BİLGİSAYAR PROGRAMLAMAYA GİRİŞ

ÖĞR. GÖR. KORAY AKİ

Program nedir?

Program, günlük hayatta bir sorunu bilgisayar ile çözmek, rutin işlemleri kolaylaştırmak için yazılan yazılımlardır.

Bilgisayar programı yazmaya yarayan bir çok programlama dili vardır.Bunlardan bazıları, Pascal,Delphi,C,C++,Visual Basic,Fortran,..gibi.

Derleyici (Compiler) nedir?

- Programlama dili ile yazılmış bir programı makine dili ile yazılmış amaç veya hedef programa çevirirler.

- Derleyicinin çevirme işlemi 3 adımda gerçekleşir.
  - Sözel Analiz (Lexical Analysis)
  - Gramer Analizi ve Tanıma (Syntax Analysis)
  - Kod Üretici (Code Generator)

Yorumlayıcı (Interpreter) nedir?

- Programın kodlarının ilk satırından son satırına kadar satır satır belirtilmiş komut ve işlemleri inceleyerek kaynak programın hatalarının düzeltilmesine imkan veren ve çalıştıran programdır.

Derleyici ve Yorumlayıcını Farkı

- Yorumlayıcıda kodun okunması ve çevrilmesi programın çalışması sırasında yapıldığından hızı düşüktür.
- Yorumlayıcı yalnızca karşılaştığı ilk hatayı rapor edebilir.
- Derleyici kaynak kodundaki bütün hataları bulabilir.

Bağlama (Linking) nedir?

- Yazılan birçok program genellikle programlama dili içerisinde bulunan kütüphane fonksiyonlarına ve yordamlara bağlantı içerdiğinden, kaynak programdan üretilen amaç program kodunun bilgisayarda çalıştırılmaya hazır olması için bu amaç program bağlama işlemine tabi tutulur. Bu işlem sonucu eksik parçaların program kodu içerisine eklenmesi ile artık yazılan program çalışmaya hazır hale gelmiş olur.

Kaynak Program (Source Program)

- Bir programlama dili ile yazılmış programdır.

- Kaynak program dosyasının uzantısı örneğin C dilinde yazıldı ise .c, C++ dilinde yazılmış ise .cpp olur.

Amaç Program (Object Program)

- Bir programlama dilinde yazılmış kaynak programın derlenmesi ile elde edilen programdır.

- Makine diline dönüştürülmüş ve çalıştırılmaya hazır programdır.

Yazılım nedir?

- Yazılım (software), programlama ve programlamayla ilgili konuların geneline verilen isimdir. Yazılım denince akla programlama dilleri bu diller kullanılarak yazılmış kaynak programlar ve oluşturulan çeşitli dosyalar gelir.

Yazılımın Sınıflandırılması

- Bilimsel yazılımlar ve mühendislik yazılımları
- Uygulama yazılımları
- Yapay zeka yazılımları
- Görüntüsel yazılımlar
- Simülasyon yazılımları
- Sistem yazılımları

Programlama Dillerinin Değerleme Ölçütleri

- Düzey
- Okunabilirlik
- Taşınabilirlik
- Verimlilik
- Kullanım Alanı

Düzey

- Bir programlama dilinin düzeyi(*level)* o programlama dilinin insan algısına ya da makineye yakınlığının ölçüsüdür.
- Bir programlama dili insan algısına ne kadar yakınsa o kadar yüksek düzeyli *(high level)* demektir.
- Yine bir programlama dili bilgisayarın elektronik yapısına ve çalışma biçimine ne kadar yakınsa o kadar düşük düzeyli *(low level)* demektir.

Düzey

- **Çok yüksek düzeyli diller ya da görsel diller ya da ortamlar (*visual languages*):** Access, Foxpro, Paradox, Xbase, Visual Basic, Oracle Forms.
- **Yüksek düzeyli diller**: Fortran, Pascal, Basic, Cobol, Algol
- **Orta düzeyli programlama dilleri:** Ada, C. (Orta düzeyli diller daha az kayıpla makine diline çevrilebildiğinden daha hızlı çalışır.)
- **Düşük düzeyli programlama dilleri:** Simgesel makine dili (*Assembly language*).
- **Makine dili:** En aşağı düzeyli programlama dili. Saf makine dili tamamen *1* ve *0*'lardan oluşur.

