Bölüm 3

C Programının Evreleri

3.1 Neler Gerekli

Bir C programının yazılıp çalışır hale gelmesi için gerekli olan ortamın sağlanması gerekir. Öncelikle, kullanacağımız bilgisayarı ve onun işletim sistemini bilmeliyiz. Tek terminalli Windows, Linux, Mac OS X ya da çok terminalli bir büyük sistemde çalışabiliriz. Hangi sistem olursa olsun, program yazmak ve geliştirmek için şunlara gerekseme vardır:

- Kaynak programı yazmak için bir metin düzenleyici (editör).
- Kaynak programı makinanın anlayacağı alt-düzeyli dile çevirmek için derleyici (compiler).
- Derlenen programı başkalarıyla birleştirmek için bağlayıcı (linker, make).
- Sözdizimi hatalarını ayıklamak için ayıklayıcı (debugger).
- Dosyaları anabelleğe yüklemek için yükleyici (loader).

Çağdaş derleyicilerin hemen hepsi son dört işlemi yapan modülleri içerirler. Dolayısıyla, onları ayrıca düşünmemiz gerekmiyor. Ancak komut satırı ile mi yoksa bütünleşik bir geliştirme aracı ile mi çalışacağımıza karar vermeliyiz. Çok uçlu (terminal) büyük sistemlerde zaten her işi akılsız terminalde yazacağımız komutlarla yapmak zorundayız.

Çok sayıda derleyici vardır (bkz [4]). Bazıları ücretli bazıları ücretsizdir. Kullanıcı istediğini seçebilir.

Bu derste GNU prjesi çerçevesinde geliştirilen ücretsiz `gcc` derleyicisini temel alacağız. Doğrudan gcc yüklenebilir. Unix türevlerinde ve Linux çekirdeği üzerine kurulu olan sistemlerin çoğunda zaten `gcc` yüklü olur. Öte yandan, `gcc` 'ye dayalı olarak yazılan arayüzler vardır. `MinGW` ("Minimalist GNU for Windows") seçeneklerden birisidir. Başka bir seçenek `Cygwin` projesidir. Bu proje, GNU ve OpenSource araçlarını Windows üzerine taşıyıp Linux sürümlerine benzer biçimde çalıştırmayı amaçlamaktadır.

Tüleşik program geliştirme araçlarına bölümün sonunda değineceğiz.

Karakter Kodlama Sistemlei

Program yazarken karakter kodlamaları hakkında da biraz bilgiye sahip olmalıyız. Eğer bu konuda temel bilgilere sahip değilseniz, neden bazı yazılımlarda Türkçe alfabede yer alan çÇ, ğĞ, ıİ, şŞ, öÖ, üÜ harflerini kullanamadığımızı anlayamazsınız. Bu geniş ve önemli konuyu bu derste ele alamayacağız. Ancak şu gerçeği bilmeliyiz. C dili yaratıldığı zamanlarda, bilgisayarlar yalnızca 128 karakterden oluşan *ascii* kodlama sistemini kullanıyordu. Farklı ülkelerin alfabelerinin bilgisayarda kullanılmalarını sağlayan *karakter kodlama sistemleri* yoktu. Üstelik, bilgisayarların günün birinde dünyaya yayılacağı ve her kültüre, her her ülkeye hizmet sunacağı öngörülmüyordu. Çok sayıda uygulama yazılımı *ascii* kodlama sistemiyle yazıldı. Buna Windows, Unix gibi işletim sistemleri de dahildir. Bunun doğal sonucu olarak, C derleyicilerinin çoğu, ascii karakterleri dışındaki karakterleri bilmez. Gerçi, farklı ülkelerin alfabelerinin kullanılmasını sağlamak için, bazı C derleyicilere modüller eklenmiştir. Gene de kaynak programın taşınabilirliğini sağlamak için, kaynak programda *ascii* dışındaki karakterlerin kullanılmamasını öneriyoruz. Özellikle adlar *ascii* karakterleriyle yazılmalıdır. (Daha ayrıtılı bilgi için bkz. [2]).

