Programlamaya Giriş

C Temelleri

Hüseyin Ahmetoğlu

C Tarihi

- Hikaye, Cambridge Üniversitesi'nden Martin Richards'ın Temel Kombine Programlama Diline (BCPL) dönüştürdüğü Ortak Programlama Dili (CPL) ile başladı. Bu, esasen kullanıcının bilgisayar belleğine doğrudan erişmesine izin veren türsüz bir dildi. Bu dil, sistem programcıları için kullanışlı hale getirdi.
- ABD, Bell Labs'tan Ken Thompson, kendi varyantını yazdı ve adını B olarak adlandırdı.
- Yine Bell Labs'ta UNIX tasarımcısı ve programcı olarak çalışan Dennis Ritchie B'den yola çıkarak 1970'lerin başında C'yi geliştirdi.
- Daha sonra, UNIX tamamen C'de yeniden yazıldı. 1983'te, C için bir ANSI standardı ortaya çıktı ve uluslararası kabulünü pekiştirdi.
- C, aynı zamanda programcıların donanımla "yakınlaşmalarını" ve bilgisayarla çok daha düşük bir düzeyde etkileşime girmelerini sağlayan üst düzey bir dildir.

Neden C Öğrenmelisiniz?

- **C çekirdek bir dildir.** Hesaplamada, C genel amaçlı, çapraz platformlu, blok yapılı, prosedürel, gerekli bir bilgisayar programlama dilidir. Bazı yaygın ve popüler bilgisayar dilleri C'ye dayanmaktadır. C'yi öğrendikten sonra, büyük ölçüde veya kısmen C'ye dayalı dilleri öğrenmek çok daha kolay olacaktır. Bu diller arasında C ++, Java ve Perl bulunur.
- **C küçük bir dildir.** C yalnızca otuz iki anahtar kelimeye sahiptir ve bunların yalnızca yirmi kadarının ortak kullanımı vardır. Bu, daha hantal dillere kıyasla öğrenmeyi nispeten kolaylaştırır.

Neden C Öğrenmelisiniz?

- **C hızlıdır.** Hızlı çalışan kodlar yazabilirsiniz ve program donanıma çok yakın seviyede olabilir. Bu, derleyici veya çalışma zamanı sisteminin potansiyel olarak tehlikeli bir şey yapmanızı engellemeden, bilgisayarınızdaki düşük seviyeli tesislere oldukça kolay bir şekil
- **C taşınabilirdir.** Bir sistem üzerinde yazılmış C programları, diğer sistemlerde çok az değişiklik yapılarak veya hiç değişiklik yapılmadan çalıştırılabilir. Değişiklikler gerekliyse, genellikle ana programa eşlik eden bir başlık dosyasındaki birkaç girişi değiştirerek yapılabilir. Ön işlemcide derleyici yönergelerinin kullanılması, bir programın birkaç farklı bilgisayar türünde derlenebilen tek bir sürümünü üretmeyi mümkün kılar. İşlev kitaplıkları tüm C sürümleri için standarttır, bu nedenle tüm sistemlerde kullanılabilirler.de erişebileceğiniz anlamına gelir.

C'DE PROGRAMLARIN GELİŞTİRİLMESİ

- C'de bir program geliştirmenin başlıca üç adımı vardır:
- 1. C programını yazma
- 2. Programı derlemek
- 3. Programın yürütülmesi

Yazma veya Düzenleme

- Bu, yeni bir program kodu yazmayı veya bir metin editörü veya bir IDE kullanarak mevcut bir kaynak programı düzenlemeyi ve .c uzantısıyla kaydetmeyi içerir.
- Çoğu programlama dili derleyicisi, programları yönetmek için olanaklar sağlayabilen belirli bir editör ile birlikte gelir. Böyle bir editör, programları yazmak, geliştirmek, değiştirmek, dağıtmak, test etmek ve hatalarını ayıklamak için eksiksiz bir ortam sunar. Bu tür yazılımlar, entegre geliştirme ortamı veya IDE olarak adlandırılır. Bir IDE tipik olarak belirli bir programlama diline adanmıştır. Bu nedenle, belirli programlama paradigmasıyla uyumlu özellikleri içerir.

