ALGORİTMA VE PROGRAMLAMA I

Yrd. Doç. Dr. Deniz KILINÇ
deniz.kilinc@cbu.edu.tr

YZM 1101

Celal Bayar Üniversitesi Hasan Ferdi Turgutlu
Teknoloji Fakültesi

Genel Bakış…

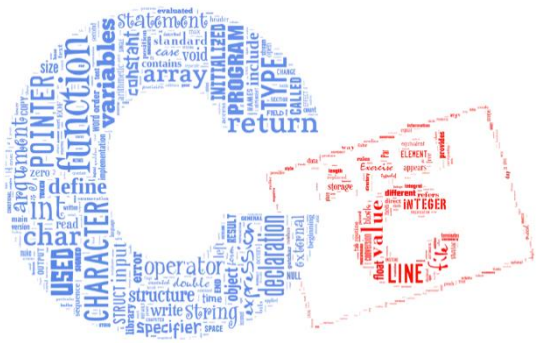
- C Programlama Dili Tarihçesi
- C Programlama Dili Tercih Nedeni
- İlk C Programı
- C Kodlarının Temel Özellikleri
- Code::Blocks
- Code:Blocks ile İlk C Programı
- C Programlama Dili Elemanları
  - Tanımlayıcılar
  - Anahtar Sözcükler
  - Veri Türleri
  - Değişkenler
  - Sabitler
  - Operatörler

1. BÖLÜM

C PROGRAMLAMA DİLİNE GİRİŞ

C Programlama Dilinin Tarihçesi

- C programlama dili, geniş amaçlı, orta seviyeli, yapısal bir programlama dilidir.
- C, temelde iki eski dile dayanır : BCPL ve B.
- BCPL, 1967 yılında Martin Richards tarafından işletim sistemleri ve derleyiciler yazmak için geliştirilmiştir.



- C programlama dili 1972 yılında Bell Laboratuvarlarında Dennis Ritchie tarafından geliştirilmiştir.
- C programlama dili, UNIX işletim sisteminin kodlama ve geliştirilmesinde kullanılarak popülerliğini arttırmıştır.

C Programlama Dilinin Tarihçesi (devam…)

- 1970'lerin sonunda C, şu anda geleneksel C olarak bilinen haline geldi. 1978 yılında Kernighan ve Ritchie tarafından yazılan, «The C Programming Language» adlı kitabın yayınlanmasından sonra, C'ye olan ilgi artmıştır.
- ANSI C, C programlama dilinin, 1989 yılında Amerika'da «American National Standards Institute (ANSI)» ve tüm dünyada «Internationals Standards Organization (ISO)» tarafından standart hale getirilmiş sürümüdür.
- ANSI C standardı, 1989 yılında onaylanmış, 1999 yılında gözden geçirilmiş ve Mart 2000'de C99: ISO/IEC 9899:1999 standardı Standart C olarak kabul edilmiştir

C Programlama Dili Tercih Nedeni

- C, en popüler dillerdin birisidir.
- C, güçlü ve esnek bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafikler çizebilirsiniz.
- C, yazılım geliştirme ortamları oldukça fazladır.
- C, özel komut ve veri tipi tanımlamasına izin verir.
- C, taşınabilir bir dildir.
- C, gelişimini tamamlamış ve standardı oluşmuş bir dildir.
- C, yapısal bir dildir. C kodları fonksiyon olarak adlandırılan alt programlardan oluşmuştur.
- C++, Java, JavaScript, JavaApplet, PHP, C# gibi diller C dilinden esinlenmiştir.

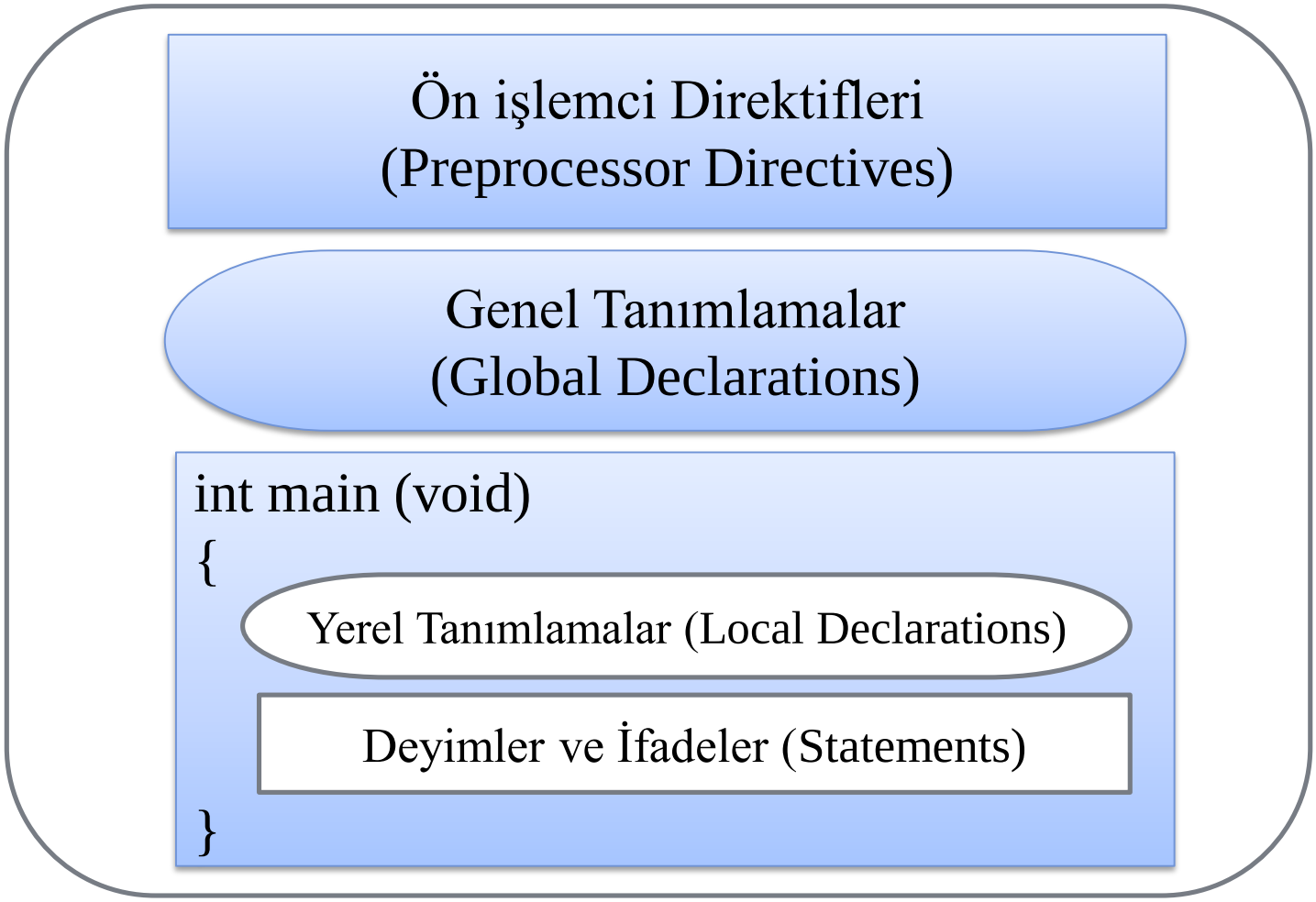
C Dili Taşınabilirdir (Portable) !!!

