

İskenderun Teknik Üniversitesi



Fakülte: Mühendislik ve Doğa Bilimleri *Fakültesi*

Bölüm: *Bilgisayar Mühendisliği Bölümü*

Ders: Algoritmalar ve Programlama

Dönem: 2020 – 2021 (Güz)

Öğretim üyesi: Dr. Öğr. Üyesi Mehmet SARIGÜL

9.

Hafta

- Yerel değişkenlere yalnızca tanımlandıkları işlevde erişilebilir.
- Bir değişken, işlev tanımlarıyla aynı seviyede herhangi bir işlevin dışında tanımlanırsa, aynı kaynak dosyada aşağıda tanımlanan tüm işlevler tarafından kullanılabilir.
 - harici değişken
- Global değişkenler: herhangi bir fonksiyon tanımından önce tanımlanan harici değişkenler
 - Kapsamı tüm program olacak

Harici Değişkenler


```

/* an example of local variables */
#include <stdio.h>

void func (void);

int main (void)
{
    int i = 5;
    printf("%d \n", i);
    func();
    printf("%d \n", i);
    func();
    return 0;
}

void func (void)
{
    int i = 5;
    printf("%d \n", i);
    i++;
    printf("%d \n", i);
}

```

- Output:
- 5
- 5
- 6
- 5
- 5
- 6

Yerel Değişkenler



```
/* an example of global variables */  
#include <stdio.h>
```

```
int i = 5;
```

```
void func (void);
```

```
int main (void)  
{  
    printf("%d \n", i);  
    func();  
    printf("%d \n", i);  
    func();  
    return 0;  
}
```

```
void func (void)  
{  
    printf("%d \n", i);  
    i++;  
    printf("%d \n", i);  
}
```

- Output:
- 5
- 5
- 6
- 6
- 6
- 7

Global değişkenler

- Bir değişkenin statik olduğu, eğer program yürütmesinin başlangıcında depolama tahsis edilirse ve depolama, program yürütme sona erene kadar tahsis edilmiş olarak kalırsa söylenebilir.
- Dış değişkenler her zaman statiktir
- Bir blok içinde, bir değişken, tür bildiriminden önce static anahtar sözcüğü kullanılarak statik olarak belirtilebilir:
 - statik tür değişken adı;
- Belirtilen statik değişken yalnızca sabit ifadelerle başlatılabilir (değilse, varsayılan değeri sıfırdır)

Statik değişkenler



```
/* an example of global variables */  
#include <stdio.h>
```

```
int i = 5;
```

```
void func (void);
```

```
int main (void)  
{  
    printf("%d \n", i);  
    func();  
    printf("%d \n", i);  
    func();  
    return 0;  
}
```

```
void func (void)  
{  
    printf("%d \n", i);  
    i++;  
    printf("%d \n", i);  
}
```

- Output:

- 5
- 5
- 6
- 5
- 6
- 7


Statik deęişken

- Matematiksel işlemler için
 - math
 - `#include<math.h>`

cmath fonksiyonları

Function	Description	Example
<code>sqrt(x)</code>	square root of x	<code>sqrt(900.0)</code> is 30.0 <code>sqrt(9.0)</code> is 3.0
<code>cbrt(x)</code>	cube root of x (C99 and C11 only)	<code>cbrt(27.0)</code> is 3.0 <code>cbrt(-8.0)</code> is -2.0
<code>exp(x)</code>	exponential function e^x	<code>exp(1.0)</code> is 2.718282 <code>exp(2.0)</code> is 7.389056
<code>log(x)</code>	natural logarithm of x (base e)	<code>log(2.718282)</code> is 1.0 <code>log(7.389056)</code> is 2.0
<code>log10(x)</code>	logarithm of x (base 10)	<code>log10(1.0)</code> is 0.0 <code>log10(10.0)</code> is 1.0 <code>log10(100.0)</code> is 2.0
<code>fabs(x)</code>	absolute value of x as a floating-point number	<code>fabs(13.5)</code> is 13.5 <code>fabs(0.0)</code> is 0.0 <code>fabs(-13.5)</code> is 13.5
<code>ceil(x)</code>	rounds x to the smallest integer not less than x	<code>ceil(9.2)</code> is 10.0 <code>ceil(-9.8)</code> is -9.0