Programı Derlemek

- Derleme, ön işleme (preprocessing), derleme(compilation), dönüştürme(assembly) ve bağlamayı(linking) içerir.
- **Preprocessing**: C derlemesinin ilk aşamasıdır. Dosyaları, koşullu derleme talimatlarını ve makroları işler. C ön işlemcisi, programı kaynak koddaki önişlemci yönergelerine göre değiştirmek için kullanılır. Önişlemci yönergesi, önişlemciye kaynak kodunun nasıl değiştirileceğine dair özel yönergeler veren bir deyimdir (#define gibi). Önişlemci, derleyici programının derleme adımının ilk bölümü olarak çağrılır. Genellikle programcıdan gizlenir çünkü derleyici tarafından otomatik olarak çalıştırılır.

Programı Derlemek

- **Derleme:** Derleme işleminin ikinci adımıdır. Ön işlemcinin çıktısını ve kaynak kodunu alır ve assembler kaynak kodunu üretir. Derleyici, kaynak programda yer alan her program ifadesini inceler ve dilin sözdizimine ve anlambilimine uygun olduğundan emin olmak için kontrol eder. Bu aşamada derleyici tarafından hatalar tespit edilirse, bunlar kullanıcıya bildirilir. Hatalar daha sonra kaynak programda düzeltilmeli (bir düzenleyici kullanılarak) ve program yeniden derlenmelidir.

Programı Derlemek

- **Assembly**: Derlemenin üçüncü aşamasıdır. assembly kaynak kodunu alır ve ofsetlerle bir assembly listesi oluşturur. Assembler çıktısı bir nesne dosyasında saklanır. Program eşdeğer bir assembly dili programına çevrildikten sonra, derleme sürecindeki bir sonraki adım, assembly dili ifadelerini gerçek makine talimatlarına çevirmektir. Çoğu sistemde, derleyici, derleme işleminin bir parçası olarak otomatik olarak yürütülür. Assembler her bir assembly dili deyimini alır ve bunu nesne kodu olarak bilinen ikili biçime dönüştürür ve bu daha sonra sistemdeki başka bir dosyaya yazılır. Bu dosya tipik olarak UNIX altındaki kaynak dosyayla aynı ada sahiptir ve son harfi "c" yerine "o" (nesne için)'dur. Windows altında, "obj" son eki genellikle dosya adındaki "c" harfinin yerini alır.

Programı Derlemek

- **Linking**: Derlemenin son aşamasıdır. Program nesne koduna çevrildikten sonra bağlanmaya hazırdır. Bağlama aşamasının amacı, programı bilgisayarda yürütmek üzere son bir forma sokmaktır. Fonksiyonlar, her C derleyicisi tarafından sağlanan standart C kütüphanesinin parçasıdır. Program, önceden derleyici tarafından işlenen diğer kaynak programları kullanabilir. Bu işlevler, nesne dosyasına bağlanması gereken ayrı nesne dosyaları olarak saklanır. Bağlayıcı bu bağlantıyı yönetir.

Programı Derlemek

- Bir programı derleme ve bağlama sürecine genellikle **building** (oluşturma) denir. Çalıştırılabilir (**executable**) bir nesne kodu biçiminde olan son bağlantılı dosya, çalıştırılmaya veya yürütülmeye(**executed**) hazır başka bir dosyada sistemde saklanır. UNIX altında, bu dosyaya varsayılan olarak **a.out** adı verilir. Windows altında, yürütülebilir dosya genellikle kaynak dosyayla aynı ada sahiptir ve **.c** uzantısı bir **.exe** uzantısı ile değiştirilir.

Programın Yürütülmesi

- Program yürütüldüğünde, programın her bir ifadesi sırayla yürütülür. Program kullanıcıdan girdi olarak bilinen herhangi bir veri talep ederse, girişin girilebilmesi için program çalışmasını geçici olarak askıya alır. Ya da program, bir fare tıklaması gibi bir olayın gerçekleşmesini bekleyebilir. Çıktı olarak da bilinen program tarafından görüntülenen sonuçlar, bazen konsol adı verilen bir pencerede görünür. Veya çıktı doğrudan sistemdeki bir dosyaya yazılabilir.

