Algoritmalar ve Programlama I

5.Hafta

Dr. Öğr. Üyesi Güncel SARIMAN
E-posta: guncelsariman@mu.edu.tr

C Programlama Dili Elemanları

- ✓ 1. Tanımlayıcılar
- ✓ 2. Anahtar Sözcükler
- ✓ 3. Veri Türleri
- ✓ 4. Değişkenler
- ✓ 5. Sabitler
- ✓ 6. Operatörler

C Tanımlayıcıları

- ✓ Programcı tarafından oluşturulurlar.
- ✓ Programdaki değişkenleri, sabitleri, kayıt alanlarını, özel bilgi tiplerini vb. adlandırmak için kullanılan kelimelerdir.
- ✓ Tanımlayıcılar, yerini tuttukları ifadelere çağrışım yapacak şekilde seçilmelidir.
- ✓ İngiliz alfabesindeki A-Z veya a-z arasındaki 26 harf ile 0-9 arası rakamlar kullanılabilir.
- ✓ Sembollerden sadece alt çizgi _ kullanılabilir.
- ✓ Tanımlayıcı isimleri harfle veya alt çizgiyle başlayabilir.
- ✓ Tanımlayıcı ismi, rakamla başlayamaz veya sadece rakamlardan oluşamaz.
- ✓ Tanımlayıcılar boşluk karakterini içeremezler.

C Anahtar Sözcükleri

- ✓ C dilinde 32 adet anahtar sözcük vardır ve hepsi küçük harfle yazılır. Anahtar sözcükler tanımlayıcı olarak kullanılamazlar.

Veri tipi	Bellek sınıfı	Deyim	İşleç
char	auto	break	sizeof
const	extern	case	
double	register	continue	
enum	static	default	
float	typedef	do	
int		else	
long		for	
short		goto	
signed		if	
struct		return	
union		switch	
unsigned		while	
void			
volatile			

C Veri Türleri

- ✓ Veri türü (data type) program içinde kullanılacak değişken, sabit, fonksiyon isimleri gibi tanımlayıcıların tipini, yani bellekte ayrılacak bölgenin büyüklüğünü, belirlemek için kullanılır.
- ✓ Bir programcı, bir programlama dilinde ilk olarak öğrenmesi gereken, o dile ait veri türleridir. Çünkü bu programcının kullanacağı değişkenlerin ve sabitlerin sınırlarını belirler

C Veri Türleri

- ✓ C programlama dilinde 5 tane temel veri tipi bulunmaktadır.
- ✓ 1. char: karakter veriler
- ✓ 2. int: tamsayı veriler
- ✓ 3. float: tek duyarlıklı kayan noktalı sayılar
- ✓ 4. double: Çift duyarlıklı kayan noktalı sayılar
- ✓ 5. void: Değer içermeyen verilerdir

C Veri Türleri-Özel Niteleyiciler

- ✓ Bazı özel niteleyiciler temel tiplerin önüne gelerek onların türevlerini oluşturur:
- ✓ Short
- ✓ Long
- ✓ Unsigned
- ✓ Niteleyiciler değişkenin bellekte kaplayacağı alanı değiştirilebilirler.
- ✓ Kısa (short), uzun (long), ve normal (int) tamsayı arasında yalnızca uzunluk farkı vardır.
- ✓ Eğer normal tamsayı 32 bit (4 bayt) ise uzun tamsayı 64 bit (8 bayt) uzunluğunda ve kısa tamsayı 16 biti (2 bayt) geçmeyecek uzunluktadır.

C Veri Türleri

- ✓ İşaretsiz (unsigned) ön eki kullanıldığı taktirde, veri tipi ile saklanacak değerin sıfır ve sıfırdan büyük olması sağlanır. İşaretli ve işaretsiz verilerin bellekteki uzunlukları aynıdır. Fakat, işaretsiz tipindeki verilerin üs limiti, işaretlinin iki katıdır.

