BMM 111
Bilgisayar Programlama-I

3. Ders

Dr. Öğr. Üyesi Mustafa İSTANBULLU

Çukurova Üniversitesi
Mühendislik Fakültesi
Biyomedikal Müh. Böl.

E-mail:mm.istanbullu@gmail.com

Not: Slaytlar, kaynakça bölümünde verilen listeden faydalanılarak hazırlanmıştır.

Yazım ve Noktalama Kuralları

- C dilinde yazılan bir programın, derleyici tarafından anlaşılabilmesi amacıyla bazı noktalama işaretlerinin kullanılması gerekir. Bu kısımda, bu özel işaretlerden bazıları özetlenmektedir.

Noktalı Virgül

- C dilinde her komutun bittiği yer noktalı virgül " ; " işareti ile belirtilir. Bağımsız komutlar noktalı virgül kullanılarak birbirinden ayrılır. Böylece, noktalı virgül kullanılarak bir satıra birden fazla komutun yazılması mümkün olur. Örneğin;

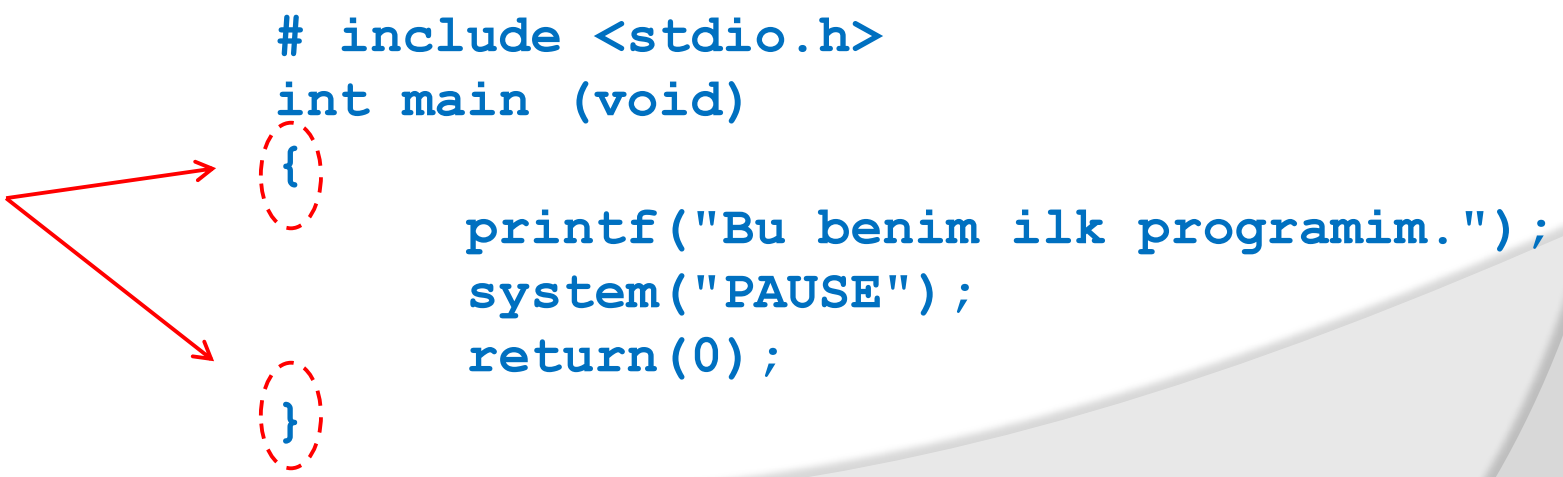
```
x=y;
y=y+1;
```

yerine, aşağıdaki gibi de yazılabilir:

```
x=y; y=y+1;
```

Ayraç İşaretleri

- C dilinin bloklardan oluştuğunu daha önce söylemiştik. Bloklar, “ { “ ve “ } ” ayraç işaretleri içinde belirtilen komutların bir araya gelmesi ile oluşur.
- Böylece, belirli bir amaca yönelik komutların bir blok içinde toplanması sağlanır.



```
# include <stdio.h>
int main (void)
{
        printf("Bu benim ilk programim.");
        system("PAUSE");
        return(0);
}
```

Açıklama Satırları

İyi Programlama Uygulaması 2.1

Her fonksiyon, kendisinden önce fonksiyonun amacını tanımlayan bir açıklamaya sahip olmalıdır.