- C dili, donanımdan ve işletim sisteminde bağımsızdır.
- C dili ile dikkatli bir biçimde yazılmış bir program, her bilgisayara taşınabilir.
  - Yani herhangi bir C programı hiçbir değişikliğe uğramadan, veya çok az bir değişimle, başka bir derleyicide ve/veya işletim sisteminde derlenebilir.
- *Sonuç olarak Windows işletim sistemlerinde yazılan bir C kodu, Linux, UNIX veya VAX gibi işletim sistemlerinde de derlenebilir.*

C Sistemleri ve Kütüphaneleri

- C sistemleri üç kısımda oluşur:
  1. Programlama ortamı
  2. C programlama dili
  3. C standart kütüphaneleri
- C programları «fonksiyon» adı verilen parçalardan ya da modüllerden oluşur.
- Fonksiyonlar C «bloklarından» oluşur.
- Her fonksiyon/blok bir veya daha fazla «deyimi» içerir.
- Her bir deyim program çalıştırıldığında belirli bir eylemi yerine getirir. Deyimler işlemleri yerine getiren komutlardır.

C Yazılımı İçeriği



İlk C Programı


```c
/* Main.c : ilk C programımız */
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    printf("Hello World!\n");
    return 0;
}
```

- Program çalıştırılması sonucunda ekrana «Hello World!» yazmaktadır.

```
Hello World!

Process returned 0 (0x0)   execution time : 0.005 s
Press any key to continue.
```

1.Satır: Yorumlar /* …. */ Comments

- 1.satır /* ile başlayıp */ ile bitmektedir. Bu işaretler arasına yorum satırları yazılır.
- Yorumlar, okunabilirliği artırmak amacıyla kullandığımız bir özelliktir. Programcı notları olarak da düşünülebilirler.
- Yorum satırlarında bilgisayar hiçbir işlem yapmaz çünkü C derleyicileri bu satırları atlar.
- Dolayısıyla, yorum satırları için makine diline çevrilmiş kodlar oluşturulmaz.

1.Satır: Yorumlar /* …. */ Comments (devam...)

- Yorumlar, diğer yazılımcıların programınızı anlamasında yardımcı olur ancak çok fazla yorum satırı programın ve kodun okunabilirliğini azaltır.
- Tek satır olan yorumlar için // karakteri de kullanılabilir.
- Yorumlar programın herhangi bir yerinde olabilirler.
- İç içe açıklamalara izin verilmez.
  - /* yorum deneme /* …. */ …..*/
- Program başlangıcının; programın yazılma amacı, yazarı, yazılma tarihi vb. yorumlarla olması programlama sürecine adaptasyon noktasında yararlı olabilir.

Yorum Satırı Örnekleri


```
/* Tek satır yorum */

// Tek satır yorum

/* İki satır yorum
                örneği */

/* Birden fazla
     satır yorum
          örneği */

Toplam = X + Y; // X ve Y toplanarak Toplam değişkenine atanır
```

2. ve 3. Satır: Ön işlemci Direktifleri (#include)

- # işaretiyle başlayan satırlar, program derlenmeden önce C ön işlemcisi tarafından işlenirler.
- Bu satır, ön işlemciye standart giriş/çıkış başlık dosyasının yani «stdio.h» dosya içeriğinin programa eklemesini söyler.
- Bu başlık dosyası, derleyicinin «printf» gibi standart giriş/çıkış kütüphane fonksiyonlarının (**ST**andar**D-I**nput-**O**utput) derlerken kullanabileceği bilgi ve bildirimleri içerir.
- Başlık dosyalarının uzantısı .h dir.

2. ve 3. Satır: Ön işlemci Direktifleri (#include) (devam…)

- Başlık dosyaları, derleyicinin kütüphane fonksiyonu çağrılarının doğru yapılıp yapılmadığını anlamasında yardımcı olan bilgiler içerir.
- ANSI C'deki standart başlık dosyaları şunlardır:

❖ assert.h	❖ locale.h	❖ stddef.h
❖ ctype.h	❖ math.h	❖ stdio.h
❖ errno.h	❖ setjmp.h	❖ stdlib.h
❖ float.h	❖ signal.h	❖ string.h
❖ limits.h	❖ stdarg.h	❖ time.h

2. ve 3. Satır: Ön işlemci Direktifleri (#include) (devam…)

- «stdio.h» başlık dosyasının eklenmesi tercihe bağlıdır fakat standart giriş/çıkış fonksiyonlarının kullanıldığı programlara eklenmelidir.
- Bu sayede, derleyici, hataları derleme anında bulabilecektir.
- Aksi takdirde, hatalar programın çalıştırıldığı anda ortaya çıkar. Bu tür hataların düzeltilmesi oldukça güç olur.

5. Satır: main( ) fonksiyonu

- C programlarının ana fonksiyonu olarak tabir edilir.
- main() kelimesinden sonraki parantezler ( ) main'in fonksiyon adı verilen program oluşturma bloklarından biri olduğunu gösterir.
- Programın yürütülmesi ilk olarak bu fonksiyonun çağrılmasıyla gerçekleşir.
- C programları bir veya birden fazla fonksiyon içerebilir ancak bunlardan biri mutlaka main() olmalıdır.

5. Satır: main( ) fonksiyonu (devam…)

- Küme parantezi, **{** , her fonksiyonun gövdesinin başına yazılır.
- Diğer küme parantezi , **}** , ise her fonksiyonun sonuna yazılmalıdır.
- Bu iki parantez arasında kalan program parçacığına *«blok»* denir.
- Bloklar C'de önemli program birimleridir.

7. Satır: printf("Hello World!\n"); fonksiyonu

- printf standart kütüphanede bulunan ekrana formatlı bilgi yazdırma fonksiyondur. Çift tırnak işareti arasındaki karakterleri ekrana yazdırır.
- Yazdırılacak karakterlerin tümüne karakter dizesi «string», mesaj ya da hazır bilgi «literal» denir.
- printf, parantezler içindeki bağımsız değişkenler (argument) ve noktalı virgülden oluşan bu satıra «ifade» denir.
- Her ifade ; (noktalı virgül) ile bitmelidir. Noktalı virgüle ifade sonlandırıcı da denir.