Function	Description	Example
<code>floor(x)</code>	rounds x to the largest integer not greater than x	<code>floor(9.2)</code> is 9.0 <code>floor(-9.8)</code> is -10.0
<code>pow(x, y)</code>	x raised to power y (x^y)	<code>pow(2, 7)</code> is 128.0 <code>pow(9, .5)</code> is 3.0
<code>fmod(x, y)</code>	remainder of x/y as a floating-point number	<code>fmod(13.657, 2.333)</code> is 1.992
<code>sin(x)</code>	trigonometric sine of x (x in radians)	<code>sin(0.0)</code> is 0.0
<code>cos(x)</code>	trigonometric cosine of x (x in radians)	<code>cos(0.0)</code> is 1.0
<code>tan(x)</code>	trigonometric tangent of x (x in radians)	<code>tan(0.0)</code> is 0.0




```
#include <stdio.h>
#include <math.h>
int main()
{
    float i=5.1, j=5.9, k=-5.4, l=-6.9;
    printf("floor of %.2f is %f\n", i, floor(i));
    printf("floor of %.2f is %f\n", j, floor(j));
    printf("floor of %.2f is %f\n", k, floor(k));
    printf("floor of %.2f is %f\n", l, floor(l));
    return 0;
}
```

floor of 5.10 is 5.000000

floor of 5.90 is 5.000000

floor of -5.40 is -6.000000

floor of -6.90 is -7.000000



```
#include <stdio.h>
#include <math.h>
int main()
{
    float i=5.1, j=5.9, k=-5.4, l=-6.9;
    printf("ceil of %.2f is %f\n", i, ceil(i));
    printf("ceil of %.2f is %f\n", j, ceil(j));
    printf("ceil of %.2f is %f\n", k, ceil(k));
    printf("ceil of %.2f is %f\n", l, ceil(l));
    return 0;
}
```

ceil of 5.10 is 6.000000

ceil of 5.90 is 6.000000

ceil of -5.40 is -5.000000

ceil of -6.90 is -6.000000

- Rastgelelik, `<stdlib.h>` başlığından, C standart kitaplık işlevi `rand` kullanılarak bilgisayar uygulamalarına dahil edilebilir.
- Şu ifadeyi düşünün:
 - `i = rand ();`
- `rand` işlevi 0 ile `RAND_MAX` arasında bir tamsayı üretir (`<stdlib.h>` başlığında tanımlanan sembolik bir sabit).


Rastgele sayı üretmek

- Standart C, RAND_MAX değerinin en az 32767 olması gerektiğini belirtir ki bu, iki baytlık (yani 16 bitlik) bir tamsayı için maksimum değerdir. (16 bit çalışan sistemlerde)
- Rand gerçekten rasgele tamsayılar üretirse, 0 ile RAND_MAX arasındaki her sayının rand her çağrıldığında seçilme şansı (veya olasılığı) eşittir.
- Doğrudan rand tarafından üretilen değerler aralığı, genellikle belirli bir uygulamada ihtiyaç duyulandan farklıdır.

- Örneğin, altı yüzlü bir zarı simüle eden bir zar atma programı, 1'den 6'ya kadar rastgele tamsayılar gerektirecektir.
- Kalan operatörünü (%) rand ile birlikte aşağıdaki gibi kullanıyoruz
 - `rand ()% 6`
- Bu da 0 ila 5 aralığında tamsayılar üretir.
- Buna ölçekleme denir.