- C dilinde yazılmış bir programın içinde istediğimiz herhangi bir yere açıklama yazabilirsiniz.
- Bunun için yapılması gereken tek şey açıklamanın başlangıç ve bitiş noktalarının belirtilmiş olmasıdır.
- Bu amaçla " /* " ve " */ " işaretleri kullanılır.
- Bu açıklamalar yürütülebilir (executable) komutlar değildir ve derleyici tarafından programın bir parçası olarak görülmezler. Kullanıcılara programı anlatmak ve program ile ilgili açıklamaları göstermek amacıyla kullanılırlar.

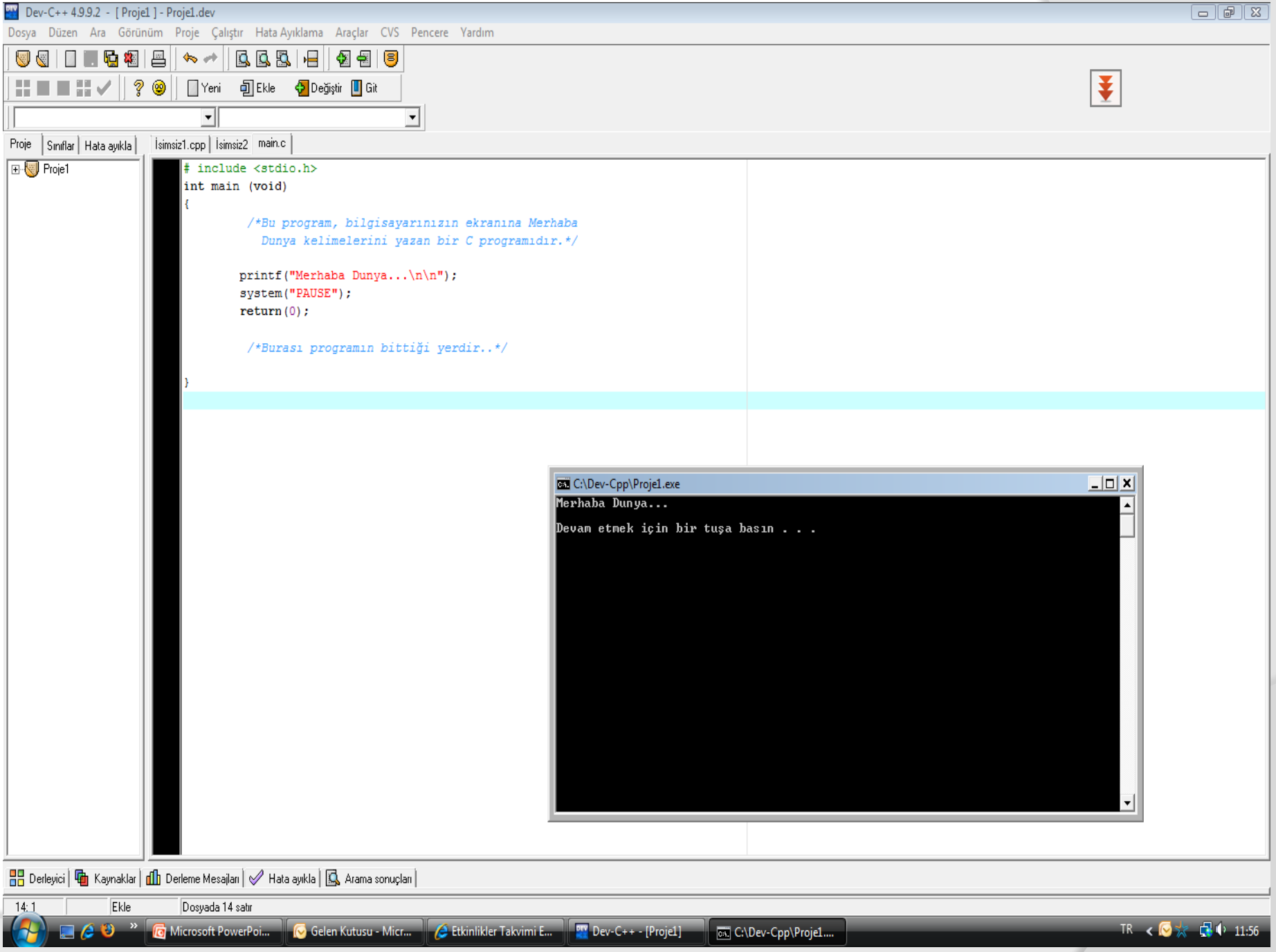
Örnek Program

```
# include <stdio.h>
int main (void)
{
        /*Bu program, bilgisayarınızın ekranına Merhaba
          Dunya kelimelerini yazan bir C programıdır.*/

      printf("Merhaba Dunya...");
      system("PAUSE");
      return(0);

        //Burası programın bittiği yerdir..
}
```

