****

**T.C. SELÇUK ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**PROJE TABANLI ÖĞRENME DERSİ**

**FİNAL RAPORU**

**CRYSTAL PROGRAMLAMA DİLİ ANLATIM**

**PROJE DANIŞMANI: Dr. Selahattin ALAN**

**PROJE SAHİBİ: Erol Can DEMİR**

**NUMARA: 163311036**

**Erol Can DEMİR**

**T.C. Selçuk Üniversitesi**

**Teknoloji Fakültesi**

**2. Sınıf 2. Öğretim Öğrencisi**

--------------------------------------------------------------------------------------------------------------------

--------------------------

**CRYSTAL PROGRAMLAMA DİLİ İLE UYGULAMA VE GELİŞTİRME**

--------------------------

2017

**İÇİNDEKİLER**

**1.GİRİŞ**

**Sayfa 4 ||** 1.1.Tarihçe

**Sayfa 4 ||** 1.2.Tanımlar

**Sayfa 5 ||** 1.3.Crystal Programlama Dili Nedir Ve Başlıca Özellikleri

**Sayfa 5 ||** 1.4.Crystal Programlama Dili İçin Ubuntu Kurulumu

**Sayfa 9 ||** 1.5.Crystal Programlama Dilinde Proje Oluşturma

**2.DEĞİŞKENLER VE VERİ TİPLERİ**

**Sayfa 11 ||** 2.1.Crystal Programlama Dilinde Değişken Oluşturma

**Sayfa 12 ||** 2.2.Crystal Programlama Dilinde Veri Tipleri

**Sayfa 12 ||** 2.2.1.İçinde Tek Veri Bulundurabilen Veri Tipleri

**Sayfa 13 ||** 2.2.2.İçinde Fazla Veri Bulundurabilen Veri Tipleri

**3.KARAR YAPILARI**

**Sayfa 17 ||** 3.1.İf-Else Karar Yapısı

**Sayfa 19 ||** 3.2.Çok Şartlı İfadeler

**Sayfa 20 ||** 3.3.Unless Karar Yapısı

**4.DÖNGÜLER**

**Sayfa 23 ||** 4.1.While Döngüsü

**Sayfa 24 ||** 4.2.Until Döngüsü

**Sayfa 25 ||** 4.3.Upto – Downto Döngüler

**Sayfa 25 ||** 4.3.1.Upto Döngüsü

**Sayfa 26 ||** 4.3.2.Downto Döngüsü

**Sayfa 27 ||** 4.4.Each Döngüsü

**5.METOTLAR**

**Sayfa 33 ||** 5.1.Metot Tanımlama

**Sayfa 34 ||** 5.2.Ana(Main) Metotlar

**6.SINIF(CLASS) YAPISI**

**Sayfa 36 ||** 6.1.Sınıf Tanımlama

**Sayfa 36 ||** 6.2.Sınıflar ile Nesne Tanımlama

**Sayfa 37 ||** 6.3.Constructor Metotlar

**Sayfa 38 ||** 6.4.Getters ve Setters Metotları

**7.TÜR KESMESİ**

**Sayfa 40 ||** 7.1.Tür Açıklama Yöntemi

**Sayfa 41 ||** 7.2.Ebedi Tür Atama Yöntemi

**8.MODÜLLER**

**Sayfa 43 ||** 8.1.Bir Ad Gibi Modül Tanımlama

**Sayfa 44 ||** 8.1.1.Include ile Modül Tanımlama

**Sayfa 45 ||** 8.1.1.Extend ile Modül Tanımlama

**Sayfa 46||** 8.2.Karışabilen Bir Tür Gibi Modül Kullanımı

**9.SABİTLER**

**10.ENUM YAPISI**

**Sayfa 50 ||** 10.1Enum Tam Sayı Değeri Çağırma

**Sayfa 51 ||** 10.2.Enum Tam Sayı Değeri Belirleme

**Sayfa 52 ||** 10.3.Tam Sayı Değeri İle Enum Çağırma

**11.ÖRNEK http SUNUCU PROGRAMI**

**12.ÖZET**

**Sayfa 58 ||** KAYNAKÇA

---------------------------------------------------

**1.BÖLÜM**

---------------------------------------------------

**GİRİŞ**

---------------------------------------------------



**1.1.TARİHÇE**

Crystal programlama dili gelişmekte olan bir programlama dilidir. Temelleri Ary Borenszweig tarafından 2011 yılında atılmaya başlanmıştır. Fakat ilk kararlı sürümüyle birlikte 2014 yılında ortaya çıkmıştır. Açık kaynak kodlu bir programlama dili olduğu için yayınlandığından günümüze kadar gelişiminde birçok kişi önemli rol oynamıştır. Fakat bu dili Ary Borenszweig ve Juan Wajnerman geliştimeye ve sunmaya başlamıştır.

