**Programlama Dili Öğreticisi**

Programlama dilleri arasında geniş bir kullanım alanı ve öneme sahip olan C dilini anlatmak ile başlayacağız. Tüm kainat ve kainatın içinde olan her şey Yaratıcı tarafından kodlanmış bir programa benzer aslında. Somut örneklerle Bir edinimin akılda daha kalıcı olacağına inandığımız için bizzat içinde bulunduğumuz “Yaşam” kavramı üzerinden programlama mantığını C dili ile anlatmaya çalışacağız. Umarız keyif alırsınız, haydi başlayalım…

**Temel Giriş/Çıkış İşlemleri**

İnsan beyni yaratıcı değildir insan beyninin bir şeyler üretmesi için dışarıdan veri alması mutlak suretle gereklidir. İnsan aldığı bu veriyi beyninin kendine has çalışma şekli ile yorumlayıp dışarıya bir fiiliyat olarak yansıtır ki buna da davranış denir. Örnek olarak bir bebeğin su isterken önceden sadece su demesi ve zaman geçtikçe susadım diyecek duruma gelmesi gibi. Beyin verileri alır işler ve buna göre çıktılar üretir. C programlama dilinde de bu böyledir. Programa kullanmasını istediğimiz bir şeyler fısıldamak istiyorsak bunu Scfan(); fonksiyonu ile yaparız. Bunu kullanarak programa dışarıdan veriler gireriz. Programın bir çıktı üretmesini ise Printf(); fonksiyonu ile ekrana yazdırarak yapabiliriz. Printf(); ise programın bize fısıldadığı bölümdür.

Dilerseniz gelin şimdi programlamanın bir katman daha derinine inelim…

#include<stdio.h>

int main( void )

{

printf("Merhaba Dünya");

return 0;

}

Yukarıda ne görüyorsunuz? Yazılar parantezler noktalama işaretleri ilk bakışta çoğu anlamsız gelen karmaşık bir yapı değil mi? Ve aralarında tek tanıdık olarak göze çarpan “Merhaba Dünya” yazısı...

#include<stdio.h>

#İnclude<stdio.h> ile açıklamaya başlayalım;

Yukarıda bahsettiğimiz bir bebeğe bir şey öğretip öğrettiğimiz şeyi söylemesi örneğini hatırlıyor musunuz? Bebeğin bunu yapması için öncelikle bizi duyması ve söylemek istediği şeyi söylemesi için öncelikle duymasını sağlayacak sağlıklı çalışan kulaklara ve söylemek için ise ses tellerine, dudaklara, ağıza ihtiyacı vardır ve bunlar arkasında duyma ve konuşma için sahip olması gereken tüm yetilere. İşte #İnclude<stdio.h> budur. Bir Programın bizi duyması ve konuşması için gerekli olan kütüphanedir. Bunu programın en üstünde tanımlamadan program ne bizi duyabilir ne de konuşabilir. Stdio.h kütüphanesi C’nin içinde geri kalan onlarca kütüphane içinden sadece birisidir. Kütüphaneler programın sağlıklı bir şekilde iletişim kurabilmesi için önceden birileri tarafından bizim için kolaylık olsun diye kodlanmış “Kod Topluluklarıdır.” Yani bu kütüphaneleri programın kullanabileceği yetenekler olarak düşünebiliriz…

Main(); Bir programın olmazsa olmazıdır. İki ayraç ifadesi içinde kalan kısım Programı çalıştıran kısımdır. Bir nevi programın beyni olarak düşünülebilir içi boş ise veya yok ise yaşamda yoktur ☺

int main( void )

{

“Programın yaptığı tüm işler bu iki parantez arasında kontrol edilir”

Return ile Ve program geriye bir değer döndürür.

}

Dikkatinizi “ printf(“Neden noktalı virgül?”); “ burada gördüğünüz parantez sonundaki “noktalı virgüle” dikkat çekmek istiyoruz. İstisnai yerler olmak ile birlikte her komut satırının sonunda noktalı virgül bulunur. Bu programa “Evet işte bu kadar bu komut bitti” anlamında bir göstergedir.

