FONKSİYONLAR 2. BÖLÜM

MATH.H KÜTÜPHANESİ

NOT:BU KÜTÜPHANEDEKİ BÜTÜN FONKSİYONLAR DOUBLE DEĞER ALIR DOUBLE DEĞER DÖNDÜRÜR.

sqrt pow fabs fonksiyonlarını 1. bölümde incelemiştik.

exp(x) e üzeri x (e ^ x) değerini hesaplar.

exp(1) = 2.718282

Bu fonksiyonlar double alır dememize rağmen int girdik ama problem yok. double int'ten kapsamlı olduğu için 1 rahatça 1.000000 ya dönüşür. Fonksiyonlar içine aldıkları değerin tipini uygun hale çevirir. Bu olaya coercion of arguements denir.

exp2(x) 2 ^ x değerini hesaplar

exp2(1) = 2.000000

log(x) doğal logaritma hesaplar log e tabanında x yani lnx

log10(x) log 10 tabanında x değerini hesaplar

NOT:başka bir tabanda logaritme bulmak için taban değiştirme kullanıyoruz

```
double yeni_log(double taban, double ic){

    double sonuc = log(ic) / log(taban);

    return sonuc;
}
```

NOT:bir sayının basamak sayısını belirlemek için döngü kullanmıştık.
(int) log10(sayi) + 1 ifadesi de basamak sayısını hesaplar.

```
int basamak_sayisi(int sayi){

    int sonuc = log10(sayi) + 1;

    return sonuc;
}
```

fmod(x, y) x in y ile bölümünden kalanı hesaplar % operatöründen farkı: fmod double argüman alabilir ama % operatörü int değerler ile kullanılır.

sin(x) cos(x) tan(x) ne iş yaptıklarını biliyorsunuz şunu bilin x değerleri RADYAN cinsinden

floor(x) içine aldığı sayıyı, sayıdan küçük en küçük int değerine yuvarlar.
Bir ondalık sayıyı en yakın tamsayıya yuvarlamak için

floor(x + 0.5);

onda birler basamğına yuvarlamak için

floor(x * 10 + .5) / 10;

rand - RASTGELE SAYI ÜRETMEK

Bir zar attığımızda üst yüzüne gelecek rakamı önceden kestiremeyiz çünkü üste rastgele bir sayı gelir. rastgele sayı yaratmak için rand fonksiyonunu kullanırız. Rand fonksiyonu 0 ile RAND_MAX isimli önceden tanımlanmış bir sayı arasında int sayılar üretir

zar örneğini düşünelim

rand fonksiyonu stdlib.h kütüphanesindedir bu yüzden stdlib.h kütüphanesini include ederiz.

ek > C ders2.c > main()

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

int main(){

    int ust = rand();


    return 0;
}
```

rand kütüphanesi bize rastgele bir sayı oluşturur. bu sayı 0, 89, 1123 gibi herhangi bir olabilir fakat bir zarın 6 yüzü vardır ve elimizde 6 tane seçenek olmasını isteriz bu yüzden oluşan rastgele sayının 6 ile bölümünden kalanını alırız.

```c
int ust = rand() % 6;
```

bu ifadenin sonucunda ortaya çıkacak sayılar
0 1 2 3 4 5 olacaktır . ama biz 1 2 3 4 5 6 dizisini istiyoruz. Bu yüzden elimizdeki diziyi 1 sağa kaydırmalıyız

```c
int ust = rand() % 6 + 1;
```

Şimdi bu sayıyı bastırabiliriz.

```
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ gcc ders2.c -o ders2
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ ./ders2
2
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ ./ders2
2
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ ./ders2
2
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ ./ders2
2
mehmet@mehmet-R580-R590:~/Masaüstü/EGITIM/bolum4/ek$ ./ders2
2
```

5 defa çalıştırdım ama hep aynı sayı geldi. demek ki aslında o kadar rastgele değilmiş. Bu durum kod test ederken çok işe yarayabilir yine de biz kod her çalıştığında farklı bir rakam görmek istiyoruz.

bunun için "program farklı zamanlarda çalıştığında farklı sayılar üret" anlamına gelen

srand(time(NULL)); komutunu yazarız.

time() fonksiyonunu kullanmak için time.h kütüphanesini ekleriz.

şimdi full programı görelim

fonksiyonek > C ders2.c > ...

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

int main(){

    srand(time(NULL));

    int ust = rand() % 6 + 1;
    printf("%d\n",ust);

    return 0;
}
```

konuyu kavramak için 2 örnek yapalım
önce 100 ile 200 arasında bir rastgele sayı seçelim(100 ve 200 dahil)

200 ile 100 arasında kaç tane olasılık var

(son - ilk + 1) 101 tane seçenek var

rand() % 101 dediğimizde oluşan sayılar 0 dan başlayacak 100 den başlamak için 100 ekleyeceğiz

rand() % 101 + 100;

---

b >a olmak üzere b ile a arasında bir tamsayı seçelim

b ile a arasında(b ve a dahil) b - a +1 olasılık var

rand % (b -a +1) dediğimizde sayılar 0 dan başlar a den başlaması için a eklemeliyiz

rand() % (b - a +1) + a

ÖRNEK:
Craps oyunu 2 zar ile oynanan bir oyundur.
Atılan zarların toplamı ilk atışta 11 veya 7 ise oyuncu kazanır
Atılan zarların toplamı ilk atışta 2 3 veya 12 ise kasa kazanır oyuncu kaybeder
Atılan zarların toplamı ilk atışta 4 5 6 8 9 10 ise bu toplam oyunucnun puanıdır.
oyuncu bu sayılardan birini yaparsa kazanır, 7 atarsa kaybeder. oyuncu bu sayıları yapana kadar zar atmaya devam eder. Craps oyununun kodunu yazın.

