Programlama: C

Doç. Dr. Alpaslan Duysak

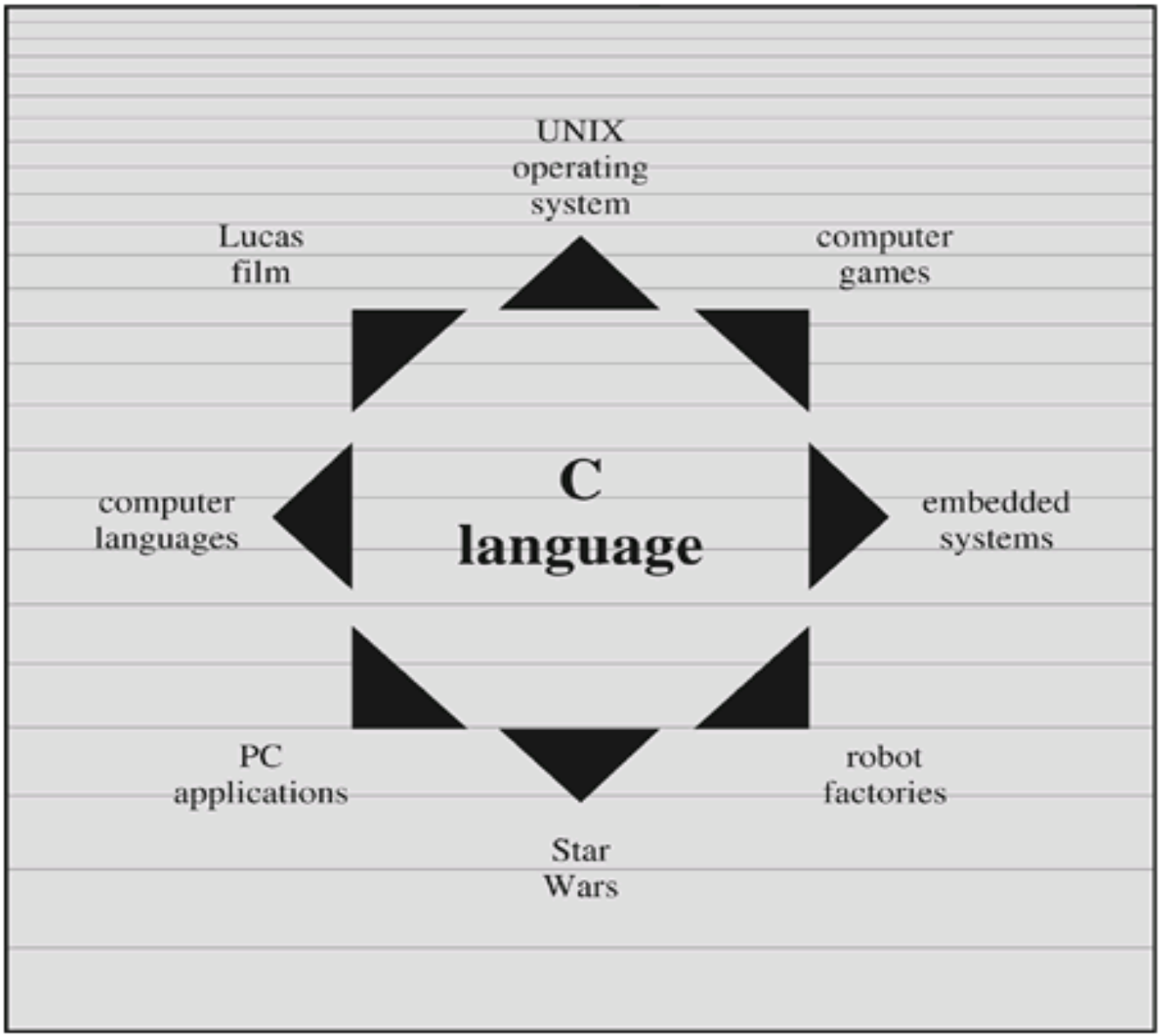
Neden C

- **Tasarım Özellikleri:** Yukarıdan-aşağıya (Top-down) tasarım, yapısal, nesnel programlama, güvenilir ve anlaşılabilir programlar.
- **Verimlilik:** CPU teknolojisinin avantajlarını iyi kullanır dolayısı ile hızlı çalışır. Program maksimum hız ve optimum hafıza için dizayn edilebilir.
- **Taşınabilirlilik:** C taşınabilir bir dildir, yani bir sistemde yazılan program hiç değiştirilmeden veya çok az değişikliklerle diğer sistemlerde de çalışır. Değişiklikler genellikle **header** file ile sınırlıdır. (IBM PC BASİC dilini Apple BASİC' e çevirmek veya IBM Fortran'ı Unix Fortrana çevirmek). C komutlarını bilgisayarın anladığı emirlere çeviren programlar 40 farklı sistem için C derleyiciler içinde vardır.
- **Güç ve Esneklik:** Unix işletim sisteminin çoğu C ile yazılmıştır. Birçok derleyici ve çevirici (compilers and interpreters), FORTRAN, Perl, Python, Pascal, LISP, Logo, and BASIC, C ile yazılmıştır.

Neden C

- **Programcı için Kullanışlı:** Donanıma müdahale etmenize olanak verir, hafıza üzerinde tam kontrol sağlar (bit operasyonlarına imkan verir) Sınırları ve eksiklikleri diğer dillere göre daha azdır, bu özelliği avantajlarının yanında tehlikeli bile olabilir. Data üzerinde tehlikeli manipülasyonlar yapılabilir. İlave olarak C çok kullanılan bir çok fonksiyonu kütüphanesinde içerir.
- **Dezavantajları :** Bazı program yanlışlarını bulmak neredeyse imkansızdır (pointer ile yapılanlar). (özgürlüğün bedeli). Takibi neredeyse imkansız programlar yazılabilir. (bu konuda yarışmalar düzenlenmektedir)
- 90 lı yıllardan itibaren özellikle büyük kullanıcılar C++ diline yöneldiler. C++ dili C nin bir üst modeli olarak tanımlanabilir. Herhangi bir C programı (hemen hemen) aynı zamanda bir C++ programıdır. Dolayısı ile siz C öğrendiğinizde aslınsa C++ da öğreniyorsunuz demektir. C günümüzde bazı alanlarda çok popüler hale gelmiştir: microcontroller programming, DVD ve bilimsel programlar…

Nerelerde kullanılır



Yüksek Seviyeli Diller ve Derleyiciler

- C gibi yüksek seviyeli diller programlamayı bir çok yönden kolaylaştırırlar: örneğin emirleri sayısal olarak ifade etmek zorunda değiliz. Emirler bilgisayarın anlayacağı formdan ziyade programcını anlayacağı şekilde ifade edilir. Bilgisayarın (CPU) izleyeceği adımlar, problem çözümü, bizim için önemli değil, biz problemi anladığımız şekilde kod olarak yazarız.

  Toplam =a+b

  Şimdi yukarıdaki işlemin assembly dili karşılığını düşünün. Ancak bilgisayar için de tam tersi geçerli. Yukarıdaki emir O'nun için anlaşılmaz bir şeydir. Tıpkı aşağıdakinin bizim için olduğu gibi

  11001111100110100101010010101010010 10

  İşte burada Compiler denilen derleyiciler devreye girer: Derleyiciler yüksek seviyeli dildeki komutları makine diline çevirirler

C’nin kullanımı

- 7 basamak yaklaşımı
  - 1: **Program Amaçlarını Tanımlamak:** programın ne yapmasını istiyoruz, hangi verilere ihtiyaç var, ne tür hesaplamalar kullanılacak ve program hangi çıktıları verecek.
  - 2: **Program Tasarımı:** problem tanımlandıktan sonra, programın problemi nasıl çözeceği planlanmalı, kullanıcı ara birimi nasıl olmalı, program nasıl organize olmalı, potansiyel kullanıcılar kim, veri nasıl kullanılmalı, hangi metotlar kullanılmalı…
  - 3: **kodun yazılması:** problemin çözümü bilindiğine göre yazımı başlayabilir. Kaynak kod oluşturulur.
  - 4: **Compile:** Kaynak kod derlenir (compile) edilir. Compiler kaynak kodunu executable koda çevirir. Executable kod, kullandığınız sistem için makine dilidir. Derleyici C dilindeki kütüphane fonksiyonlarını koda ekler. Derleyici ek olarak programın geçerli bir C dili olup olmadığını kontrol eder.

C’nin kullanımı

- 5: **programın Çalıştırılması:** Çoğu durumda, compilerın ürettiği executable kod sadece program ismi yazılarak çalıştırılabilir.
- 6: **Programın testi ve Hataların Giderilmesi:** program çalışabilir fakat bu hatasız olduğu anlamına gelmez. İstenilen bazı durumlar için hatalar olabilir. Bunlar bulunur ve düzeltilir. Bu işleme Debugging denir. Hatalara Bugs denir. Bu hatalardan bazıları; tasarım hatası, veri manipülasyon hataları, C dilinin kullanımından gelenler, yazım hataları…..
- 7: **Programın Bakımı ve Geliştirilmesi:** programların genellikle güncellenmesi, bulunan hataların düzeltilmesi, yeni durumlar veya veriler için uyarlanması ve geliştirilmesi gerekebilir.

Programlama Makineleri

- C programı oluşturmanın yolu hangi sistemi kullandığınıza göre değişir. C taşınabilir olduğundan birçok ortam için hazırlanmıştır; Unix, Linux, Windows, Machintos… C programının text dosyasında yazılması ve kaydedilmesi gerekir. “Topla.c” formatında bir dosya oluşturulur. Oluşturulan bu dosya, program, compile ve link edilir. Compiler kodu önce arakod (intermediate kod) çevirir, linker diğer kodlar ile ara-kodu birleştirir ve executable, çalıştırılabilir kod üretilir. Her bir program (kaynak kod) ayrı ayrı compile edilip linker ile sonradan birleştirilebilir. Bu bir avantajdır; herhangi bir modül de değişiklik gerekirse diğer modüller bundan etkilenmezler. Linker C kütüphane kodlarını ana programa ekler.
- Ara-kod oluşturmanın değişik yolları vardır. En çok kullanılan metot kaynak kodu makine diline çevirmek ve sonucu object kod olarak saklamaktır. Object kod makine dilinde olmasına karşın çalışmaya hazır bir program değildir. Object file den eksik olan startup koddur.

Programlama Makineleri

- Startup kod yazılan C programı ile işletim sistemi arasında bir arabirim olarak çalışır. Örnek olarak IBM PC programı DOS ortamında veya Linux altında çalışır. Çünkü donanım aynıdır ve aynı object kod hepsinde çalışır ve fakat farklı startup programlarına ihtiyaç vardır.
- İkinci eksik nokta kütüphane dosyalarını manipüle eden kodlardır. Bütün C programları fonksiyonlar denilen kütüphane dosyalarını kullanırlar. Kütüphane dosyaları object kod halindedirler. Linker'in rolü bu üç elemanı bir araya getirmektir

  sizin yazdığınız

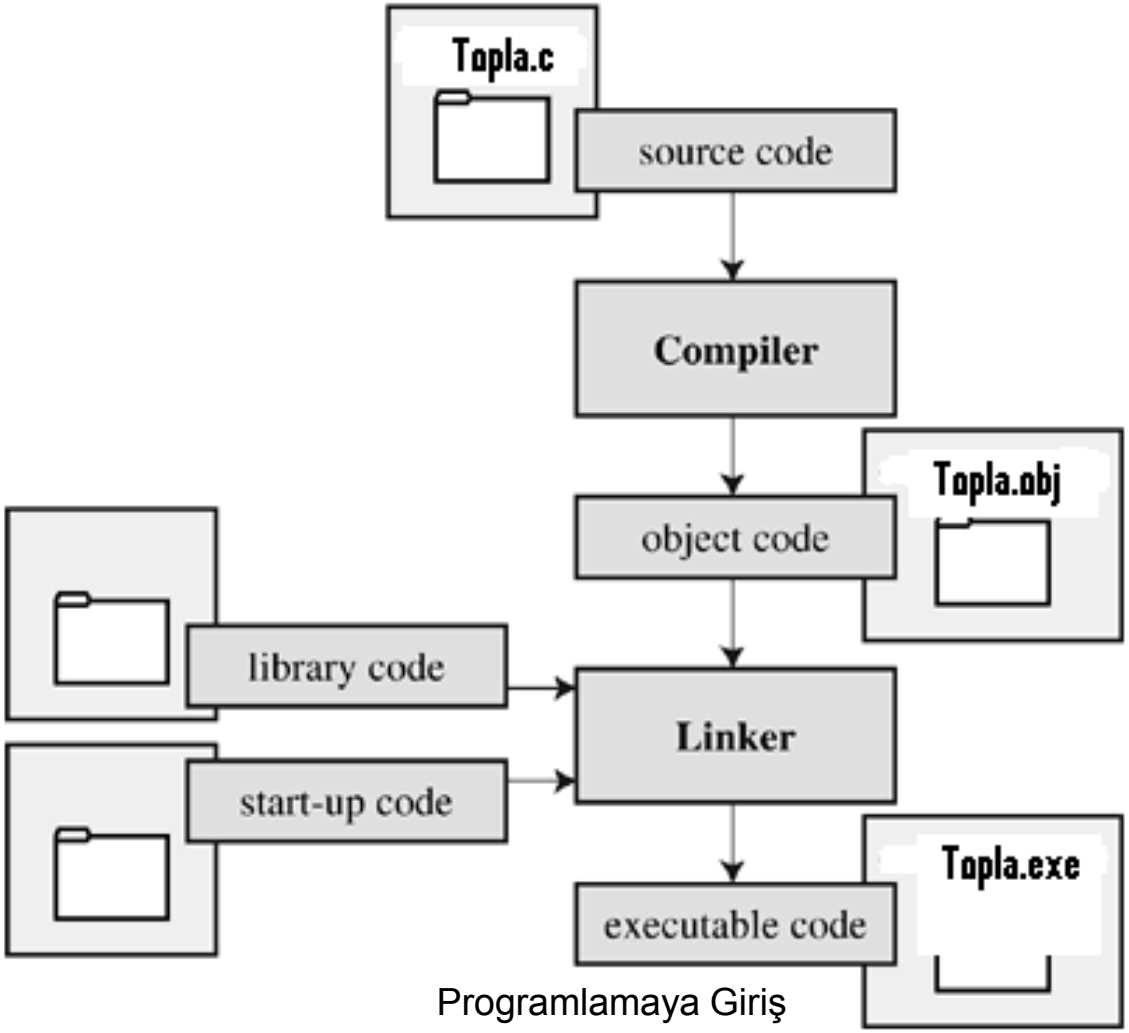
  compile ederek elde ettiğiniz object kod,

  kullandığınız sistem için startup kod,

  kütüphane kodları

Bunları bir tek file haline getirir: Executable file.