Crystal programlama dili insanlar tarafından basit anlaşılan ve derlemesi kolay bir dildir. Ary Borenszweig tarafından 2014 yılında ilk olarak “Crystal 0.18.0 “ sürümü çıkarıldı. Daha sonra meraklıları ve isteklileri tarafından günümüze kadar gelişerek sürekli yeni sürümleri çıkmış ve halen gelişimi devam etmektedir. En son 13 Temmuz 2017 tarihinde Crystal 0.23.1 sürümü ile en gelişmiş sürümü piyasaya sürülmüştür.

**1.2.TANIMLAR**

Günlük hayatta karşımıza çıkan bir problemi bilgisayar üzerinde çözmek için kurduğumuz sıralı algoritmalara **program** denir. Bu programları kodlayan ve kullanıcıların isteklerini karşılamasını sağlayan kişilere ise **programcı** denir. Programcının bir bilgisayara ne yapmasını istediğini anlatmasının yoluna da **programlama dili** denir.

Bilgisayarlar üzerinde yapılacak bütün işlemlerin makine koduyla yazılmasını isterler. Programlama dilleri ise programcılara bu işlemin bir programlama dili ile yapılmasını sağlarlar. Peki bizim belirli bir programlama diliyle yazdığımız programları bilgisayara nasıl anlatacağız? Bu çevirme işlemine **derleme(compile)** ya da **yorumlama** denir. Programlama dilini makine koduna çevirme işini yapan programa ise **derleyici(compiler)** denir.

Tek bir problemi çözecek davranışın, temel işlemleri yapan komutların veya belirli bir amaca ulaşmak için çizilen yola **algoritma** denir. Algoritmaların bariz belirli bir sonu ve bir başı vardır. Algoritmalar programın tasarımını yapmamızı ve izlememiz gerek yolları bize gösterirler.

**1.3.CRYSTAL PROGRAMLAMA DİLİ NEDİR VE BAŞLICA ÖZELLİKLERİ**

Crystal programlama dili kullanıcısına sağladığı hız ve performans kazançları ile göz doldurmuştur. Açık kaynaklı bir programlama dilidir. Söz dizimi Ruby programlama dili ile benzerlik göstermektedir. Dilin gelişimi hala sürmektedir ve gelişimine katkı sağlamak için <https://github.com/crystal-lang/crystal-book> adresine ulaşabilirsiniz.

Crystal programlama dili bilgisayarımızda performans kazancı sağlar. Buna örnek olarak programcının bu dil ile tasarladığı ve günde ortalama 10 Milyon talep karşılayan bir program RAM belleğimizi 185 MB ve işlemcimizi sadece %30 oranında kullanarak yormuştur.

Aşağıdaki tabloda bir 10.000 web soket bağlantısını karşılayan programın Node JS platformunda ve Crystal programlama dilinde yazılışlarının karşılaştırılması yapılmıştır.

|  |  |  |
| --- | --- | --- |
|  | **RAM** | **İşlemci (CPU)** |
| **Node JS** | 900 MB | %38 |
| **Crystal Programlama Dili** | 11 MB | %1.85 |

Tablodan da anlayabileceğimiz gibi Crystal programlama dili performans açısından bize önemli katkılar sağlamaktadır. En avantajlı yönü budur. Öte yandan Crystal programlama dili ile genel olarak diğer dillerde yaptığımız işleri tabi ki gerçekleştirebiliriz. Fakat her dil gibi bu dilinde gerçekten iyi olduğu bir yöne vardır. Crystal programlama dili web sunucusu(server) programlamada iyidir. Diğer dillere göre kod karmaşası daha azdır. Yani kullanıcıya daha az satırda daha çok iş sağlayabilir.

Crystal programlama dilinin Ruby programlama dili dışında Python, Go, C# ve Rust programlama dillerinden etkilenmiştir. Crystal programlama dili nesne tabanlı bir dildir. Derleyici olarak ise LLVM derleyicisini kullanmaktadır.