Kaçış dizisi	Tanım
\n	Yeni satır. İmleci sonraki satır başına konumlandırır.
\t	Yatay sekme. İmleci sonraki sekme noktasına kaydırır.
\a	Uyarı. İmlecin o andaki konumunu değiştirmeden bir ses veya görülebilir uyarı üretir.
\\	Ters bölü. Dizi içine ters bölü karakterini yerleştirir.
\"	Çift tırnak. Dizi içine çift tırnak karakteri yerleştirir.


Dev-C++ 4.9.9.2 - [Proje1] - Proje1.dev
Dosya Düzen Ara Görünüm Proje Çalıştır Hata Ayıklama Araçlar CVS Pencere Yardım
Yeni Ekle Değiştir Git
Proje Sınıflar Hata ayıkla
İsimsiz1.cpp İsimsiz2 main.c
Proje1
include <stdio.h>
int main (void)
{
/*Bu program, bilgisayarınızın ekranına Merhaba
Dunya kelimelerini yazan bir C programıdır.*/
printf("Merhaba Dunya...\n\n");
system("PAUSE");
return(0);
/*Burası programın bittiği yerdir..*/
}
C:\Dev-Cpp\Proje1.exe
Merhaba Dunya...
Devam etmek için bir tuşa basın . . .
Derleyici Kaynaklar Derleme Mesajları Hata ayıkla Arama sonuçları
14: 1 Ekle Dosyada 14 satır
Microsoft PowerPoi... Gelen Kutusu - Micr... Etkinlikler Takvimi E... Dev-C++ - [Proje1] C:\Dev-Cpp\Proje1....
TR 11:56

```
1  // Fig. 2.3: fig02_03.c
2  // Printing on one line with two printf statements.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     printf( "Welcome " );
9     printf( "to C!\n" );
10 } // end function main
```

```
Welcome to C!
```

Fig. 2.3 | Printing one line with two `printf` statements.

```
1  // Fig. 2.4: fig02_04.c
2  // Printing multiple lines with a single printf.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     printf( "Welcome\nto\nC!\n" );
9  } // end function main
```

```
Welcome
to
C!
```

Fig. 2.4 | Printing multiple lines with a single `printf`.

Yazım Kuralları

- C dilinde program yazarken bazı komutları içeriye doğru girintili olarak yazarız.

```
# include <stdio.h>
int main (void)
{
        /*Bu program, bilgisayarınızın ekranına Merhaba
          Dunya kelimelerini yazan bir C programıdır.*/

        printf("Merhaba Dunya...\n\n");
        system("PAUSE");
        return(0);

        /*Burası programın bittiği yerdir..*/
}
```

- Aslında, C komutların nereye yazıldığı önemli değildir. Satırın neresine yazarsanız yazın, komutlar bilgisayar tarafından aynı şekilde anlaşılır.

- Ancak, yüzlerce satırdan oluşan programlar yazıldıkça, bunların okunaklığı ve anlaşılırlığı zorlaşır.
- Bu nedenle C dilinde yıllardan beri kullanılan geleneksel bir yazım stili oluşmuştur.
- Buna göre, bir bloğu oluşturan tüm komutlar ve açıklama satırları ayraç işaretlerine göre daha içeride yazılır.
- Böylece blokların birbirinden ayrılması ve yazılan program kodunun daha anlaşılır bir hale gelmesi sağlanır.
- Bu durumu göstermek üzere, aynı kaynak koduna sahip iki program göz önüne alınsın:

1. Örnek Program

```
# include <stdio.h>
int main (void)

{         /*Bu program, toplam ödemeyi hesaplar..*/

          float ucret, vergi, lux, toplam;
          lux=0.0;

          printf("Ucreti Giriniz: ");
          scanf("%f", &ucret);

          vergi=ucret*0.06;

          if (ucret>40000.0)
              lux=ucret*0.005;

          toplam=ucret+vergi+lux;

          printf("Toplam Odeme %0.2f", toplam);
          system("PAUSE");
          return(0);

}
```