7. Satır: printf("Hello World!\n"); fonksiyonu (devam…)

- printf ifadesindeki tırnak işaretleri arasındaki karakterler aynen ekrana yazdırılır. Ancak \n karakterlerinin yazdırılmamaktadır.
- Ters eğik çizgi (\ ), çıkış karakteri olarak adlandırılır ve printf'in farklı bir iş yapması gerektiğini belirtir.
- printf, ters çizgi işaretiyle karşılaştığında, bu işaretten sonraki karaktere bakar ve bu karaktere göre bazı özel işler yapar.
- Ters çizgi işareti (\ ) ve bu işaretten sonra gelen karaktere çıkış sırası denir.
- \n çıkış sırası, yeni satır anlamına gelir ve imlecin yeni satıra geçmesine sebep olur.

7. Satır: printf("Hello World!\n"); fonksiyonu (devam…)

- Çıkış karakterleri (escape sequence) aşağıdaki gibidir:

Çıkış	Anlamı
\0	null karakteri temsil eder (sonlandırıcı karakter)
\n	Yeni satır
\r	Satırbaşı
\t	Yatay sekme
\v	Düşey sekme
\f	İleri besleme
\b	Geri boşluk (space)
\a	Alarm karakteri
\"	Çift tırnak
\\	Ters bölü

7. Satır: printf("Hello World!\n"); fonksiyonu (devam…)


```
1   /* Main.c : ilk C programımız */
2   #include <stdio.h>
3   #include <stdlib.h>
4
5   int main ()
6   {
7       printf("1.Hello World!\n");
8       printf("2.Hello ");
9       printf("World!\n");
10      printf("\n");
11      printf("aaaa\t");
12      printf("bbbb\n");
13      printf("cccc\t");
14      printf("dddd\t");
15      printf("\n");
16      return 0;
17  }
18
```

Ekran Çıktısı Ne Olacak

```
"C:\D\Akademik\Courses\Algoritma ve Programlama I\Labs\Hafta 2\HelloW
1.Hello World!
2.Hello World!

aaaa    bbbb
cccc    dddd

Process returned 0 (0x0)   execution time : 0.007 s
Press any key to continue.
```

C Kodlarının Temel Özellikleri - Özet

- Yazılımda kullanılacak olan her fonksiyon için ilgili başlık dosyası programın başına ilave edilmelidir.
- Her C programı main() fonksiyonunu içermelidir.
- Program içinde kullanılacak olan değişkenler ve sabitler mutlaka tanımlanmalıdır.
- Her ifade satırının sonuna ; işareti konmalıdır.

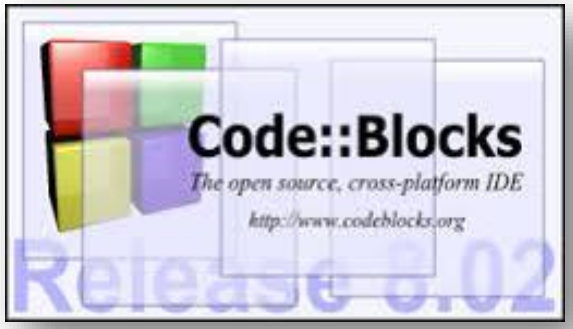
C Kodlarının Temel Özellikleri – Özet (devam…)

- Her bloğun ve fonksiyonun başlangıcı ve bitişi sırasıyla { ve } sembolleridir.
- C dilinde yazılan kodlarda küçük-büyük harf ayrımı vardır (case sensitive).
  - Örneğin; "A" ile "a" derleyici tarafından farklı değerlendirilir.
- Yorum satırı operatörü /* */ veya // sembolleridir.

Code::Blocks Nedir?

- Code:Blocks özellikle C ve C++ gibi programlama dillerinde kodlama geliştirme yapmamıza imkan veren, açık kaynak kodlu ve cross-platform bir IDE'dir (Integrated Development Environment).

- Code:Blocks IDE ortamının kendisi de C++ ile geliştirilmiştir.



Code::Blocks Derleyici Özellikleri

- Çoklu derleyici desteği:
  - GNU GCC
  - MS Visual C++
  - Dijital Mars
  - Borland C++ 5,5
  - OpenWatcom
  - LLC
  - Intel C++
  - SDDC, Tiny C, GDC D
  - GNU ARM, GNU ARV

Code::Blocks Derleyici Özellikleri (devam…)

- Çok hızlı özelleştirilmiş derleme ve bağlama (makefiles gerekli değildir).
- Paralel derleme ve bağlama desteği (var ise CPU'nun diğer çekirdeklerini kullanarak).
- Çoklu hedef proje desteği.
- Workspace kullanarak birden fazla proje ile çalışma.

Code::Blocks Debugger (Hata Ayıklayıcısı) Özellikleri

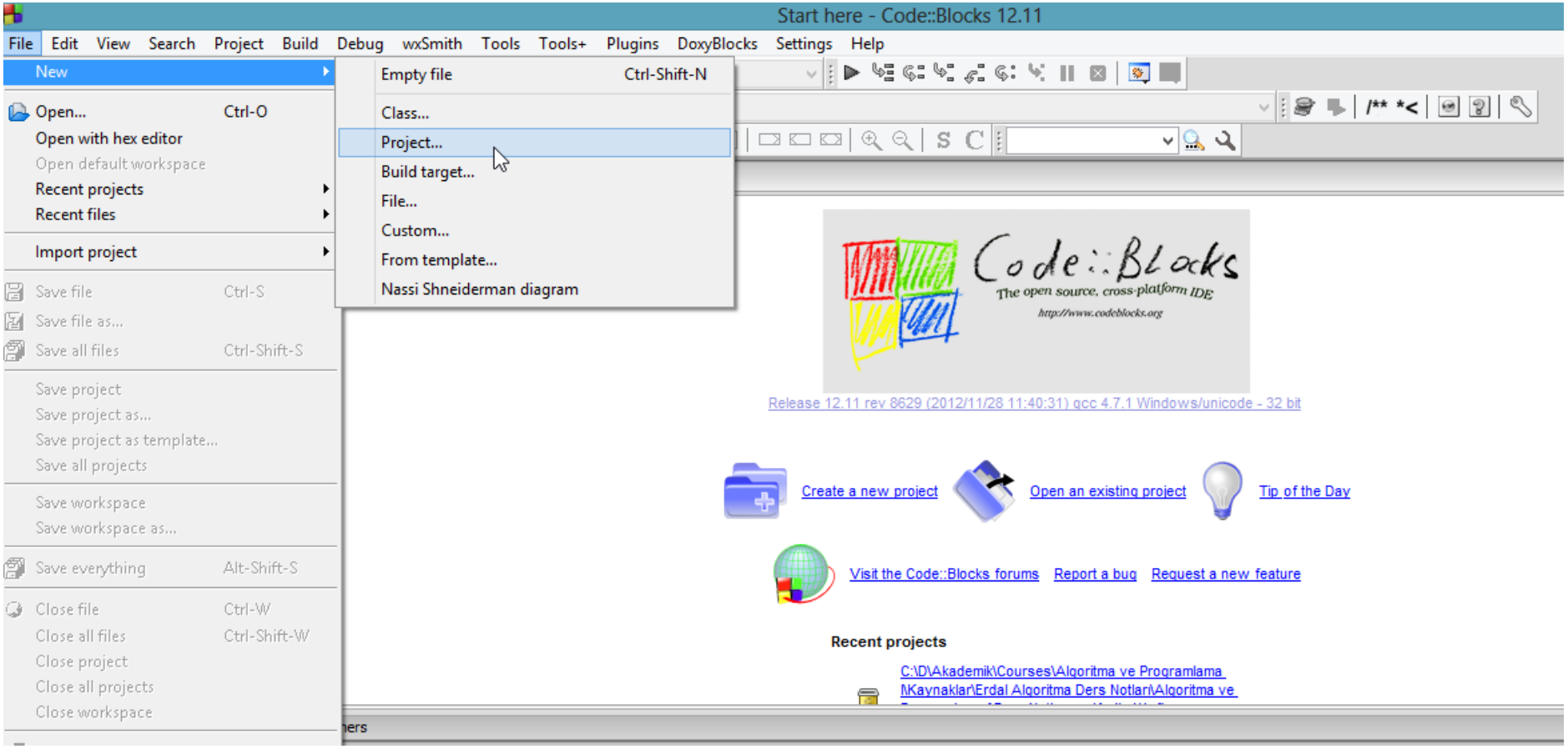
- GNU GDB (Tüm özellikleriyle GNU'nun hata ayıklayıcısını desteklemektedir.).
- MS CDB destekler (Tüm özellikleriyle birlikte değil).
- Tam breakpoint desteği.
- Görünen yerel fonksiyonlar, semboller ve argümanlar.
- Disassembly desteği.
- Özel bellek dökümü.
- CPU görünümü.

