Tarihçe

1972 yılında Bell Laboratuvarlarında Unix işletim sistemi ile kullanılmak için tasarlanmış, genel amaçlı, orta seviyeli ve yapısal bir dildir. Özellikle sistem programlama konusunda tercih edilmektedir. İşletim sistemleri ve derleyici gibi sistem programlarının yazılımında yoğun olarak C programla dili kullanılır. Unix işletim sistemlerinin yaygınlaşması ile birlikte C programlama dili de yaygınlaşmış ve gelişmeler göstermiştir. Günümüzde kullanılan nesne yönelimli programlama dilleri C# ve Java, script dilleri JavaScript bu dilden esinlenerek ortaya çıkmışlardır. Bunlardan başka C programlama dilinin en önemli özelliklerinden bir tanesi de taşınabilir olmasıdır. Taşınabilirlikten kastımız farklı bir derleyici veya farklı bir işletim sisteminde de hiçbir değişiklik yapmadan veya çok az bir değişiklikle kodlarımızı çalıştırmamızdır. C programlama dili bu konuda standart hale gelmiş işletim sisteminden bağımsız bir dildir. Günümüzde kullanılan birçok işletim sisteminde derleyicisi vardır. Windows işletim sistemlerinde çalıştığı gibi Linux ve IOS işletim sistemlerinde de kullanabilirsiniz.

Neden C?

C programlama dilini kullanmamız için önemli nedenler aşağıda sıralanmıştır.

- ✓ C, güçlü ve esnek bir dildir. C ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafik çizebilirsiniz.
- ✓ C özel komut ve veri tipi tanımlamasına izin veren bir dildir.
- ✓ Ortamdan bağımsız, aşınabilirlik konusunda iyi bir dildir.
- ✓ Gelişimini tamamlamış ve standart hale gelmiş bir dildir.
- ✓ Yapısal bir dildir. Fonksiyon adı verilen alt programlardan oluşur.
- ✓ Birçok programlama diline esin kaynağı olduğu için diğer programlama dillerinin öğreniminde temel teşkil eder.

C Programlama Dilinin Genel Kullanımı

```c
/*C Programlama Dilinin Genel Kullanımı*/
//C Programlama Dilinin Genel Kullanımı

#include<stdio.h>

main()
{
    printf("Merhaba");
}
```

Genel kullanım şeklini inceleyecek olursak ilk satırda yer alan /*……*/ simgeleri ve ikinci satırda yer alan // simgeleri açıklama satırları olarak karşımıza çıkmaktadır. Aralarındaki fark ise ilk simge çoklu satırlar içinde geçerlidir. İkinci simge ise tek satır açıklamalarında kullanılır. Bu satırlar derleyici tarafından derlenmezler. Yani oluşacak çıktı içinde yer almazlar. Bu simgelere genel manada açıklama operatörleri denilir.

Kullanım şeklimizi incelemeye devam edersek karşımıza #include <stdio.h> ifadesi gelecektir. Buradaki #include ifadesi bir kütüphane ekleneceğini belirtirken, <….> sembolleri arasındaki kısım ise eklenecek kütüphanenin ismini belirtir. Bu ifadeden genel manada çıkaracağımız şey ise bir kütüphane ekleniyor ve biz yazmış olduğumuz kod içerisinde bu kütüphaneye ait fonksiyonlar kullanacağız. Yani kütüphanelerin içerisinde fonksiyonlar barınmaktadır.

Bir önceki konuda C programlama dilinin fonksiyonlardan oluştuğunu söylemiştik. main() de bu fonksiyonlardan bir tanesidir. Fakat bu fonksiyon özel bir fonksiyondur. Ana program bu fonksiyonda saklanır. Program çalıştırıldığında ilk işlenecek fonksiyon budur. Yani program her zaman main()'den başlar. Dolayısıyla her C programında bir tane main() fonksiyonu bulunur. Bu fonksiyon içerisinde printf("Merhaba") komutu bulunmaktadır. Bu komutu kodumuzda kullanmak için ilgili kütüphane olan stdio.h kodumuzun başlangıcına eklenmiştir. printf fonksiyonu ile ekrana bir şeyler yazdırabiliriz. Örneğin:

printf("Adınız: Fatih"); yazdığımızda ekranda Adınız: Fatih yazacaktır.

printf("Yaşınız: %d",35); yazdığımızda ekranda Yaşınız: 35 yazacaktır. Burada ki %d tam sayılar için kullanılır. Virgül işaretinden sonra gelecek olan değer %d'nin yazılmış olduğu yere gelecektir.

printf("Kilonuz: %f",110.5); yazdığımızda ekranda Kilonuz: 110.5 yazacaktır. %f ondalık sayıların gösteriminde kullanılacaktır.