Okunabilirlik

- Okunabilirlik *(readability)* kaynak kodun çabuk ve iyi bir biçimde algılanabilmesi anlamına gelen bir terimdir.
- Kaynak kodun okunabilirliği söz konusu olduğunda sorumluluk büyük ölçüde programı yazan programcıdadır.
- En iyi program kodu, sanıldığı gibi "en zekice yazılmış fakat kimsenin anlayamayacağı" kod değildir.

Taşınabilirlik

- Taşınabilirlik (*portability*) bir sistem için yazılmış olan kaynak kodun başka bir sisteme götürüldüğünde, hatasız bir biçimde derlenerek, doğru bir şekilde çalıştırılabilmesi demektir.
- Taşınabilirlik standardizasyon anlamına da gelir.
- Programlama dilleri *(ISO International Standard Organization)* ve *ANSI (American National Standard Institute)* tarafından standardize edilir.

Verimlilik

- Verimlilik (e*fficiency*) bir programın hızlı çalışması ve daha az bellek kullanma özelliğidir.
- Programın çalışma hızı ve kullandığı bellek miktarı pek çok etkene bağlıdır.
- Kullanılan algoritmanın da hız ve kullanılan bellek üzerinde etkisi vardır.
- Verimlilik üzerinde rol oynayabilecek diğer etkenler sabit bırakıldığında, kullanılan programlama dilinin tasarımının da verim üzerinde etkili olduğu söylenebilir. Bu açıdan bakıldığında C verimli bir dildir.

Kullanım Alanı

- Bazı diller özel bir uygulama alanı için tasarlanır.
- Sistem programlama yapay zeka uygulamaları, simülasyon uygulamaları, veritabanı sorgulamaları, oyun programlarının yazımı amacıyla tasarlanan ve kullanılan programlama dilleri vardır.
- Bazı diller daha geniş bir kullanım alanına sahiptir.
- Örneğin veritabanı sorgulamalarında kullanılmak üzere tasarlanan bir dil mühendislik uygulamalarında da kullanılabilir.
- C ana uygulama alanı "sistem programcılığı" olan bir dildir. Ancak neredeyse tüm uygulama alanları için C dilinde programlar yazılmıştır.

Uygulama Alanlarına Göre Sınıflandırma

- Bilimsel ve mühendislik uygulama dilleri: Pascal, C, FORTRAN.
- C Programlama dili üniversitelerdeki akademik çalışmalarda da yoğun olarak kullanılır.
- Veri tabanı dilleri: XBASE, (Foxpro, Dbase, CA-Clipper), Oracle Forms, Visual Foxpro.
- Genel amaçlı programlama dilleri: Pascal, C, Basic.
- Yapay zeka dilleri: Prolog, Lisp
- 5. Simülasyon dilleri GPSS, Simula 67
- Makro Dilleri (Scripting languages) Awk, Perl, Python, Tcl, JavaScript.
- Sistem programlama dilleri: BCPL, C, C++, occam.

Alt Programlama Yeteneği

- Bir bütün olarak çözülmesi zor olan problemlerin parçalara ayrılması, bu parçaların ayrı ayrı çözülmesinden sonra parçalar arasındaki bağlantının sağlanması programlamada sık başvurulan bir yöntemdir.
- Bir programlama dili buna olanak sağlayan araçlara sahipse programlama dilinin alt programlama yeteneği vardır denilebilir. Bu, bir programlama dilinin, programı parçalar halinde yazmayı desteklemesi anlamına gelir.
- Alt programlama algılamayı kolaylaştırır, okunabilirliği artırır. Kaynak kodun test edilmesini kolaylaştırır, daha kolay güncelleştirilmesini sağlar. Alt programlamanın en önemli faydalarından biri de oluşturulan alt programların birden fazla projede kullanılabilmesidir (*reusability*).
- *C*, alt programlama yeteneği yüksek bir dildir. C'de alt programlara *işlev* (*function*) denir. İşevler *C* dilinin temel yapı taşlarıdır.

Öğrenme ve Öğretme Kolaylığı

- Her programlama dilini öğrenmenin ve öğrenilen programlama dilinde uygulama geliştirebilmenin zorluk derecesi aynı değildir.
- Genel olarak programlama dillerinin düzeyi yükseldikçe, bu programlama dilini öğrenme ve başkalarına öğretme kolaylaşır.
- Bugün yaygın olarak kullanılan yüksek düzeyli programlama dillerinin bu derece tutulmasının önemli bir nedeni de bu dillerin çok kolay öğrenilebilmesidir.