2. Örnek Program

```
# include <stdio.h>
int main (void){/*Bu program, toplam ödemeyi hesaplar..*/ float
ucret, vergi, lux, toplam; lux=0.0;
printf("Ucreti Giriniz:");scanf("%f", &ucret); vergi=ucret*0.06;if
(ucret>40000.0)lux=ucret*0.005;
toplam=ucret+vergi+lux;printf("Toplam Odeme
%0.2f",toplam);system("PAUSE");return(0);}
```

- Görüldüğü gibi, ele alınan her iki programda aynı şekilde çalışmaktadır ve aynı işi yaparlar.
- Ancak, ilk örnekte verilen programın, ayraç işaretlerinin ve kaynak kodun düzenli bir biçimde hazırlanmış olması nedeniyle, okunaklığı ve anlaşılırlığı diğerine göre çok daha yüksektir.
- İkinci örnekte verilen programın ise okunaklığına dikkat edilmeden, komutlar ardışık olarak sıralanmıştır.
- Bu durum yazdığımız programın, gerek bizim tarafımızdan gerekse bizim dışımızdaki kişiler tarafından kolay anlaşılmasını sağlayacaktır.

C Kütüphaneleri

- **"Merhaba Dunya"** programımızı anlatırken, **printf ()** fonksiyonunun C derleyicisini geliştiren kişiler tarafından oluşturulduğunu söylemiştik.
- Bu fonksiyon, C dilinin bir parçası değildir, ancak, hemen hemen tüm C programlarında kullanılır.
- Öyleyse bu fonksiyon nereden gelmiştir ve neden böyle bir fonksiyona ihtiyaç duyulmuştur ?

- Bütün C derleyicileri, yoğun olarak kullanılan fonksiyonlardan oluşan bir standart kütüphaneye sahiptirler.
- Kütüphane içinde, kodu daha önce yazılmış, programcının kullanımına hazır fonksiyonlar bulunur. Böylece bu fonksiyonların her defasında tekrar tekrar yazılması engellenmiş olur ve daha standart bir yapıda yaygın olarak kullanımı sağlanır.
- Ancak standart kütüphaneler dışında da bir çok kütüphane, yapılan işin özelliğine göre C programına dahil edilebilir ve bu kütüphaneye ait özel fonksiyonlar program içinde kullanılabilir.
- Herhangi bir kütüphane içinde tanımlanmış olan bir fonksiyonun program içinden kullanımının sağlanması için ilgili kütüphanenin mutlaka programa tanıtılması gerekir.
- Bunun için `#include` ifadesi kullanılarak bir tanımlamanın yapılması gerekir.

- Örneğin, `printf()` fonksiyonu standart kütüphane `<stdio.h>` da tanımlandığından, bu fonksiyonu kullanmak için programın başında bu kütüphaneyi aşağıdaki şekilde tanıtmamız gerekir.

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

- Bu kütüphanenin dışında, örneğin matematiksel işlemlerin kolaylıkla yapılmasında kullanılmak üzere (`<math.h>` ve `<string.h>` gibi) birçok farklı kütüphane bulunmaktadır.

- Yazılacak olan programın ihtiyacına göre bu farklı kütüphanelerden yararlanılması mümkündür.

C Dilindeki Sözcükler

- C dilindeki sözcükler, bir C programının en temel elemanlarından birisidir. Bu sözcükleri daha iyi anlayabilmek için kullanıcıdan yarıçap değerini alan ve bu değeri kullanarak bir dairenin alanını hesaplayan aşağıdaki C programını inceleyelim..

```
/*Bu program, dairenin alanını hesaplar..*/        açıklama
# include <stdio.h>                               kütüphane
# define PI 3.14                                  isim sabiti
int main (void)                                   ana fonksiyon
{
        double yaricap, alan;                     tanımlama komutu
        printf("Yaricapi giriniz:");
        scanf("%lf", &yaricap);
        alan=PI*yaricap*yaricap;
        printf("Alan=%lf", alan);
        return(0);
}
```

Yuvarlak içine alınanlar: özel amaçlı sözcükler.
Altı çizgili olan sözcükler: mor olan tanıtıcı isimleri,
mavi olanlar: değişken isimleri, siyah olanlar: fonksiyon isimleri.