3.2 Programın Hazırlık Evreleri

metin düzenleyici -editör Bir (metin düzenleyici (editör) aç. Her işletim sisteminde kaynak program yazmaya yarayan basit bir editör vardır. *MS Word* gibi gelişkin editörlerle kaynak program yazmayınız. Çünkü onlar, bizim göremediğimiz biçemlendirme (format) komutlarını metne eklerler. Bizim göremediğimizi derleyici görür. Onlar derleyicinin anlayacağı komutlar olmadığından, kaynak programı derleyemez. Windows işletim sisteminde *Notepad* ya da onun biraz daha gelişmişi olan *npp* gibi bir editör yeterlidir. Özellikle kaynak program

yazmak için geliştirilmiş daha iyi editörler de vardır. Ama bizim yazacağımız programlar için şimdilik Notepad yeterlidir.

kaynak - source- Kaynak programı, seçtiğimiz editörle yazacağız. Kaynak programın en basit yapısından başlayarak, adım adım ilerleyeceğiz.

Bu kesimde basit bir örnek olarak kullanılmak üzere Program 3.1'yi yazalım. Bu program yazılabilecek en basit C programıdır. Bu programı *enbasit.c* adıyla kaydedelim. Aslında bu program hiç bir iş yapmaz. Ama kaynak programın bazı temel niteliklerine sahiptir.

Program 3.1.

```
main()
{

}
```

Her kaynak programa bir `ad` verip, kaydediyoruz. Kaynak programın adı, genel adlandırma kuralına uymalıdır; yani ad harf, sayak (digit) ve alt çizgiden (_) den oluşan bir string'dir. Ama sayakla başlayamaz. En çok 32 karaktere kadar ayırıcı niteliği vardır. Bu demektir ki, ilk 32 karakterleri aynı olan iki adı, derleyici birbirinden ayıramaz (bkz. 5.4.1).

Kaynak program adının sonuna *.c* uzantısı mutlaka konulmalıdır. Derleyici bu uzantıyı görünce kaynak programın C dili ile yazıldığını anlar.

1.satırdaki `main()` bir fonksiyondur. `main` fonksiyonun adıdır. Her C kaynak programında olmalıdır.

Bir ad'ın (identifier) sonuna () parantezi gelirse, derleyici onun bir fonksiyon olduğunu anlar.

Bazı derleyicilerde `main` adının önüne `int` konulur. Her fonksiyonun vereceği (döndüreceği, return) değeri vardır; o değerin veri_tipi fonksiyon adının önüne yazılır. Ama buradaki `return 0` deyiminin özel bir işlevi vardır. `main()` içindeki son deyimdir. Aslında 0 değeri çıkış birimine gitmez. Son deyime kadar olanlar hatasız işlenirse, `return 0` deyimine ulaşılır. Böylece `main()` fonksiyonunu kullanan kabuğa (shell), görevin hatasız yapıldığını bildirir. Son deyim olan `return 0` deyiminden önce hata varsa, zaten son deyime ulaşılamaz.

2.satırda { parantezi açılıyor, 4.satırda } ile kapanıyor. { } parantezinin içi, main fonksiyonunun gövesidir. { } parantezi içindeki deyimler

sırasıyla işlenir. Bu özeliği nedeniyle { } parantezine bir blok denilir. İleride fonksiyon bloklarını, akışı yönlendiren blokları ve döngü yapan başka blokları göreceğiz. Program akışı bir blok içine girince, blok içindeki deyimleri sırasıyla işler. Aksi istenmedikçe, bloktaki işi bitmeden akış blok dışına çıkmaz.

Bu söylediklerimize göre Program 3.1 şöyle de yazılabilir:

Program 3.2.

```
1  int main()
     {
      return 0;
     }
```

Bu program yukarıdaki ile aynıdır. Her ikisi de hiç bir iş yapmazlar. Şimdi programımıza bir metin yazmasını söyleyelim. C dilinde standart çıkışa (bizim sistemlerde ekrandır) bir çıktı göndermek için printf() fonksiyonunu kullanırız. () parantezi içine yazılanlara parametre ya da argüman denilir. Parametre string ise çift tırnak (" ") arasına yazılır.