Programlama Tekniklerine Verilen Destekler

- Programlamanın tarihsel gelişim süreci içinde, bazı programlama teknikleri *(paradigmaları)* ortaya çıkmıştır.
- Programlamanın ilk dönemlerinde yazılan programların boyutları çok küçük olduğundan program yazarken özel bir teknik kullanmaya pek gerek kalmıyordu.
- Programların büyümesiyle birlikte, program yazmaya yönelik farklı teknikler, yöntemler geliştirildi.
- "Prosedürel programlama" ile "yapısal programlama" çoğu zaman aynı anlamda kullanılır. Yapısal programlama bir programlama tekniğidir.
- Yapısal programlama tekniği dört ana ilke üzerine kuruludur:
  - Böl ve üstesinden gel (*divide and conquer*)
  - Veri gizleme (*Data hiding*)
  - Tek giriş ve tek çıkış (*single entry single exit*)
  - Döngüler, diğer kontrol yapıları

Giriş / Çıkış Kolaylığı

- Sıralı, indeksli ve rasgele dosyalara erişme, veritabanı kayıtlarını geri alma, güncelleştirme ve sorgulama yeteneğidir.
- Veritabanı programlama dillerinin bu yetenekleri, diğerlerinden daha üstündür. Bu dillerin en tipik özelliklerini oluşturur.
- *C* giriş çıkış kolaylığı güçlü olmayan bir dildir.
- *C*'de veri tabanlarının yönetimi için özel kütüphanelerin kullanılması gerekir.

C Nasıl Bir Programlama Dilidir?

- *C* orta düzeyli bir programlama dilidir. Diğer yapısal programlama dillerine göre *C* dilinin düzeyi daha düşüktür.
- *C* dili hem yüksek düzeyli dillerin kontrol deyimleri, veri yapıları gibi avantajlarına sahipken hem de bitsel işleçler gibi makine kodu deyimlerini yansıtan işleçlere sahiptir.
- *C* dili hem makinenin algısına hem de insanın algılamasına yakın bir dildir. C makineye yeterince yakındır ama programcıya da uzak değildir. Tercih edilmesinin ana nedenlerinden biri budur.
- Bir programı *C* dili kullanarak yazmak, aynı programı makine dilinde yazmaya göre çok daha kolay olmasına karşın, *C*'de yazılmış bir programın verimi aynı oranda düşmez.
- *C* dilinin ana uygulama alanı "Sistem programlama" dır.

C Nasıl Bir Programlama Dilidir?

- *C* doğal bir dildir. *C* bilgisayar sistemi ile uyum içindedir.
- *C* algoritmik bir dildir. *C* dilinde program yazmak için yalnızca dilin sözdizimini ve anlamsal yapısını bilmek yetmez genel bir algoritma bilgisi de gerekir.
- *C* diğer dillerle kıyaslandığında taşınabilirliği çok yüksek olan bir dildir. Çünkü 1989 yılından bu yana genel kabul görmüş standartlara sahiptir.
- C ifade gücü yüksek, okunabilirlik özelliği güçlü bir dildir.
- *C* çok esnek bir dildir. Diğer dillerde olduğu gibi programcıya kısıtlamalar getirmez. Makinanın olanaklarını programcıya neredeyse tamamen yansıtır.
- C güçlü bir dildir, çok iyi bir biçimde tasarlanmıştır. *C*'ye ilişkin işleçlerin ve yapıların bir çoğu daha sonra başka programlama dilleri tarafından da benimsenmiştir.