- ✓ Kısa ve uzun tamsayı tutacak tanımlayıcılar için int anahtar kelimesinin yazılmasına gerek yoktur.

- ✓ short s; /* short int s; anlamında */
- ✓ long k; /* long int k; anlamında */

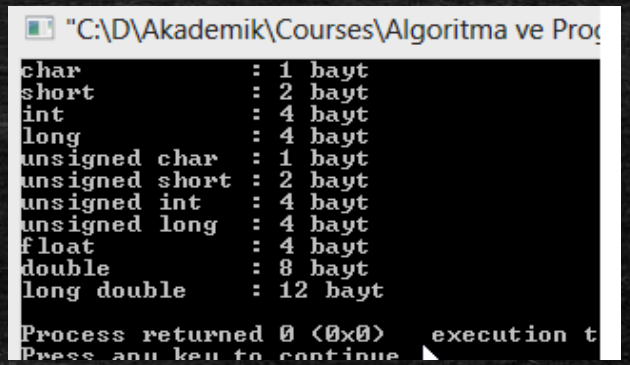
C Veri Türleri

- ✓ Bir C programı içerisinde, veri tiplerinin bellekte kapladığı alan sizeof operatörü ile öğrenilebilir. İ
- ✓ Bu alanların derleyiciye??? ve işletim sistemine??? bağlı olarak değişiklik gösterir.

Veri Tipi	Açıklama	Bellekte işgal ettiği boyut (bayt)	Alt sınır	Üst sınır
char	Tek bir karakter veya küçük tamsayı için	1	-128	127
unsigned char			0	255
short int	Kısa tamsayı için	2	-32,768	32,767
unsigned short int			0	65,535
int	Tamsayı için	4	-2,147,483,648	2,147,483,647
unsigned int			0	4,294,967,295
long int	Uzun tamsayı için	8	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long int			0	18,446,744,073,709,551,615
float	Tek duyarlı gerçel sayı için (7 basamak)	4	-3.4e +/- 38	+3.4e +/- 38
double	Çift duyarlı gerçel sayı için (15 basamak)	8	-1.7e +/- 308	+1.7e +/- 308

C Veri Türleri

```c
/*
*********************************************
* Veri türlerinin bellekte kapladığı alanı  *
* bulmak için yazılan C programıdır.        *
*********************************************
*/
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf( "char           : %d bayt\n", sizeof(char));
    printf( "short          : %d bayt\n", sizeof(short));
    printf( "int            : %d bayt\n", sizeof(int));
    printf( "long           : %d bayt\n", sizeof(long));
    printf( "unsigned char  : %d bayt\n", sizeof(unsigned char));
    printf( "unsigned short : %d bayt\n", sizeof(unsigned short));
    printf( "unsigned int   : %d bayt\n", sizeof(unsigned int));
    printf( "unsigned long  : %d bayt\n", sizeof(unsigned long));
    printf( "float          : %d bayt\n", sizeof(float));
    printf( "double         : %d bayt\n", sizeof(double));
    printf( "long double    : %d bayt\n", sizeof(long double));
    return 0;
}
```



C Değişkenleri

- ✓ Değişken, program içinde kullanılan değerlere bellek üzerinde açılan alanlardır. Bu alanlar bir değişken ismi ile anılırlar.
- ✓ Değişken isimlendirilmeleri, tanımlayıcı kurallarına uygun biçimde yapılmalıdır.
- ✓ C'de tüm değişkenler kullanılmadan önce programa bildirilmelidir.
- ✓ Bu bildirim esnasında, değişkenin veri türü belirlenir.
- ✓ Örnek:
- ✓ veri_türü değişken_adı;
- ✓ int sayac;

C Değişkenleri

Örnekler

- ✓ int x;
- ✓ int x1, y1, z1;
- ✓ long d, d1;
- ✓ char c;
- ✓ char c1, c2, c3;
- ✓ float a;
- ✓ float a1, a2, a3;
- ✓ int u[3];
- ✓ float k[10*20];

Örnekler

```
int x = 1;
int x1 = 10, y1 = 20,
z1 = 30;
char c = 'a';
float a = 123.45;
```

C Sabitleri

Sabit bildirimi, başlangıç değeri verilen değişken bildirim gibi yapılır.