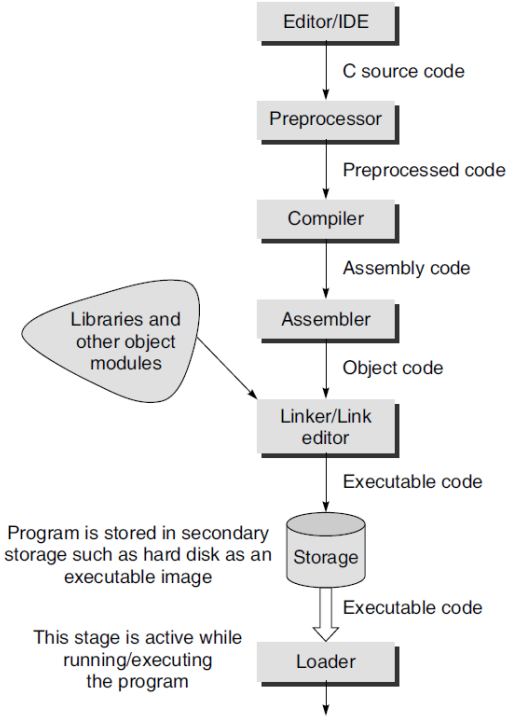
Hatalar

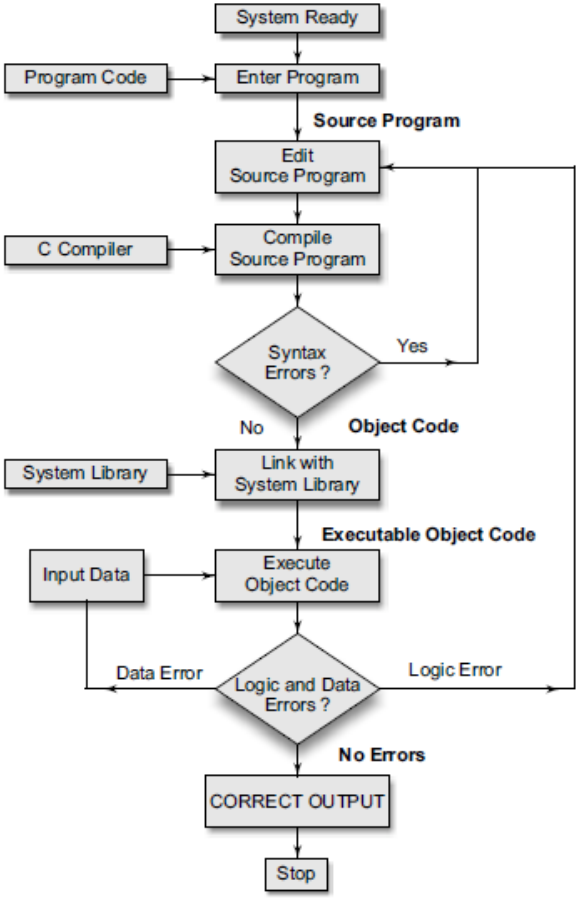
- Her şey yolunda giderse, program amaçlanan görevi yerine getirir. Program istenen sonuçları vermezse, geri dönüp programı yeniden analiz etmek gerekir. Üç tür hata meydana gelebilir:
- **Derleme hataları** Bunlar derleyici tarafından verilir ve programın çalışmasını engeller.
- **Bağlantı hataları** Bunlar, bağlayıcı tarafından veya çalışma zamanında verilir ve programı sonlandırır. Bağlayıcı ayrıca hataları algılayabilir ve raporlayabilir, örneğin, programın bir parçası eksikse veya var olmayan bir kitaplık bileşenine başvurulursa.
- **Çalışma zamanı hataları** Bunlar işletim sistemi tarafından verilir.

Hata ayıklama (Debugging)

- Bir programdan hataları gidermeye hata ayıklama(Debugging) denir. Bir programdaki her türlü hata, «**bug**» olarak bilinir. Hata ayıklama sırasında, bilinen tüm sorunları veya hataları programdan kaldırmak için bir girişimde bulunulur. Programcı, programı adım adım izleyerek, her değişkeni takip ederek programın durumunu izler. Program durumu, basitçe, programın yürütülmesinde belirli bir noktada tüm değişkenlerin değerlerinin kümesidir. Mevcut hesaplama durumunun anlık görüntüsüdür.
- **Debugger**, programcının başka bir programı adım adım çalıştırmasını ve o programın değişkenlerinin değerini incelemesini sağlayan bir programdır. Hata ayıklayıcılar, çeşitli kullanım kolaylığı ve karmaşıklık düzeylerinde gelir. Daha gelişmiş hata ayıklayıcılar, hangi kaynak kod satırının yürütüldüğünü gösterir.