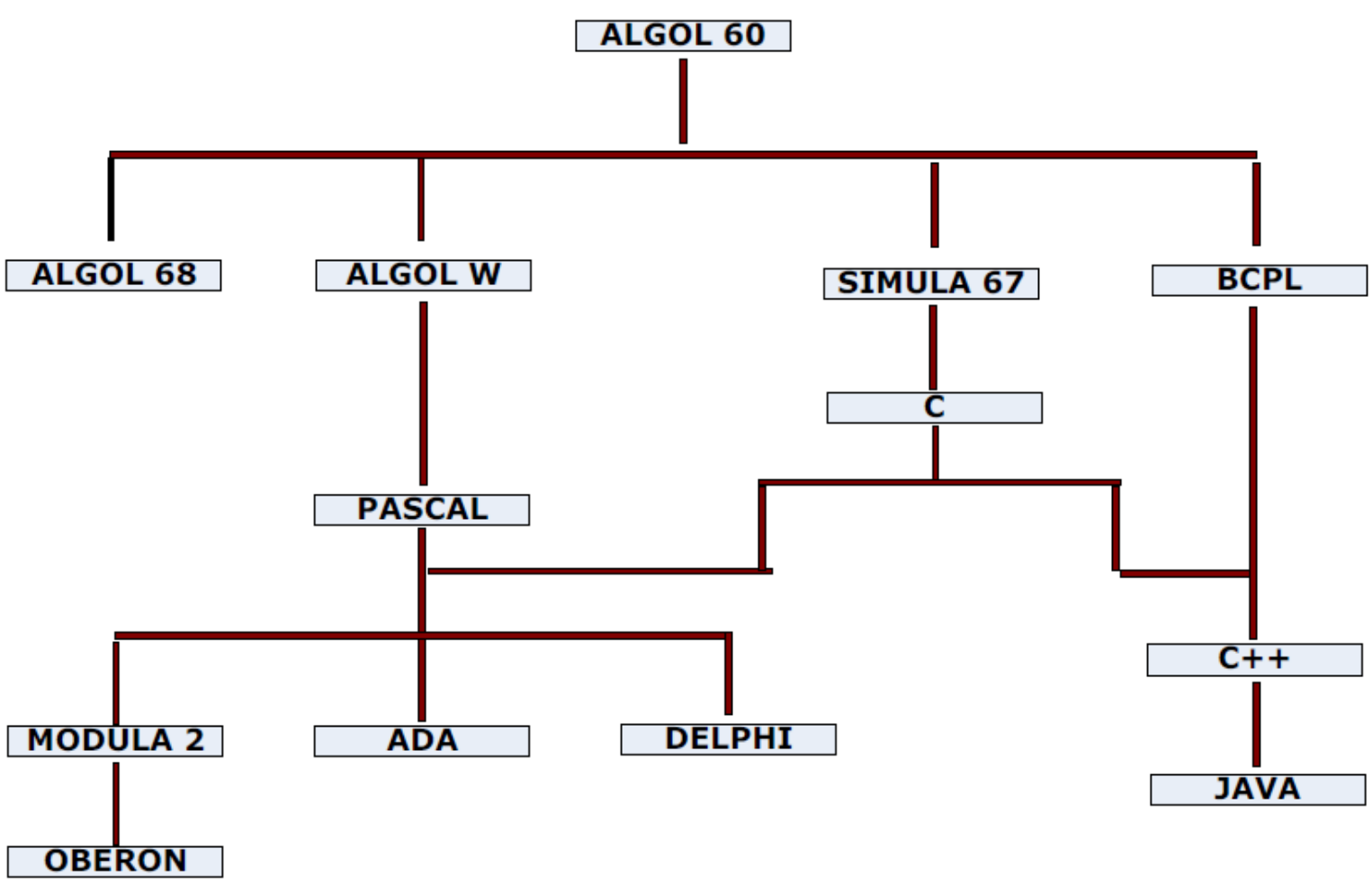
C Nasıl Bir Programlama Dilidir?

- *C* verimli bir dildir. C de yazılan programlar dilin düzeyinin düşük olması nedeniyle hızlı çalışır. Verimlilik konusunda *assembly* diller ile rekabet edebilir.
- *C* küçük bir dildir. Yeni sistemler için derleyici yazmak zor değildir.
- C'nin standart bir kütüphanesi vardır. Bu kütüphane ile sık yapılan işlemler için ortak bir arayüz sağlanmıştır.
- *C*'nin eğitimi diğer bilgisayar dillerine göre daha zordur.
- C dili *UNIX* işletim sistemi ile bütünleşme içindedir. *UNIX* işletim sisteminde kullanılan bazı araçlar kullanıcının C dilini bildiğini varsayar.
- Diğer tüm bilgisayar programlama dillerinde olduğu gibi C dilinin de zayıf tarafları vardır. Esnek ve güçlü bir dil olması programcının hata yapma riskini artırır. C dilinde yazılan kodlarda yapılan yanlışlıkların bulunması diğer dillere göre daha zor olabilir.