Ancak, veri tipinin önüne const anahtar sözcüğü konmalıdır.

Sabit içerikleri program boyunca değiştirilemez. Yalnızca kullanılabilir.

Genellikle, sabit olarak bildirilen değişken isimleri büyük harflerle, diğer değişken isimlerinin ise küçük harflerle yazılması (gösterilmesi) C programcıları tarafından geleneksel hale gelmiştir

C Sabitleri

Örnekler:

```
const float PI = 3.142857;
const double NOT= 12345.8596235489;
const int EOF= -1;
const char[] = "devam etmek için bir tuşa basın...";
```

printf () - Tip belirleyici (conversion specifier)

% işareti ile başlar ve bir veya iki karakterden oluşur (%d gibi).

Ekrana yazdırılmak istenen değişkenin tipi, % işaretinden sonra belirtilir.

Ayrıca biçim ifadesinin içine, sola - sağa yaslama, noktadan sonra x basamak yaz vb gibi isteklerimizi belirten karakterler de ekleyebiliriz.

Gerçek sayıların yazdırılmasında, noktadan sonra yazılacak basamak sayısı durumların ifade edilmesi için ve tamsayıların aynı hizada yazdırılması için nokta operatörü veya rakamlar kullanılır.

Aynı şekilde karakter katarlarının sağa ya da sola dayalı yazdırılması için veya bir karakter katarındaki karakterlerin kaç tanesinin yazdırılacağını belirtmek için de yine nokta, eksi gibi operatörlerin ve rakamların çeşitli kombinasyonları kullanılır.

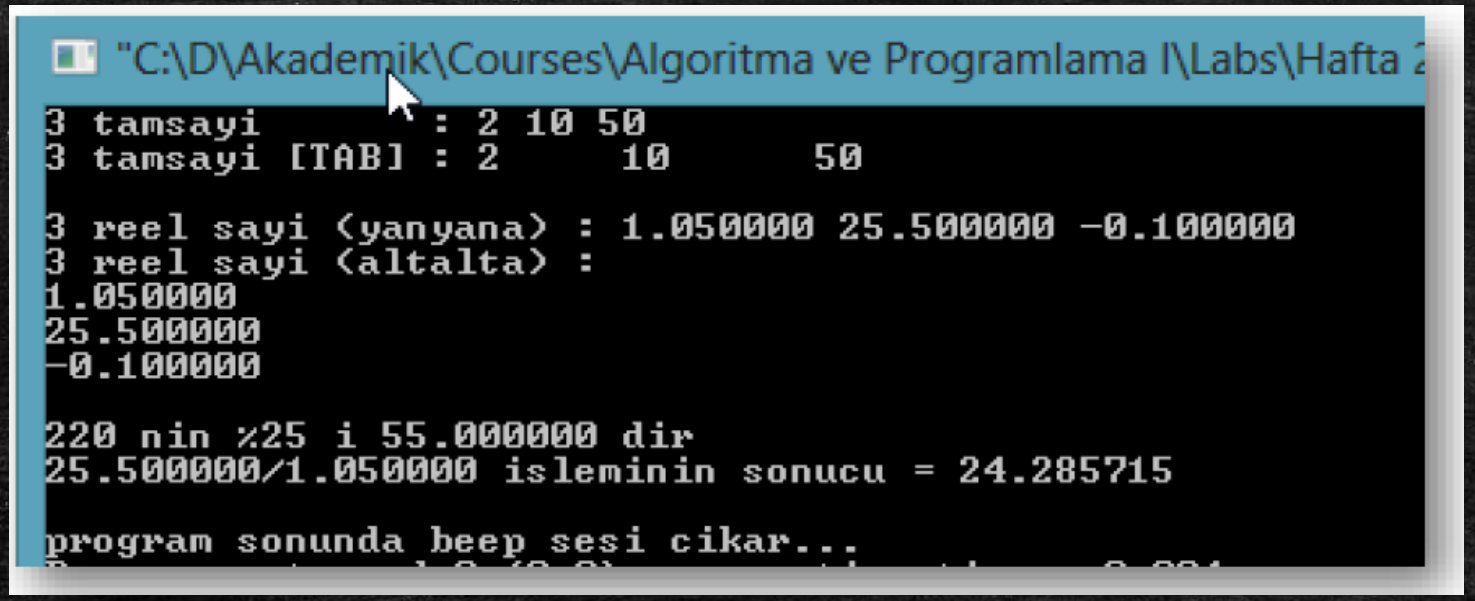
printf () - Tip belirleyici (conversion specifier)