Program 3.3.

```
1  main()
     {
       printf("Merhaba C!\n");
     }
```

Bunu derleyip koşturmak istersek, bazı derleyiciler printf() fonksiyonunu tanımadığını bildirir ve derlemeyi yapmaz. Bu derste kullandığımız Dev-C++ tümleşik geliştirme aracı,

```
1  3 4 D:\Dev-CppPrj\01week\enbasit.c  [Warning] incompatible
   implicit declaration of built-in function 'printf' [enabled by
   default]
```

uyarısını vererek ekrana *Merhaba C!* yazar. Böyle olmasının nedeni, `printf()` fonksiyonunun C dili içinde olmayışıdır. Dolayısıyla onu derleyici tanımıyor. Bunun gibi, ileride kullnacağımız bir çok fonksiyon, C dili içinde değildir. Onlar C kütüphanesindedirler. Programımıza ilgili kütüphaneyi çağırırsak, artık, derleyici onun nerede olduğunu bilir ve itiraz etmeden çalıştırır. `printf()` fonksiyonu `stdio.h` adlı başlık dosyası (paketi) içindedir. Onu çağırmak için, programın en başına

#include <stdio.h>

yazıyoruz. Benzer işi, bilgisayara giriş işlemini yapan `scanf()` fonksiyonu içinde yapmalıyız. Genelleştirisek, bütün giriş/çıkış (G/Ç, input/output - I/O) işlemlerini yapan fonksiyonlar *stdio.h* paketindeir. *std* standart'ın kısaltmasıdır. *io* ise input-output'un kısaltmasıdır. *.h* uzantısı *stdio* paketinin bir *başlık dosyası* (header file) olduğunu belirtir.

Program 3.4.

```
#include <stdio.h>

  main()
{
  printf("Merhaba C!\n");
}
```

Merhaba C! parametresinin sonuna gelen \n *kaçış* karakteridir (escape character). Ekrana metni yazdıktan sonra imleci satır başına gönderir. Böylece, arkadan gelecek yeni çıktılar varsa, onlar yeni satırın başından başlanarak yazılır.

C dilinde, diğer bütün dillerde olduğu gibi, bilgisayara bir iş yaptıran sözlere deyim (ifade -expression) denilir. C dilinde deyimleri bilgisayara verilen birer komut (buyruk) olarak görebilirsiniz.

Son olarak, gövdedeki tek deyimin sonuna noktalı virgül (;) konulduğuna dikkkat ediniz. C dilinde, başka bir çok dilde olduğu gibi, deyimlerin bittiği sonlarına konulan (;) ile belirtilir. Bunu koymazsak, derleyici deyimin bitiğini anlayamaz; derleme hatası verir.

ad -identifier C dilinde kullandığımız her şeye (sabit, değişken, fonksiyon, program, vb) bir ad vermeliyiz. Yazdığımız kaynak programa *deneme* adını verip kaydedelim (flopy disket, sabit disk, teyp vb bir kayıt ortamı).

C kaynak programları adı daima *.c* uzantısını alır. Derleyici onun C kaynak programı olduğunu, uzantısına bakarak anlar. Bazı C editörleri *.c* uzantısını kendiliğinden koyabilir. Kullandığımız editör bunu yapmıyorsa, program adının sonuna *.c* uzantısını eklemeyi unutmayınız. Dolayısıyla, kaynak programımız kayıt ortamına *deneme.c* olarak kaydedilecektir. Notepad kullanırkan, öntanımlı olarak *.txt* uzantısının gelmemesine özen gösteriniz.

Şimdi *deneme.c* kaynak programını derleyip koşturmayı öğreneceğiz.

derleyici -compiler Kaynak programı derlemek için C derleyicisini (compiler) çağıracağız.