C PROGRAMLAMA DİLİNE GİRİŞ

- C dili *Ken Thompson* ve ekip arkadaşı *Dennis Ritchie tarafından* Bell Telefon Laboraturvarında Unix işletim sistemi ile kullanılmak için geliştirilmiştir.
- C, özellikle sistem programlamada sembolik makine dili (Asembler) ile tercih edilmektedir. İşletim sistemleri, derleyiciler ve debug gibi aşağı seviyeli sistem programlarının yazılımında yoğun olarak C programlama dili kullanılır.
- C Programlama Dili, hemen her alanda kullanılmaktadır. Günümüzde nesneye yönelik programlama dilleri (C++, Java) ve script dilleri (JavaScript, JavaApplet, PHP) gibi programlama dilleri C Programlama Dili'nden esinlenmiştir.

C Programlama Dilinin Tarihi



C Programlama Dilinin Tarihi

Ken Thompson ve Dennis Ritchie Unix Ġşletim Sistemi üzerinde çalışırken (Yıl: 1972)

Atomlar ve Türleri

- Bir programlama dilinde yazılmış kaynak dosyanın (*program*) anlamlı en küçük parçalarına "atom" (*token)* denir.
- Bir programlama dilinde yazılmış bir metin, atomlardan oluşur.
- Bir kaynak dosya, derleyici program tarafından ilk önce atomlarına ayrılır. Bu işleme "atomlarına ayırma" *(Tokenizing - Lexical analysis*) denir. Farklı programlama dillerinin atomları birbirlerinden farklı olabilir.

Anahtar Sözcükler

- Anahtar sözcükler (*keywords - reserved words)* programlama dili tarafından önceden belirlenmiş anlamlara sahip atomlardır.
- Bu atomlar kaynak dosya içinde başka bir anlama gelecek biçimde kullanılamaz. Örneğin, bu atomların değişken ismi olarak kullanılmaları geçerli değildir.
- *Standart ANSI C* (*C 89*) dilinde *32* anahtar sözcük vardır:

Anahtar Sözcükler

Auto	break	case	char	const	continue
default	do	double	else	enum	extern
float	for	goto	if	int	long
register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void
volatile	while				

İsimler

- Değişkenler, işlevler, makrolar, değişmezler gibi yazılımsal varlıklara programlama dili tarafından belirlenmiş kurallara uyulmak koşuluyla isimler verilebilir. Bazı atomlar yazılımsal bir varlığa verilen isme (*identifier)* ilişkindir .
- Her programlama dilinde olduğu gibi *C* dilinde de kullanılan isimlerin geçerliliğini belirleyen kurallar vardır.

Bağımsız Atomlar (Punctuators)

- Bağımsız atomlar, anlamsal açıdan değer taşıyan atomlardır. Bağımsız atomlar işleç (*operator*) ya da ayıraç (*separators*) olarak görev yapabilir. Bu atomlardan bazıları önişlemci programa (*preprocessor)* ilişkindir.

```
[ ] ( ) { } . ->
++ -- & * + - ~ !
/ % << >> < > <= >= == != ^ | && ||
? : ; ...
= *= /= %= += -= <<= >>= &= |=
, # ##
<: :> <% %> %: %:%:
```

Sabitler

- Değişmezler (*constants*) doğrudan işleme sokulan, dinamik olarak değişebilen bilgi içermeyen atomlardır.

Örneğin:

sayac = son + 10

gibi bir ifadede *10* değişmezi doğrudan *son* isimli değişken ile işleme sokulur.

Dizgeler

- İki tırnak içindeki ifadelere "dizge" *(string / string literals)* denir.
- Dizgeler programlama dillerinin çoğunda tek bir atom olarak alınır, daha fazla parçaya bölünemez.

"**DİZGELER DE BİRER ATOMDUR**" ifadesi bir dizgedir.