d	int türden bir ifadeyi onluk sistemde yazar
ld	long türden bir ifadeyi onluk sistemde yazar
o	unsigned int türden bir ifadeyi sekizlik sistemde yazar
x, X	unsigned int türden bir ifadeyi onaltılık sistemde yazar; x için küçük harfleri, X için büyük harfleri kullanır
lx	unsigned long türden bir ifadeyi onaltılık sistemde yazar
c	int veya char türden bir ifadeyi karakter olarak yazar
s	char * türden bir ifadeyi null karakter ile karşılaşıncaya kadar, ya da duyarlılıkla belirtilen sayı kadar yazar
u	unsigned int türden bir ifadeyi onluk sistemde yazar
f	double türden bir ifadeyi yazar
lf	double veya long double türden bir ifadeyi onluk sistemde yazar
e	gerçek sayıları üstel olarak yazar
%	dönüştürülmez, % olarak yazdırılır

printf () - Tip belirleyici (conversion specifier)

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

int main()
{
    int   a = 2,    b = 10,   c = 50;
    float f = 1.05, g = 25.5, h = -0.1, yuzde;

    printf("3 tamsayi       : %d %d %d\n", a, b, c);
    printf("3 tamsayi [TAB] : %d \t%d \t%d\n", a, b, c);

    printf("\n");

    printf("3 reel sayi (yanyana) : %f %f %f\n", f, g, h);
    printf("3 reel sayi (altalta) : \n%f\n%f\n%f\n\n", f, g, h);

    yuzde = 220 * 25/100.0;
    printf("220 nin %%25 i %f dir\n", yuzde);
    printf("%f/%f isleminin sonucu = %f\n", g, f, g / f);

    printf("\nprogram sonunda beep sesi cikar...\a");

    return 0;
}
```

printf () - Tip belirleyici (conversion specifier)



C Operatörleri

Operatörler, değişkenler veya sabitler üzerinde matematiksel ve karşılaştırma işlemlerini yapan simgelerdir. Yani bir operatör bir veya daha fazla değişken üzerinde işlem yapan semboldür.

C programlama dilinde 4 tip operatör bulunmaktadır.
1. Aritmetik Operatörler
2. Atama Operatörleri
3. Karşılaştırma Operatörleri
4. Mantıksal Operatörler

Aritmetik Operatörler

Operatör	Açıklama	Örnek	Anlamı
+	toplama	x + y	x ve y nin toplamı
-	çıkarma	x - y	x ve y nin farkı
*	carpma	x * y	x ve y nin çarpımı
/	bölme	x / y	x ve y nin oranı
%	mod alma	x % y	x / y den kalan sayı