Bunun için bilgisayarımızda bir *C* derleyicisinin önceden yüklenmiş olduğunu varsayalım. Örneğin, GNU projesi çerçevesinde yazılan `gcc` derleyicisi bilgisayarımızda yüklü olsun.

Ayrıca, kaynak program(larımız)ın bulunduğu dizin ile derleyicinin bulunduğu dizinin PATH ortam değişkenine eklendiğini varsayalım.

Derleyiciyi yüklemek ve PATH ortam değişkenine onun patikasını (path) eklemek işi, maalesef çok standart değildir. Kullandığınız işletim sistemine bağlıdır. Bu derste o konulara giremeyeceğiz. Ama her işletim sistemi için, derleyicinin nasıl yükleneceğini ve onun bulunduğu yerin PATH ortam değişkenine nasıl ekleneceğini anlatan web dökümanlarını internette yığınla bulabilirsiniz.

Derleyiciyi çağırmak demek, onu anabelleğe yüklemek demektir. Bu çok kolaydır. Derleyicinin adını yazmak yetecektir. Tabii, derleyicinin *deneme.c* adlı kaynak programı derlemesini istediğimize göre

```
gcc deneme.c
```

yazmalıyız. Tabii, yukarıda söylediğimiz gibi, `gcc` derleyicisinin bulunduğu yere giden patikayı ve *deneme.c* programına giden patikayı PATH ortam değişkenine eklemiş olmalısınız. Bunu yapmadıysanız, hem `gcc` 'nin önüne hem kaynak programın önüne kendi patikalarını yazmanız gerekir:

```
patika1/gcc}  patika2/deneme.c
```

yazmalısınız. Burada patika1 ve patika2, sırasıyla, derleyicinin ve kaynak programın bulunduğu adreslerdir. Her program derleyişte bu zahmetli işten kurtulmak için PATH ortam değişkenine ilgili patikalar eklenmelidir.

3.2.1 Kaynak Programın Derleyiciden Geçişleri

Genellikle C derleyicilerde kaynak program derlenirken ard arda geçişler olur. Bu geçişler bazı derleyicilerde kendiliğinden sırasıyla olur; kullanıcı geçişleri farketmez. Bazılarında geçişler için kullanıcı komut yazar.

C kaynak programının yürütülebilir hale gelmesi için şu evrelerden geçer:

Önişlemciler -preprocessor: C kaynak programının başlangıcına yazılan satırlar için derleyicinin yaptığı önişlemlerdir. Derleme işlemi başlamadan önce bunların yapılması gerekir. Programın başlangıcına ya-

zılan satırların, programa dahil edilebilmesi için yapılan bu işlemler dört tanedir:

1. Dosya içerme komutları
2. Makro tanımlar ve yerine koymalar
3. Koşullu metin içerme
4. Satır numaralama

Önişlemcilerin nasıl çalıştığını ileride örneklerle açıklayacağız.

Derleme - compile: İkinci geçişte C diline ait deyimler geçer; simge tablosu oluşturulur; varsa sözdizimi yanlışları listelenir. Bu geçişte oluşan dosyaya değişik işletim sistemlerinde değişik adlar verilir. Genelde, derleyicinin yarattığı bu kodlara *object* kodlar denilir. Windows işletim sisteminde object (.obj uzantılı) dosya yaratılır.

Bağlama -Linker: C kaynak programının başında, kütüphaneden çağrı yapan satırlar var olabilir. Linker, önceki geçişte yaratılan object kodu, kütüphaneden çağrılanlarla birleştirir. Yürütülebilir (executable) dosyayı yaratır ve kaydeder. Linker'in yarattığı bu son birleşik kodlar yürütülebilir biçime getirilmiş programlardır. Windowsta yürütülebilir dosyalar `.exe` uzantısını alır. Onları koşturmak için uzantılarını yazmaya gerek yoktur.

Linker object dosyada olmayan çağrılı fonksiyonlar için *library*'ye bakar; bulunca onları birleştirir. Birleşik biçim için makina kodunu yaratır ve kaydeder. Linker birbirlerinden ayrı olarak yazılmış kaynak programlardan yaratılan *object* dosyaları da birleştirerek tek bir program haline getirebilir. Bu birleşme işinin nasıl yapılacağı işletim sistemine, derleyiciye ve linkere bağlı olarak farklılıklar gösterir.