Başlık Dosyaları

- **#include <stdio.h>** #include deyimi, programda eklenecek olan başlık dosyanını işaret eder. Bu örnekte verilen başlık dosyası (header file) stdio.h dir. #include <stdio.h> ifadesi stdio.h dosyasının derleme işlemine dahil edileceğini anlatır.
- **main() özel bir fonksiyondur**. Ana program bu dosyada saklanıyor anlamındadır. Programın yürütülmesine bu fonksiyondan başlanır. Dolayısıyla her C programında bir tane main() adlı fonksiyon olmalıdır.
- **printf()** standart kütüphane bulunan ekrana formatlı bilgi yazdırma fonksiyondur. stdio.h dosyası bu fonksiyonu kullanmak için program başına ilave edilmiştir. Aşağıda printf() fonksiyonunun basit kullanımı gösterilmiştir.
- **getchar()** fonksiyonu ile standart girişten bir karakter okunur. Programı istenen bir yerde durdurup, bir karekter girinceye kadar bekletir.
- **return o** main() fonksiyonuna işinin bittiğini gösterir.

Veri Tipleri

- C programlama dilinde dört tane temel veri tipi bulunmaktadır.

❑ Bunlar:

- char
- int
- float
- Double

- Bazı özel niteleyiciler vardır ki bunlar yukarıdaki temel tiplerin önüne gelerek onların türevlerini oluşturur.

❑ Bunlar:

- short
- long
- unsigned

Veri Tipleri

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

Giriş çıkış fonksiyonlari

printf(); çıkış için , **scanf();** giriş için kullanılır.

printf() fonksiyonu:
Fonksiyonun yapısı aşağıdaki gibidir:
printf("çıkış biçimi karakterleri", değerler listesi);

Buradaki "çıkış biçimi karakterleri" şunlar olabilir:
a) Açıklama karakterleri:
Olduğu gibi ekranda görünmesini istediğimiz karakterlerdir.

b) Escape düzeni karakterleri:

İmlecin yerini ve bazı özel karakterleri yazdırmaya yarar. Bu karakterler \ ile başlar. Aşağıdaki listede bu karakterler ve anlamları bel

Karakter	Anlamı
\n (new line)	Kürsörü alt satıra geçirir
\r (return)	Kürsörü satır başına geçirir
\b (back space)	Kürsörü bir önceki sütuna çeker
\a (alarm, bell)	Bip sesi vermeye yarar
\t (tab)	Kürsörü belirli bir miktar sağa kaydırır
\\	\ koymaya yarar
\'	' koymaya yarar
\"	" koymaya yarar
\?	? koymaya yarar
\0	NULL karakteri

c) Çıkış biçimi belirten karakterler:

Değerler listesi bölümündeki değişkenlerin değerlerinin ekrana yazılış biçimlerini belirten ifadelerdir. Bu ifadeler % ile başlar.

Aşağıdaki listede bu karakterler ve anlamları belirtilmiştir.

Karakter	Anlamı
%d	İşaretli tamsayı
%ld	Uzun tamsayı
%f	Ondalıklı (reel) sayı
%lf	Double (uzun) ondalıklı sayı
%e	Exponantiel (üstel) sayı
%x	Hexadecimal sayı
%c	Tek karakter
%s	String (karakter grubu)

Değerler listesi:

Değerler listesine, ekrana yazdırılacak değişkenler veya işlemler yazılır.

scanf() fonksiyonu:

Tanımlanmış değişkenlerin temsil ettiği bellek bölgelerine değer okutmak amacıyla kullanılır. Fonksiyonun yapısı aşağıdaki gibidir:

scanf("giriş biçimi karakterleri",adresler listesi);

Giriş biçimi karakterleri:

Buradaki giriş biçimi karakterleri, printf() fonksiyonundaki çıkış biçimi karakterlerine karşılık gelir.

Adresler listesi:

Bellekte atama yapılacak değişkenlerin adreslerini belirtmeye yarar. Bunun için değişken adının önüne & konur.

Program 1.1: *Derlendikten sonra ekrana 'Merhaba Dünya!' yazar*

```
01: /* ilk.c: ilk C programi */
02: #include <stdio.h>
03:
04: main()
05: {
06:    printf("Merhaba Dünya!\n");
07: }
```

Programda, 1. satırda /* ... */ sembolleri görülmektedir. Bu ifadeler arasında yazılan herhangi bir metin, işlem vb. satırlar, derleyici tarafından işlenmez (değerlendirilmez). Yani /* */ ifadeleri açıklama operatörüdür.

#include <stdio.h>

2. satırdaki #include deyimi, programda eklenecek olan başlık dosyasını işaret eder. Bu örnekte verilen başlık dosyası (header file) stdio.h dir.

