), pa presložiti elemente u skladu sa novim redosljedom indeksa.

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

void random_indices(int *, int);
void shuffle(int *, int *, int);

int main()
{
    int indices[10] = {0};
    int x[10] = {11, 22, 23, 38, 49, 55, 101, 103, 177, 202};
    // Pokrenite program bez ove naredbe nekoliko puta. Sta uocavate?
    srand(time(NULL));

    puts("Niz je: ");
    for (int i = 0; i < 10; i++)
        printf("%d ", x[i]);

    random_indices(indices, 10);
    puts("\nIndeksi su: ");
    for (int i = 0; i < 10; i++)
        printf("%d ", indices[i]);

    shuffle(x, indices, 10);
    puts("\nPreslozeni niz je: ");
    for (int i = 0; i < 10; i++)
        printf("%d ", x[i]);
}

void random_indices(int *x, int length)
{
    int i, t, index;

    for(i = 0; i < length; i++)
        x[i] = i; // redom upisemo sve moguce vrijednosti indeksa

    // vrsimo zamjenu elemenata na tekucoj poziciji i nasumicnoj poziciji index.
    for(i = 0; i < length; i++)
    {
        index = rand() % length; // generisemo nasumican indeks
        t = x[i];
        x[i] = x[index];
        x[index] = t;
    }
}

void shuffle(int *arr, int *indices, int length)
{
    int i;
    int temp[length]; // pomocni niz
    for (i = 0; i < length; i++)
        temp[i] = arr[indices[i]]; // vrsimo upis po novom redosljedu
    for (i = 0; i < length; i++)
        arr[i] = temp[i]; // slozimo nazad u odredisni niz
}
```