Code::Blocks Ara Yüz (IDE, Editör) Özellikleri

- Özelleştirilebilir ve genişletilebilir, söz dizimi renklendirilmesi (syntax highlighting)
- C++ ve XML için yazım editöründe kod katlama desteği.
- Ara yüzde sekme desteği.
- Kod tamamlama (Code Completion).
- Sınıf tarayıcı (Class Browser).
- Akıllı satır içe alma (Smart Indent).
- Birçok özelleştirilebilir araç.
- Farklı kullanıcılar için **TODO** liste yönetimi.

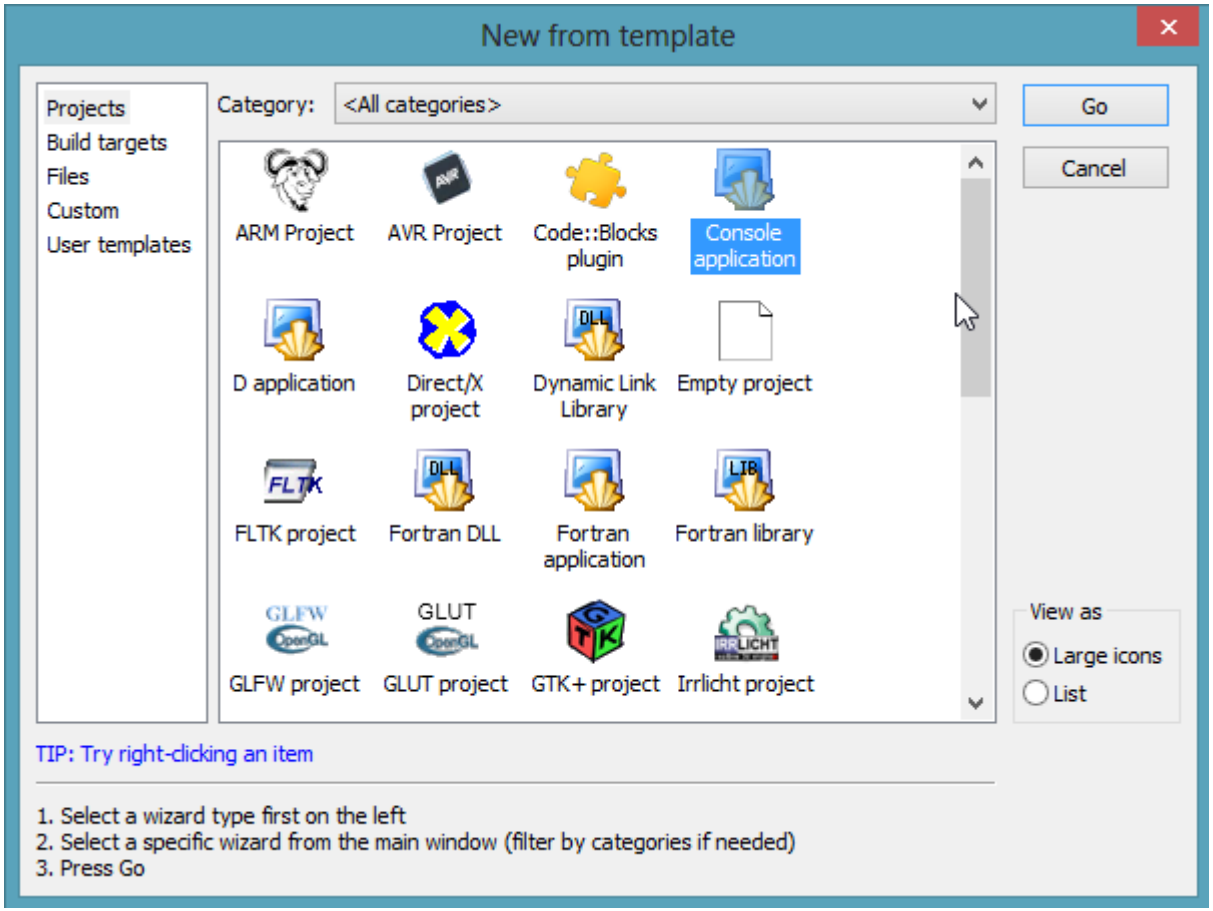
Code::Blocks ile İlk C Programı

- Ana menüden **“File → New → Project”** seçilir ve proje oluşturma sihirbazı başlatılır.