C programlarını yazmak, derlemek ve yürütmek için tipik adımlar



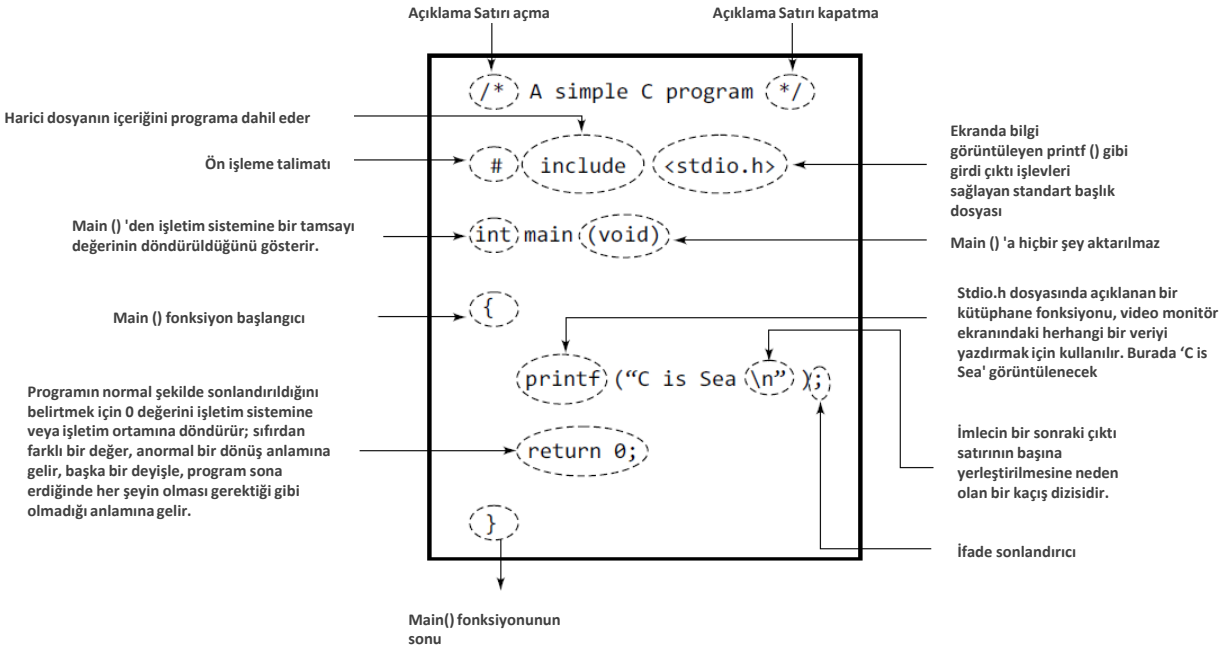

System Ready
Program Code
Enter Program
Source Program
Edit Source Program
C Compiler
Compile Source Program
Syntax Errors ?
Yes
No
Object Code
System Library
Link with System Library
Executable Object Code
Input Data
Execute Object Code
Data Error
Logic and Data Errors ?
Logic Error
No Errors
CORRECT OUTPUT
Stop

‘Hello World!’

[*] main.c

```c
#include <stdio.h>

/* Bu ilk C programım... */

int main(void) {
    printf("Merhaba Dünya!");
    return 0;
}

```

```c
/* A Simple C Program */
#include <stdio.h>
int main(void)
{
  printf(“C is Sea\n”);
  return 0;
}
```