#include <stdio.h> ifadesi stdio.h dosyasının derleme işlemine dahil edileceğini anlatır

main()

4. satırdaki main() özel bir fonksiyondur.

Ana program bu dosyada saklanıyor anlamındadır.

Programın yürütülmesine bu fonksiyondan başlanır. Dolayısıyla her C programında **bir tane** main() adlı fonksiyon olmalıdır

Örnek:

```
#include<stdio.h>
int a,b;
void main() {
    printf("ilk sayınız");
    scanf("%d",&a);
    printf("ikinci sayınız");
    scanf("%d",&b);
    printf("%d",a+b);
}
```

Not 1) Programın sonucunda iki sayının toplamının bulunduğunu belirtmek için;
printf("Toplam=%d",a+b); yazılır.

Not 2) Girilen sayıları ve toplamı yazdırmak istersek bunu; printf("%d + %d = %d dir",a,b,a+b); biçiminde yazarız.

Not 3) Programın her çalıştırılmasında ekranda, daha önceden kalan görüntünün silinmesini istersek, programda clrscr(); fonksiyonunu kullanırız.

Not 4) Programın çalışması bittiğinde, hemen tümleşik ortama dönülür.Bu durumda ekrandaki görüntüyü tam göremeyiz.Şayet programın çalışmasını bir tuşa basana kadar bekletmek istersek; bunu system("Pause"); fonksiyonu yardımıyla yaparız.

Buna göre programın son durumu şöyle olmalıdır:

```
#include<stdio.h>
int a,b;
void main() {
    clrscr();
    printf("ilk sayınız");
    scanf("%d",&a);
    printf("ikinci sayınız");
    scanf("%d",&b);
    printf("%d + %d = %d dir",a,b,a+b);
    system("Pause");
    return 0;
}
```

C de Atama ve Matematiksel İşlemler:

C de değer atama;

değişken=atanan_değer; biçiminde yapılır.

Örnek:

a) a = 3; b = 7; c = a + b;

b) a = 3; b = 7; a = a + 5; c = a + b;

c) a = 3; b=7; a = a - 1;

b = a + b - 2; c = d = a * b + b; d = d - 1;

değişken=değişken+değer; ifadesi ile değişkenin değeri değer kadar arttırılır.

değişken=değişken-değer; ifadesi ile değişkenin değeri değer kadar azaltılır.

Matematiksel İşlemler:

Aşağıdaki tabloda, matematiksel işlemler ve C deki karşılıkları ile anlamları verilmiştir.

İşlem	Anlamı
+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Modül
++	Bir arttırma
--	Bir eksiltme

Matematiksel İşlemler:

Matematiksel ifadeler hesaplanırken izlenen adımlar:

1. Önce parantez içindeki ifadeler hesaplanır. İç içe parantezler var ise hesaplamaya en içteki parantezden başlanır.
2. İlk önce * , / ve % işlemleri daha sonra + ve - işlemleri yapılır.Öncelik sırası aynı olan işlemlerde hesaplama soldan sağa doğru yapılır.

Örnek:

a) a = 5;

```
a = a + 1;
printf("a = %d", a);
```

b) a =5;

```
a++;
printf("a = %d", a);
```

c) a =5;

```
b = -13;
a++; b--;
a = a - b;
b = a * b;
printf("a=%d b=%d",a,b);
```

- C dilinde tamsayı değerini ekrana yazdırmak için %d kullanılıyor.

```c
#include <stdio.h>

int main()
{
int a;
a=10;
printf("a sayisinin degeri %d\n",a);
getchar();
return 0;

}
```

SORU -1

- Kullanıcının klavyeden gireceği 2 sayının toplamını bulan programı yazın.

```
#include<stdio.h>
#include<conio.h>
main()
{   int a,b;
    printf("1.sayiyi giriniz: ");
    scanf("%d",&a);
    printf("2.sayiyi giriniz: ");
    scanf("%d",&b);
    printf("Sonuc= %d",a+b);
    getch();
}
```

- scanf() fonksiyonu klavyeden veri okumak için kullanılan fonksiyondur. Klavyeden yazılan değeri bir değişkene atar.