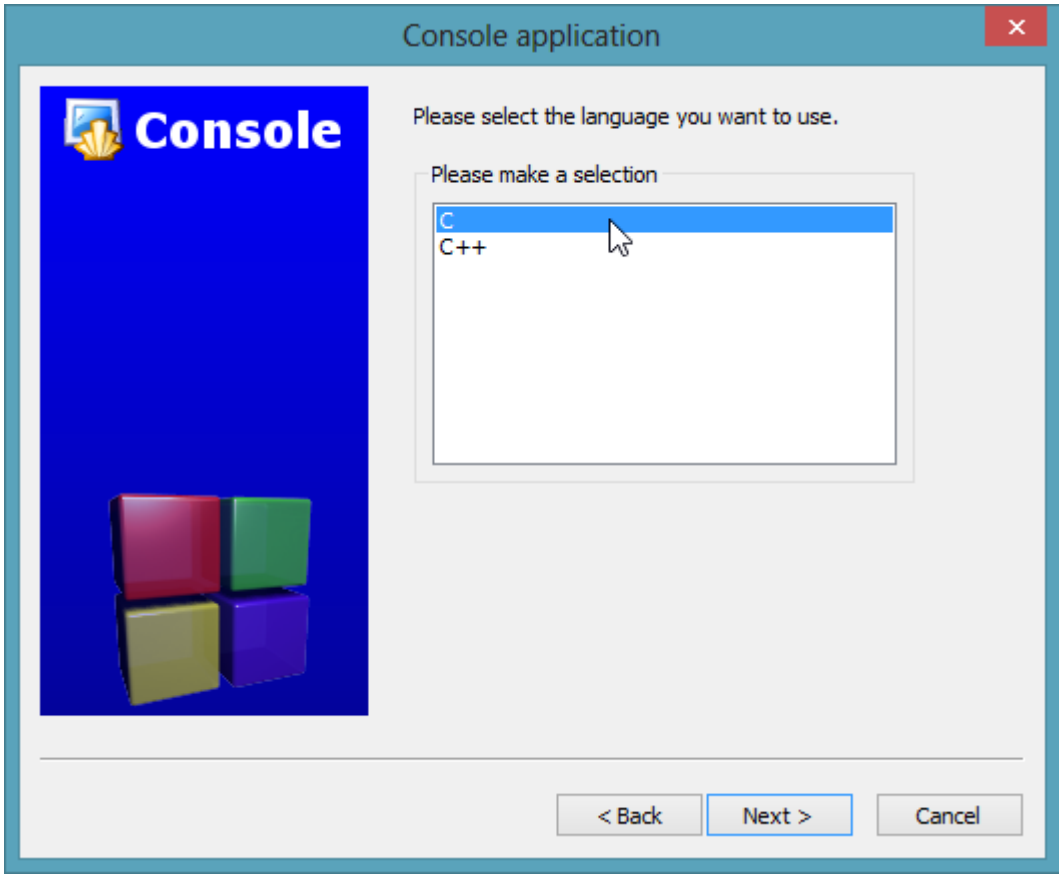
Code::Blocks ile İlk C Programı (devam…)

- Karşımıza çıkan şablon seçim ekranında **“Console application”** şablonu seçilir **“Go”** düğmesine basılır.



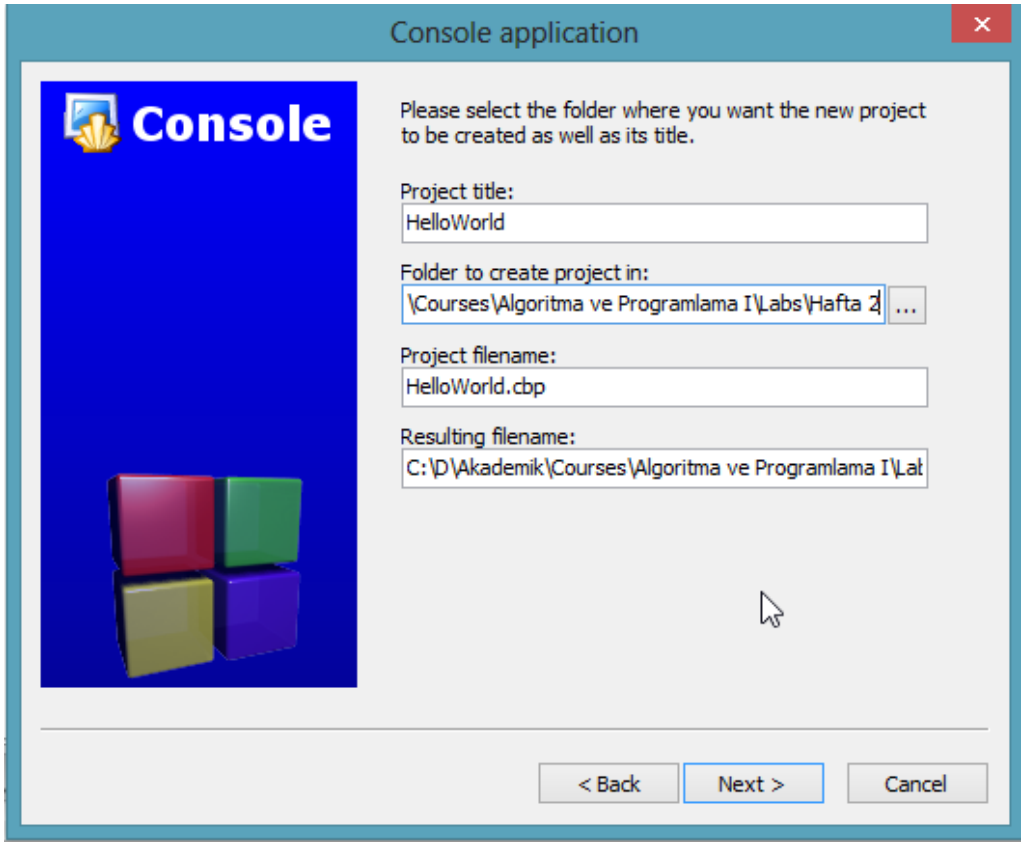
Code::Blocks ile İlk C Programı (devam…)

- Karşımıza çıkan Programlama Dili Seçim ekranında **“C”** programlama dili seçilir ve **“Next”** düğmesine basılır.