2. Bölüm C Dilinin Temelleri

- C dilinde, (A-Z, a-z, 0-9 ve _) gibi karakterler bazı sistematik kurallar çerçevesinde bir araya gelerek sözcükleri oluştururlar.
- C dilindeki sözcükler, özel amaçlı sözcükler ve tanıtıcılar olarak 2'ye ayrılırlar.
- Şimdi, bu elemanların anlamı ve C dili içinde kullanımını inceleyelim.

Özel Amaçlı Sözcükler

- C dilindeki özel amaçlı sözcükler (reserved words) derleyici tarafından kullanılan özel kelimelerdir.
- Bunlar, program içinde doğru yerlerde ve doğru şekilde kullanılmak zorundadırlar.
- Bu sözcüklerin her birinin derleyici için özel bir anlamı vardır.
- Standart C dilinde, toplam 32 adet özel amaçlı sözcük bulunmaktadır.
- Ancak, derleyici yazan firmalar, bu sözcükleri kendi derleyicileri için yenilerini ekleyebilmektedirler.
- Özel amaçlı sözcükler, ilerleyen zamanlarda tek tek kullanılacaktır.

Özel Amaçlı Sözcükler

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

Tanıtıcılar

- Bir program yazılırken bu özel amaçlı sözcüklerin dışında, birçok tanımlamanın da yapılması gerekir.
- Bu tanımlamalar tanıtıcılar (identifiers) olarak isimlendirilir.
- Tanıtıcılar bazen programcı tarafından programın ihtiyacına göre çeşitli şekillerde tanımlanabilirken, bazen de C dilinin kütüphanelerinde bulunan isimler olabilirler.
- Örneğin, daha önce gördüğümüz `printf()` fonksiyonu ve diğer standart fonksiyon isimleri, kütüphaneler içinde yer alan tanıtıcılardır.

- C dilinde kullanılan tanıtıcıların isimlerinin geçerli olabilmesi için aşağıdaki kurallara uygun olarak oluşturulması gereklidir:

1. Tanıtıcı içinde harf (a ..z, A .. Z), sayı (0 .. 9) veya alt çizgi (underscore "_") bulunabilir.

2. Bir tanıtıcı, bir harf ya da alt çizgi "_" işareti ile başlamalıdır.

3. Tanıtıcı içinde özel karakterler (#, &, ö, ş, v.b.) bulunamaz.

4. Tanıtıcı, C dilinde kullanılan özel amaçlı sözcüklerden biri olamaz.

5. C dili büyük-küçük harf duyarlı (case-sensitive) bir dildir.

- Bir tanıtıcıyı belirlerken, kullanım alanına göre anlamlı bir sözcüğün seçilmesi, programın okunabilir ve anlaşılabilir olmasını sağlar.

- Örneğin, toplamını bulmak için kullanacağımız bir tanıtıcı ismini **`xyz`** gibi bir isim olarak tanımladığımızda, daha sonra bu tanıtıcının hangi amaçla kullanıldığını anlamamız çok zor olacaktır.

- Bunun yerine **`toplam`** isimli bir tanıtıcı kullanmamız daha anlamlı olacaktır.

- C dilinde büyük ve küçük harflerin farklı anlamlar içerdiğinden bahsetmiştik. Buna göre,aşağıdaki sözcüklerden herbiri farklı bir tanıtıcıyı tanımlamaktadır:

`Toplam` **`toplam`** **`TOPLAM`** **`toplaM`**

Değer Sabitleri

- C dilinde kullanılan diğer bir eleman değer sabitleri (literal constant) dır.
- C dilinde tamsayı, reel sayı, karakter ve dizgi sabitleri gibi birçok farklı değer sabiti tanımı bulunmaktadır.

Tamsayılar

- Tamsayı sabitleri (integer constants) 0-9 rakamlarından oluşan ve ondalık değerleri olmayan sayılardır.
- Tamsayı sabitleri 0 ile başlayamaz. Bu sabitler pozitif yada negatif olarak tanımlanabilirler. Bu durumda sayının başına "+" ya da "-" işareti konulur.
- İşaret kullanılmaması durumunda tamsayı sabiti pozitif olarak kabul edilir.

- Örneğin, aşağıdaki değerler geçerli birer tamsayı sabitleridir.

```
8          +345          -58
```

- Aşağıdaki değerler ise geçersiz tamsayı sabitleridir.