C Kütüphane Dosyaları

C programlama dilinde kullanılan bazı kütüphane dosyaları aşağıda belirtilmiştir. Bu dosyalar #include<.....> noktalı kısıma yazılarak kullanılır. C programlama diline ait tüm kütüphanelerin sonunda .h uzantısı bulunur. Kodlarımızı yazdığımız dosyanın uzantısı ise .c'dir.

assert.h, ctype.h, errno.h, float.h, limits.h, locale.h, math.h, setjmp.h, signal.h, stdarg.h, stddef.h, stdio.h, stdlib.h, string.h, time.h başlıca C kütüphaneleridir.

Başlangıçta vermiş olduğumuz örnekteki printf fonksiyonu stdio.h kütüphanesine aittir. Bu kütüphane eksik olduğu zaman kodumuz hata verecektir.

Kaynak Kodun Derlenmesi

C programları veya kaynak kodları (source code) uzantısı .c olan dosyalarda saklanır. Kaynak kod, bir C derleyicisi (C compiler) ile nesne koduna (object code) daha sonra uygun bir bağlayıcı (linker) programı ile işletim sisteminde çalıştırılabilen (executable yani .exe) bir koda dönüştürülür. Sonuç olarak kaynak kodumuz kullandığımız işletim sistemine ait derleyici sayesinde işletim sisteminin anlayacağı bir dosyaya dönüştürülecektir.

C Kodlarının Taşıması Gereken Özellikler

Bir C programı aşağıda verilen özellikleri mutlaka taşımalıdır.

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

- ✓ Her bloğun ve fonksiyonun başlangıcı ve bitişi sırasıyla { ve } sembolleridir.
- ✓ C dilinde yazılan kodlarda küçük-büyük harf ayrımı vardır. Yani "A" ile "a" birbirinden farklıdır.
- ✓ Açıklama operatörü /* */ sembolleridir.

Veri Tipleri, Değişkenler ve Sabitler

Veri tipi 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 programlama dilinde dört tane temel veri tipi bulunmaktadır. Bunlar:

- ✓ int
- ✓ char
- ✓ float
- ✓ double

Fakat bazı özel niteleyiciler vardır ki bunlar yukarıdaki temel tiplerin önüne gelerek onların türevlerini oluşturur. Bunlar:

- ✓ short
- ✓ long
- ✓ unsigned

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 programlama dilinde değişkenler, sabitler gibi bilgileri saklayacak yapıların kullanılmadan önce tanımlanması gerekmektedir. Tanımlama işleminin kullanılmadan önce yapılması yeterlidir. Tanımlamalar yapılırken önce veri türü adı yazılır daha sonra tanımlama ismi yazılır. Tanımlama ismini verirken C programlama dili özelliklerine uymak gerekmektedir.

- ✓ Tanımlama isimleri İngiliz alfesinin karakterlerini kullanır.
- ✓ Tanımlama isimleri "_" veya harflerden oluşmalıdır.
- ✓ İlk karakterden sonraki karakterler sayısal değer olabilir.
- ✓ Birden fazla kelimeden oluşan tanımlama isimleri arasına boşluk bırakılmaz.
- ✓ Büyük harf ve küçük harf yazımlarına dikkat edilmelidir.
- ✓ C programlama dilinin kullandığı komut ve ifadeler Tanımlama ismi olarak kullanılamazlar.
- ✓ Değişken isimleri 32 karakterden oluşabilir.

Doğru	Yanlış
_ad	$ad
adSoyad	ad soyad
ad_soyad	Ad$soyad
_1ad	1ad
PRINTER	if

Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır. Ancak, veri tipinin önüne “const” anahtar sözcüğü konmalıdır. Örneğin:

- ✓ const float PI = 3.142857;
- ✓ const double NOT = 1254.45778;
- ✓ const char[] =”fatih Albayrak”;

gibi sabit bildirimleri geçerli olup bunların 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. Birçok C programında sabitler #define önişlemci komutu ile de tanımlandığını görebilirsiniz.