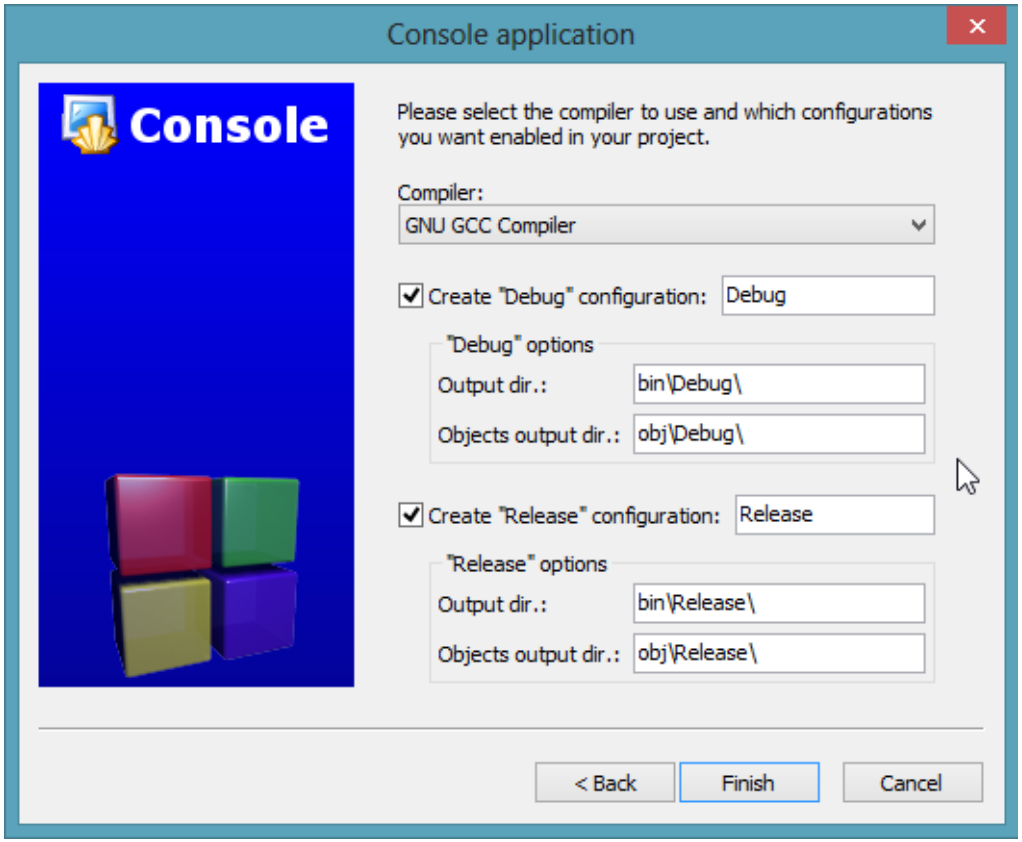
Code::Blocks ile İlk C Programı (devam…)

- Karşımıza çıkan yeni ekranda **“Project Title”** alanına Proje İsmi girilir. **“Folder to create project in”** kısmında ise Proje klasörünün oluşturulacağı ve proje dosyalarının yaratılacağı dosya yolu seçilir.



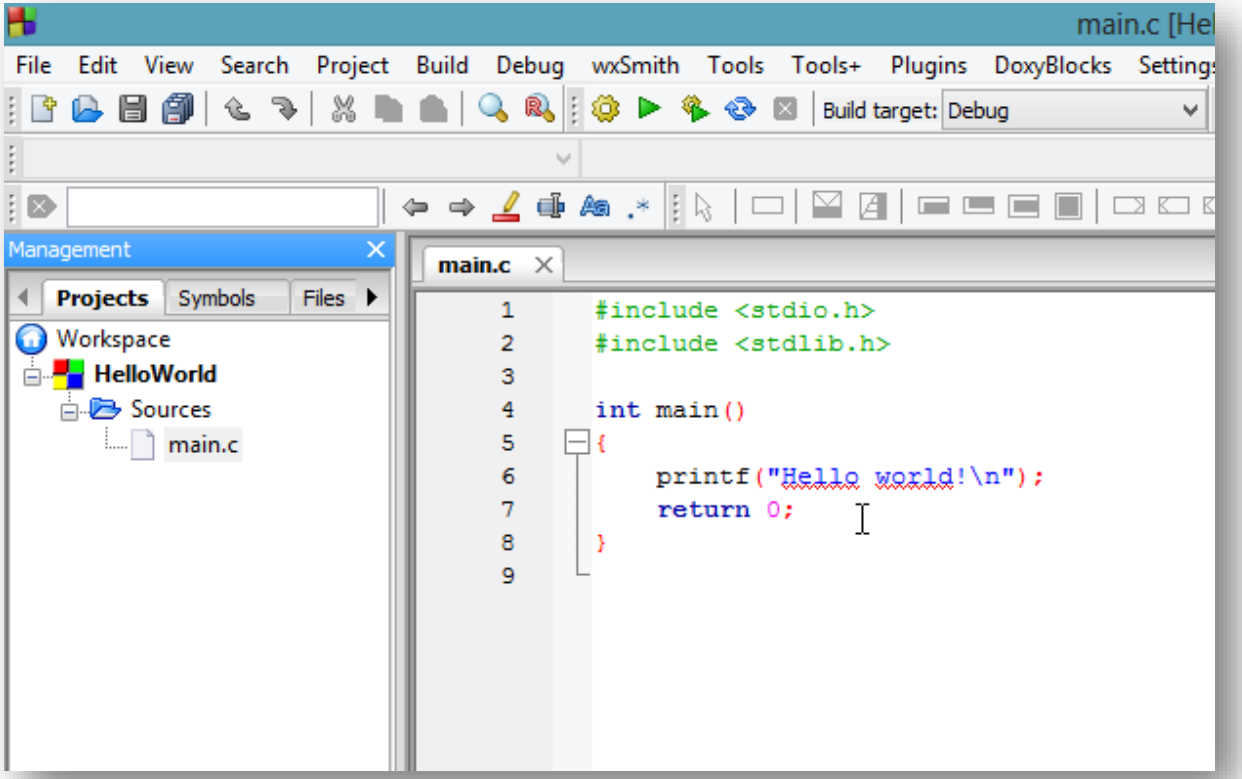
Code::Blocks ile İlk C Programı (devam…)

- Karşımıza çıkan Derleyici (Compiler) seçim ekranı varsayılan değerleriyle bırakılır. Böylece **“Debug”** ve **“Release”** modlarının ayarları da tamamlanmış olur. Son olarak **“Finish”** düğmesine basılır.