Performans: Bazı derleyiciler, yukarıdakilere ek olarak yeni bir geçiş daha yaparak, optimum performans için kodları yeniden düzenler.

Yükleme : Yürütülebilir hale gelen programın koşabilmesi (run) için onun anabelleğe yüklenmesi gerekir. Bunu yükleyici (loader) yapar. Bunun için, yürütülebilir programın adının yazılıp Enter'e basılması yeterlidir. .exe uzantısını yazmak gerekmez.

Koşturma : Yukarıdaki işlem yapılınca, CPU devreye girer ve programdaki komutları sırayla ve birer birer işler. Böylece kaynak programda istenenler gerçekleşir.

Artık, *deneme.exe* adını almış olan programı koşturabiliriz:

```
deneme
```

Bu program koşunca ekrana

```
Merhaba C
```

metni yazılacaktır.

Tümleşik Yazılımlar (IDE)

Son yıllarda yazılan bazı tümleşik yazılımlarda (IDE), yukarıda sıralanan evreler kısmen ya da tamamen kendiliğinden yapılmaktadır. Birçoğunun içinde kendi editörü vardır. Ana menüden kullanıcının istedikleri seçilebilir. Bu derste kullanacağımız `Dev C++` yazılımının kullanılışına bakınız. İsteyenler *Eclipse*, *Netbeans*, *Visual Studio* vb yazılımları da kullanabilirler.

Hata ayıklayıcı (debugger)

Derleyicinin gördüğü yanlışları ayıklayıp düzeltmeye yarayan programlardır. İyi derleyicilerin hemen hepsi, kaynak programdaki yanlışları satır numaraları ile bildirir.

3.3 Derleyiciler

C dili ile yazılan programları koşturmak için, öncelikle onların derlenip makina koduna dönüştürülmesi gerekir. C dili için ücretli ve ücretsiz çok sayıda derleyici vardır. Bazıları IDE (Integrated DevelopmentEnvironment) adı verilen görsel bir arayüze sahiptir. IDE içinde programları yazmak için bir editör vardır. Editör, genellikle anahtar sözcükleri ve başka tipleri renkli gösterir. Satır numaralarını gösterir. Kelime tamamlama yeteneği olabilir. Bir komutu yazmaya başlayınca, editör onu tamamlayabilir. Hata ayıklamayı ve güncelleştirmeyi kolaylaştırmak için, satırlara sıra numarası verir. Ücretsiz IDE'ye örnek olarak *Eclipse*, *DevC++* önerilebilir. Ücretli olanlar arasında *Microsoft VisualC++* ya da *Visual Studio* önerilebilir. Kısa süreli deneme kullanımları için internetten ücretsiz indirilebilir, ama ticari program yazmaya başlarsanız lisans almak zorundasınız.

Çalışırken herhangi bir derleyici kullanabilirsiniz. Ücretsiz bazı derleyicileri

```
http://delorie.com/djgpp/
```

sitesinden görebilirsiniz. Buradan indireceğiniz derleyicilerle yazacağınız yazılımları lisanssız olarak pazarlayabilirsiniz.

Biz, derslerde Dev-C++ yazılımını kullanacağız. Ücretsiz bir IDE'dir. Bizim yapmak istediğimiz her işi yapabilir. Bu IDE'yi

```
http://www.bloodshed.net/devcpp.html
```

web sitesinden indirebilirsiniz.

Şimdi basit bir örnek inceleyelim:

Program 3.5.

```
#include <stdio.h>

int main()
{
    printf("C dili ile yazdığım ilk program!\n");
    return 0;
}
```

Açıklamalar:

1.satır: #include <stdio.h> deyimi, standart giriş/çıkış (I/O) kütüphanesini yazdığınız programa çağırır. Sanki *studio.h* içindeki her şeyi kendiniz yazmışsınız gibi programınıza katılır.