Crystal programlama dilinde bir değişken birden fazla veri tipi olabilir. Fakat program çalıştığı anda yalnız tek veri tipi kullanılır.

Crystal programlama dilinin başlıca özellikleri aşağıda gösterilmiştir.

* Ruby dili ile benzerliği
* C dili kadar performanslı olmayı hedeflemektedir
* Kontrol ve bağımsız değişkenlerin türünü belirlemeden kullanma
* Verimli yerel kod derleme
* Kod derleme zamanındaki avantajları
* Statik söz dizimine sahip olma
* Kolay öğrenim sunma
* Kodlama kolaylığı sağlama

**1.4.CRYSTAL PROGRAMLAMA DİLİ İÇİN UBUNTU KURULUMU**

Crystal programlama dili Windows işletim sistemi üzerinden desteklenmemektedir. Linux, Ubuntu, Redhat ve CentOs, Arch Linux, Mac OS X ve Windows üzerinden Ubuntu Bash işletim sistemi ve yöntemleri desteklemektedir.

Windows üzerinden 2 seçenek vardır. Bunlar;

* Windows üzerine Ubuntu Bash kurma
* Windows üzerinde VM ware programı üzerinden sanal bilgisayara Crystal destekleyen işletim sistemleri kurarak

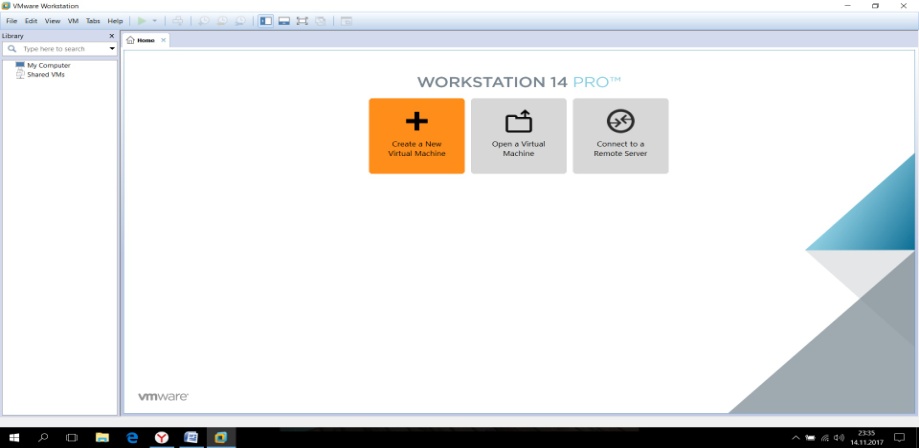
Yukarda anlattığımız gibi Windows üzerinden erişme yöntemlerinden olan VMware Workstation programına Ubuntu işletim sistemi kurmayı göstereceğiz. Aşağıda öncelikle VMware programı üzerinden Ubuntu işletim sistemine sahip sanal bilgisayar kurmayı daha sonra da Crystal programlama dili derleyicisine ulaşmayı adım adım anlatacağız.

***1.Adım***

Bilgisayarımızı VM ware Workstation programını indirip kuruyoruz. VM ware Workstation’un son versiyonu olan VMware Workstation 14 Pro versiyonunu kuruyoruz.

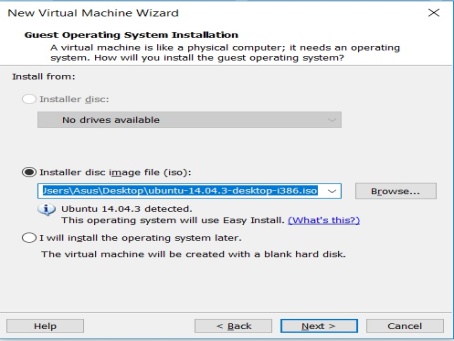
***2.Adım***

Açılan VmWare pernceresinden “Create a New Virtual Machine “ seçeneği ile sanal bilgisayarımızı kurmaya başlayabiliriz.



***3. Adım***

Bu seçeneği seçtikten sonra karşımıza çıkan pencerede 1. Seçenek ile diske Yerleştirdiğiniz işletim istemi kurulum kaynağı ile kuruluma başlayabilirsiniz. 2. Seçenek ile ise bilgisayarınızdaki işletim sistemi iso dosyasını seçerek bu iso dosyası ile kuruluma başlayabilirsiniz.