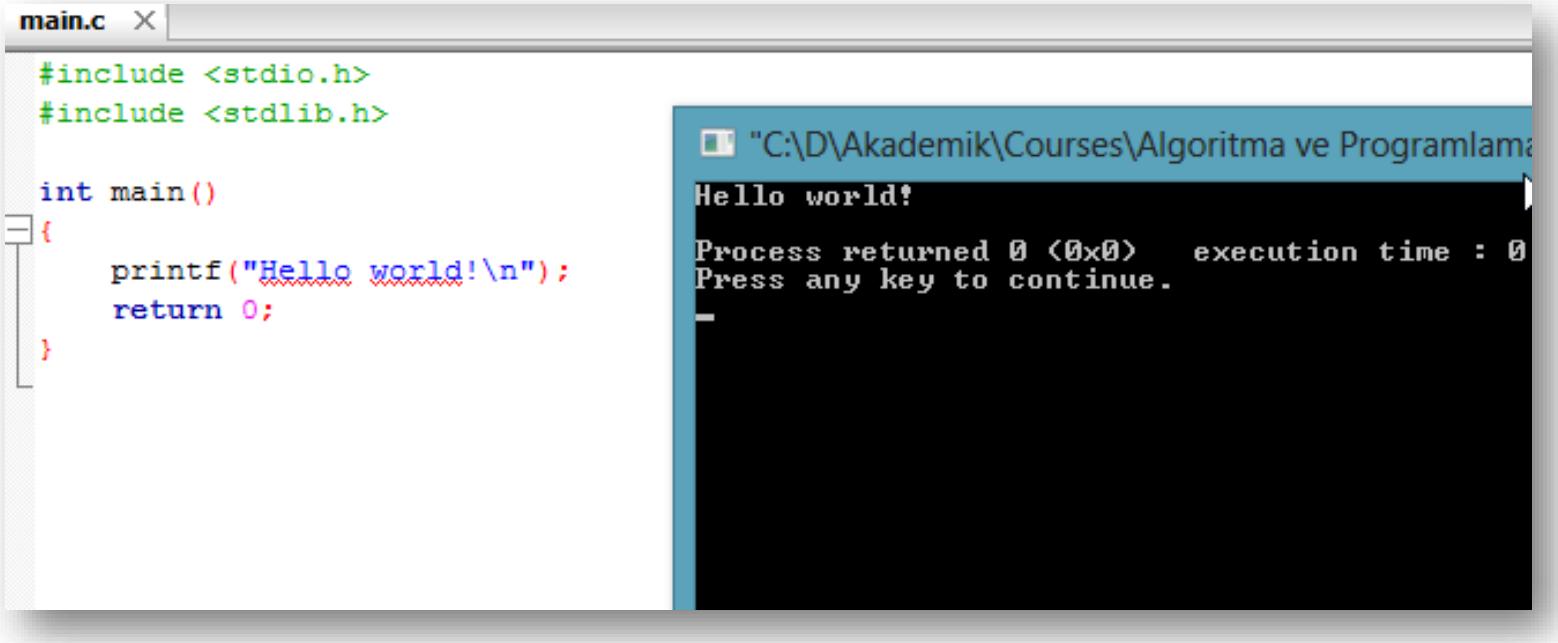
Code::Blocks ile İlk C Programı (devam…)

- Proje oluşturma sihirbazının son adımının tamamlanması ile birlikte aşağıda görüldüğü gibi Proje sekmesinde HelloWorld isimli projemiz ve editörde de **“main.c”** isimli dosyada ilk C programımız otomatik olarak yerleştirilir.



Code::Blocks ile İlk C Programı (devam…)

- Son olarak derleme ve çalıştırma işlemi için **“F9”** kısa yolu veya **“Build → Build and run”** menü adımı seçilir.



Code::Blocks ile HelloWorld Uygulaması

C Programlama Dili Elemanları

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

C Tanımlayıcıları

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

C Anahtar Sözcükleri

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

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

C Veri Türleri

- Veri tipi (data type) program içinde kullanılacak değişken, sabit, fonksiyon isimleri gibi tanımlayıcıların tipini, yani bellekte ayrılacak bölgenin büyüklüğünü, belirlemek için kullanılır.

- Bir programcı, bir programlama dilinde ilk olarak öğrenmesi gereken, o dile ait veri tipleridir. Çünkü bu, programcının kullanacağı değişkenlerin ve sabitlerin sınırlarını belirler.

C Veri Türleri (devam…)

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

C Veri Türleri (devam…)

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

C Veri Türleri (devam…)

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

- Kısa ve uzun tamsayı tutacak tanımlayıcılar için int anahtar kelimesinin yazılmasına gerek yoktur.
  - short s; /* short int s; anlamında */
  - long k; /* long int k; anlamında */

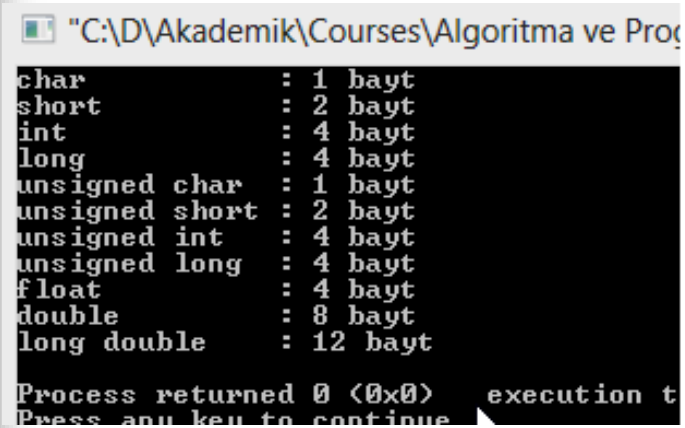
C Veri Türleri (devam…)

- Bir C programı içerisinde, veri tiplerinin bellekte kapladığı alan sizeof operatörü ile öğrenilebilir. İlgi çekici olan, bu alanların derleyiciye??? ve işletim sistemine??? bağlı olarak değişiklik göstermesidir.

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

C Veri Türleri (devam…)


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

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



C Değişkenleri

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

  veri_türü değişken_adı;

  int sayac;

C Değişkenleri (devam…)

Değişken/Sabit/ Fonksiyon/Yapı Adı	Geçerlilik	Açıklama
asal	geçerli	-
Momentum	geçerli	-
ivme	geçerli	-
olasilik	geçerli	-
IsikHizi	geçerli	-
isik_hizi	geçerli	Alt çizgi karakteri '_' kullanılabilir
isik hizi	geçersiz	Boşluk karakteri kullanılamaz
ışık_hızı	geçersiz	Türkçe karakter kullanılamaz
1Bit	geçersiz	rakam ile başlanamaz
typedef	geçersiz	Anahtar kelimelerden birisi kullanılamaz

C Değişkenleri (devam…)

- **Örnekler**

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

- **Örnekler**

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

C Sabitleri

- Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır.
- Ancak, veri tipinin önüne const anahtar sözcüğü konmalıdır.
- Sabit içerikleri program boyunca değiştirilemez. Yalnızca kullanılabilir.
- Genellikle, sabit olarak bildirilen değişken isimleri büyük harflerle, diğer değişken isimlerinin ise küçük harflerle yazılması (gösterilmesi) C programcıları tarafından geleneksel hale gelmiştir.

C Sabitleri (devam…)

- Örnekler:

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

printf () - Tip belirleyici (conversion specifier)

- % işareti ile başlar ve bir veya iki karakterden oluşur (%d gibi).
- Ekrana yazdırılmak istenen değişkenin tipi, % işaretinden sonra belirtilir.
- Ayrıca biçim ifadesinin içine, sola - sağa yaslama, noktadan sonra x basamak yaz vb gibi isteklerimizi belirten karakterler de ekleyebiliriz.
- Gerçek sayıların yazdırılmasında, noktadan sonra yazılacak basamak sayısı durumların ifade edilmesi için ve tamsayıların aynı hizada yazdırılması için nokta operatörü veya rakamlar kullanılır.
- Aynı şekilde karakter katarlarının sağa ya da sola dayalı yazdırılması için veya bir karakter katarındaki karakterlerin kaç tanesinin yazdırılacağını belirtmek için de yine nokta, eksi gibi operatörlerin ve rakamların çeşitli kombinasyonları kullanılır.