#define MAX 100

#define YARICAP 14.22

Karakter sabitler, bir harf için tek tırnak, birden çok karakter için çift tırnak içinde belirtilirler.

'A' /* bir karakter */

"Merhaba Dunya" /* bir karakter kümesi */

Değişkenler ve sabitlerin ekrana yazdırılırken kullanım şekilleri;

```c
#include <stdio.h>

#define PI 3.141593

int main()
{
    const int MAX = 100;
    char c = 'a';
    char *s = "Bu bir sicim";
    int i = 22;
    float f = 33.3;
    double d = 44.4;

    printf("PI = %f\n",PI);
    printf("MAX= %d\n", MAX);
    printf("c = %c\n", c);
    printf("s = %s\n", s);
    printf("i = %d\n", i);
    printf("f = %f\n", f);
    printf("d = %f\n",d);
    printf("d = %.2f\n", d);
    return 0;
}
```

Değişkenlerin Tanım Aralıkları

Değişkenler tanım aralıklarına göre iki sınıfta gruplandırılabilirler.

Yerel (Local) Değişkenler: Yerel değişkenler kullanıldıkları fonksiyon veya alt program içerisinde tanımlanır ve yalnızca tanımlandıkları alan içerisinde tanınır ve kullanılır.

```c
#include <stdio.h>

int topla(int a, int b)
{
    int c = a + b;
    return 0;
}

int main()
{
    topla(2,5);
    printf("toplam= %d", c);
    return 0;
}
```

Yandaki kod bloğunda topla isimli fonksiyon içerisinde tanımlanan değişkenler vardır. main fonksiyonunda topla fonksiyonunu çağırıp kendisine değerler gönderdik. Fakat sonucu ekrana yazdırırken c değişkeni main bloğu içerisinde tanımlı olmadığı için program altını çizdi. Hata verdi.

Genel(Global) Değişkenler: Genel değişkenler bütün fonksiyonların dışında bildirilir. Bir değişken program boyunca sürekli olarak kullanılıyorsa genel olarak bildirilmelidir.

```c
#include <stdio.h>

int c;

int topla(int a, int b)
{
    c = a + b;
    return 0;
}

int main()
{
    topla(2,5);
    printf("toplam= %d", c);
    return 0;
}
```

Yazmış olduğumuz kod üzerinde c değişkeni üzerinde küçük bir değişiklikle kodumuzu çalışır hale getirebiliriz. C değişkenini topla fonksiyonu içerisinde değil de global alanda tanımladığımızda programımız çalışır hale gelir.

Tip Dönüşümleri

```c
#include <stdio.h>

int main()
{
    int a = 11;
    int b = 4;
    float c, d, e;

    c = a / b;
    d = (float)a / b;
    e = a / (float)b;

    printf("c= %f\n", c);
    printf("d= %f\n", d);
    printf("e= %f\n", e);
    return 0;
}
```

Kod yazarken yapmış olduğumuz işlemler içerisinde birçok değişken ve sabit olabilir. Bu değişken ve sabitlerin veri tipleri yapacağımız işlem için önemlidir. Veri tipi karmaşasının önüne geçebilmek için elde edilecek sonuç tipine dönüşümler yapmak gerekir. Bir işlem içerisinde değişime uğrayan değişkenler kendi veri tiplerini korurlar. Tip dönüşümü geçici olarak yapılır. Dönüştürülen tip için bellekte geçici bir yer ayrılır. İşlem sona erdiği zaman bellekte ayrılan alan serbest bırakılır.

Yandaki kod bloğunu incelediğimizde a ve b isimli iki tam sayıyı birbirine böldüğümüzde sonucun çeşitli durumlarının ekrana yazdırıldığı görülecektir. Aynı bölüm değerlerinin c, d ve e float değişkenlerine atanmasının sonucunu aynı beklerken sonucun farklı çıktığı gözlemlenir. Bunun nedeni veri tipi değişikliğidir. Başlangıçta int olan değerleri olduğu gibi bıraktığımızda sonucun virgülden sonraki kısmının alınmadığını görürüz. Fakat ikinci ve üçüncü durumlarda virgülden sonraki kısmın doğru olarak sonuca yansıdığı görülür. Tip dönüşümü düzgün yapılmayan hesaplama işlemlerinde yanılabiliriz. Bu yüzden tip dönüşümü işlemini örnekler üzerinde tekrarlamalıyız.