`06`	Tamsayı sabitleri sıfır ile başlayamaz…
`124,7`	Tamsayı sabitleri özel karakter içeremez.
`2.4`	Tamsayı sabitlerinin ondalık değeri yoktur.

Reel Sayılar

Reel sayı (gerçek sayı) sabitleri, tam ve ondalık kısmı olan sabitlerdir. Kayan Nokta (floating point) Sabitleri olarak da adlandırılırlar.

- Örneğin, `+5.7` bir reel sayı sabitidir.
- Buradaki `5` reel sayının tam kısmını, `7` ise ondalık kısmını temsil eder.
- Reel sayı sabitlerinde, sayının tam ve ondalık kısmı nokta "." ile birbirinden ayrılır.
- Tıpkı tamsayı sabitleri gibi reel sayı sabitleri de pozitif ya da negatif olarak tanımlanabilir.
- Reel sayı sabitlerinde, eğer noktadan sonraki kısım yazılmazsa, sıfır olarak kabul edilir.

- Aşağıdaki değerler geçerli birer reel sayı sabitidir.

```
23.823      +97.2      -568.0          0.0          .52
```

- Aşağıdaki değerler ise geçersiz reel sayı sabitleridir.

`213` Nokta kullanılmamış..

`124,7` Ondalık kısım nokta ile ayrılmalıdır.

- Bunların dışında, bilimsel çalışmalarda, büyük sayıları gösterebilmek amacıyla, 10 üzeri gösterimler de kullanılır.
- Örneğin, 23.898 sayısı, 0.23898×10^2 olarak da gösterilebilir.
- Aynı sayıyı, matematiksel olarak 2.3898×10^3 ya da 238.98×10^{-1} olarak da gösterilebilir.
- C dilinde 10 üzeri gösterimleri tanımlamak şu ana kadar öğrendiğimiz yapılar ile mümkün değildir.
- Bu amaçla bilimsel gösterim şekli kullanılır.
- Bu gösterime göre, 10 üzeri tanımlaması `e` yada `E` sembolü kullanılarak gerçekleştirilir. Bu sembolü izleyen tamsayı, 10 üzeri olarak tanımlamak istediğimiz üst için kullanılır.

- Şimdi aşağıdaki örnekleri inceleyelim:

Matematiksel Gösterim	C Dilinde Kullanım
0.23898×10^{2}	0.23898e2
2.3898×10^{3}	2.3898E3
238.98×10^{-1}	238.98e-1

- Aşağıdaki gösterimler ise yanlış birer bilimsel gösterimdir:

Gösterim	Açıklama
.45e	e'nin tamsayı değeri tanımlanmamış
21e-0.3	e'nin tamsayı değeri (üst) yanlış tanımlanmış, reel sayı olamaz
12,7e5	virgül yerine nokta kullanılmalıdır

Karakterler

- C dilinde tek tırnak işareti arasında bulunan karaktere, karakter sabiti (character constant) adı verilir.
- Burada karakter olarak tanımlanabilecek olan değerle ASCII kod tablosundaki gösterimlerden birisi olabilir.
- Örneğin, aşağıdaki değerler geçerli birer karakter sabitidir:

```
'a'        '2'        ','        '␣'
```

- Bu listenin en sonunda dikdörtgen işareti ile gösterilmiş olan sembol, boşluk karakterini temsil eder. Bu örnekte, tek karelik bir boşluk alanı tanımlanmıştır.

Karakterler

- Aşağıdaki değerler ise geçersiz karakter sabitlerini göstermektedir:

`'abc'` : Karakter sabiti sadece tek bir karakter değeri için tanımlanabilir.

`'' a ''` : Karakter sabiti tek tırnak işaretleri ile tanımlanabilir.

`A` : Karakter sabitinin tek tırnak işaretleri ile tanımlanması gerekir.

Dizgiler

- Bu sabitler çift tırnak işareti “ ” ile tanımlanırlar ve birden fazla karakterin bir araya gelmesinden oluşurlar.
- Aşağıdaki değerler geçerli birer dizgi sabitleridir:

```
“Ali”    “aa12aa”    “Bengisu”    “2”    “□”
```

- Bir dizgi sabiti (string constant) birden çok karakter içerebileceği gibi, tek bir karakterden de oluşabilir.
- Aşağıdaki değerler geçersiz birer dizgi sabitidir.

```
‘merhaba’:Dizgi sabitleri çift tırnak ile tanımlanır.
C dersi :Dizgi sabitleri çift tırnak ile tanımlanır.
“Örnek” :Dizgi sabitleri ASCII tablosunda yer almayan
 karakterleri içermez.
```

- Şimdi aşağıda tanımlanmış olan sabitlerin ne anlama geldiğini anlamaya çalışalım.