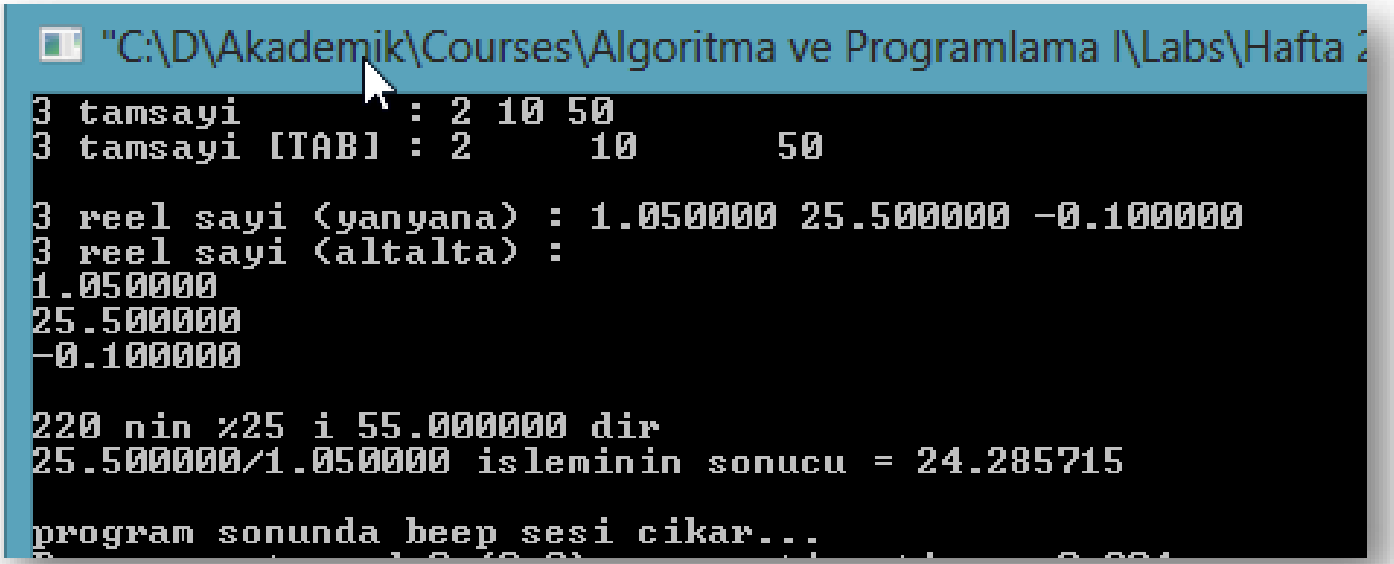
printf () - Tip belirleyici (conversion specifier)
(devam…)

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

printf () - Tip belirleyici (conversion specifier) (devam…)


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

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

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

    printf("\n");

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

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

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

    return 0;
}
```

printf () - Tip belirleyici (conversion specifier)
(devam…)



C Operatörleri

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

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

C Operatörleri – Aritmetik Operatörler

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

- ***Örnekler:***

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

C Operatörleri – Atama Operatörleri

- Bu operatörler bir değişkene, bir sabit veya bir aritmetik ifade atamak (eşitlemek) için kullanılır.
- Birleşik atama: bazı ifadelerde işlem operatörü ile atama operatörü birlikte kullanılarak, ifadeler daha kısa yazılabilir.

  Eğer ifade

  değişken = değişken [operatör] aritmetik ifade;

  şeklinde ise, daha kısa bir biçimde

  değişken [operatör]= aritmetik ifade;

  olarak yazılabilir.

C Operatörleri – Atama Operatörleri (devam…)

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

C Operatörleri – Atama Operatörleri (devam…)

Örnek	Anlamı

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

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

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

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

C Operatörleri – Atama Operatörleri (devam…)

- Örnek: Aşağıdaki işlemlerden sonra a, b ve c'nin son değerleri ne olur?

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

```
a = 7
b = 5
c = 7
```

C Operatörleri – Atama Operatörleri (devam…)

- Örnek: Aşağıdaki işlemlerden sonra i'nin son değerleri ne olur?

```
int i = 1;
i++;
++i;
i += 1 + i++;
i = i + 1;
```

i = 9

scanf() Fonksiyonu

- Birçok programda ekrana verilerin bastırılmasının yanı sıra klavyeden veri okunması gerekebilir.
- scanf() fonksiyonu klavyeden veri okumak için kullanılan fonksiyondur.
- Tip belirleyicileri printf fonksiyonu ile aynı mantıkta kullanılır ve % sembolü ile ifade edilir.
- Örneğin klavyeden bir x tamsayısı okumak için aşağıdaki ifade kullanılır:

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

Örnek 00: Değişken Tanımla, Toplama Yap

Sabit girilen 2 tam sayının

- Toplamını

bulup yazdıran C programını yazınız.

- 3 değişken kullanılacaktır.
- Değişkenlerin ilk değerleri 0 olarak atanacaktır.

Örnek01: printf () Tip Belirleyecileri


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

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

    return 0;
}
```

Örnek 02: İki Gerçel Sayı ile Aritmetik İşlemler

Klavyeden girilen 2 gerçel sayının

- Toplamını
- Çıkartılmasını
- Çarpımını
- Bölümünü

bulup yazdıran C programını yazınız.

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

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

Örnek 03: Operator Test

Aşağıdaki işlemleri teker teker gerçekleştiriniz ve her işlemden sonra değişlenlerin değerini ekrana yazdırınız.

a = 5;

a değerini YAZ

b = a++;

a ve b değerlerini YAZ

c = ++a;

a, b ve c değerlerini YAZ

Örnek 04: Girilen Sınav ve Ödevlere Göre Ders Notu Hesaplama

Klavyeden girilecek aşağıdaki sınav ve ödevlere göre Ders notu hesaplanacaktır:

- Ödev: %9 (3 tane)
- Quiz: %21 (3 tane)
- Ara Sınav: %30 (2 tane yazılı sınav)
- Final: %40 (1 tane genel yazılı sınav)

Not1: Toplam 9 tane giriş yapılacaktır. Girişler Gerçel sayı olacaktır.

Not2: Ders notu 2 ondalıklı olarak gösterilecektir.

KAYNAKLAR

- Okt. Tuna GÖKSU Bilgisayar ve Programlama Sunumu
- N. Ercil Çağıltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Milli Eğitim Bakanlığı "Programlamaya Giriş ve Algoritmalar Ders Notları", 2007
- http://tr.wikipedia.org/wiki/Code::Blocks
- http://www.codeblocks.org
- http://www.AlgoritmaveProgramlama.com
- http://www1.gantep.edu.tr/~bingul/c

İYİ ÇALIŞMALAR…

Yrd. Doç. Dr. Deniz KILINÇ
deniz.kilinc@cbu.edu.tr