Programlama Makinaları



Programlama Makineleri

- Her iki file, Object ve Executable, makine dili komutlarını içerir fakat Object file sadece sizin yazdığınız kodların makine diline çevrilmiş hallerini içerir. Executable file sizin yazdıklarına ek olarak, kütüphane fonksiyonlarını ve starup kodlarını da içerir.
- Bazı sistemlerde siz compile ve link işlerine ayrı ayrı yapmak durumundasınız, bazı diğer sistemlerde ise compiler linkeri otomatik olarak çalıştırır.
- UNIX System:

UNIX sisteminin kendi C editörü yoktur siz genel amaçlı UNIX editörlerinden birini kullanmak durumundasınız, bunlar: emacs, jove, vi, veya X Window System text editor . Sonra dosyanızı uzantısı ".c" olacak şekilde bir ad vererek saklarsınız. Programınızı Compile etmek için;

cc Topla.c

Yazmanız yeterli. UNIX C compileri cc olarak adlandırılmıştır.

Programlama Makineleri

- UNIX executable file olarak “a.out” üretir. Programınızı çalıştırmak için

a.out

Yazmanız yeterlidir. Object kod ise “Topla.o” olarak üretilir.

LINUX Systems: LINUX de C programı yazmak ve çalıştırmak bu işi UNIX de yapmak gibidir, önemli farkı C Compileri olarak “gcc” kulanırsınız.

gcc Topla.c

WINDOWS: C Compiler standart windows paketlerinden biri değildir ve siz yüklemek durumundasınız. Günümüzde birçok firma Windows için C Compiler üretmektedir: Microsoft, Borland, Metrowerks, ve Digital Mars. Bunların en önemli özelliği ise kendi C veya C++ editörlerinin olmasıdır. Bunların menüleri vardır programınızı isimlendirmek, kaydetmek, debug yapmak veya bulunduğunuz ortamı terk etmeden compile etmek ve çalıştırmak olanağınız vardır.

Biraz da Eğlenin

- 1: C programlamada “Portability” ne demektir
- 2: kaynak file (source file), object ve executable file arasındaki farkları belirtiniz
- 3: Compiler ne iş yapar
- 4: Linker ne iş yapar

Örnek C Programı

Aşğıda basit bir C programı verilmiştir:

```
#include <stdio.h>
 int main(void)                          /* basit bir program */
 {
 int num;                                /*num adlı bir degisken tanımla */
 num = 1;                                /*num degiskenine bir deger ata*/
 printf("Ben basit bir ");               /* printf() fonksiyonunu kullan*/
 printf("Bilgisayarim.\n");
 printf("Benim favori numaram %d O bir ilk.\n",num);
 return 0;
 }
```

Örnek C Programı

- Verilen C programının her bir satırına yakından bakalım:

#include <stdio.h>

Bu satır compiler'a stdio.h dosyasında ne bilgiler varsa programa ekle demektir. stdio.h dosyası standart bir **C paketidir ve klavye ve monitör bilgilerini içerir.** Dolayısı ile yazdığınız C Programı klavyeyi veya monitörü kullanacaksa bu dosyayı yukarda verilen şekilde programınıza eklemelisiniz.

int main(void)

C programı en az bir veya daha fazla fonksiyondan oluşur. Yukarıdaki program bir fonksiyondan oluşur: main(), bu fonksiyonun da ismidir. Yukarıdaki örnekte "int" main fonksiyonun integer (tam sayı) döndürdüğünü, sonuç verdiğini, belirtir. Parantez içinde kullanılan "void" ise fonksiyonun herhangi bir girişi, argümanı olmadığını belirtir.

Örnek C Programı

/* ..... */

Yukardaki format C programında kullanıcı açıklamalarının program içersine yazılmasına olanak veren kodlardır. C Programı /* başlayarak */ ulaşıncaya kadar yazılmış her şeyi açıklama kabul eder ve compile etmez.

{

Fonksiyonun başlangıcını ifade eder, gösterir.

int num;

Bu ifade bir değişken tanımladığınızı (num) ve bu değişkenin türünün int (tam sayı) olduğunu belirtir.

num = 1;

Num adlı değişkene değer atama. Num değişkeni şimdi 1 değerini almıştır.

Örnek C Programı

printf(“Ben basit bir ");

“printf()” standart bir C kütüphane fonksiyonudur. Yukarıdaki fonksiyon ekrana “Ben basit bir” ifadesini yazar ve kursörü aynı satırda bırakır.

printf("Bilgisayarim.\n");

Önceki printf() fonsiyonunun kaldığı yerden “Bilgisayarım” ifadesini yazar. Sondaki “\n” ifadesi kursörü yeni bir satıra götürür.

printf("Benim favori numaram %d O bir ilk.\n",num);

num değerini yazar, bu değer “1”. Burada kullanılan “%d” komutu num değerinin nereye ve hangi formatta yazılacağını belirtir.

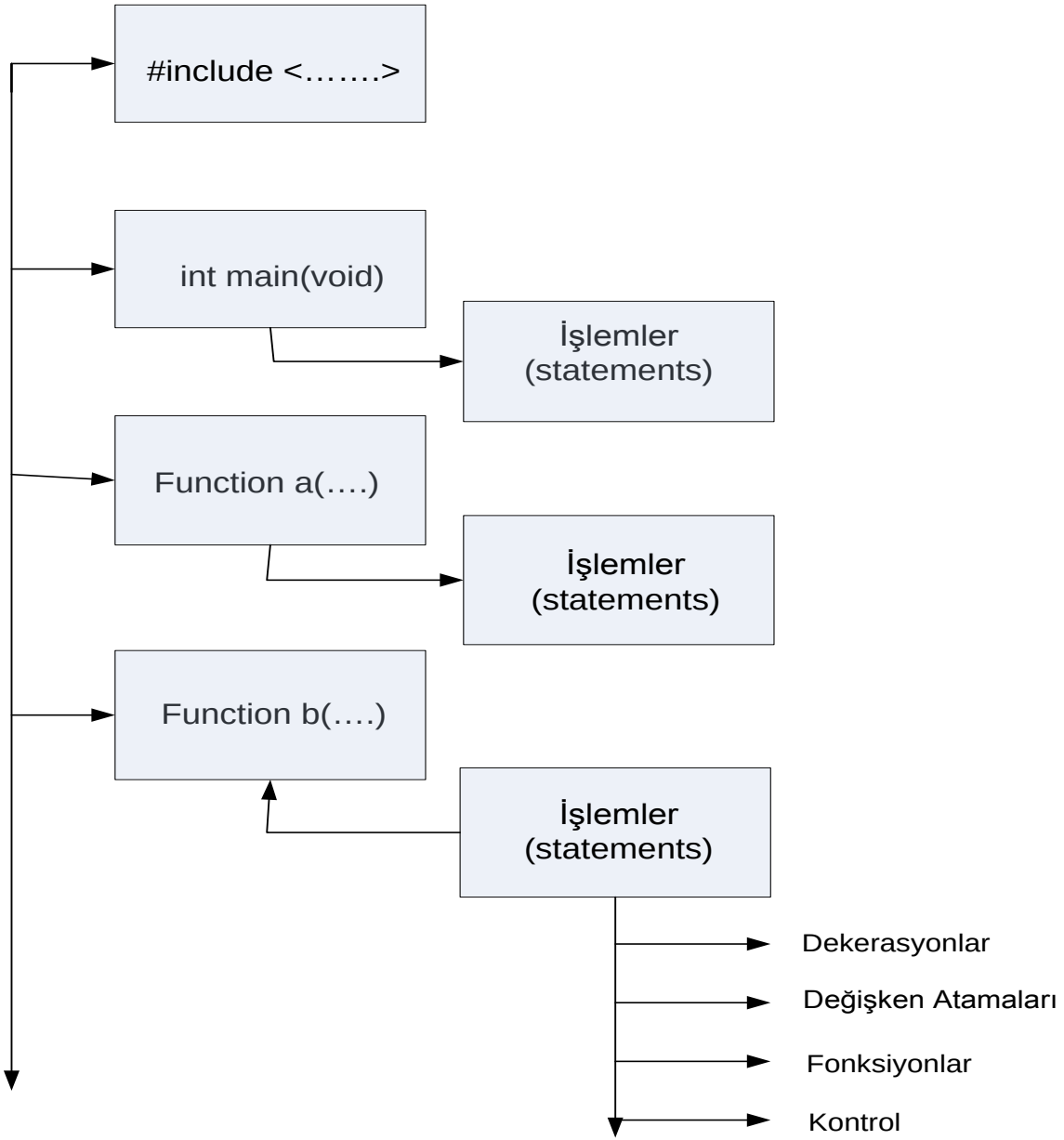
return 0;

C programı işlem sonunda kullanıcıya sonucu iletir. Bu C programının standart bir rutinidir.

}

Program, fonksiyon, sonu.

C Programının Detayları



C Programının Detayları

#include <stdio.h> /*Standart input/output header*/

Ortak programları paylaşmanın çok kestirme bir yoludur. stdio.h İçindeki programları sizin yazdığınız programa ekler. Bu program bir preprocessor kodudur: C compiler'i yazdığınız kodu compile etmeden önce bazı hazırlıklar yapar, bu işleme preprocessing denir. Header file içindeki bilgiler (kod) derleme sırasında sizin yazdığınız kod ile birleştirip executable kod oluşturulur.

Header file içinde değişken tanımlamaları ve atamaları ve fonksiyon isimleri ve nasıl kullanılacaklarını içerir. Gerçek fonksiyonlar kütüphane dosyaları içindedirler (precompiled files). Header file bu fonksiyonların yerlerini ve nasıl kullanılacaklarını compiler'a iletir ve çalışan bir kod üretilmesinde compiler'a yardım eder.

İnt main(): C programı çalışmaya daima main() fonksiyonu ile başlar. Diğer fonksiyonlara istediğiniz ismi verebilirsini, fakat C programı işlemlere daima main() fonksiyonu ile başlar. Parantezler bunun bir

C Programının Detayları

fonksiyon olduğunu belirtir. Parantez içine fonksiyon argümanları, parametreleri, konur. Bunlar fonksiyona giren ve fonksiyondan gelen değişkenler ve sabitlerdir. Başındaki **int** fonksiyonun geri döndürdüğü değerin türünü ifade eder, integer= tam sayı. Nereye geri döndürdüğü… İşletim sistemine. Örnekteki **main(void)** fonksiyonu void=hiç bir şey, değeri almaktadır. Bazı compiler lar **main()** biçiminde kullanıma izin verirler.

/*…….*/: program içine açıklayıcı bilgiler ve görüşlerinizi yazmak istediğinizde verilen format uygun yorum satırlarını program içine ekleyebilirsiniz. Kullanımı;

```
/*Buraya istediğinizi yazın…….*/
/*
buraya da
İstediğinizi
Yazın
*/
```

C Programının Detayları

Yorum aşağıdaki formatta da yazılabilir;

//yorumlarınız….

int main () // program başı

Deklerasyonlar: örneğimizde **(int num)** iki önemli noktayı deklere eder. Birincisi fonksiyonun içinde bir değişken tanımladığımızı compiler a bildirir. İkincisi bu değişkenin türünün integer yani tam sayı olduğunu belirtir. Compiler bu bilgileri alarak hafızada bu değişken için uygun bir yer ayarlar.

Deklerasyonlar **“;”** ile son bulurlar bu bir C kuralıdır.

Burada int bir C tanımı (keyword) olarak verilir. Keyword bir C tanımı olarak belirlenmiştir ve biz bunları istediğimiz şekilde kullanamayız. Örneğin **int** kelimesini bir fonksiyon ismi olarak veya bir değişken olarak kullanmayız. Örneğimizde num bizim değişkenimiz için bir tanımlayıcıdır, ismidir. Deklerayon sonunda, belli bir tanımlayıcı (değişken ismi) hafızada belli bir yer ile irtibatlandırılır ve belli tür data, veri, burada saklanır. **C de bütün değişkenler kullanılmadan önce mutlaka tanımlanmalıdırlar.**

C Programının Detayları

Dolayısı ile programda kullandığınız bütün değişkenler, türleri ve değerleriyle bir liste olarak programa verilmelidir. C de genel kullanım bütün değişkenler program gövdesinden önce tanımlanırlar;

```
int main()
{
        int kapı;
        int kalem;
        kapı=5;
        kalem=100000;
        // diğer tanımlama ve fonksiyonlar.
        ……..
}
```

Ancak siz deklerasyonu ve değer atamayı programın istediğiniz anında yapabilirsiniz, o nu kullanmada önce.

C Programının Detayları

Data Türü: C dilinde kullanılan birçok veri türü tanımı vardır, örnek olarak integer, floating point,charecters. Bilgisayar belirtilen türe göre bilgiyi saklar ve işler.

İsim seçimi: Değişkenler için en önemli özellik mantıklı isim kullanmaktır. Örneğin koyun sayımı yapan bir program için

int Koyun_sayısı;

İyi programlama için, eğer isim yeterince açık değilse ***//***açıklayınız.

İsim 63 karaktere kadar olabilir. Bazı durumlarda (hafıza işlemleri) bu sayı 31 karakter olarak belirtilmiştir. 63 karakterden sonrasını bilgisayar tanımaz.

Kullanılabilen karakterler: küçük-büyük harfler, sayılar ve underskore (_). İlk karakter bir harf veya undeskore olmalıdır.

Doğru isimler: alpaslan_duysak, masa2, oranTı…

Kullanımı yanlış olanlar: %alp, 2masa, alpaslan-duysak

C Programının Detayları

Değişken isimleri atanırken ilk karakter olarak tek veya çift underskore kullanmamaya özen gösterelim. İşletim sistemleri ve C kütüphaneleri bu tür kullanımları çokça yaptığından, bu bir hata değil fakat isim karışmaları olabilir. C dilinde büyük/küçük harf ayırımı vardır.

C Programının Formatı

```
#include <stdio.h>
int main(void)
    {
        statements
        return 0;
    }
```

C Programının Formatı

```
#include <stdio.h>
 int main(void)
 {
 int num1;                    // birinci sayının deklerasyonu
 int num2;                    // ikinci sayının deklerasyonu
 int carpim;                  // sonuc değişkeni deklerasyonu
    printf("Birinci sayiyi giriniz ");      // sayı girişini sor
    scanf("%ld", &num1);                 // birinci sayıyı gir
    printf("Ikinci sayiyi giriniz ");       // sayı girişini sor
    scanf("%ld", &num2);                 // birinci sayıyı gir
          carpim=num1*num2;              // sayıları çarp
          printf("Sayilarinizin carpimi.....%d",carpim); //Yazdır
return 0;
 }
```

Hatalar: Debugging

Syntax Errors: C dilinin kurallarına uymazsanız syntax denilen hataları yaparsınız (konuşma dillerinde gramer hataları gibi)

```
#include <stdio.h>
int main(void)
 (
 int n, int n2, int n3;
 /* this program has several errors
 n = 5;
 n2 = n * n;
 n3 = n2 * n2;
printf("n = %d, n squared = %d, n cubed = %d\n",
    n, n2, n3)
return 0;
)
```

Hatalar: Debugging

Semantic Errors: Anlam, mana hatalarıdır. Bir sayının karesi yerine o sayının üçüncü kuvvetini alabilirsiniz örnek olarak. Bu tür hatalar bilgisayar tarafından bulunamazlar. Siz burada devreye giriyorsunuz. Bu hataları bulmanın bazı metotları vardır.