```
a = b + 10;
c = d + c * e –f /g + h % j;
z = u[1] * u[2];
x = 10;
a = b = c = 0;
```

Atama Operatörleri

Bu operatörler bir değişkene, bir sabit veya bir aritmetik ifade atamak (eşitlemek) için kullanılır.

Birleşik atama: bazı ifadelerde işlem operatörü ile atama operatörü birlikte kullanılarak, ifadeler daha kısa yazılabilir.

Eğer ifade
değişken = değişken [operatör] = aritmetik ifade;
şeklinde ise, daha kısa bir biçimde
değişken [operatör]= aritmetik ifade;
olarak yazılabilir.

Atama Operatörleri

Bu operatörler bir değişkene, bir sabit veya bir aritmetik ifade atamak (eşitlemek) için kullanılır.

Operatör	Açıklama	Örnek	Anlamı
=	atama	x = 7;	x = 7;
+=	ekleyerek atama	x += 3	x = x + 3
-=	eksilterek atama	x -= 5	x = x - 5
*=	çarparak atama	x *= 4	x = x * 4
/=	bölerek atama	x /= 2	x = x / 2
%=	bölüp, kalanını atama	x %= 9	x = x % 9
++	bir arttırma	x++ veya ++x	x = x + 1
--	bir azaltma	x-- veya --x	x = x - 1

Atama Operatörleri

Örnek	Anlamı

x = y++;	y'nin değeri önce x'e aktarılır sonra bir arttırılır.
	x = y;
	y = y + 1;

x = ++y;	y'nin değeri önce bir arttırılır sonra x'e aktarılır .
	y = y + 1;
	x = y;

x = y--;	y'nin değeri önce x'e aktarılır sonra bir azaltılır.
	x = y;
	y = y - 1;

x = --y;	y'nin değeri önce bir azaltılır sonra x'e aktarılır .
	y = y - 1;
	x = y;

```
a = 5;
b = a++;
c = ++a;
```

```
a = ?;
b = ?;
c = ?;
```

scanf() Fonksiyonu

Birçok programda ekrana verilerin bastırılmasının yanı sıra klavyeden veri okunması gerekebilir.

scanf() fonksiyonu klavyeden veri okumak için kullanılan fonksiyondur.

Tip belirleyicileri printf fonksiyonu ile aynı mantıkta kullanılır ve % sembolü ile ifade edilir.

Örneğin klavyeden bir x tamsayısı okumak için aşağıdaki ifade kullanılır:

```
scanf("%d", &x);
```

Printf() Tip Belirleyicileri

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

int main()
{
    printf ("Karakterler: %c %c \n", 'a', 65);
    printf ("Decimal Sayilar: %d \n", 1977);
    printf ("Oncesinde Bosluk: %10d \n", 1977);
    printf ("Oncesinde 0 yazmak: %010d \n", 1977);
    printf ("float Sayilar: %.2f\n", 3.1416);
    printf ("%s \n", "Bir String");

    return 0;
}
```

İki Gerçel Sayı ile Aritmetik İşlemler

Klavyeden girilen 2 gerçel sayının
Toplamını
Çıkartılmasını
Çarpımını
Bölümünü
bulup yazdıran C programını yazınız.

Not-1: Her aritmetik işlem için birer değişken tanımlayınız ve girilen iki sayının aritmetik işlemini gerçekleştirip, bu değişkene atayınız.

Not-2: Toplama ve çıkartma işleminin sonucu 2 ondalıklı, çarpma işleminin sonucu 4 ondalıklı ve bölme işleminin sonucu 6 ondalıklı olmalıdır.

Örnek:3

Aşağıdaki işlemleri teker teker gerçekleştiriniz ve her işlemden sonra değişkenlerin değerini ekrana yazdırınız

```
a = 5;
            a değerini YAZ
b = a++;
            a ve b değerlerini YAZ
c = ++a;
            a, b ve c değerlerini YAZ
```

Örnek : 4

Örnek 04: Girilen Sınav ve Ödevlere Göre Ders Notu Hesaplama

Klavyeden girilecek aşağıdaki sınav ve ödevlere göre Ders notu hesaplanacaktır:
Ödev: %9 (3 tane)
Quiz: %21 (3 tane)
Ara Sınav: %30 (2 tane yazılı sınav)
Final: %40 (1 tane genel yazılı sınav)

Not1: Toplam 9 tane giriş yapılacaktır. Girişler Gerçel sayı olacaktır.
Not2: Ders notu 2 ondalıklı olarak gösterilecektir.