***4. Adım***

Bu pencerede kuracağınız sanal bilgisayarın adını ve kendi şifrenizi girerek devam edebilirsiniz.

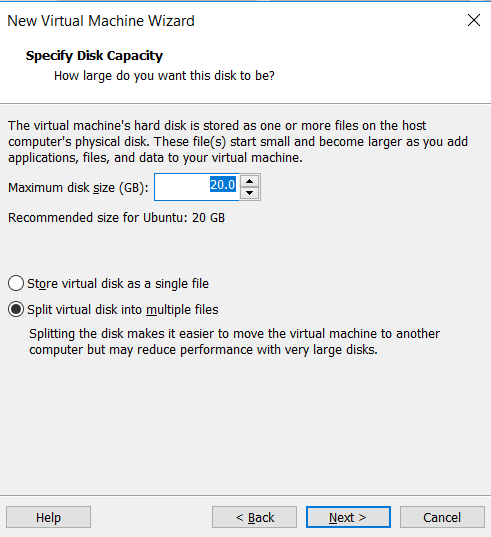
***5. Adım***

Bu pencerede ise sanal bilgisayarı asıl bilgisayarınız da nereye kuracağınızı seçerek devam

edebilirsiniz.

***6. Adım***

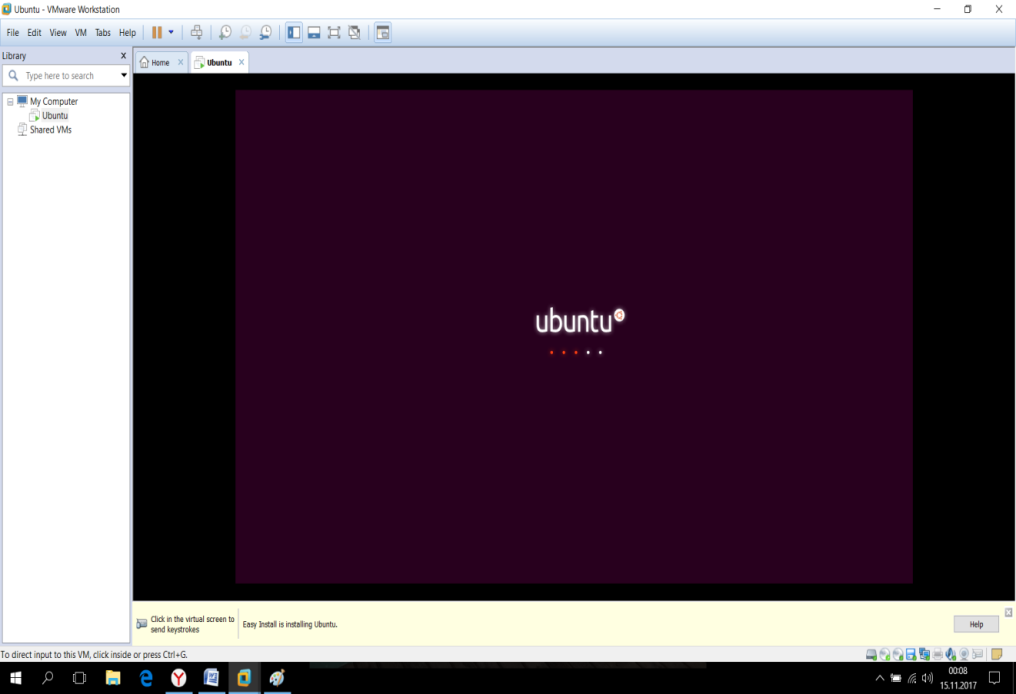
Açılan pencerede ise kuracağınız sanal bilgisayarınızın maksimum hard disk boyutunu belirleyiniz.



***7. Adım***

Açılan son pencerede de kuracağımız sanal bilgisayarın özelliklerine bakarak finish seçeneği

kuruluma başlayabiliriz.