Programı manuel olarak adım adım çalıştırabilirsiniz ve sonuçları kontrol edersiniz.

Programda bazı yerlere printf() koyarak çıktıları kontrol edebilirsiniz.

Debugging bir tür programdır ve sizin yazdığınız kodları adım adım çalıştırarak sonuçları kontrol etmenize olanak verir.

Keywords

ISO/ANSI C Keywords			
auto	**enum**	*restrict*	unsigned
break	extern	return	**void**
case	float	short	**volatile**
char	for	**signed**	while
const	goto	sizeof	*_Bool*
continue	if	static	*_Complex*
default	*inline*	struct	*_Imaginary*
do	int	switch	
double	long	typedef	
else	register	union	

Biraz da eğlenin

1) Syntax ve semantic hataları tanımlayıp bu hataları içeren birer kısa örnek program yazınız
2) Aşağıdaki programdaki hataları bulunuz

```
include studio.h
int main{void} /* this prints the number of weeks in a year /*
(
    int s
    s := 56;
    print(Bir yılda s hafta vardır.);
     return 0;
```

3) Hangileri C dilinde keyword olarak tanımlanmışlardır:

main, int, function, char, =

Biraz da eğlenin

4) 7., 8. ve 9. satırlardan sonra program sonucu ne olur?

```
#include <stdio.h>
 int main(void)
 {
 int a, b;
 a = 5;
 b = 2;          /* line 7 */
 b = a;           /* line 8 */
 a = b;           /* line 9 */
 printf("%d %d\n", b, a);
 return 0;
 }
```

Veri Türleri (data types)

- Bilgisayara gerekli veriyi (data) girerseniz ve yapılması gerekenleri tam tarif ederseniz (program) sizin için aklınıza gelebilecek her şeyi yapar.
- Günlük hayatta kullanılan birçok farklı tür veri (sayılar, karakterler…) bilgisayar için tanımlanmalıdır.
- Bilgisayar ortamında datalar sabitler ve değişkenler içinde saklanırlar ve bu değişkenler data türleri ile tanımlanırlar.
- Sabitleri biliyoruz

  42, 35.200, 10000
- Değişkenin türü değişken tanımlanırken belirlenir.

  int num
- Data türü değişkenin ne tür bilgi depoladığını, bu bilgiyi depolamak için ne kadar hafıza gerekli olduğunu ve veri üzerinde ne tür operasyonlar yapılabileceğini belirtir. (Örneğin datanız karakter ise çarpma işlemi gerçekleştiremezsiniz)

Data Types

Original K&R Keywords	C90 Keywords	C99 Keywords
int	signed	_Bool
long	void	_Complex
short		_Imaginary
unsigned		
char		
float		
double		

Data Types

- **İnt** C dilinin temel veri türüdür ve tam sayıları ifade eder. Diğerleri
  long,
  short,
  unsigned,
  signed

int türünün farklı şekillerde temsil edilmesini sağlar.

- **char** alfabedeki harflerin, bazı işaret ve simgelerin temsil edilmesinde kullanılır. Ek olarak küçük tam sayılar için de kullanılabilir. (niye?)

- **float, double, long double** türleri tam olmayan (kesirli) sayılar için kullanılır.
- **_Bool** türü Boolen aritmatiği için kullanılır (True-False)
- **_Complex** ve **_Imaginary** yenidir ve complex ve imagenery sayılar (sanal ve reel kısımları olan) için kullanılır.

int

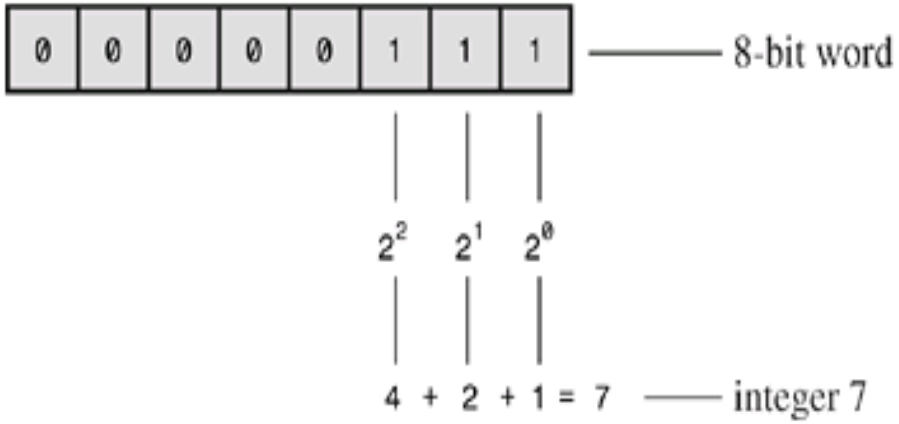
- Tamsayılar **int** ile tanımlanır. Sayı negatif, pozitif veya sıfır olabilir.

  -23 11 112 980

- Bilgisayarda tam sayılar binary olarak depolanırlar.

  Örnek: 8-bit sistemde (word=8) 7 rakamı nasıl depolanır?

  7=111

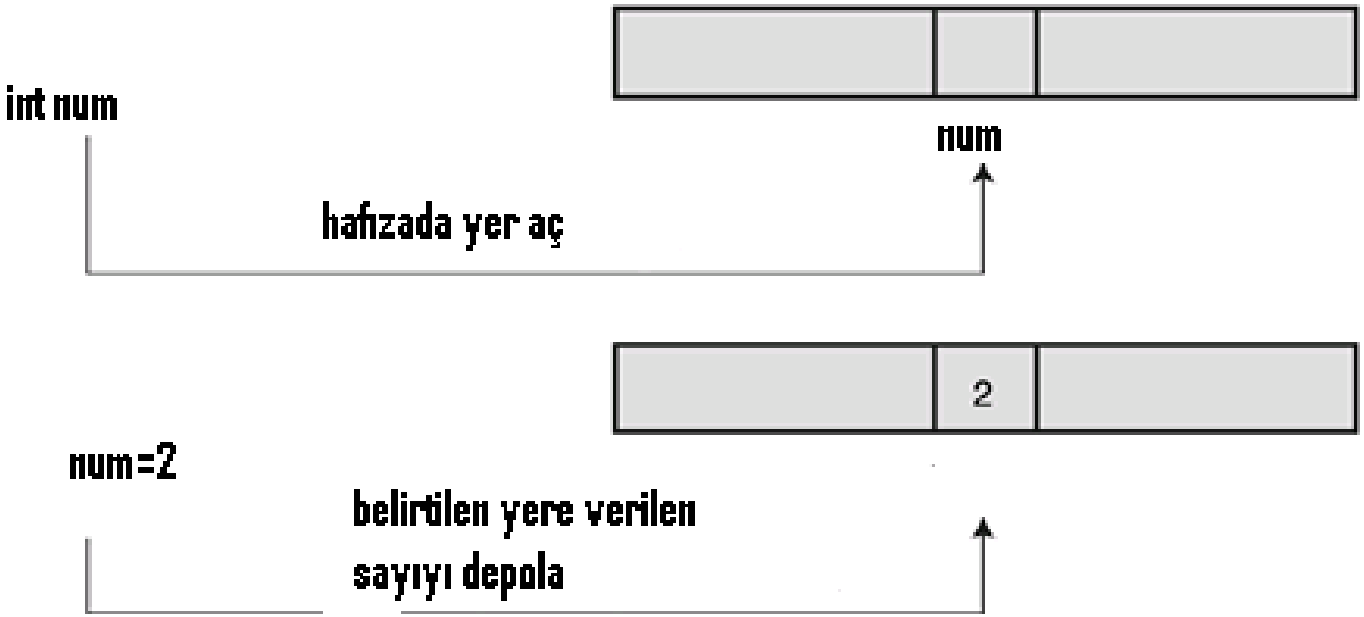


int

- Depolama bit olarak kullanılan makine donanımın bağlıdır, makinanın kullandığı word. IMB PC türü makinalar için 16-bit word kullananlar **int** depolamak için 16 bit kullanırlar. (–32768 to 32767). Günümüz bilgisayarları 32/64 bit word' e sahip ve **int** türü veriler 32764 bit ile temsil edilir.

Type	Macintosh Metrowerks CW (Default)	Linux on a PC	IBM PC Windows XP and Windows NT	ANSI C Minimum
`char`	8	8	8	8
`int`	32	32	32	16
`short`	16	16	16	16
`long`	32	32	32	32
`long long`	64	64	64	64

İnt ve değer ataması



İnt ve printf()

- **Printf()** fonksiyonu kullanılarak **int** değerleri yazdırılabilir. **%d** (format specifier) format ayarlayıcısı; **int** değerinin satırda nereye yazılacağını belirtir. Her bir **%d** değeri kadar değişken (veya **int** sabiti) **printf()** fonksiyonu içinde olmalıdır.

```
#include <stdio.h>
 int main(void)
 {
 int on = 10;
 int iki = 2;
 printf(“dogru kullanim: ");
 printf("%d eksi %d = %d\n", on, 2, on - iki );
 printf(“yanlıs kullanim: ");
printf("%d eksi %d = %d\n", on );          // iki argüman unutuldu
return 0;
 }
```

Octal and Hexadecimal

- Farklı tabanlı sayı sitemleri kullanılabilir: 8 (octal) ve 16 (hexadecimal). İkilik sistemle 8 ve 16 sistemler arasında bağ kurmak daha kolaydır. Bu sayı sistemlerini bilgisayara nasıl anlatırız?
- **0x** veya **0X** hexadecimal sayı girildiğini ifade eder. Ör:

  16=0X10
- **0** ise octal kullanımı ifade eder.Ör:

  16=020
- Nasıl yazılırsa yazılsın **int** sayıların depolanması aynıdır.
- Gösterimleri:

  Octal **%o**

  Hexadecimal **%x**

Octal and Hexadecimal

- Eğer C prefix' i görmek istiyorsanız: %#o, %#x, ve %#X gösterimi: **0, 0x, and 0X** .

```
#include <stdio.h>
 int main(void)
 {
 int x = 100;
printf("dec = %d; octal = %o; hex = %x\n", x, x, x);
printf("dec = %d; octal = %#o; hex = %#x\n", x, x, x);
return 0;
}
```

Diğer **int** türleri

- C dili 3 sıfat kullanarak 3 ayrı **int** türü daha üretmiştir: short, long, ve unsigned.
- **Short int**: küçük sayılar ile kullanılır ve **int** e göre daha az yer kaplar. Aynen **int** gibi negatif veya pazitif (işaret) olabilir.
- **Long int:** büyük sayılar için kullanışlıdır, daha fazla yer kaplar, işaretlidir.
- **Long long int**: daha da büyük sayılar içindir ve daha fazla yer kaplar.
- **Unsigned int:** negtif olmayan sayılar içindir. 0-65535 arası ayıları kapsar (**int** -32768 ile 32767).
- **unsigned long int, unsigned long, unsigned short int, unsigned short, unsigned long long int, ve unsigned long long.**
- Signed sıfatı açık olarak kullanılabilir: **short int---signed short int**

Niye farklı İnteger Türleri?

- IBM PC, Windows 3.1: **int** ve **short** her ikisi de aynı 16 bit ve **long** 32 bit.
- Windows XP

  short 16 bit

  int ve **long** 32 bit
- IBM Itanium, AMD Opteron, Power PC 65

  long long kullanılabilir 64 bit

Günümüzde en çok kullanılan: long long=64 bit

long = 32 bit

short = 16 bit

int = 16 veya 32 bit

Sınırlar

- **short ve int** : –32,767 ile 32,767, (16-bit unit),
- **long** : –2,147,483,647 ile 2,147,483,647, (32-bit unit).
- **unsigned short ve unsigned int** : 0 ile 65,535,
- **unsigned long** : 0 ile 4,294,967,295.
- **long long type** : –9,223,372,036,854,775,807 ile 9,223,372,036,854,775,807,
- **unsigned long long** : 0 ile 18,446,744,073,709,551,615. (eighteen quintillion, four hundred and forty-six quadrillion, seven hundred forty-four trillion, seventy-three billion, seven hundred nine million, five hundred fifty-one thousand, six hundred fifteen)

Sınırlar

- **İnt** kullanımında ilk önce **unsigned** kullanmaya çalışın daha büyük pozitif rakamlara ulaşırsınız
- Gerekli değilse **long** kullanmayın, makinayı yavaşlatır.
- **Long** ve **int** aynı uzunlukta olan makine ile çalışıyorsanız, **long** kullanın. (16 bit makinaya geçtiğinizde problem olmaz)
- 64 bit gerekliyse **long long** kullanın, yakın gelecekte bu çok yaygın kullanılacak KULLANILIYORRR
- Daha az hafıza kullanmak istiyorsanız, **short** kullanın. Bazı durumlarda **short** register uzunluğuna karşılık gelir ve **short** kullanmak avantaj olabilir (hesaplamları registerlarda yaptırıyorsunuz)

Sınırlar aşılırsa ne olur?

```
#include <stdio.h>
 int main(void)
{
 int i = 2147483647;
unsigned int j = 4294967295;
printf("%d %d %d\n", i, i+1, i+2);
 printf("%u %u %u\n", j, j+1, j+2);
return 0;
 }
```

- İnt bir arabanın km kadranına benzetilebilir: 100.000 km yi aştı mı, tekrar sıfırdan başlar. İnt ise – minimum değerinden başlar

Long, long long int

- Normal olarak sayılar, örneğin 2345, **int** olarak saklnır, eğer sayı **int** sınırları dışında ise bilgisayr sayının **long int** olduğunu varsayar.
- Eğer sayı **long int** türünün maximum değerinden fazla ise bilgisayar bu sefer **usigned long int** olarak saklar. Dahada büyük ise **long long int** veya **unsigned long long int** olarak saklar.

- Bazen küçük bir sayıyı büyük bir sayıymış gibi depolamak gerekebilir. O durumlarda sayının sonuna “l” veya “L” konur: 16 bit **int** ve 32 bit **long** olan bir sistemde, 7=16 bit ve 7L=32 bit olarak temsil edilir.
- **Long long int** ise 7LL ile
- **unsigned long long int** 7ull

Printing short, long, long long,and unsigned Types

- Unsigned int: %u
- Long int: %ld
- Long integer (hexedecimal) : %lx, (octal): %lo
- Short: %h…(%hd=short int)

char

- Harfleri ve sembolleri depolamak için kullanılır. **Char** aslında bir integer türü olarak depolanır (**char** integer depolar) Bilgisayar karekterleri belirlenmiş integerlar ile temsil eder. ASCII. Örneğin A=65
- **Char** =bit
- Unicode: 96000 karakter.
- Deklerasyon:

char alp='A' // doğru kullanım

char alp=A // A'yı bir değişken zanneder.

char alp="A" // A'yı bir kelime zanneder.

Diğer char kullanımları (kaçış kodları)

Kullanım	Anlam
\a	Alert (ANSI C).
\b	Backspace.
\f	Form feed.
\n	Newline.
\r	Carriage return.
\t	Horizontal tab.
\v	Vertical tab.
\\	Backslash (\).
\'	Single quote (').
\"	Double quote (").
\?	Question mark (?).
\Ooo	Octal value. (o represents an octal digit.)
\xhh	Hexadecimal value. (h represents a hexadecimal digit.)