3.satır: *int main()* deyimi `main` fonksiyonunun bildirimidir (declare). Her C programı `main` fonksiyonunu içermelidir. main foksiyonunun önüde `int` anahtar sözcüğü, sonunda () parantezi var. Her fonksiyon bildiriminde bu ikisi gereklidir. Fonksiyonun vereceği (döndüreceği -return-) veri tipi fonksiyon adının önüne yazılır. Onun fonksiyon olduğunu derleyiciye bildirmek için sonuna () parantezi konulur. `main` özel bir fonksiyondur. Onun önüne `int`, sonuna () konulması genel kurala uyar. Ama bazı derleyiciler main fonksiyonunun önüne int koymayı gerekli görmez, ama sonundaki () parantezi mutlaka konulur. () içine gerekirse parametreler konulabilir. Bu derste kullandığımız Dev-C++ derleyicisi `main` fonksiyonunun önüne `int` konulsa da konulmasa da çalışır.

4.satırdaki açılış ({) parantezi `main()` fonksiyonunun gövdesini oluşturan blokun başladığı yerdir. 7.satırdaki (}) kapanış parantezi main() fonksiyonunun gövdesini oluşturan blokun bittiği yerdir. main() fonksiyonunun yapacağı her iş { } bloku içine sırasıyla yazılır. İlk yazılan deyim ilk işleyen deyim, son yazılan deyim, son işleyen deyimdir. Önişlemciler dışında, programın başka yerlerinde ne yazılırsa yazılsın, ancak bu blok içine yazılan deyimler çalışır.

5.satır: `printf()` fonksiyonu () parantezi içine yazılan parametreyi ya da parametreleri standart çıktıya gönderir. Çoğu sistemde olduğu gibi,

bizim için standart çıktı birimi ekrandır. Standart çıktı yazıcı, dosya vb olarak da tanımlanabilir.

`printf()` fonksiyonunun () parantezi, fonksiyon parametrelerinin yazıldığı yerdir. Fonksiyon tanımlarında hiç parametre yoksa bile () parantezleri yazılır. Derleyici onun fonksiyon olduğunu () parantezinden anlar. () içine her veri tipinden parametre konulabilir. Parametre string ise mutlaka " " içine alınır; çünkü C dilinde stringler " " içine yazılır.

6.satır: `return 0` değeri, fonksiyonun döndürdüğü (bize verdiği) değerin 0 olduğunu gösteriyor. Burada 0 değeri *int* tipindendir dolayısıyla *main()* fonksiyonunun önüne, döndüreceği değeri belirtmek üzere konulan `int` veri tipi ile uyumludur. Bir fonksiyonun return değeri daima gövdedeki son deyimdir. *return 0* olması iyidir; çünkü burada 0 değeri, işi isteyen kabuğa *"hata yok"*, istediklerinin hepsini yaptım, diyor.

3.4 derleme

C/C++ kaynak programlarını yazmak, derlemek (compile) ve koşturmak (run) için çok seçenek vardır. C/C++ dilini öğrenmek isteyen ya da öğrendiğini uygulamak isteyen kişi, istediği derleyiciyi seçebilir. Kaynak programı yazmak için, kullanıcı kendi işletim sisteminde uygun bir editör seçebilir. Editörler de hünerlerine göre sınıflandırılabilir.

Kaynak programın yazılıp kaydedildiğini varsayalım. Programı derleyip koşturmak için önümüzdeki olanakları iki ana gruba ayırabiliriz. Kullanıcı iki yoldan birisini seçmelidir.

1. Komut satırı kullanmak
2. IDE kullanmak

3.5 Komut Satırı Kullanmak

Komut satırından yönetilebilen çok derleyici vardır. Bunlar arasında en yaygın olanı GNU projesi çerçevesinde geliştirilen `gcc` derleyicisidir.