SORU -2

- Kullanıcının klavyeden gireceği 3 sayının çarpımını bulan programı yazın.

```
#include<stdio.h>
#include<conio.h>
main()
{   int a,b,c;
    printf("1.sayiyi giriniz: ");
    scanf("%d",&a);
    printf("2.sayiyi giriniz: ");
    scanf("%d",&b);
    printf("3.sayiyi giriniz: ");
    scanf("%d",&c);
    printf("Sonuc= %d",a*b*c);
    getch();
}
```

SORU - 3

- Kullanıcının klavyeden gireceği 3 sayının ortalamasını bulan programı yazın.

```
#include<stdio.h>
int main( void )
{
    float sayi1,sayi2,sayi3,ortalama;
    printf("Uc sayi giriniz> ");
    scanf("%f%f%f",&sayi1,&sayi2,&sayi3);
    ortalama = ( sayi1 + sayi2 +sayi3  ) / 3;
    printf("Ortalama sonucu: %f'dir",ortalama);
    getch();
    return 0;
}
```

SORU - 4

- Kullanıcının klavyeden girdiği tek bir karakterin ne olduğunu ekrana yazan program .

```
#include<stdio.h>
int main( void )
{
    char karekter;
    printf("Klavyeden bir harf giriniz. ");
    scanf("%c",&karekter);
    printf("Klavyeden girdiğiniz karekter: %c'dir",karekter);
    getch();
    return 0;
}
```

NOT :Bir değişkende tek bir karekter saklamak istediğimizde değişken türünü char olarak belirtebiliriz.

SORU - 5

- Kullanıcının klavyeden girdiği tek bir karakterin ne olduğunu ekrana yazan program .

```c
#include<stdio.h>
int main( void )
{
    char karekter;
    printf("Klavyeden bir harf giriniz. ");
    scanf("%c",&karekter);
    printf("Klavyeden girdiğiniz karekter: %c'dir",karekter);
    getch();
    return 0;
}
```

İf – else Deyiminin Kullanımı

`if` deyiminin `else` ile birlikte kullanımı şu şekildedir:

```
if(koşul){
   ...
   deyimler; (küme1)
   ...
}

else{
   ...
   deyimler; (küme2)
   ...
}
```

Switch-Case Yapısı Kullanımı

- Bu deyim bir *değişken*in içeriğine bakarak, programın akışını bir çok seçenekten birine yönlendirir. case (durum) deyiminden sonra değişkenin durumu belirlenir ve takip eden gelen satırlar (deyimler) işleme konur. Bütün durumların aksi söz konu olduğunda gerçekleştirilmesi istenen deyimler default deyiminden sonraki kısımda bildirilir

Switch-Case Yapısı Kullanımı

```
switch(değişken)
{
      case sabit1:
          ...
          deyimler;
          ...
      case sabit2:
          ...
          deyimler;
          ...
      .
      .
      .
      case sabitn:
          ...
          deyimler;
          ...
      default:
          ...
          hata deyimleri veya varsayılan deyimler;
          ...
}
```

Döngüler

- Döngü (loop) deyimleri, bir kümenin belli bir koşul altında tekrar edilmesi için kullanılır. C programlama dilinde, while, do...while ve for olmak üzere üç tip döngü deyimi vardır. Diğer programlama dillerinde olduğu gibi, bu deyimlerle istenildiği kadar iç-içe döngü yapısı kullanılabilir.

For Döngüsü

- Bu deyim, diğer döngü deyimleri gibi bir kümeyi bir çok kez tekrarlamak için kullanılır. Koşul sınaması while da olduğu gibi döngüye girmeden yapılır. Bu döngü deyimin içinde diğerlerinden farklı olarak başlangıç değeri ve döngü sayacına sahip olmasıdır.

```
for( başlangıç ; koşul ; artım )
{
 ...
  döngüdeki deyimler;
 ...
}
```

While Döngüsü

Tekrarlama deyimidir. Bir küme ya da deyim while kullanılarak bir çok kez yinelenebilir. Yinelenmesi için koşul sınaması döngüye girilmeden yapılır. Koşul olumlu olduğu sürece çevrim yinelenir.

Do – while döngüsü

Bu deyim while deyiminden farkı, koşulun döngü sonrasında sınanmasıdır. Yani koşul sınanmadan döngüye girilir ve döngü kümesi en az bir kez yürütülür. Koşul olumsuz ise döngüden sonraki satıra geçilir.

```
do{
        .....
        döngüdeki deyimler;
        .....
}while(koşul);
```