			Tanım	ASCII No
'\0'	'\x0'	'\0'	NULL karakter	0
'\a'	'\x7'	'\07'	çan sesi (alert)	7
'\b'	'\x8'	'\010'	geri boşluk (back space)	8
'\t'	'\x9'	'\011'	tab karakteri (tab)	9
'\n'	'\xA'	'\012'	aşağı satır (new line)	10
'\v'	'\xB'	'\013'	düşey tab (vertical tab)	11
'\f'	'\xC'	'\014'	sayfa ileri (form feed)	12
'\r'	'\xD'	'\015'	satır başı (carriage return)	13
'\"'	'\x22'	'\042'	çift tırnak (double quote)	34
'\\'	'\x5C'	'\134'	ters bölü (back slash)	92

Print char

- **%c** kullanılır. **(%d** kullanımı **int** içindi)

```
#include <stdio.h>
 int main(void)
{
char ch;
 printf(“Lutfen bir karekter girin\n");
 scanf("%c", &ch); /* kullanici girisi */
printf(“char ve int %c ile %d.\n", ch, ch);
return 0;
 }
```

char

- Char işaretli bir data türü olarak tanımlanmıştır. -128 ile 127 arasında değerler alır. Compiler char türünü yeniden tanımlayıp 0-255 arasında değerler almasını sağlayabilir.
- Signed veya unsigned char ile beraber kullanılabilir. Compiler tanımı ne olursa olsun sizin tanımınız geçerli olur. Unsigned char artık sadece unsigned char olarak işlem görür.
- **-Bool türü**: true (1) vefa false (0) ifade etmek için tanımlanmıştır. 1 bit lik hafıza kullanır. Türü int
- İnt türü bu kadar mı? Tabi ki hayır.

int16_t = 16-bit signed integer

uint32_t = 32-bit unsigned integer.

float

- Kesirli sayılar, gerçek sayılar, float ile gösterilir.

  -12.56 1.00 500.1 980.980

- Float data türünü bilgisayarda depolamak int türü dataları depolamaktan farklıdır. Sayı önce parçalarına ayrılır ve tam ve kesirli kısım ayrı ayrı depolanır.
- C startlarına göre float kullanımda en azından 6 basamak hassasiyeti olmalı.

  22,123456

- Aralığı 10^{-37} to 10^{+37}. (çok yararlıdır, örneğin güneşin ağırlığı 2^{30} kg veya proton yükü $1,6*10^{-19}$ coulombs)
- Flot türü 32 bit kullanılarak depolanır. 24 bit sayıyı, 8 bir üssü ve işareti depolar.

float

Number	Scientific Notation	Exponential Notation
1,000,000,000	= 1.0×10^{9}	= 1.0e9
123,000	= 1.23×10^{5}	= 1.23e5
322.56	= 3.2256×10^{2}	= 3.2256e2
0.000056	= 5.6×10^{-5}	= 5.6e-5

float

- Flot türünün bir de double olanı vardır. Range aynı olmasına karşın hassasiyet 10 dijite çıkarılmıştır:

  22,1234567890

Bu haasasiyet genellikle 13 dijit olarak verilir. C minimum 10 dijiti garanti eder.

- 64 bit kullanır
- Deklerasyonu kolaydır:

```
float a;
a=12.35;
```

3.14159

.2

4e16

.8E-5

100.

float

- Compiler float hassasiyetini defult olark double kabül eder. Yani işlemlerinizi 64 bit üzerinden yapar ve saklar. Bu tabi ki hesap hassasiyeti için güzel fakat daha fazla yer kaplar ve yavaştır.
- Bu durumu düzeltebilirsiniz: float sayıların sonuna f veya F eklerseniz bilgisayar onları float alır ve işlemlerini o na göre yapar.

  0.7f

  1.35F
- Yazdırılmaları:
- Printf() fonksiyonu %f kullanır float ve double sayıları onluk tabanda yazmak için.
- Sayıların exponensiyel (üs) olarak yaılmasını istiyorsanız %e kullanın (bazı sistemlerde %a ve %A).

Float

```
#include <stdio.h>
 int main(void)
 {
 float a1 = 32000.0;
 double a2 = 2.14e9;
long double a3 = 5.32e-5;
printf("%f yazilabilir %e\n", a1, a1);
 printf("%f yazilabilir %e\n", a2, a2);
 printf("%f yazilabilir %e\n", a3, a3);
 return 0;
}
```

Float

- float toobig = 3.4E38 * 100.0f;
- printf("%e\n", toobig);
- Ne olur???

```
#include <stdio.h>
int main(void)
 {
 float a,b;
 b = 2.0e20 + 1.0;
 a = b - 2.0e20;
printf("%f \n", a);
 return 0;
 }
```

Aralıklar

Type	Macintosh Metrowerks CW (Default)	Linux on a PC	IBM PC Windows XP and Windows NT	ANSI C Minimum
`float`	6 digits	6 digits	6 digits	6 digits
	-37 to 38	-37 to 38	-37 to 38	-37 to 37
`double`	18 digits	15 digits	15 digits	10 digits
	-4931 to 4932	-307 to 308	-307 to 308	-37 to 37
`long double`	18 digits	18 digits	18 digits	10 digits
	-4931 to 4932	-4931 to 4932	-4931 to 4932	-37 to 37

```
#include <stdio.h>
int main(void)
{
 printf("Type int %u kadar bytes icerir.\n", sizeof(int));
 printf("Type char %u kadar bytes icerir.\n", sizeof(char));
 printf("Type long %u kadar bytes icerir.\n", sizeof(long));
printf("Type double %u kadar bytes icerir.\n", sizeof(double));
return 0;
 }
```

Karekter serileri (Charecter Strings)

- Seri olarak verilen bir dizi harfi ifade eder.

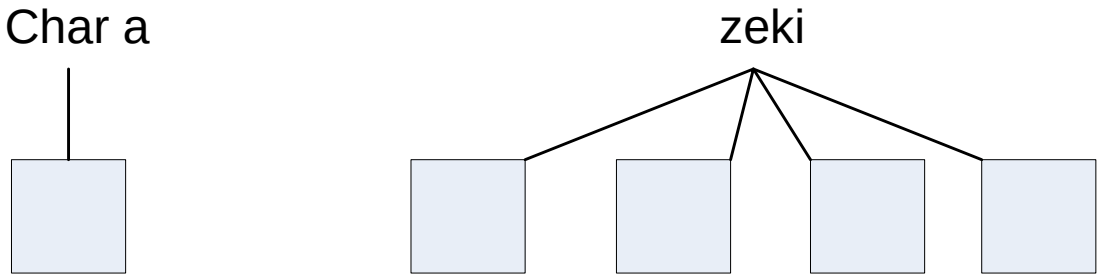
  “Zekiii öğrenciler sınıfı”
- Çift tırnak işareti derleyiciye bunun bir string olduğunu söyler. (Tek tırnak işareti, hatırlayınız, sadece bir karakter için kullanılır).
- Karakter serileri için özel bir tür yoktur, bir array (kolon matrix) içersinde **char** olarak saklanırlar.
- Seri haldeki karakterler ardışık hafıza ünitelerinde saklanırlar (her bir harf için bir ünite=1Byte)

Z	e	k	i	l	e	r		S	ı	n	ı	f	ı	!	!	\0

- Serinin sonunda kullanılan karaktere null denir ve seri sonunu markalar

Karekter serileri (Charecter Strings)

- Null yazdırılamayan (nonprinting) bir karakterdir ve ASCII kod karşılığı 0 dır.
- String her zaman yukarıdaki formatta depolanır ve bir array (seri, sıra haldeki hafıza üniteler) içerisinde saklanır.



- char a; char b[5]
- Sonuna \0 eklemeyi unutmayın!!!!.

Örnek program

```
#define  _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define LATIFE “Ne harkulade bir isim!"
int main(void)
 {
 char isim[40];
 printf(“isminiz nedir ?\n");
 scanf("%s", isim);
 printf(“Merhaba, %s. %s\n", isim, LATIFE);
 return 0;
 }
```

Örnek program

- Null karakterini açık olarak yazmassınız, derleyici onu kendi hallediyor. (scanf() okurken)
- scanf() string okurken boşluk-tab-yeni satır ile karşılaşırsa okumayı keser.
- Genel olarak scanf() ve %s bir kelime bütünlüğünü okutmak için kullanılır.
- Genel amaçlı string okutmaları için gets() fonksiyonu kullanılır. (ilerde göreceğiz.)

Karakter serisinin boyutu

- Strlen() fonksiyonu: bir karakter serisindeki harf sayısını verir.

```
#include <stdio.h>
#include <string.h>
#define LATIFE “Ne harkulade bir isim!"
int main(void)
 {
 char isim[40];
 printf(“isminiz nedir?\n");
 scanf("%s", isim);
printf(“Merhaba, %s.\n", isim);
printf(“isminiz icerindeki %d harf %d kadar hafıza, byte,
    kaplar.\n", strlen(isim), sizeof isim);
printf(“LATIFE’de %d harf var %d kadar hafıza, byte,
    kaplar.\n", strlen(LATIFE), sizeof LATIFE);
return 0;
}
```

Sabitler ve C Preprocessor

- **#define** ISIM deger

Sabitleri (şimdilik) program başlamadan tanımlamanıza ve kullanmanıza olanak verir.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
 #define PI 3.14159
int main(void)
{
float alan, cevre, yaricap;
 printf(“Pizzanizin yaricapi nedir?\n");
 scanf("%f", &yaricap);
 alan = PI * yaricap * yaricap;
 cevre = 2.0 * PI * yaricap;
 printf(“Pizza Paremetreleriniz asagidaki gibidir:\n");
 printf(“cevresi = %f, alan = %f\n", cevre, alan);
 return 0;
 }
```

Sabitler ve C Preprocessor

- Sabit tanımlama karakter ve semboller içinde geçerlidir:

```
#define BEEP '\a'
#define TEE 'T'
#define ESC '\033'
#define OOPS “Oldu mu yani!"
```

- Sabit tanımlamanın bir başka yolu da

  const int Ay = 12;

- Ay değeri artık 12 dir ve program içersinde değiştirilemez.

Symbolic Constant	Represents
CHAR_BIT	Number of bits in a char
CHAR_MAX	Maximum char value
CHAR_MIN	Minimum char value
SCHAR_MAX	Maximum signed char value
SCHAR_MIN	Minimum signed char value
UCHAR_MAX	Maximum unsigned char value
SHRT_MAX	Maximum short value
SHRT_MIN	Minimum short value
USHRT_MAX	Maximum unsigned short value
INT_MAX	Maximum int value
INT_MIN	Minimum int value
UINT_MAX	Maximum unsigned int value
LONG_MAX	Maximum long value
LONG_MIN	Minimum long value
ULONG_MAX	Maximum unsigned long value
LLONG_MAX	Maximum long long value
LLONG_MIN	Minimum long long value
ULLONG_MAX	Maximum unsigned long long value

Symbolic Constant	Represents
FLT_MANT_DIG	Number of bits in the mantissa of a float
FLT_DIG	Minimum number of significant decimal digits for a float
FLT_MIN_10_EXP	Minimum base-10 negative exponent for a float with a full set of significant figures
FLT_MAX_10_EXP	Maximum base-10 positive exponent for a float
FLT_MIN	Minimum value for a positive float retaining full precision
FLT_MAX	Maximum value for a positive float
FLT_EPSILON	Difference between 1.00 and the least float value greater than 1.00

Giriş/Çıkış Fonksiyonları

- Printf() ve scanf() çok yaygın kullanımı olan giriş/çıkış fonksiyonlar (daha başkaları da var)
- Printf() fonksiyonu bir sonraki slayttta verilen format tanımlayıcılar ile beraber aşağıdaki formatta kulalanılır.

Printf(	“Bu renk %s sana çok yakıştı\n”,	Renk)
	Kontrol bölgesi	Değişkenler

Sembol	Kullanımı
%a	Floating-point number, hexadecimal digits and p-notation (C99).
%A	Floating-point number, hexadecimal digits and P-notation (C99).
%c	Single character.
%d	Signed decimal integer.
%e	Floating-point number, e-notation.
%E	Floating-point number, e-notation.
%f	Floating-point number, decimal notation.
%g	Use %f or %e, depending on the value. The %e style is used if the exponent is less than -4 or greater than or equal to the precision.
%G	Use %f or %E, depending on the value. The %E style is used if the exponent is less than -4 or greater than or equal to the precision.
%i	Signed decimal integer (same as %d).
%o	Unsigned octal integer.
%p	A pointer.
%s	Character string.
%u	Unsigned decimal integer.
%x	Unsigned hexadecimal integer, using hex digits 0f.
%X	Unsigned hexadecimal integer, using hex digits 0F.
%%	Prints a percent sign.

Ekstra kontrol

- Önceki slaytta verilen format belirleyicilerine ek olarak % işareti ile format tanımı yapan harfler arasına (%...d) aşağıda tabloda verilen ek format tanımlayıcılarını koyabilirsiniz. Sadece tabloda verilen olasılıklar geçerlidir.

Modifier	Meaning
flag	The five flags (`-`, `+`, space, `#`, and `0`) are described in Table 4.5. Zero or more flags may be present. Example: `"%-10d"`
digit(s)	The minimum field width. A wider field will be used if the printed number or string won't fit in the field. Example: `"%4d"`
.digit(s)	Precision. For `%e`, `%E`, and `%f` conversions, the number of digits to be printed to the right of the decimal. For `%g` and `%G` conversions, the maximum number of significant digits. For `%s` conversions, the maximum number of characters to be printed. For integer conversions, the minimum number of digits to appear; leading zeros are used if necessary to meet this minimum. Using only `.` implies a following zero, so `%.f` is the same as `%.0f`. Example: `"%5.2f"` prints a `float` in a field five characters wide with two digits after the decimal point.
`h`	Used with an integer conversion specifier to indicate a `short int` or `unsigned short int` value. Examples: `"%hu"`, `"%hx"`, and `"%6.4hd"`
`hh`	Used with an integer conversion specifier to indicate a `signed char` or `unsigned char` value. Examples: `"%hhu"`, `"%hhx"`, and `"%6.4hhd"`

`j`	Used with an integer conversion specifier to indicate an `intmax_t` or `uintmax_t` value. Examples: `"%jd"` and `"%8jX"`
`l`	Used with an integer conversion specifier to indicate a `long int` or `unsigned long int`. Examples: `"%ld"` and `"%8lu"`
`ll`	Used with an integer conversion specifier to indicate a `long long int` or `unsigned long long int`. (C99) Examples: `"%lld"` and `"%8llu"`
`L`	Used with a floating-point conversion specifier to indicate a `long double` value. Examples: `"%Lf"` and `"%10.4Le"`
`t`	Used with an integer conversion specifier to indicate a `ptrdiff_t` value. This is the type corresponding to the difference between two pointers. (C99) Examples: `"%td"` and `"%12ti"`
`z`	Used with an integer conversion specifier to indicate a `size_t` value. This is the type returned by `sizeof`. (C99). Examples: `"%zd"` and `"%12zx"`

Flag	Meaning
-	The item is left-justified; that is, it is printed beginning at the left of the field. Example: `"%-20s"`
+	Signed values are displayed with a plus sign, if positive, and with a minus sign, if negative. Example: `"%+6.2f"`
space	Signed values are displayed with a leading space (but no sign) if positive and with a minus sign if negative. A `+` flag overrides a space. Example: `"% 6.2f"`
#	Use an alternative form for the conversion specification. Produces an initial `0` for the `%o` form and an initial `0x` or `0X` for the `%x` or `%X` form, respectively. For all floating-point forms, `#` guarantees that a decimal-point character is printed, even if no digits follow. For `%g` and `%G` forms, it prevents trailing zeros from being removed. Examples: `"%#o"`, `"%#8.0f"`, and `"%+#10.3E"`
0	For numeric forms, pad the field width with leading zeros instead of with spaces. This flag is ignored if a `-` flag is present or if, for an integer form, a precision is specified. Examples: `"%010d"` and `"%08.3f"`

format karakteri	Anlamı
%d	int türünü desimal sistemde yazar.
%ld	long türünü desimal sistemde yazar
%x	unsigned int türünü hexadecimal sistemde yazar.
%X	unsigned int türünü hexadecimal sistemde yazar.(semboller büyük harfle)
%lx	unsigned long türünü hexadecimal sistemde yazar.
%u	unsigned int türünü decimal sistemde yazar.
%o	unsigned int türünü oktal sistemde yazar.
%f	float ve double türlerini desimal sistemde yazar.
%lf	double türünü desimal sistemde yazar.
%e	gerçek sayıları üstel biçimde yazar.
%c	char veya int türünü karakter görüntüsü olarak yazdırır.
%s	string olarak yazdırır.
%lf	long double türünü desimal sistemde yazdırır.

Örnekler