GNU derleyicisi

Bilindiği gibi, GNU projesi çerçevesindeki yazılımlar açık kaynak kodlu genel kamu lisanslıdır. GNU projesi UNIX'e benzeyen, onun kadar yetenekli

olan ama UNIX kodu içermeyen açık kaynak kodlu ve ücretsiz bir işletim sistemini kullanıcıların emrine sunmak amacıyla 1984 yılında başlatıldı. Orijinal GNU C derleyicisi GNU projesinin kurucusu olan Richard Stallman tarafından yazıldı. Geçen zaman içinde bu derleyici *C++, Objective-C, Java, Fortran ve Ada* dillerini de içine alacak biçimde genişletildi.

Bu gün `gcc` adıyla anılan derleyici en yaygın kullanılan derleyici olmuştur. *Linux* çekirdeği üzerine kurulu işletim sistemlerinin hemen hepsi `gcc` derleyicisini standart olarak içerir. Ayrıca , isteyen herkes `gcc` derleyicisini internetten serbestçe indirip kurabilir. Sürekli geliştirilen bu derleyiciyi yüklemek isteyenler, son sürümünü kurmalıdırlar. Kurulum farklı işletim sistemleri için farklılıklar gösterir. Onların hepsini burada açıklama olanağı yoktur. Ama isteyenler, internetten her işletim sistemi için kurulumun nasıl yapılacağını öğrenebilirler. Zaten üniversitelerin sunucu makinalarında `gcc` kuruludur. Sunuculara erişim izniniz varsa, `gcc` ile programlarınızı derleyip koşturabilirsiniz.

Burada, komut satırından derleme ve koşturma eylemlerinin nasıl yapılacağını gösteren basit bir komut yazacağız.

Uygun bir editörle C/C++ kaynak programını yazıp *deneme.c* (C++ için *deneme.cpp*) adıyla *cprog* adlı dizin içine kaydedelim. `gcc` derleyicisinin ve programlarımızın bulunduğu dizininin PATH çevre değişkenine eklendiğini varsayıyoruz.

Programı derlemek için, komut penceresinden

```
gcc deneme.c -o deneme
```

yazmak yetecektir. Bu komut, `deneme.c` adlı kaynak programı derleyecek ve yürütülebilir `deneme` dosyasını yaratacaktır. Unix benzeri işletim sistemlerinde `gcc` çok amaçlı kullanılabilir. Seçenekler (option) kullanılarak, `object` dosya yaratılabilir ve *linker* ile istenen başka dosyalara bağlanabilir.

`gcc`, hata ayıklama için kullanıcıya geniş olanaklar sunar.

3.6 IDE Kullanmak

Tek kullanıcılı sistemler için bazı tümleşik geliştirme araçları (IDE - Integrated Development Environment) vardır. Olardan birisini sisteminize yükleyebilirsiniz. Tümleşik araçlar, yukarıda söylenenleri, fare tıklamasıyla yapan görsel bir arayüzle gelir. İçlerinde editör, derleyici, hata ayıklayıcı vardır. Kullanıcı arayüzdeki düğmelere basarak, programını yazabilir, kaydedebilir, geliştirebilir, hataları ayıklayabilir, güncelleyebilir, koşturabilir. Bazı editörler, anahtar sözcükleri renkli yazar, sözcük tamamlama yeteneğine sahiptir.

Hata olduğunda, hata ayıklayıcı (debugger) hatalı satırı gösterir. Tümleşik sistemler, program geliştirmeyi çok kolaylaştırır. Ama yeni başlayanlar, düğmelere tıkladığında, arayüzün gerisinde çalışan kodun ne olduğunu bilmelidir.

Tümleşik araçlar arasında Eclipse, Netbeans, Visual C++, Dev-C++ vb sayılabilir. Bunların hepsi `gcc` derleyicisine dayanır. Kullanıcı istediğini seçebilir. Ortak derslerde birliği sağlamak için Bloosshed firmasının `Dev-C++` tümleşik yazılımını kullanacağız. Bu yazılımı internetten ücretsiz indirebilirsiniz. Yazılımın kullanımı, öteki Windows arayüzleri gibidir. Menü ile yönetilebilir Bkz . Bölüm 12.