***8. Adım***

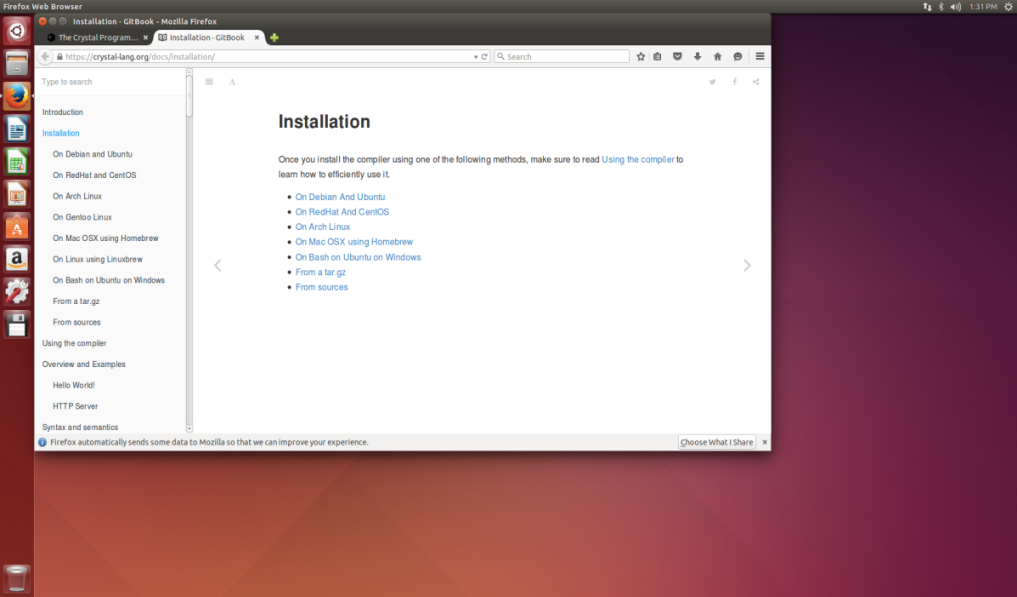
İşletim sistemi kurulumu sona erdikten sonra başarıyla sanal bilgisayarımıza Ubuntu işletim

sistemimizi kurmuş oluyoruz.

***9. Adım***

Şimdi de programlama dilimizin derleyicisini kurmak için <https://crystal-lang.org/>

Sitesine girerek Ubuntu işletim sisteminde kurma seçeneğini seçiyoruz.

******

***10. Adım***

***“On Debian and Ubuntu”*** seçeneğini seçtiğimizde karşımıza bir setup kodu çıkacaktır. ***“curl***

***https://dist.crystal-lang.org/apt/setup.sh”*** bu kodu Ubuntu terminaline yazarak Crystal

depomuzu oluşturmuş oluyoruz. Crystal’i oluşturmak ve üzerinde işlemler yapmak için mutlaka

Crystal deposuna ve bu deponun yapılandırılmasına ihtiyacımız vardır. Daha sonra Depo

yapılandırması için alt satırda yazan kod satırını da ***“apt-key adv-keyserver.gnupg.net – recv***

***keys 09617FD37CC06B54 echo “deb*** [***https://dist.crystal-lang.org/apt***](https://dist.crystal-lang.org/apt) ***crystal main” >***

***/etc/apt/sources list.d/crystal.listapt-get update***” terminale yazıyoruz ve bu sayede Crystal

depomuzu yapılandırmış oluyoruz.

***11. Adım***

Daha sonra Crsytali yükleme aşamasına geçeriz. ***“sudo apt-get install crystal”*** komutunu

Terminale yazınca crystal yüklenmeye başlamıştır. Ayrıca “***sudo apt-get update”*** kodunu da

terminale yazdığımızda Crystal için en son güncellemeyi de yapmış oluruz. Son adımla birlikte

Crystal programlama dilini bilgisayarımıza kurmuş olduk. Bundan sonra Crystal programlama

dilinde projelerimizi oluşturup üzerinde programlar yazacağız.

**1.5.CRYSTAL PROGRAMLAMA DİLİNDE PROJE**

**OLUŞTURMA**

Crystal’i yükledikten sonra artık istediğimiz gibi proje oluşturabilir ve projeyi çalıştırabiliriz. Proje oluşturma ve çalıştırma adımları terminal üzerinden yapılır fakat programımız oluşturduğumuz projenin içindeki ***programadi.cr*** dosyamız içerisine yazılır. Terminal üzerinden proje oluşturmak için öncelikle ***library(kütüphane)*** ***veya application(uygulama)*** projesi oluşturmamıza göre farklı komutlar yazmalıyız. Oluşturmak için ise ***“init”*** komutunu kullanmamız gerekir.