Açıklama Satırı açma
Açıklama Satırı kapatma
/* A simple C program */
Harici dosyanın içeriğini programa dahil eder
Ön işleme talimatı
include <stdio.h>
Ekranda bilgi görüntüleyen printf () gibi girdi çıktı işlevleri sağlayan standart başlık dosyası
Main () 'den işletim sistemine bir tamsayı değerinin döndürüldüğünü gösterir.
int main (void)
Main () 'a hiçbir şey aktarılmaz
Main () fonksiyon başlangıcı
{
Stdio.h dosyasında açıklanan bir kütüphane fonksiyonu, video monitör ekranındaki herhangi bir veriyi yazdırmak için kullanılır. Burada 'C is Sea' görüntülenecek
printf (“C is Sea \n”);
Programın normal şekilde sonlandırıldığını belirtmek için 0 değerini işletim sistemine veya işletim ortamına döndürür; sıfırdan farklı bir değer, anormal bir dönüş anlamına gelir, başka bir deyişle, program sona erdiğinde her şeyin olması gerektiği gibi olmadığı anlamına gelir.
return 0;
İmlecin bir sonraki çıktı satırının başına yerleştirilmesine neden olan bir kaçış dizisidir.
İfade sonlandırıcı
}
Main() fonksiyonunun sonu

main() Fonksiyonu

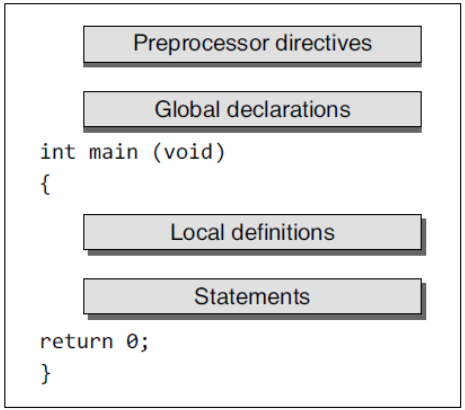
- main (), kullanıcı tanımlı bir metottur.
- main (), program çalıştırıldığında çağrılan programdaki ilk metottur.
- Başlangıç kodu main () metodunu çağırır.
- Programcı, main () metodunun adını değiştiremez.

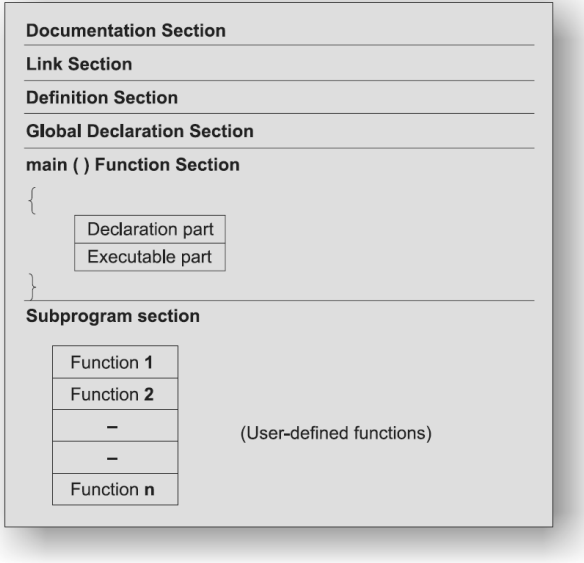
```
int main(void) { /* ... */ }

int main(int argc, char *argv[ ]) { /* ... */ }
```

main() Fonksiyonu

- Main () 'den dönmek veya exit ()' e geçmek için tamamen standart ve taşınabilir yalnızca üç değer vardır:
  - Düz sıradan tam sayı değeri 0
  - Stdlib.h dosyasında tanımlanan EXIT_SUCCESS sabiti
    - 0 veya EXIT_SUCCESS kullanılırsa, derleyicinin çalışma zamanı kitaplığının bunu işletim sisteminin başarılı olarak değerlendirdiği bir sonuç koduna çevirmesi garanti edilir.
  - Stdlib.h dosyasında tanımlanan EXIT_FAILURE sabiti
    - EXIT_FAILURE kullanılırsa, derleyicinin çalışma zamanı kitaplığının bunu işletim sisteminin başarısız olarak değerlendirdiği bir sonuç koduna çevirmesi garanti edilir.

C PROGRAMININ YAPISI





KAYNAKLAR

- Goel, A., & Mittal, A. (2016). Computer Fundamentals and Programming in C (RMK). Pearson Education India. Retrieved from https://www.oreilly.com/library/view/computer-fundamentals-and/9789332579200