```
#include <stdio.h>
#define PAGES 931
int main(void)
 {
 printf("*%d*\n", PAGES);
printf("*%2d*\n", PAGES);
printf("*%10d*\n", PAGES);
printf("*%-10d*\n", PAGES);
return 0;
 }
```

```
#include <stdio.h>
 int main(void)
 {
 const double RENT = 3852.99;
 printf("*%f*\n", RENT);
 printf("*%e*\n", RENT);
printf("*%4.2f*\n", RENT);
 printf("*%3.1f*\n", RENT);
 printf("*%10.3f*\n", RENT);
 printf("*%10.3e*\n", RENT);
 printf("*%+4.2f*\n", RENT);
printf("*%010.2f*\n", RENT);
 return 0;
 }
```

```
#include <stdio.h>
 int main(void)
 {
printf("%x %X %#x\n", 31, 31, 31);
printf("**%d**% d**% d**\n", 42, 42, -42);
printf("**%5d**%5.3d**%05d**%05.3d**\n", 6, 6,
   6, 6);
return 0;
 }
```

```
#include <stdio.h>
 #define AD “Alpaslan Duysak!"
int main(void)
{
 printf("/%2s/\n", AD);
printf("/%24s/\n", AD);
printf("/%24.5s/\n", AD);
printf("/%-24.5s/\n", AD);
return 0;
 }
```

Format tanımlayıcıları

- Kullanılan bu format tanımlayıcıları (%d…) nasıl çalışır?
- Binary olarak depolanmış bir değeri belirtilen formata uygun olarak görüntüler: örneğin 76 sayısı 01001100 depolanmış olabilir; %d bunu 76 olarak çevirir. %x ise sayıyı hexadecimal e dönüştürür ve 4c olarak verir. Aynı sayı %c ile okunduğunda L olarak elde edilir.

Scanf()

Klavyeden girilen değerleri değişik formatlara çevirip değişkenlere atama yapar, saklar. Değişken tanımlamaları için **scanf()** pointer kullanır.

Şimdilik pointer için **&** işareti kullanacağız. Eğer basit C veri türlerini okuyorsak & işaretini kullanacağız. Karakter serisini bir karakter array a okuyorsak & kullanılmaz.

Pointer ileride detaylı olarak işlenecek.

Scanf() ve &

```
#include <stdio.h>
 int main(void)
 {
int yas;
 char hayvan[30];
Float kilo;
printf(“Yasinizi, kilonuzu ve favori ev hayvaniniz.\n");
 scanf("%d %f", &yas, &kilo);
scanf("%s", hayvan);
printf("%d %.2f %s\n", yas, kilo, hayvan);
return 0;
 }
```

Scanf()

- **Scanf()**, boşluk, yeni satır veya tab kullanarak girişleri birbirinden ayırır. Veya siz her bir giriş için farklı **scanf()** kullanabilirsiniz.

- Tek istisna %c format ayarlayıcısı, her bir karakteri tek tek okur. Bu tab veya boşlukta olabilir, ayırım yapmaz.

- **Scanf()** fonksiyonu da aynen **printf()** de olduğu gibi format ayarlayıcıları kullanır. Bunlar bir sonraki slaytta verilmiştir.
- Format ayarlayıcılarına ek olarak format

Conversion Specifier	Meaning
%c	Interpret input as a character.
%d	Interpret input as a signed decimal integer.
%e, %f, %g, %a	Interpret input as a floating-point number (%a is C99).
%E, %F, %G, %A	Interpret input as a floating-point number (%A is C99).
%i	Interpret input as a signed decimal integer.
%o	Interpret input as a signed octal integer.
%p	Interpret input as a pointer (an address).
%s	Interpret input as a string. Input begins with the first non-whitespace character and includes everything up to the next whitespace character.
%u	Interpret input as an unsigned decimal integer.
%x, %X	Interpret input as a signed hexadecimal integer.

Modifier	Meaning
*	Suppress assignment (see text). Example: "%*d"
digit(s)	Maximum field width. Input stops when the maximum field width is reached or when the first whitespace character is encountered, whichever comes first. Example: "%10s"
hh	Read an integer as a signed char or unsigned char. Examples: "%hhd" "%hhu"
ll	Read an integer as a long long or unsigned long long (C99). Examples: "%lld" "%llu"
h, l, or L	"%hd" and "%hi" indicate that the value will be stored in a short int. "%ho", "%hx", and "%hu" indicate that the value will be stored in an unsigned short int. "%ld" and "%li" indicate that the value will be stored in a long. "%lo", "%lx", and "%lu" indicate that the value will be stored in unsigned long. "%le", "%lf", and "%lg" indicate that the value will be stored in type double. Using L instead of l with e, f, and g indicates that the value will be stored in type long double. In the absence of these modifiers, d, i, o, and x indicate type int, and e, f, and g indicate type float.

Eğlence

- Hataları bulun!

```
define B booboo
define X 10
main(int)
{
int yas;
 char ad;
printf(“ilk adinizi girin.");
scanf("%s", ad);
printf(“okey, %c, yasiniz kac?\n", ad);
 scanf("%f", yas);
 xp = yas + X;
printf(“bu %s! En azindan %d.\n", B, xp);
 rerun 0;
}
```

Temel İşlem Operatörleri

- C progrmları temel matematik işlemlerini bazı operatörler ile simgelerler:

+ - * / =

- Değer atama operatörü: =

alp=1;

(alp değişkeninin değeri şimdi bir olarak atandı)

i=i+1;

(i değişkeninin değerini bul ve ona 1 ekle, sonra toplam sonucunu i değişkenine ata)

Temel İşlem Operatörleri

```
İnt i;
i=20;
i=i+10;
```

i değişkeninin değeri şimdi? 30

25=i ???? Olamaz, değer atama operatörünün sol tarafında değişken olmalıdır, sabit değil.

Temel İşlem Operatörleri

- Toplama operatörü: **+**

  C=a+b;

  printf("%d", 4 + 20);
- Çıkarma operatörü: **-**

  A=30-25;
- + ve – operatörleri aynı zamanda sayıların işaretini belirtmek için kullanılır. (-20, +4)
- Çarpma operatörü: ***

  A=10*C;
- Bölme operatörü: **/**

  A=20.7/12.123;

C’de int/int, float/float bölmeleri yapılır. (5/3 anlamsızdır çünkü kesirli kısımlar atılır.)

Temel İşlem Operatörleri

```
#include <stdio.h>
 int main(void)
 {
 printf("integer division: 5/4 is %d \n", 5/4);
 printf("integer division: 6/3 is %d \n", 6/3);
printf("integer division: 7/4 is %d \n", 7/4);
 printf("floating division: 7./4. is %f \n", 7./4.);
printf("mixed division: 7./4 is %f \n", 7./4);
return 0;
 }
```

Temel İşlem Operatörleri

- Gerçekte compiler iki farklı tür veriyi bölme işlemine tabi tutmaz, önce tek tip veriye dönüştürür sonra böler.
- İnt/float işleminde iki tür de float olarak alınır ve işlem yapılır.
- Bölme işlemi yapılır ve kesirli kısım atılır:5/3=1.

- Kullanılan operatörlerde öncelik sırası vardır.

  A=10+20*2/4=?

- **A=(10+20*2)/4=?**
- **A=10+20*(2/4)=?**
- **y = 6 * 12 + 5 * 20;**

Operators	Order of evaluation	Remarks
[] ( ) ->	Left to right	Array subscript, function call
– + sizeof( ) ! ++ ––		
& * ~ (cast)	Right to left	Unary
* / %	Left to right	Binary Multiplicative
+ -	Left to right	Binary Additive
>> <<	Left to right	Shift operators
< <= > >=	Left to right	Relational operators
== !=	Left to right	Equality operators
&	Left to right	Bitwise And operator
^	Left to right	Bitwise Xor operator
|	Left to right	Bitwise Or operator
&&	Left to right	Logical And operator
||	Left to right	Logical Or operator
?:	Left to right	Conditional operator
= += -= *= /= %=		
&= -= |= <<= >>=	Right to left	Assignment
,	Right to left	Comma

Temel İşlem Operatörleri

```
#include <stdio.h>
int main(void)
 {
 int top;
 top = -(2 + 5) * 6 + (4 + 3 * (2 + 3));
 printf("top = %d \n", top);
return 0;
 }
```

Temel İşlem Operatörleri

- C’de 40 kadar operatör vardır.
- Modulus operatör: % Mod operatörü

13%5= 13 mod 5 diye okunur.Sonuc olarak 3 değerini geri döndürür. (13 mod 5 de iki kez 5 var kalan 3)

- Float ile çalışmaz

Temel İşlem Operatörleri

- Artırma (++) ve eksiltme (--) operatörleri:
- İki şekilde, **++i** veya **i++** değişkenin değerini 1 artırmak için kullanılır.

```
#include <stdio.h>
int main(void)
{
 int ultra = 0,super = 0;
 while (super < 5)
{
 super++;   // super=super+1;
++ultra;     //ultra=ultra+1;
 printf("super = %d, ultra = %d \n", super, ultra);
 }
return 0;
 }
```

Aynı sonucu verdiler.

Temel İşlem Operatörleri

```
#include <stdio.h>
int main(void)
 {
 int a = 1, b = 1;
 int aplus, plusb;
aplus = a++;
 plusb = ++b;
 printf("a aplus b plusb \n");
 printf("%1d %5d %5d %5d\n", a, aplus, b, plusb);
 return 0;
 }
```

a	aplus	b	plusb
2	1	2	2

Temel İşlem Operatörleri

Q=2*++a;

A değişkenini 1 artır ve 2 ile çarp, sonucu q'ya ata.

Q=2*a++;

a'yı iki ile çarp, sonucu q'ya ata ve a değişkenini 1 artır.

- -- azaltma operatörü:
- --say,,, say - -
- Sadece parantez () opetörünün önceliği ++ veya - - operatörünün önündedir.

X*Y++=X*(Y++)

y = 2; n = 3;

nextnum = (y + n++)*6;

Temel İşlem Operatörleri

ans = num/2 + 5*(1 + num++); ????

i = 3;

j =4;

k = i++ + -j;

k=???

n = 3;

y = n++ + n++;

```
while (num < 21)
{
printf("%d %d\n", num, num*num++);
 } ????
```

Kural

- Eğer bir değişken bir operasyonda birden fazla sefer kullanılıyorsa, o değişken üzerinde ++ veya – operatörü kullanmayın

- Fonksiyonlarda artırma veya eksiltme operatörünü kullanmayın eğer değişken birden fazla argümanın parçası ise.

Biraz da Eğlenin

1) Aşağıdaki işlemlerin sonuçlarını bulun

a) x = (2 + 3) * 6;

b) x = (12 + 6)/2*3;

c) y = x = (2 + 3)/4;

d) y = 3 + 2*(x = 7/2);

Biraz da Eğlenin

2) Aşağıdaki program ekrana ne yazar?

```
#include <stdio.h>
///#define FORMAT "%s! C guzel bir dil!\n“
 int main(void)
{
 int num = 10;
///printf(FORMAT,FORMAT);
printf("%d\n", ++num);
 printf("%d\n", num++);
 printf("%d\n", num--);
printf("%d\n", num);
return 0;
 }
```

Biraz da Eğlenin

3) Aşağıdaki program ekrana ne yazar?

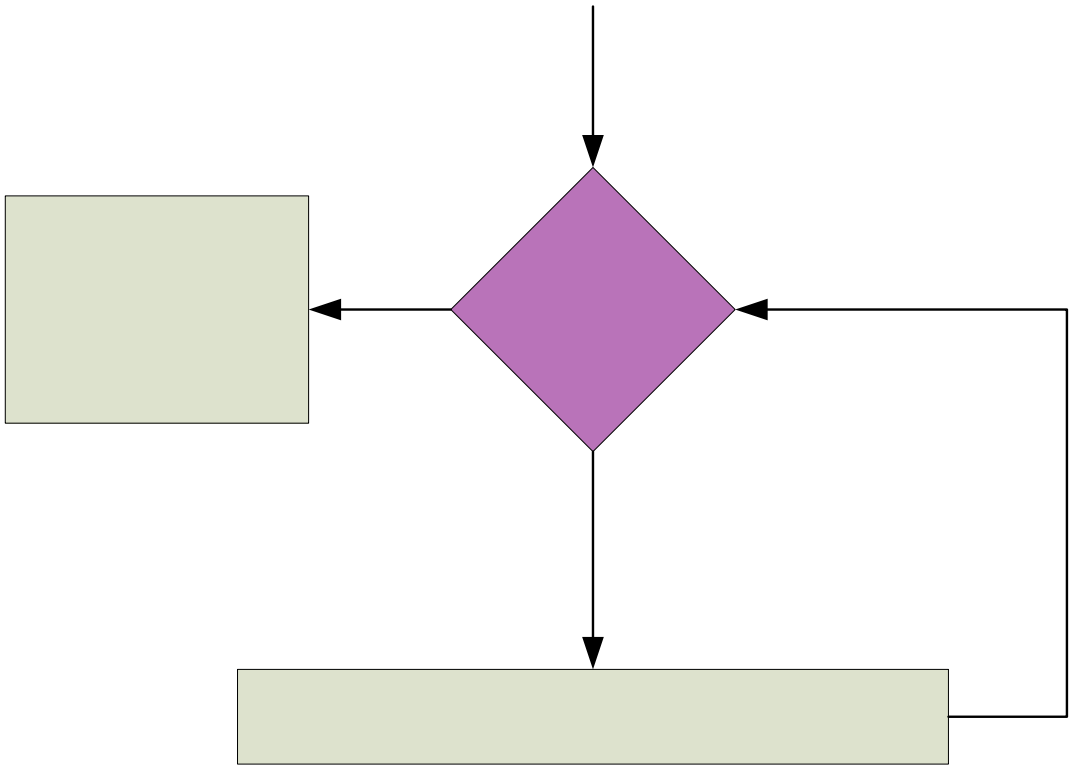
```
#include <stdio.h>
int main(void)
 {
 char c1, c2;
int diff;
float num;
 c1 = 'S'; c2 = 'O';
 diff = c1 - c2;
 num = diff;
 printf("%c%c%c:%d %3.2f\n", c1, c2, c1, diff, num);
return 0;
 }
```

Kontrol Yapıları

- While() kontrol yapısı:

while(şartlar)

işlemler



Kontrol Yapıları

- While(): Mutlaka bir test değeri içermelidir ki while döngüsü bir sefer yanlış olmalı ve döngü sona ermelidir. Aksi halde sonsuz döngü oluşur.