1. Kütüphane Projesi Oluşturma: Kütüphane projesi oluşturmak için ***“ $ crystal init lib kutuphaneadi “*** komutu ile oluştururuz. Komutu yazdıktan sonra sırasıyla

create my\_cool\_lib/.gitignore

create my\_cool\_lib/LICENSE

create my\_cool\_lib/README.md

create my\_cool\_lib/.travis.yml

create my\_cool\_lib/shard.yml

create my\_cool\_lib/src/my\_cool\_lib.cr

create my\_cool\_lib/src/my\_cool\_lib/version.cr

create my\_cool\_lib/spec/spec\_helper.cr

create my\_cool\_lib/spec/my\_cool\_lib\_spec.cr

bu komutlar ekranımızda görünecektir ve library projesi oluşturulmuş olur.

1. Application (Uygulama) Projesi Oluşturma: Uygulama projesi oluşturmak için ***“ $ Crystal init app projeadi “*** komutunu uygularız. Bu kod parçasını yazdıktan sonra sırasıyla

create my\_cool\_lib/.gitignore

create my\_cool\_lib/LICENSE

create my\_cool\_lib/README.md

create my\_cool\_lib/.travis.yml

create my\_cool\_lib/shard.yml

create my\_cool\_lib/src/my\_cool\_lib.cr

create my\_cool\_lib/src/my\_cool\_lib/version.cr

create my\_cool\_lib/spec/spec\_helper.cr

create my\_cool\_lib/spec/my\_cool\_lib\_spec.cr

bu komutlar ekranımızda görünecektir be böylece belirlediğimiz yolda içinde ***“projeadi.cr”*** dosyası olan bir proje dosyası oluşturulmuş olur.

Projemizi oluşturduktan sonra artık program oluşturma kısmına gelmiş oluyoruz. Programımızı yazmak için projeyi oluşturduğumuz yolda ***“src”*** dosyasının içindeki ***“projeadi.cr”*** dosyamızda programımızı yazabiliriz.

Programı çalıştırmak için ise terminalde ***“ $ crystal run projeadi/src/projeadi.cr“*** komutunu yazarak terminal ekranında programımızı çalıştırırız.

---------------------------------------------------

**2.BÖLÜM**

---------------------------------------------------

**DEĞİŞKENLER VE**

**VERİ TİPLERİ**

---------------------------------------------------

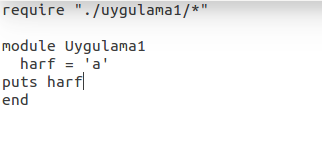
**2.1.CRYSTAL PROGRAMLAMA DİLİNDE DEĞİŞKEN OLUŞTURMA**

Crystal programlama dilinde değişken oluştururken değişken adını ve karşısına değeri yazılır. Crystal veri tipini kendisi değişkenin değerine göre tanımlar.

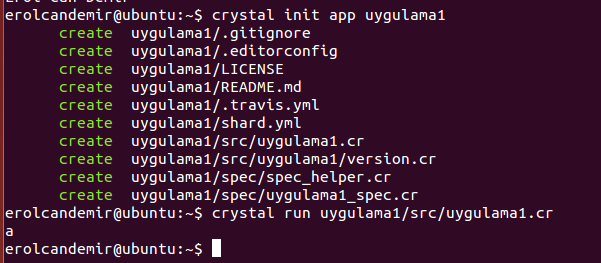
***sayi = 10 🡪 sayi*** değişkenine 10 değerini atadık.

***puts sayi 🡪 sayi*** değişkenini yazdırdık.

**Örnek - Uygulama1**

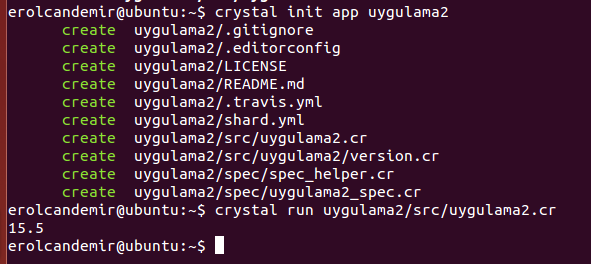
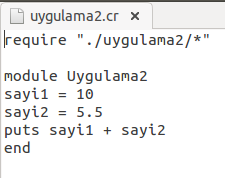


Bu kısımda **.cr** uzantılı dosyamıza programımızı yazıyoruz. **harf** değişkenine **‘a’** karakterini atadık. Daha sonra **puts** komutu ile **harf** değişkenini yazdırdık.