Sabit	Açıklama
`4.0`	`Reel Sayı Sabiti`
`4`	`Tamsayı Sabiti`
`"4"`	`Dizgi sabiti`
`'4'`	`Karakter Sabiti`

Veri Tipleri

Veri Tipi	Açıklama	Bit	Aralık
int	Tamsayı (integer)	16 32	-32768 ile 32767 -2147483648 ile 2147483648
double	Çift duyarlı reel sayı	64	Yaklaşık 12 basamak duyarlı
char	Karakter	8	0 ile 255
short int	Kısa tamsayı	8	-128 ile 127
long int	Uzun tamsayı	32	-4294967296 ile 4294967295
unsigned int	İşaretsiz tamsayı	16	0 ile 65535
float	Reel Sayı	32	Yaklaşık 6 basamak duyarlı

Veri Tipleri

- Veri tipleri, yukarıda verilen sabit türlerine özgü tanımlamaların yapılması amacıyla kullanılır.
- Bu veri tiplerinin özellikleri ve bellekte kapladıkları alanlar birbirinden farklıdır. Ancak, veri tiplerinin bu özellikleri kullanılan teknoloji altyapısına ve derleyici özelliklerine göre de farklılık göstermektedir.

Tamsayı Veri Tipi - `int`

- Tamsayı değerleri C programı içinde **`int`** özel amaçlı sözcüğü kullanılarak tanımlanır.
- Tamsayı veri tipleri, tamsayı sabitlerinde anlatıldığı gibi, ondalık kısmı olmayan sayıların bellekte saklanması için kullanılır.

Tek/Çift Duyarlı Reel Sayı Veri Tipi – `float, double`

- Çift duyarlı reel sayılar ile yaklaşık 12 basamak duyarlı reel sayıların tanımlanması sağlanır.

- Genelde, bu veri tipi bellekte toplam 8 bayt (64 bit) lik bir alan kullanır. Bu veri tipine ait değişkenlerin tanımlanabilmesi için **`double`** özel amaçlı sözcüğü kullanılır.

- Reel sayı (**`float`**) veri tipi ise, çift duyarlıklı reel sayılara (double) oranla daha küçük sayılarla işlem yapılması gerektiği durumlarda kullanılmalıdır.

Karakter Veri Tipi – `char`

- Bir karakter veri tipinin tanımlanabilmesi için C dilinde **`char`** özel amaçlı sözcüğü kullanılır.
- Bu veri tipi ile bir bayt uzunluğunda (8 bit) tek bir karakter değerinin tanımlanması mümkün olur. Karakter veri tipi ile karakter değerlerinin bellekte saklanması sağlanır.
- Her karakter bir tamsayı değeri ile temsil edilir. Karakterler arasında yapılan karşılaştırma işlemleri sırasında bu tamsayı değerleri kullanılır.
- Örneğin 'J' karakterinin tamsayı karşılığı 74 iken, 'V' karakterinin karşılığı 86 dır. Yani 'J' karakteri, 'V' karakterinden daha önce sıralanmıştır.

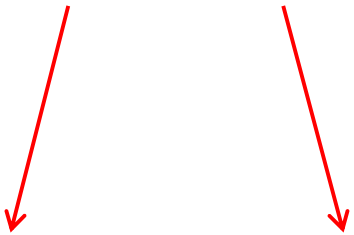
Değişkenler

- Değişkenler (variables) bir program içinde kullanılan temel nesnelerdir.
- Bir programda farklı tipteki veriler (karakter, sayı gibi) program içinde kullanılmak üzere bellekte tutulmalıdır.
- İşte, verileri bellekte saklamak için kullanılan bu bellek alanlarına ulaşmak, bellek hücrelerine verilen isimlerle mümkündür.
- Bu isimlerle bellek hücrelerine veri aktarabilir veya saklanan verileri bu hücrelerden alarak program içinde kullanabiliriz.
- Bu hücrelerin içeriği, program akışı sırasında değiştirilebileceğinden, değişken olarak adlandırılırlar.
- Bellek hücrelerine programcı tarafından verilen isimler doğru bir şekilde tanımlanmış birer tanıtıcı ismi olmalıdır.