```
Sayi=1;
While(sayi<20)
{
        Printf("sayıyoruz işte,%d\n",sayi)
}
```

- Döngü ne zaman biter?

Kontrol Yapıları

```
#include <stdio.h>
int main(void)
{
 int n = 5;
while (n < 7)                          // satır5
 {
 printf("n = %d\n", n);
 n++;                                  // satır 8
printf(“Simdi n = %d\n", n);      // satır 9
}
 printf(“Dongu biter, n=%d.\n“,n);
 return 0;
}
```

Kontrol Yapıları

Döngüye giriş koşulu iyi ayarlanmalı:

```
index = 10;
while (index++ < 5)
printf(“iyi gunler dileriz.\n");
```

Döngüden çıkış olmalı

```
#include <stdio.h>
 int main(void)
{
 int n = 0;
        while (n < 3)
        printf("n = %d\n", n);
        n++;
         printf(“yapilanin hepsi bu\n");
 return 0;
 }
```

Kontrol Yapıları

```
#include <stdio.h>
 int main(void)
{
 int n = 0;
        while (n++ < 3)
        printf("n = %d\n", n);
         printf(“yapilanin hepsi bu\n");
 return 0;
 }
```

Kontrol Yapıları

```
#include <stdio.h>
int main(void)
{
int num = 1;
    while (num < 21)
    {
        printf("%d %d\n", num, num * num);
        num = num + 1;
    }

return 0;
}
```

```
#include <stdio.h>
#define MAX 100
int main(void)
{
 int count = MAX + 1;
   while (--count > 0)
   {
   printf("%d sise su rafta duruyor, " "%d sise
   su!\n", count, count);

   printf(“bir tanesi kirildi,\n");
    printf("%d sise su kaldi!\n\n", count - 1);
    }
 return 0;
 }
```

Biraz da Eğlenin

4) Hataları bulun

```
int main(void)
 {
 int i = 1, float n;
printf(“Bazı hatlar olabilir, bulun onları!\n");
 while (i < 30)
 n = 1/i;
 printf(" %f", n);
 printf(“Hepsi bu kadar!\n");
 return;
 }
```

Biraz da Eğlenin

5) Aşağıdaki kod parçaları (a ve b) bir programdan alınmıştır. Verilen kod ekrana ne yazar?

a)

```
int x = 100;
while (x++ < 103)
        printf("%d\n",x);
        printf("%d\n",x);
```

Biraz da Eğlenin

b)

```
char ch = 's';
while (ch < 'w')
{
        printf("%c", ch);
        ch++;
}
printf("%c\n",ch);
```

While() ve Mantık

Operator	Meaning
<	Is less than
<=	Is less than or equal to
==	Is equal to
>=	Is greater than or equal to
>	Is greater than
!=	Is not equal to

While() ve Mantık

- Dikkat edilmesi gereken noktalar:
  = ve == birbirinden farklı
- Bazı öngüler sonsuz bazıları sayılı olarak tanımlıdırlar;

```
#include <stdio.h>
 int main(void)
 {
 const int NUMBER = 22;
 int count = 1;
        while (count <= NUMBER)
        {
        printf("Arkadaşım olurmusun!\n");
        count++;
         }
 return 0;
 }
```

While()

- Dikkat edilmesi gereken noktalar:

Bir sayıcı olmalı,

Bu sayıcı bir sınır değeri ile karşılaştırılmalı

Bu sayıcı döngünün her seferinde artırılmalı

Note: Sayıcı her zaman döngü dışında tanımlanır ve değer atanır. Sayıcının değerinin artırılması

while(sayıcı++<Limit)

şeklinde olabilir veya açık olarak döngü içerisinde yapılır.

Bu durumları for döngüsü ortadan kaldırır.

For() döngüsü

- For() döngüsü sayıcı tanımlama-değer atama, test ve sayıcı değer artırımını aynı yerde yapar.

```
#include <stdio.h>
 int main(void)
 {
 const int NUMBER = 22;
 int count;
        for (count = 1; count <= NUMBER; count++)
        printf("Arkadaşım olurmusun!\n");
 return 0;
}
```

For() döngüsü

```
#include <stdio.h>
int main(void)
 {
 int num;
printf("       n       n kupu\n");
        for (num = 1; num <= 6; num++)
        printf("%d %d\n", num, num*num*num);
 return 0;
 }
```

For() döngüsü

- For() döngüsü değişik şekillerde kullanılabilir:

```
#include <stdio.h>
int main(void)
 {
 int saniye;
        for (saniye = 5; saniye > 0; saniye --)
        printf("%d saniyeler!\n", saniye);
 printf(“Booooooom!\n");
 return 0;
 }
```

For() döngüsü

- Artırmayı bir yapmak zorunda değilsiniz.

```
#include <stdio.h>
int main(void)
 {
 int n;
        for (n = 2; n < 60; n = n + 13)
        printf("%d \n", n);
 return 0;
 }
```

For() döngüsü

- Sayılar yerine karakterleri sayabilirsiniz

```
#include <stdio.h>
 int main(void)
 {
 char ch;
   for (ch = 'a'; ch <= 'z'; ch++)
   printf(" %c nin ASCII değeri %d.\n", ch, ch);
return 0;
 }
```

For() döngüsü

- For() döngüsünü test için kullanabilirsiniz (sadece saymaları yapmıyor yani)

```
#include <stdio.h>
int main(void)
 {
 int num;
printf("       n       n kupu\n");
          for (num = 1; num*num*num <= 216; num++)
          printf("%d %d\n", num, num*num*num);
 return 0;
 }
```

For() döngüsü

- Artırım aritmetik veya geometrik olarak seçilebilir:

```
#include <stdio.h>
int main(void)
 {
 double d;
   for (d = 100.0; d < 150.0; d = d * 1.1)
 printf(“d niz simdi %f\n", d);
 return 0;
 }
```

For() döngüsü

- Artırma yerine istediğiniz legal işlemi koyabilirsiniz:

```
#include <stdio.h>
 int main(void)
 {
 int x;
 int y = 55;
        for (x = 1; y <= 75; y = (++x * 5) + 50)
        printf("%d %d\n", x, y);
 return 0;
}
```

For() döngüsü

- Sonsuz döngü

```
for (; ; )

printf(“Biri beni durdursun\n");
```

For() döngüsü

- İlk sayı değer ataması olmak zorunda değildir.
- Bu kısım sadece bir defa işlem görür.
- Buraya istediğiniz başka bir işlem koyabilirsiniz

```
#include <stdio.h>
 int main(void)
 {
 int num = 0;
 for (printf(“benim istedigim sayiyi girin!\n"); num != 6; )
 scanf("%d", &num);
printf(“İste istedigim sayi bu!\n");
 return 0;
 }
```

For() döngüsü

- Döngü parametreleri döngü içerisinde döngü çalışırken değiştirilebilirler:

for (n = 1; n < 10000; n = n + degisken)

Değer atama operatörleri (ilaveler)

+=

-=

***=**

/=

%=

For() döngüsü

- For() döngüsünü birden fazla değer için kullanabilirsiniz

```
#include <stdio.h>
 int main(void)
 {
 const int ilk= 37;
 const int ikinci = 23;
int kg, eder;
 printf(" kg eder\n");
 for (kg=1, eder=ilk; kg <= 16; kg++, eder += ikinci)
printf("%5d $%4.2f\n", ounces, cost/100.0);
 return 0;
 }
```

Do while döngüsü

```
#include <stdio.h>
 int main(void)
 {
 const int sifre = 13;
 int girilen_sifre;
 do
 {
 printf(“Bilgisayar kulubune uye olmak icin,\n");
 printf(“gizli sifreyi giriniz: ");
 scanf("%d", & girilen_sifre);
 } while (girilen_sifre != sifre);
 printf(“Tebrikler artik uyesiniz!\n");
 return 0;
 }
```

Hangisi

- **for()** ve **while()** döngüleri giriş-şartlı döngülerdir.
- Şart daha başlangıçta test edilir, eğer giriş şartı sağlanmıyorsa döngü işlemez.
- **do() while()** döngüsü ise çıkış-şartlı bir döngüdür. İşlem en az bir kez yapılır.
- Giriş şartlı: while or for?
- Hangis size uyarsa!!

Hangisi

- For(;test;)=while(test)

İlk
While(test)
{
İşlemler
Artır
}
=
for(ilk; test; artır)
işlemler

Biraz da Eğlenin

1) scanf() ve %c kullanarak bir karakter serisini girişi olarak olan ve çıkış olarak girilen seriyi tersten yazan bir program yazınız. Aynı programı tek bir kelimeyi okuyarak aynı kelimeyi tersten yazacak şekilde değiştirin (strlen fonksiyonunu kullanın)

İç İçe Döngüler

```
#include <stdio.h>
#define SAT 6
#define CHARS 10
int main(void)
 {
 int sat;
 char ch;
    for (sat = 0; sat < SAT; sat++)
    {
          for (ch = 'A'; ch < ('A' + CHARS); ch++)
                printf("%c", ch);
                printf("\n");
    }
 return 0;
 }
```

```
#include <stdio.h>
int main(void)
 {
 const int SAT = 6;
 const int CHARS = 6;
 int sat;
 char ch;
        for (sat = 0; sat < SAT; sat++)
        {
                for (ch = ('A' + sat); ch < ('A' + CHARS); ch++)
                printf("%c", ch);
                printf("\n");
        }
 return 0;
}
```

```
#include <stdio.h>
#define SAT 6
#define CHARS 10
int main(void)
 {
 int sat;
 char ch;

    for (sat = 0; sat <SAT;sat++)
    {
            for (ch = 'A'; ch <('A'+sat); ch++)
                    printf("%c", ch);
                    printf("\n");

    }
 return 0;
 }
```

Dizilere giriş

- Aynı tip verilerin seri olarak depolandığı üniteler (hafızada ayrılmış bölümler)
- Dizinin bir adı (adresi) vardır ve her bir parçasına elemen denir.
- Dizinin her elemana dizi indeksi ile ulaşılır.

float alp[20]

- Yukarıdaki komut alp'in bir dizi (array) olduğunu, 20 elmanı bulunduğunu ve bunların tiplerinin float olduğunu belirtir.

Dizilere giriş

- Dizinin birinci ve ikinci elemanları sırasıyla

  alp[0] ve alp[1]

- Dizinin son elemanı

  alp[19]

- Eğer alp dizisinin 5. elamanının 5 olmasını istiyorsak

  alp[5]=5.0;

- Veya

  scanf(“%f”,&alp[5]);

Dizilere giriş

alp[25]=?

- Dizi türü C türlerinden her hangi biri olabilir.

```
#include <stdio.h>
#define BOYUT 10
int main(void)
 {
 int index, skore[BOYUT];
 int sum = 0;
 float average;
 printf(“Futbol sonuçlarını %d gir:\n", BOYUT);
 for (index = 0; index < BOYUT; index++)
 scanf("%d", &skore[index]); // 10 sonucu gir
 printf(“Girdiğiniz sonuçlar aşağıdadır:\n");
```

```
for (index = 0; index < BOYUT; index++)
 printf("%5d", skore[index]); // sonuçların
   sağlaması
 printf("\n");
for (index = 0; index < BOYUT; index++)
sum += skore[index]; // sonuçların toplamı
 average = (float) sum / BOYUT; // ortalama
 printf(“Sonuçların toplamı = %d, average =
   %.2f\n", sum, average);
 return 0;
}
```

Eğlence

1)Aşağıdaki program nasıl çalışır? Compile etmeden cevaplayınız

```
#include <stdio.h>
 int main(void)
 {
 char ch;
 scanf("%c", &ch);
 while ( ch != 'g' )
 {
 printf("%c", ch);
 scanf("%c", &ch);
 }
 return 0;
 }
```

Eğlence

2) Aşağıdaki diziyi ele alalım:

```
double kuyu[20];
```

a) Dizinin adı?

b) Eleman sayısı?

c) Hangi tür bilgi saklanabilir?

d) Scanf() ile kullanımlarından hangisi doğru?

- scanf("%lf", kuyu[2])
- scanf("%lf", &kuyu[2])
- scanf("%lf", &kuyu)

Eğlence

3)Aşağıdaki çıktıyı veren programı yazınız (kolay, zoru bekleyin o zaman)

F

FE

FED

FEDC

FEDCB

FEDCBA

Eğlence

3) Aşağıdaki çıktıyı veren programı yazınız (iyi şanslar, he he…)

A

ABA

ABCBA

ABCDCDA

ABCDEDCBA

Lab

4) Geçen haftaki 3. problemin ilk kısmı

5) Kullanıcıdan alt ve üst limitleri isteyen ve limitlerin kareleri arasındaki sayıların karelerinin toplamlarını bulan bir program yazın. Kullanıcıya programdan çıkmayı isteyip istemediğini sorsun.

6)Hocanız lotodan 1 milyon dolar kazandı (varsayım), ve parayı %8 faiz veren bir bankaya yatırdı. Hocanız her yılın sonunda 100.000 dolar nakit para çekiyor hesaptan. Hocanız kaç yılda parayı bitirir. (tekrar ediyorum bu bir varsayım !)

Döngü Kontrolleri: Dallanmalar (seçmeler) ve Atlamalar

İf (karşılaştırma)

işlemler

diğer işlemler

Karşılaştırma doğru sonuç verirse işlemler yapılır. Yanlış olursa diğer işlemler yapılır.

- else eklenmesi:

İf (karşılaştırma)

işlemler1

Else

işlemler2

Döngü Kontrolleri: Dallanmalar (seçmeler) ve Atlamalar

- Çoktan seçmeli

```
İf(a<1000)
    Top=0;
Else if(a<1500)
    Top=1;
Else if(a<2000)
    Top=2;
Else
    Top=3;
```

Döngü Kontrolleri: Dallanmalar (seçmeler) ve Atlamalar