Program komut satırını algılar okumaya devam eder ve noktalı virgülü görünce “Tamam burası bu kadarmış” deyip programı çalıştırmaya çalışır.

Bir programın en temel davranışı olan girdi işlem ve çıktı mantığı ile ilgili şöyle bir örnek verirsek daha iyi oturacaktır;



Işık ağaçlar için girdidir. Basitçe, ağaçlar, içlerinde Yaratıcı tarafından kodlanmış işlemler ile bu girdiyi, ışığı işleme sokarak sonunda bir çıktı yani Oksijen üretir ☺

Programlama mantığı tamamen budur. Girdiler vardır işlemler vardır ve sonunda beklediğimiz bir çıktı vardır.

#### Değişken nedir? Tanımı nasıl yapılır?

Değişkenler programın veri tutucularıdır. Yani ne? Şöyle ki programa sayısal bir ifademi eklemek istiyoruz eklemek istediğimiz sayısal ifade rasyonel bir sayı mı tam sayı mı veya bunların hiçbiri bir karakter mi yada kelime mi eklemek istiyoruz işte tüm bunları önceden bilgisayara “Bak ben şimdi şu tarzda bir veri ekleyeceğim” dediğimiz kısma değişken ifadeleri denir. Eğer bunu demez isek programımız hata verir “Böyle bir şey yazdın bu hangi tarz da bir veri?” der.

Görüyorsunuz ki bilgisayara neredeyse her şeyi biz anlatmak ve öğretmek zorundayız tıpkı bir bebeğe her şeyi anlatır, öğretir gibi. ☺



Bir örnekle açıklamaya çalışalım;

#include<stdio.h>

int main( void )

{

int x;

x = 25;

printf("x sayısı %d",x);

return 0;

}

Veya

#include<stdio.h>

int main( void )

{

int x = 25;

printf("x sayısı %d",x);

return 0;

}

İnt : İngilizce “İnteger” yani Tam sayı kelimesinin kısaltılmışıdır. Yani bilgisayara “sana bir tam sayı değeri gönderiyorum” dedik ve sayımızı girdik. Artık x değeri içinde 25 sayısını tutuyor. Örneğin programda x + 5 gibi bir işlem yaparsak sonucu 30 çıkacaktır. Gayet basit. Printf fonksiyonun içinde daha önce görmediğimiz bir şey çıkıyor karşımıza %d ifadesi. Bu ekrana bir tam sayı yazdırılacağını belirtiyor dikkat etmemiz gereken bir diğer husus tırnaklardan sonra virgül koyup yazdırmak istediğimiz ifadeyi yazmak.

Önemli Bir Uyarı! Her zaman eşitliğin sağındaki ifade solundaki ifadeye eşitlenir. Eşitliğin sol tarafında sadece sağ tarafındaki işlemin tutulacağı değişken ismi olmalıdır. Sol tarafta işlem yapılmaz. Şöyle ki;

{

int x = 25;

int y;

y + 5 = x; <- Yanlış!!!! Doğrusu -> x = y + 5

return 0;

}

Burada Kullanıcıdan değişkenlere basit atama işlemleri yapması ve ekrana bastırması istenir.

Bu kısmı anladıysak şimdi değişken tanımı ile ilgili başka bir örnek üzerinde detaylıca gösterelim;

#include<stdio.h>

int main( void )

{

int sayi; <- Tamsayı olarak “sayi” isimli bir veri oluşturduk.

printf("Değer giriniz"); <- Kullanıcıdan sayı girmesini istedik.

scanf("%d",&sayi); <- Dışarıdan değer alıyoruz. &d ile tam sayı

olduğunu belittik, &sayi diyerek bu tam sayı olan önceden

bilgisayara tanıttığımız verinin ismini giriyoruz. & <- Bu kalıp

bir ifadedir C programlama dilini yapan büyüklerimiz böyle bir

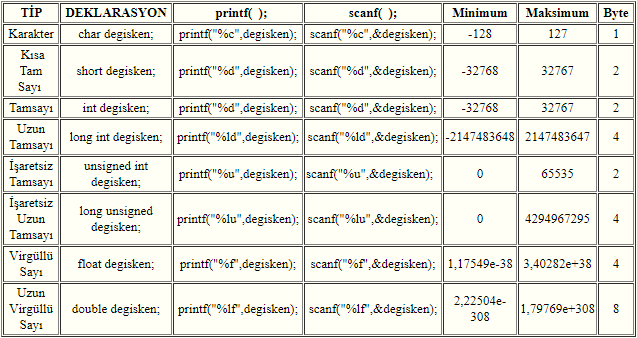
kural koymuş ☺

printf("Girilen değer: %d",sayi); <- Artık programımız bize bir

şeyler fısıldıyor ☺

return 0; <- Geriye herhangi bir değer döndürüyoruz.

