How to generate random numbers

- <stdlib.h> içindeki rand() fonksiyonu kullanılır.
- rand() fonksiyonu 0 ile 2147483647 arasında bir integer değer üretir.
- Üretilen random integerı "%" operatörüyle istediğimiz değer arasına çekebiliriz.
- rand() kullanmadan önce <stdlib.h> headerını dahil etmeyi unutma!

ÖRNEK

```
int a = rand() % 100;       // a in the range 0 to 99
int b = rand() % 100 + 1;   // b in the range 1 to 100
int c = rand() % 30 + 1985; // c in the range 1985-2014
```

Enumerations *(a.k.a. Enum)*

- Stringlerle ifade edilen constant integerlar kümesi olarak düşünülebilir.
- Değişkenler(stringler) arkaplanda integerlara atanır(0'dan başlayarak 1'er 1'er artar).
- Programmer kendi de integer ataması yapabilir.
- Stringler büyük harfle yazılır.
- Aşağıdaki örnekte METU'nun değeri 0, ITU'nun 1 ve BOGAZICI'nin 2'dir.
- Bu stringler yerine, integer değerleri de kullanılabilir.

ÖRNEK

```
#include <stdio.h>

enum school { METU, ITU, BOGAZICI };

int main(){

      enum school mySchool = ITU;

      if(mySchool == ITU) // ITU yerine 1 de yazılabilirdi.
            puts("Your school is Istanbul Technical University! :)");

      else if(mySchool == METU)  // else if(mySchool == 0) ile aynı.
            puts("Your school is Middle East Technical University");

      else
            puts("Your school is Bogazici University");

}
```

Storage Classes

- Bunlar bir değişkenin nerelerden ulaşılabileceğini ve ne kadar süre hafızada tutulacağını belirleyen classlardır.
- Değişkenler default olarak auto storage classından türetilirler.

Auto

- Değişken sadece o bloğa girildiğinde oluşturulur, bloktan çıkınca yok edilir.
- Mesela kendi yazdığın her fonksiyonun içinde tanımladığın değişkenler(local variables) auto classına aittir, fonksiyon bittiğinde değişken yok edilir.

Static

- Değişken programın başlangıcından sonuna kadar hafızada tutulur.
- Execution başlamadan değişkenin hafızada yeri allocate edilir ve bu değişkene sadece 1 kez atama yapılır!
- Bir fonksiyon içinde static değişken tanımlarsan(local değişken yani auto), fonksiyondan çıkıldığında yok edilir ama değeri korunur(bu fonksiyon yeniden çağırıldığında bir önceki değer hatırlanır).
- A static local variable exists for the duration of the program but is visible only in the function body.
- Bütün nümerik static değişkenler, oluşturulduklarında sen initialize etmezsen otomatik olarak 0'a initialize edilirler.

- Global değişkenler bütün fonksiyonlardan önce tanımlanırlar. Her yerden ulaşılabilirler.
- Scope: Bir değişkenin ulaşılabileceği kısımdır.

Recursion

- Base case'i determine edebilmek önemli kısım.

Arrays

- Bracketlerle array tanımlayınca malloc'a gerek yok.
  - int x[10]; // 10luk bi yer allocate eder memory'de ve bunlar garbage value içerir.
- " int n[10] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 }; " şeklinde initialize edilebilir.
- Ya da bi for döngüsü ile initialize edilebilir element by element.
- Declare ederken sizeını belirtmezsen ve liste ile initialize edersen hata almazsın, listedeki eleman sayısı kadar size atar arraye.
- " #define SIZE 10 " diyip int n[SIZE]; diye declare edebilirsin.
- 2 katlı arraylerde => int variableName [Row Index][Column Index]
- Array'in adı, ilk elemanının adresini yani arrayin adresini verir.
- Bu yüzden scanf'te array isminden önce "&" kullanılmaz. *scanf("%s",myArray);*
- Static arraylerin elemanları 0'a initialize edilir otomatik olarak.

ÖRNEK

```
int x[] = {2,3,4,5};
printf("Address of x:  %p\nAddress of x[0]:   %p\n", x, &x[0]);
```

Yukarıdaki örnekte ikisinin de aynı adresi gösterdiğini görebilirsin.

- Fonksiyon array alacaksa, fonksiyonun input kısmına arrayin sizeını yazmaya gerek yok.
- Individual array elements are passed-by-value.
- Eğer bir array fonksiyona gönderilirken const olarak gönderilirse, elemanlarında değişiklik yapmaya çalışınca compile-time error yersin.

- Fonksiyonların içinde array elemanları değiştirilirken * kullanılmaz.

  ÖRNEK

```
#include <stdio.h>

void modifyArray( int b[] );

      int main( void ){
         int a[] = { 10, 20, 30 };
         modifyArray( a );
         printf("%d %d %d\n", a[ 0 ], a[ 1 ], a[ 2 ] );
      }

      void modifyArray( int b[] ){
            b[ 0 ] /= 2;
            b[ 1 ] /= 2;
            b[ 2 ] /= 2;
      }

// OUTPUT: 5 10 15
```