```
if (şart1)
        işlem1
if (şart2)
        işlem2
else
        işlem3
```

Else hangi if()'e ait

Eğlence

7) Kullanıcıya “Analiz için bir sayı girin, çıkış için q” diye soran,

Girilen sayıların bölenlerini bulup yazan, sayı tek ise bunu ekrana yazıp yeni sayı girilmesini isteyen,

“q” girildiğinde güle güle deyip çıkan

bir program yazınız.

Mantık Opertörleri

&& **VE**

|| **VEYA**

! **DEĞİL**

Öncelik

! Önceliği çok yüksektir, **&&** ‘nın önceliği **||** ‘nın önceliğine göre daha yüksektir

a > b && b > c || b > d

((a > b) && (b > c)) || (b > d)

C mantıksal işlemleri soldan sağa doğru yapar.

Mantık Opertörleri

Örnek:

a) **while ((c = getchar()) != ' ' && c != '\n')**

b) **if (number != 0 && 12/number == 2)**

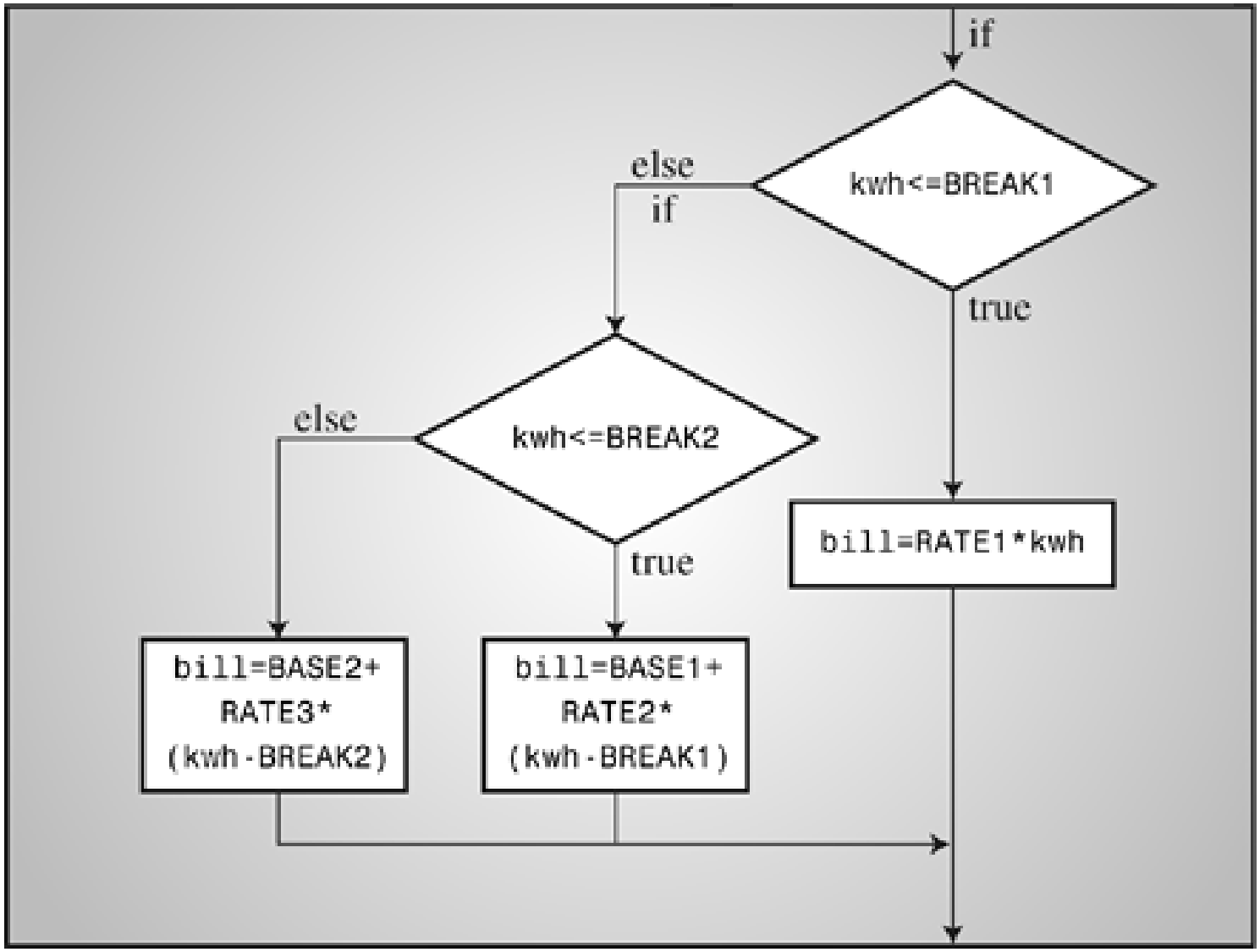
printf(“sayınız 6.\n");

c) **if (range >= 90 && range <= 100)**
printf(“OK!\n");

d) **if (90 <= range <= 100)**

printf("Good show!\n"); ?????????

```
#include <stdio.h>
#define RATE1   0.12589      /* rate for first 360 kwh     */
#define RATE2   0.17901      /* rate for next 320 kwh      */
#define RATE3   0.20971      /* rate for over 680 kwh      */
#define BREAK1  360.0        /* first breakpoint for rates  */
#define BREAK2  680.0        /* second breakpoint for rates */
#define BASE1   (RATE1 * BREAK1)
#define BASE2  (BASE1 + (RATE2 * (BREAK2 - BREAK1)))
 int main(void)
{
   double kwh;              /* kilowatt-hours used        */
   double bill;             /* charges                    */
   printf("Please enter the kwh used.\n");
   scanf("%lf", &kwh);      /* %lf for type double        */
   if (kwh <= BREAK1)
      bill = RATE1 * kwh;
   else if (kwh <= BREAK2)   /* kwh between 360 and 680    */
      bill = BASE1 + (RATE2 * (kwh - BREAK1));
   else                      /* kwh above 680              */
      bill = BASE2 + (RATE3 * (kwh - BREAK2));
   printf("The charge for %.1f kwh is $%1.2f.\n", kwh, bill);
   return 0;
}
```


if
kwh<=BREAK1
else
if
true
kwh<=BREAK2
else
true
bill=RATE1*kwh
bill=BASE2+
RATE3*
(kwh-BREAK2)
bill=BASE1+
RATE2*
(kwh-BREAK1)

```
#include <stdio.h>
int main(void)
{
    unsigned long num;          // number to be checked
    unsigned long div;          // potential divisors
    int isPrime;                // prime flag
    printf("Please enter an integer for analysis; ");
    printf("Enter q to quit.\n");
    while (scanf("%lu", &num) == 1)
    {
```

```
for (div = 2, isPrime= 1; (div * div) <= num; div++)
      {
         if (num % div == 0)
         {
            if ((div * div) != num)
            printf("%lu is divisible by %lu and %lu.\n",  num, div, num / div);
            else
               printf("%lu is divisible by %lu.\n",  num, div);
            isPrime= 0; // number is not prime
         }
      }
      if (isPrime)
         printf("%lu is prime.\n", num);
      printf("Please enter another integer for analysis; ");
      printf("Enter q to quit.\n");
   }
   printf("Bye.\n");
   return 0;
}
```

```
#include <stdio.h>
#define PERIOD '.'
int main(void)
{
    int ch;
    int charcount = 0;

   while ((ch = getchar()) != PERIOD)
    {
        if (ch != '"' && ch != '\'')
            charcount++;
    }
    printf("Burada %d tane cift tirnak olmayan karakter var.\n", charcount);
   return 0;
}
```

Kelime, karakter ve satır sayıcı

- Kullanıcıdan karakter olarak cümleleri alan, önceden tanımlanmış karakteri gördüğünde duran ve çıktı olarak satır sayısı, kelime sayısı ve karakter sayısını veren bir program yazalım.

Kelime, karakter ve satır sayıcı

- Pseudocode:

```
Karakter oku
   while (hala giriş var)
        karakter sayıcıyı artır
        satır okundu ise satır sayıcıyı artır
        kelime okundu ise kelime sayıcıyı artır
        bir sonraki karakteri oku
```

Kelime, karakter ve satır sayıcı

Girş öngüsü:

```
while ((ch = getchar()) != STOP)
{
...
 }
```

STOP: giriş sonlandırma.

*Getchar() karakter girişi için kullanılır ve her bir döngüde karakter sayıcı artırılır.

Kelime, karakter ve satır sayıcı

- Satır sayma: satır sayıcı newline karakterine (\n) bakar ve sayar
- Kelime sayma:kelime ilk karakter ile başlar ve ilk boşluğa ulaşınca sonlanır; boşluk bulma testi,

c!=‘ ’ && c!=‘\n’ && c!=‘\t’ (true eğer boşluk değilse)

c == ' ' || c == '\n' || c == '\t' (true eğer c boşluksa)

Daha kısa yolu var:

ctype.h header file ile isspace() fonksiyonunu kullanmak.

Kelime, karakter ve satır sayıcı

- isspace(c) true değerini verir eğer c boşluk ise.
- Karakterin kelime olup olmadığının takibi için bir bayrak kullanılır. İlk karakter görüldüğünde bayrak 1 yapılır ta ki ilk boşluk görülünceye kadar. İlk boşluk oluştuğunda bayrak 0 yapılır.

```
#include <stdio.h>
#include <ctype.h>          // isspace() için
#define STOP '|'
#define false 0
#define true 1
int main(void)
{
    char c;                 // okunan karakter
    long s_karakter = 0L;       // karakter sayisi
    int s_satir = 0;         // satir sayisi
    int s_kelime = 0;         // kelime sayisi
    int kelime = false;     // kelime de ise, dogru
    printf("Analiz edilecek metni giriniz (| cikis icin):\n");
```

.

```
while ((c = getchar()) != STOP)
    {
        s_karakter++;          // karakter sayici
        if (c == '\n')
            s_satir++;      // satir sayici
        if (!isspace(c) && !kelime)
        {
            kelime = true;  // yeni bir kelime baslangici
            s_kelime++;      // kelime say
         }
       if (isspace(c) && kelime)
            kelime = false; // kelimenin sonu
     }
     printf("karakter = %ld, kelime = %d, satir = %d, ",
          n_karakter, n_kelime, n_satir);
      return 0;
}
```

```
#include <stdio.h>
int main(void)
{
    long num;
    long sum = 0L;
    int status;
    printf("Toplanacak tam sayiyi girin ");
    printf("(q to quit): ");
    status = scanf("%ld", &num);
    while (status == 1)
    {
        sum = sum + num;
        printf("Diger sayi lutfen (q to quit): ");
        status = scanf("%ld", &num);
    }
    printf("Toplam: %ld.\n", sum);
    return 0;
}
```

Şart operatörü (? :)

- C dili "if else.." işleminin kısa yolunu tanımlamıştır:

Durum1 ? Durum2 : Durum3

(eğer Durum1 doğru ise işlem sonucu Durum2 dir.
Eğer Durum1 yanlış ise işlem sonucu Durum3 tür)

```
x = (y < 0) ? -y : y;
```

```
if (y < 0)
   x = -y;
 else
   x = y;
```

Şart operatörü (? :)

max = (a > b) ? a : b;

Eğer a>b ise max=a

Eğer a<b ise max=b

```
#include <stdio.h>
#define ORT 200       /* mkare/kutu */
int main(void)
{
    int mkare;
    int kutu;
    printf("boyanacak yerin metrekaresini girin:\n");
    while (scanf("%d", &mkare) == 1)
    {
        kutu = mkare / ORT;
        kutu += ((mkare % ORT == 0)) ? 0 : 1;
        printf("Ihtiyaciniz olan %d %s boya.\n", kutu,
               kutu == 1 ? "kutu" : "kutular");
        printf("Enter next value (q to quit):\n");
    }
   return 0;
}
```

Döngü yardımcıları: Continue ve Break

- Döngü içinde bazı bölümlerin atlanması ve hatta durdurulması için kullanılır.
- **Continue** : döngü içinde kalan kısmın atlanması ve yeni bir iterasyonun başlamasını sağlar.
- Eğer iç içe döngüler ile beraber kullanılırsa sadece en içteki döngüyü etkiler.
- Üç döngü formu ile de kullanılabilir.
- Continue döngünün başa (döngü şartına) dönmesini sağlar

Continue

- Örnek: 1 den 10 a kadar sayıları yazmak istiyorsunuz fakat 4 ve 7 hariç:

```
#include <stdio.h>
int main(void)
{
    int i;
    for(i = 0; i < 11; i++)
    {
        if ((i == 4) || (i == 7)) continue;
        printf(" sayilariniz i = %d\n", i);
    }
   return 0;
}
```

Break

- Döngünün döngü şartına bağlı kalmadan sonlanmasına ve bir sonraki işlemin icra edilmesini sağlar.

```
#include <stdio.h>
int main(void)
{
    int i=0;
while (1)
    {
    i = i + 1;
    printf(" Sayimiz i= %d\n",i);
    if (i>5) break;
    }
  return 0;
}
```

```
#include <stdio.h>
int main(void)
{
int p, q;
    scanf("%d", &p);
while ( p > 0)
{
    printf("%d\n", p);
    scanf("%d", &q);
    while( q > 0)
    {
        printf("%d\n",p*q);
        if (q > 100)
            break;
        scanf("%d", &q);
    }
    if (q > 100)
        break;
    scanf("%d", &p);
}
    return 0;
}
```

```
#include <stdio.h>
int main(void)
{
    const float MIN = 0.0f;          const float MAX = 100.0f;
    float score;                     float total = 0.0f;
    int n = 0;        float min = MAX;        float max = MIN;
    printf("Ilk scoru gir (q to quit): ");
    while (scanf("%f", &score) == 1)
    {
        if (score < MIN || score > MAX)
        {
            printf("%0.1f Gecersiz tekrar dene: ", score);
            continue;
        }
        printf("Tamam %0.1f:\n", score);
        min = (score < min)? score: min;
        max = (score > max)? score: max;
        total += score;
        n++;
        printf("Bir sonraki score lutfen (q to quit): ");
    }
```

```
if (n > 0)
    {
       printf(" %d tane scorun ortalamasi %0.1f.\n", n,
    total / n);
        printf("En dusuk = %0.1f, En yuksek = %0.1f\n",
    min, max);
    }
    else
        printf("Gecerli score girilmedi.\n");
    return 0;
}
```

Çoktan Seçmeli: switch ve break

- İki alternatif arasından seçim yaparken if-else çok kullanışlıdır. Bazen ikiden fazla seçenek olabilir: (bir den fazla if-else tabi ki kullanılabilir fakat alternatif ve kolay yol vardır).
- Break kullanılmaz ise bütün case ‘ler taranır.