Değişken Tanımı

- C dilinde kullanılan bütün değişkenler kullanılmadan önce mutlaka tanımlanmalıdır.
- Bu tanıtım sırasında değişkenin ismi ile birlikte saklayabileceği değerlerin veri tipinin belirtilmesi gerekir.
- Örneğin, tamsayı değeri alabilecek olan `toplam` isimli bir değişkeni tanımlamak için aşağıdaki tanıtım komutu kullanılır.

```
int toplam;
```



Veri tipi değişken ismi

- Bu komut derleyiciye bellekte `toplam` isimli bir hücrenin tamsayı değerler alabilecek şekilde oluşturulması gerektiğini söyler.
- Böylece derleyici, `toplam` değişkeni için gerekli büyüklükteki bellek alanını rezerve eder.
- Bu değişkenin sakladığı değerin `10` olduğunu düşünelim.
- Bilgisayarımız bellekte `toplam` değişkeni için bir yer ayıracak ve `toplam` değişkeni için tanımlanan `10` değerini de bellekte bu alana yerleştirecektir.
- Aslında, bu değişkenin bellekte tutulduğu alanın ilk adresi bilgisayar tarafından işaretlenir ve daha sonra bu değişkene ait değere ulaşılmak istendiğinde bu adres kullanılır.
- Ancak, programcılar tarafından bu adres çok da anlamlı değildir ve hatırlanması zordur. Bu nedenle, değişkenlerin içerdiği değerlere ulaşabilmek için değişkenlere verilen isimin kullanılması tercih edilir.

- Bu isimler genelde değişkenimizin program içindeki göreviyle ilişkilendirilir ve kolay hatırlanabilir bir isimdir.
- Bu durumda bilgisayarımız, değişkene bizim verdiğimiz bu ismi, kendisinin verdiği adres ile birlikte tutar ve böylece değişkenin değerine erişim sağlanır.
- Aşağıda da görüldüğü gibi bellek hücresinde tutulan `10` değerine, bu hücre için tanımladığımız değişken ismi ile ulaşmamız mümkün olur.

Bellek görüntüsü

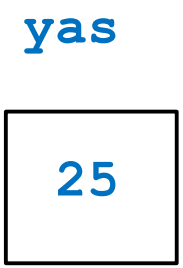
- Değişkenleri daha iyi anlayabilmek için şimdi aşağıdaki örnekleri birlikte inceleyelim.

```
int yas;
```

'`int yas;`' ile bellekte sadece tamsayı veri tipindeki değerleri saklayabilecek bir alan ayrılır.

Bu bellek hücresine **`yas`** değişken ismi ile erişim sağlanır. **`yas`** değişkeninin değeri henüz belli değildir. Bu nedenle '?' ile gösterilmiştir.

- Daha sonra **yas** değişkeninin değeri örneğin **25** olarak atandığında aşağıdaki gibi saklanacaktır



- Aynı veri tipli değişkenler için tek bir tanımlama cümlesi yazmak yeterlidir. Örneğin;

```
int a;
int b;
```

yerine

```
int a, b;
```

yazılabilir.

```c
// Fig. 2.5: fig02_05.c
// Addition program.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   int integer1; // first number to be entered by user
   int integer2; // second number to be entered by user

   printf( "Enter first integer\n" ); // prompt
   scanf( "%d", &integer1 ); // read an integer

   printf( "Enter second integer\n" ); // prompt
   scanf( "%d", &integer2 ); // read an integer

   int sum; // variable in which sum will be stored
   sum = integer1 + integer2; // assign total to sum

   printf( "Sum is %d\n", sum ); // print sum
} // end function main
```

Fig. 2.5 | Addition program. (Part 1 of 2.)

KAYNAKÇA

- Prof.Dr. İbrahim DEVELİ, Bilgisayar Programlama Ders Notları, Erciyes Üniv. Elektrik-Elektronik Müh. Böl.
- H.Turgut UYAR, Programlamaya Giriş Ders Notları,İTÜ, 2004.
- Fedon KADİFELİ,Standart C Programlama Dili, (Tercüme),2000.
- Doç. Dr. Soner ÇELİKKOL, Programlamaya Giriş ve Algoritmalar, Murathan Yayınevi, TRABZON; 2009
- N. Ercil Çağıltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Çeşitli kişilerin internette paylaşıma açtığı notlardan faydalanılmıştır.