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

enum Oyun {KAZANC, KAYIP, DEVAM};

int zar_at();

int main(){

    srand(time(NULL));

    enum Oyun durum;
    int puan;

    int toplam = zar_at();

    switch (toplam)
    {
    case 7:
    case 11:
        durum = KAZANC;
        break;

    case 2:
    case 3:
    case 12:
        durum = KAYIP;
        break;

    default:
        durum = DEVAM;
        puan = toplam;
        printf("puan: %d\n",puan);
        break;
    }

    while(durum == DEVAM){
        toplam = zar_at();

        if(toplam == puan){
            durum = KAZANC;
        }
        else if(toplam == 7){
            durum = KAYIP;
        }
```

5. satırda yeni bir ifade var: enum. enum, numaralandırma anlamına gelen enumeration kelimesinin kısaltmasıdır.

5. satırın anlamı: Oyun isminde bir liste var ve bu listenin elemanları olan KAZANC KAYIP ve DEVAM define komutuyla yazılmış gibi sırasıyla 0 1 ve 2 olarak numaralandırılmış.

13. satırda ise bu numaralandırılmış Oyun isimli listenin elemanlarını tutacak durum isminde bir değişken tanımladık

46. satırdan sonrası diğer sayfada

```
47
48      }
49
50      if(durum == KAZANC)
51          printf("tebrikler\n");
52      else
53          printf("seni loser\n");
54
55      return 0;
56  }
57
58  int zar_at(){
59
60      int zar1 = rand() % 6 + 1;
61      int zar2 = rand() % 6 + 1;
62
63      int toplam = zar1 + zar2;
64
65      printf("toplam: %d zarlar:%d %d\n",toplam,zar1,zar2);
66
67      return toplam;
68
69  }
```

ÇALIŞMA SORULARI

1) bir madeni parayı 10000 defa atın kaç defa yazı kaç defa tura geldiğini hesaplayın ve yazı gelme olasılığını bastırın.

2)craps oyununu modifiye ederek 10000 defa craps oyununu oynatın kaç defa kazanç ve kayıp olduğunu bastırın.Oyuncu açısından bu oyunu oynamak mı yoksa oynamamak mı daha karlıdır

3)Computer Assisted Instruction "bilgisayar destekli eğitim" anlamına gelir. biz de öğrencilere toplama işlemi yaptıran bir program yazacağız.

program aşağıdaki gibi çalışacaktır:

öğrenciye 10 tane kolay soru sorulacak(kolay soru 1 basamaklı sayılarla işlem 5 + 9 gibi)

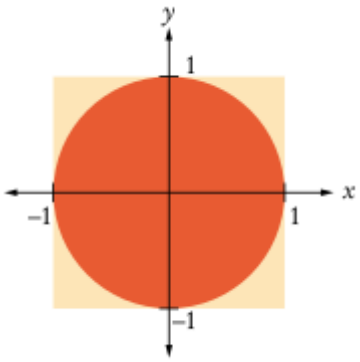
herbir soru için doğru cevap verildiğinde "tebrikler" "harikasın" "bravo" kelimelerinden biri, yanlış cevap için de "bir daha dene" "diğer soruyu yapabilirsin" "vazgeçmek yok" mesajlarından biri yazılacak.

eğer 10 soruda 7 veya daha az sayıda doğru varsa "bu asamayi gecemedin ogretmeninden yardim iste" mesajı ile çıkış yapılacak

eğer 10 soruda 8 veya daha fazla doğru varsa "zor sorulara devam etmek ister misin (e / h): " şeklinde sorulup h girilirsen "gule gule" mesajı ile çıkış yapılacak. e girilirse 10 tane zor soru(iki bsamaklı sayılarla toplama işlemi 25 + 69 gibi) daha sorulup çıkış yapılacak

4)Monte carlo yöntemi ile pi sayısını hesaplayın:
Monte carlo yöntemi tam olarak sonuç alınamayacak görevler için yakın çözümler üretmektir. Mesela: pi sayısını tam olarak hesaplamak zordur fakat pi sayısına çok yakın bir sayı aşağıdaki yöntemle bulunabilir:

yarıçapı 1 olan bir çemberin, kenar uzunluğu 2 olan bir kare içine tam olarak oturtulduğu bir sistemle dart oynadığımızı düşünelim. Kare ve çemberin koordinat düzlemine aktarılmış hali aşağıdaki gibidir



Attığımız dart eğer hedefe ulaşıyorsa, bu çemberin içinde rastgele bir yere düşecek demektir. Çok fazla sayıda dart attığımızda dart sisteminin oklarla dolacağını hayal edebilirsiniz.(okların kordinatları rastgele ondalık sayılar olacak)
Bu durumda
çemberin_içindeki_oklar / tüm_oklar oranı = pi / 4
tür.

Sizin göreviniz:

```
double rastgele_double(double alt_limit, double ust_limit){
    return alt_limit + (ust_limit - alt_limit) *  rand() / RAND_MAX;
}
```

fonksiyonu ile atılan bir okun koordinatlarını belirlemek

double pi_hesapla(int atis_sayisi) fonksiyonu ile çemberin içine düşen ve düşmeyen okların sayısı bulmak ve çemberin_içindeki_oklar / tüm_oklar oranı = pi / 4 hesabını yapmak

pi sayısını yazdırmak