Rastgele sayı üretimi

- 6 rakamı ölçekleme faktörü olarak adlandırılır.
- Ardından, önceki sonucumuza 1 ekleyerek üretilen sayı aralığını değiştirebiliriz.
 - $1 + \text{rand} () \% 6$



```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    for (int i = 1; i <= 20; i++)
    {
        printf("%5d", 1 + rand() % 6);
        if (i % 5 == 0)
            printf("\n");
    }
    return 0;
}
```

Output:

6	6	5	5	6
5	1	1	5	3
6	6	2	4	2
6	2	3	4	1

- Bu programı tekrar çalıştırmak tamamen aynı değerler dizisini üretir.
- Bunlar nasıl rastgele sayılar olabilir? İronik olarak, bu tekrarlanabilirlik rand fonksiyonunun önemli bir özelliğidir.
- Rand işlevi aslında sözde rasgele sayılar üretir.
- Rand'ı tekrar tekrar çağırmak rastgele görünen bir sayı dizisi üretir.
- Ancak, program her çalıştırıldığında sıra kendini tekrar eder.

Rastgele sayı üretmek

- Bir program tamamen hata ayıklandıktan sonra, her yürütme için farklı bir rasgele sayı dizisi üretmeye koşullandırılabilir.
- Buna rasgeleleştirme denir ve standart kitaplık işlevi srand ile gerçekleştirilir.
- Srand işlevi, işaretsiz bir tamsayı bağımsız değişkenini alır ve programın her çalışması için farklı bir rasgele sayı dizisi üretmek için rand işlevini tohumlar.(seed)

Rastgele sayı üretmek



```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int seed;
    printf("Enter a seed:");
    scanf("%d", &seed);
    srand(seed);
    for (int i = 1; i <= 10; i++)
    {
        printf("%5d", 1 + rand() % 6);
        if (i % 5 == 0)
            printf("\n");
    }
    return 0;
}
```

Enter seed: **67**

6	1	4	6	2
1	6	1	6	4

Enter seed: **867**

2	4	6	1	6
1	1	3	6	2

Enter seed: **67**

6	1	4	6	2
1	6	1	6	4

- Her seferinde bir tohum girmeden rastgele seçim yapmak için aşağıdaki gibi bir ifade kullanın
- `srand (time(NULL));`
- Bu, bilgisayarın tohumun değerini otomatik olarak almak için saatini okumasına neden olur.
- İşlev zamanı, 1 Ocak 1970 gece yarısından bu yana geçen saniye sayısını döndürür.
- Bu değer işaretsiz bir tam sayıya dönüştürülür ve rastgele sayı üreticinde çekirdek olarak kullanılır.
- `time` işlev prototipi `<time.h>` içindedir.



```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
    srand(time(NULL));
    for (int i = 1; i <= 10; i++)
    {
        printf("%5d", 1 + rand() % 6);
        if (i % 5 == 0)
            printf("\n");
    }
    return 0;
}
```



```

• /* rand example: guess the number */
• #include <iostream>
• #include <cstdlib>      /* srand, rand */
• #include <ctime>        /* time */
• using namespace std;
• int main()
• {
•     int iSecret, iGuess;

•     srand(time(NULL));
•     iSecret = rand() % 10 + 1;
•     do {
•         cout<<"Tahmin et (1 to 10): "<<endl;
•         cin >> iGuess;
•         if (iSecret < iGuess) cout<<"Gizli numara daha
dusuk"<<endl;
•         else if (iSecret > iGuess) cout<<"Gizli numara daha
yuksek"<<endl;
•     } while (iSecret != iGuess);

•     cout<<"Tebrikler!!!";
•     return 0;
• }

```

- Bilgisayara karşı bir sayı oyunu oynanacaktır. Bu oyunda bilgisayar 4 haneli bir sayı tutacaktır.
- Her turda bir tahmin yapılacaktır. Eğer rakam doğru ve yeri de doğru ise bunun sayısı, eğer rakam doğru ama yeri yanlış ise bunun sayısı da verilecektir.
- Örnek:
 - Tutulan sayı 1453 -> tahmin 1257 -> 2 doğru rakam doğru yerde diyecektir.
 - Tutulan sayı 1453 -> tahmin 1549 -> 1 doğru rakam doğru yerde 2 doğru rakam yanlış yerde.
- Bu oyunu yazınız.

Ödev



Ders Sonu