```
Switch(int değer)
{
Case sabit1:
    işlem
Case sabit2:
    işlem
Defult:
    işlem
}
```

```
#include <stdio.h>
int main()
{
int a;
printf("bir sayi gir\n");
scanf("%d", &a);
switch (a) {
        case 1: printf("bir\n");
        case 2: printf("iki\n");
        case 3: printf("uc\n");
        case 4: printf("dort\n");
        default: printf("hicbiri\n");
}
    return 0;
}
```

```
#include <stdio.h>
int main()
{
int a;
printf("bir sayi gir\n");
scanf("%d", &a);
switch (a) {
        case 1: printf("bir\n");break;
        case 2: printf("iki\n");break;
        case 3: printf("uc\n");break;
        case 4: printf("dort\n");break;
        default: printf("hicbiri\n");break;
}
    return 0;
}
```

```
if ( a == 1) {
    ifade_1;
    ifade_2;
}
else if (a == 2) {
    ifade_3;
    ifade_4;
}
else if (a == 4) {
    ifade_5;
}
else {
    ifade_6;
    ifade_7;
}
```

```
switch (a) {
case 1:
        ifade_1;
        ifade_2;
        case 2:
        ifade_3;
        ifade_4;
case 4:
        ifade_5;
default:
        ifade_6;
        ifade_7;
}
```

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{
    char ch;
    printf("alfebeden bana bir harf ver ");
    printf("bende sana o harf ile baslayan hayvan ismini
vereyim.\n");
    printf("Lutfen harfi giriniz; cikis icin # yaziniz.\n");
    while ((ch = getchar()) != '#')
    {
        if('\n' == ch)
            continue;
        if (islower(ch))     /* sadece küçük harf        */
```

```
switch (ch)   {
        case 'a' :
            printf("Kiral ASLAN\n");
            break;
        case 'b' :
            printf("Beyaz et BALIK\n");
            break;
        case 'c' :
            printf("Guzel hayvan CEYLAN\n");
            break;
        case 'd' :
            printf("Yenmez, DOMUZ\n");
            break;
        case 'e' :
            printf("Guzel gozlu, ESEK\n");
            break;
        case 'f' :
            printf("Korkulmaz, FARE\n");
            break;
        default :
            printf("Zamanim azdi butun harfleri koyamadim!\n");
    }
```

```
else
        printf("sadece küçük harfler gecerli.\n");
    while (getchar() != '\n')
        continue;
    printf("Bir harf veya  #.\n");
  }
  printf("Hoscakalin!\n");
  return 0;
}
```

Çoktan Seçmeli: switch ve break

- Hayvanlar alemi örneğinde karakterlerin nasıl okunduğuna dikkat ediniz:Yalnızca ilk karakter dikkate alınıyor, “enter” dikkate alınmıyor. İnter-aktif programlarda çok önemlidir.

GOTO işlemi

- BASIC ve Fortran komutu, C de var fakat kullanılması gerekli değil (tavsiye edilmiyor).
- C GOTO olmadan da çok iyi çalışıyor. Kullanımı:

Önce bir yer tanımlanır

Part2: printf(…)

……

Goto part2;

GOTO işlemi

```
if (boyut > 12)
 goto a;
 goto b;
 a: eder = eder * 1.05;
 bayrak = 2;
b: atura = eder *bayrak;
```

GOTO işlemi

```
if (boyut > 12)
 {
 eder = eder * 1.05;
 bayrak = 2;
 }
 fatura = eder *bayrak;
```

Biraz Eğlence

1) Hangileri Doğru (true) veya yanlış (false)

   100 > 3 && 'a'>'c'

   100 > 3 || 'a'>'c'

   !(100>3)

2) Aşağıdaki şartları sağlayacak işlemleri yazınız

   a) sayı 90 ‘a eşit veya ondan büyük fakat 100 den küçük

   b)ch karakteri q değil veya k değil

   c) sayı 1 ve 9 arasında fakat 5 değil

Biraz Eğlence

3) Sonuçları bulunuz

- 3 + 4 > 2 && 3 < 2
- x >= y || y > x
- d = 5 + (6 > 2)
- 'X' > 'T' ? 10 : 5
- x > y ? y > x : x > y

Biraz Eğlence

4) Tam sayıları giriş alan “0” girildiğinde sonlanan bir program yazınız. Bu program girilen tam sayılar içinden, toplam kaçının tek olduğunu bulsun, tek ve çift sayıların ortalamasını çıktı olarak versin

5) Metin okuyan bir program yazınız, program # ile sonlansın. Program metin içinde kullanılan “e” harflerinin toplam sayısını bulsun.

6) Bir lokanta için menü programı hazırlayınız.

****

Lezzet lokantasına Hoş geldiniz

****

Otomatik sipariş için 1’e, garson çağırmak için 2’ye basınız.

Menüden çıkmak için 3’e basınız.

****

(1’e basıldı ise)

Lütfen aşağıdaki menülerden birini seçiniz

1) Soğuk ve sıcak mezeler
2) Çorbalar
3) Ev yemekleri
4) Kebap türleri
5) Tatlılar
6) İçecekler

(2’ye basıldığını farz edelim)

****

Çorba menüsüne hoş geldiniz,
Seçeneklerimiz aşağıda verilmiştir.

a) Ezo gelin
b) Mercimek
c) İşkembe
d) Sebze çorbası

****

(Ek puan: Seçimi çıktı olarak yazınız.)

Fonksiyonlar

- Fonksiyon belli bir gorevi yerine getirmek icin dizayn edilmis,kendi icinde yeterli, program parcalarina denir.
- Neden: Sizi tekrardan kurtarir, bir islemi defalarca yapacaksaniz fonksiyon kullanin. Ana program icinde istediginiz kadar cagirin. Baska programlar icinde de kullanabilirsiniz.
- Fonksiyionlar kara kutular olarak dusunulebilirler ve kendilerine gonderilen ve kendilerinden alinan bilgiler ile tanimlanirlar. Iclerinde neyin nasil yapildigi fonksiyonu ilgilendirir. Ornegin printf() fonksiyonu: biz sadece nasil kullanilacagini biliyoruz, icinde neler oldugu bizi fazla alakadar etmiyor.

Fonksiyonlar

- Ana programi anlamli is parcalarina ayirin: her bir parca bir isi icra etsin.
- Her bir parcanin program ile olan iliskisini tespit edin (programdan ne aliyor, programa ne geri veriyor)
- Fonksiyon nasil tanimlanir, nasil cagrilir, birbirleriyle nasil haberlesirler.

```
#include <stdio.h>
#define SON 40
 void yildizlar(void); /* fonksiyon prototipi, deklerasyon */
 int main(void)
{
 yildizlar();
 printf("%s\n", Benim adin Alpaslan Duysak);
 yildizlar(); /* fonksiyonun kullanimi */
 return 0;
 }
////////////////////////////////////////////////////////////////////////////////////////////////////
 void yildizlar(void) /* fonksiyonun tanimlanmasi */
 {
 int count;
 for (count = 1; count <= SON; count++)
 putchar('*');
 putchar('\n');
 }
```

Ornegin Analizi

- Fonsiyon prototipi: Compilar a ne tip bir fonksiyon oldugunu soyler
- Fonksiyonun kullanimi: fonksiyon cagrilir ve sonucu alinir
- Fonksiyonun tanimlanmasi: ne is yapacagi tanimlanir.
- Degiskenlerde oldugu gibi, her fonksiyonunda bir turu olamalidir.

void yidizlar(void);

- () parantezler bunun bir fonksiyon oldugunu belirtir

Ornegin Analizi

- Ilk kullanilan void fonksiyonun turunu ifade eder (bu ornek icin fonksiyon bir deger geri dondurmuyor, void kullanilir)
- Parantez icindeki void fonksiyonun bir arguman almadigini ifade eder.
- ; ise fonksiyonun deklare edildigini ifade eder (tanimlandigini degil)
- Fonksiyon deklerasyonu main() den hemen once gelir. Main() icine de, degiskenlerin tanimlandigi yer, olabilirler.

Ornegin Analizi

- main() icinde fonksiyonlar adlari ile cagrilirlar,

  yildizlar();

- Fonksiyon icra edilir ve cagrildigi yere geri doner, isminden sonraki satirdan program devam eder.
- Ana program ve fonksiyon ayni file icinde olabilirler, bu durumda compile etmek daha kolaydir.
- Farkli dosyalar icinde olabilirler, bu durumda da fonksiyonu farkli programlarin kullanmasi kolaydir.

Fonksiyon Argumanlari

void dubs(int x, int y, int z) ;

3 tane rguman aliyor: x,y,z

void show_n_char(char ch, int num);

void show_n_char(char, int);

Kullanimi:

show_n_char(SPACE, 12);

Argumanlari; SPACE ve 12

```
#include <stdio.h>
#include <string.h>                                   /* strlen() */
#define SON 40
#define SPACE ' '
void show_n_char(char ch, int num);
int main(void)
{
    int spaces;
    show_n_char('*', SON);                            /* arguman bir sabit */
    putchar('\n');
    show_n_char(SPACE, 12);                           /* sabit */
    printf(" Alpaslan Duysak\n");
    printf("\n");
    spaces = SON - 28;                                /* arguman hesaplaniyor */
        show_n_char(SPACE, spaces);                   /* degisken bir arguman */
    printf(" Ya Sizin ki\n");
    show_n_char('*', SON);
    return 0;
}
void show_n_char(char ch, int num)
{
    int count;
    for (count = 1; count <= num; count++)
        putchar(ch);
}
```

Fonksiyonun deger geri dondurmesi

- Int fonksiyon (float a, float b)

Int: dondurulen degerin turu, fonksiyon iki float data aliyor ve bir integer datayi ana programa geri donduruyor.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int imin(int, int);
int main(void)
{
    int say1, say2;
    printf("Bir cift sayi giriniz(q cikis) :\n");
    while (scanf("%d %d", &say1, &say2) == 2)
    {
printf("Iki sayidan %d ve %d kucugu %d.\n", say1, say2, imin(say1,say2));
printf(" Bir cift sayi giriniz (q cikis):\n ");
    }
    return 0;
}
int imin(int n, int m)
{
    int min;
    if (n < m)
        min = n;
    else min = m;
    return min;
}
```

- kucuk=imin(sayi1,sayi2); evet
- Kucuk=min; hayir
- y=2+2*imin(sayi1,sayi2)+25; evet
- Return turu ile fonksiyon geri dondurme turu farkli ise….
- Fonksiyon illk return gordugunde cagrildigi yere geri doner. Birden fazla return kullanabilirsiniz.

- Fonksiyonlar geri dondurdukleri tur ile ayni tur olarak deklare edilmeliler
- Fonksiyon geri bir sey dondurmuyorsa void olarak tanimlanmali
- int imax(int, int); evet
- int imax(int a, int b); evet
- Arguman kullanilmiyorsa void yazilmali

```
#include <stdio.h>
int imax(int, int); /* prototip, deklerasyon */
int main(void)
{
   printf("Iki sayinin %d ve %d buyugu %d.\n", 3, 5,
   imax(3.0, 5.0));
        return 0;
}
int imax(int n, int m)
{
   int max;
   if (n > m)
        max = n;
   else max = m;
   return max;
}
```

- Kisa fonksiyonlarda deklerasyon yerine fonksiyon tanimi konulabilir:

```
int imax(int a, int b) { return a > b ? a : b; }
 int main()
{
 ...
z = imax(x, 50);
 ...
 }
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int menu_choice(void);
main() {
    int choice;
    choice = menu_choice();
    printf("Seceneginiz: %d\n", choice);
    return 0;
}
int menu_choice(void)
{
    int selection = 0;
    do
    {
        printf("\n");
        printf("\n1  Kayit Ekle");
        printf("\n2 - Kayit Degistir");
        printf("\n3 - Kayit Sil");
        printf("\n4 - Cik");
        printf("\n");
        printf("\nSeciminizi girin: ");
        scanf("%d", &selection);
    } while (selection < 1 || selection > 4);
    return selection;
}
```

Recursion

- Bir fonksiyon keni kendini cagirabilir.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
void up_and_down(int);
int main(void)  {
        up_and_down(1);
        return 0;
}

void up_and_down(int n) {
        printf("seviye %d: n yer %p\n", n, &n); /* 1 */
        if (n < 4)
                up_and_down(n + 1);
        printf("seviye %d: n yer %p\n", n, &n); /* 2 */
}
```

```
#include <stdio.h>
void display(char cr, int lines, int width);
int main(void)
{
    int ch;                                  /* character to be printed    */
    int rows, cols;                          /* number of rows and columns */
    printf("Enter a character and two integers:\n");
    while ((ch = getchar()) != '\n')
    {
        scanf("%d %d", &rows, &cols);
        display(ch, rows, cols);
        printf("Enter another character and two integers;\n");
        printf("Enter a newline to quit.\n");
    }
    printf("Bye.\n");
    return 0;
}
```

```
void display(char cr, int lines, int width)
{
    int row, col;
    for (row = 1; row <= lines; row++)
   {
        for (col = 1; col <= width; col++)
            putchar(cr);
        putchar('\n');  /* end line and start a new one */
    }
}
```

```
#include <stdio.h>
void display(char cr, int lines, int width);
int main(void)
{
    int ch;             /* character to be printed      */
    int rows, cols;     /* number of rows and columns   */
    printf("Enter a character and two integers:\n");
    while ((ch = getchar()) != '\n')
    {
        if (scanf("%d %d",&rows, &cols) != 2)
            break;
        display(ch, rows, cols);
        while (getchar() !=  '\n')
            continue;
        printf("Enter another character and two integers;\n");
        printf("Enter a newline to quit.\n");
    }
    printf("Bye.\n");
    return 0;  }
```

```
void display(char cr, int lines, int width)
{
    int row, col;
    for (row = 1; row <= lines; row++)
    {
        for (col = 1; col <= width; col++)
            putchar(cr);
        putchar('\n');  /* end line and start a new one */
    }

}
```

```
#include <stdio.h>
char get_choice(void);
char get_first(void);
int get_int(void);
void count(void);
int main(void)
{
    int choice;
    void count(void);
    while ( (choice = get_choice()) != 'q')
    {
        switch (choice)
        {
            case 'a' :  printf("Buy low, sell high.\n");
                        break;
            case 'b' :  putchar('\a');  /* ANSI */
                        break;
            case 'c' :  count();

                        break;
            default  :  printf("Program error!\n");
                        break;
        }
    }
    printf("Bye.\n");
    return 0;   }
```

```
void count(void){
    int n,i;
    printf("Count how far? Enter an integer:\n");
    n = get_int();
    for (i = 1; i <= n; i++)
        printf("%d\n", i);
    while ( getchar() != '\n')
        continue;
}
char get_choice(void){
    int ch;
    printf("Enter the letter of your choice:\n");
    printf("a. advice          b. bell\n");       printf("c. count          q. quit\n");
    ch = get_first();
    while (  (ch < 'a' || ch > 'c') && ch != 'q')
    {
        printf("Please respond with a, b, c, or q.\n");
        ch = get_first();
    }
    return ch;
}
char get_first(void){
    int ch;
    ch = getchar();
    while (getchar() != '\n')
        continue;
    return ch;
}
int get_int(void){
    int input;     char ch;
    while (scanf("%d", &input) != 1)    {
        while ((ch = getchar()) != '\n')
            putchar(ch);  // dispose of bad input
        printf(" is not an integer.\nPlease enter an ");        printf("integer value, such as 25, -178, or 3: ");
    }
    return input;
}
```