}

****

**Şekil 1.**

Şekil 1’de kullanılabilecek tüm sayısal değişkenler ve sahip oldukları özellikleri gösterilmiştir. Bunları en az birer kez kullanarak alıştırma yapmanızı öneririz.

#### Aritmetik Operatörler ve İfadeleri

( + ) : Artı

( - ) : Eksi

( / ) : Bölme

( \* ) : Çarpma

( ^ ) : Üs alma

( % ) : Modül

#### Bilgisayarın matematiği iyidir en azından ona bu ifadelerin neye yaradığı daha önceden öğretilmiştir. örneğin +’nın toplama -‘nin çıkarma işlemini olduğunu bilmektedir. Bize ise sadece bunları kullanarak işlem yapmak kalıyor. İşlem önceliği ise tıpkı matematikteki gibidir. ☺

#### Basit bir örnek vereceğiz bu örneği inceleyip farklı operatörler kullanara kendi örneklerinizi oluşturup kodlamanızı tavsiye ederiz.

#include<stdio.h>

int main( void )

{

float sayi1,sayi2,ortalama;

printf("İki sayı giriniz");

scanf("%f%f",&sayi1,&sayi2);

ortalama = ( sayi1 + sayi2 ) / 2;

printf("Ortalama sonucu: %f'dir",ortalama);

return 0;

}

#### Bu kısımdan sonra kullanıcıya buraya kadar olan bölüm ile ilgili basit uygulamalar yapması istenir. Basit sorular sorulur. Kullanıcının anlamasını pekiştirmek için…

#### Açıklama Satırları

#### Hatırlıyor musunuz önceki bazı örneklerimizde kod satırının yanına o kod satırının ne iş yaptığını açıklayan yazılar eklemiştik

#include<stdio.h>

int main( void )

{

int sayi; <- Tamsayı olarak “sayi” isimli bir veri oluşturduk.

printf("Değer giriniz"); <- Kullanıcıdan sayı girmesini istedik.

scanf("%d",&sayi); <- Dışarıdan değer alıyoruz. &d ile tam sayı

olduğunu belittik, &sayi diyerek bu tam sayı olan önceden

bilgisayara tanıttığımız verinin ismini giriyoruz. & <- Bu kalıp

bir ifadedir C programlama dilini yapan büyüklerimiz böyle bir

kural koymuş ☺

printf("Girilen değer: %d",sayi); <- Artık programımız bize bir

şeyler fısıldıyor ☺

return 0; <- Geriye herhangi bir değer döndürüyoruz.

}

#### Bakın burada! En az kod yazmak kadar önemli olan başka bir şeyde yazılan kodun anlaşılır olmasıdır. Buna gerek duyulur ki bizden sonra bizim kodumuza bakan başka bir programcı yapmak istediğimiz şeyi anlasın ve programı kaldığı yerden geliştirmeye devam edebilsin diye veya uzun zaman sonra yazdığımız bir kodu incelediğimizde “Burasıda neymiş?” dememek için açıklama satılarına ihtiyaç duyarız.

#### Açıklama satırı şöyle yazılır;

/\*Çok satırdan oluşan açıklamalar için yıldızlar arasında kalan bütün

alan, yorum olarak değerlendirilir

ve derleyici (compiler) tarafından

işlenmez.

\*/

#include<stdio.h>

int main( void )

{

//Tek satırlık bir açıklama.

printf("Hello World\n");

}

**Burada Kullanıcıdan yorum satırı kullanarak bir kod yazması ve derlenmesi istenir. Kod satırının derleyici tarafından işlenmediğini görmesi sağlanır.**