Bu kısımda da projeyi çalıştırıyoruz ve ekrana **a** karakteri yazdırılıyor.

**Örnek – Uygulama2**



**2.2.CRYSTAL PROGRAMLAMA DİLİNDE VERİ TİPLERİ**

**2.2.1.İçinde Tek Veri Bulundurabilen Veri Tipleri**

**Nill Veri Tipi**

Diğer programlama dillerinde NULL (boş) veri tipi Crystal programlama dilinde NİLL olarak kullanılmaktadır. NİLL veri tipi bir değişkene boşluk atamak boş değer atamakta kullanılır.

***VeriTipi = nill*** 🡪 ***VeriTipi*** isimli değişkenimize ***nill*** değer atıldı.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

**Bool Veri Tipi**

**Bool** veri tipi doğru/yanlış değerleri simgeler. Doğru ve yanlış değerleri true ve falseözel amaçlı kelimeler ile tanımlanır. Yani bool tipindeki bir değişken sadece true ya da false değer alabilir.

***VeriTipi = true*** 🡪 ***VeriTipi*** isimli değişkene ***true*** değeri atandı. Artık bu değişken **Bool** veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 true false

**Char Veri Tipi**

Char veri tipi bütün unicode karakterleri içinde barındırır. Char karakterinde bir değişken tanımlanırken atanan karakter mutlaka tek tırnak içerisinde yazılmalıdır. Uzunluğu 2 byte(16bit)’dır.

***VeriTipi = ‘a’*** 🡪 ***VeriTipi*** isimli değişkene ***a*** karakteri atandı. Artık bu değişken Char veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 a

**String Veri Tipi**

String veri tipi uzun veya kısa her türlü metini içerisinde tutar. Atanan karakterler mutlaka çift tırnak içerisinde atanmalıdır.

***VeriTipi = “Crystal”*** 🡪 ***VeriTipi*** isimli değişkene ***Crystal*** metni atandı.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 Crystal

**Integer Veri Tipi**

Hemen hemen bütüm programlama dillerinde bulunan en basit veri tiplerindendir. Kapsama alanı bütün tam sayılardır. Uzunluğu 4 byte (16 bit)‘dır.

***VeriTipi = 5*** 🡪 ***VeriTipi*** isimli değişkene ***5*** değeri atandı. Artık bu değişken Int veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 5

**Float Veri Tipi**

Ondalık sayıları kapsayan iki temel veri tipinden birisidir. Tamsayıları da içerisinde kapsar.

***VeriTipi = 5.6*** 🡪 ***VeriTipi*** isimli değişkene ***5*** değeri atandı. Artık bu değişken Float veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 5.6

**Symbol Veri Tipi**

Symbol veri tipi herhangi bir kelimeyi veya içinde sembol barındıran değişkenleri tanımlamak için kullanılır. İçinde boşluklu yapıyı da kabul eder yani tanımlarken boşluklu ifade de kullanabilme imkanı sağlar.

***VeriTipi = : ”Sembol veri tipi”*** 🡪 ***VeriTipi*** isimli değişkene ***“Sembol Veri Tipi”*** değeri atandı. Artık bu değişken symbol veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 Sembol Veri Tipi

**2.2.2.İçinde Fazla Veri Bulundurabilen Veri Tipleri**

**Array Veri Tipi**

Diziler veri tipidir. Array(dizi) veri tipi içerisinde birden fazla veri bulundurabilen veri tipidir. Bir sınırlaması yoktur içerisine istenilen kadar veri atılabilir.

***VeriTipi = [1,5,7,30]*** 🡪 ***VeriTipi*** isimli değişkene ***[1,5,7,30]*** değerleri atandı. Artık bu değişken Array veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 [1,5,7,30]

\*\*\* VeriTipi = [] of nill 🡪 Dizinin boş değer döndürmesi için of nill komutu kullanılmalıdır. Bu komuttan sonra VeriTipi değişkenini yazdırırsak ekran çıktısı nill olacaktır. Bu kullanım tipi sadece nill için değil istediğimiz diğer veri türlerini de kullanabiliriz. Bunun anlamı dizinin içinde o veri türünden elemanlar vardır.

**Hash Veri Tipi**

Hash veri tipi bir anahtar ve o anahtarın değerini tutan veri tipidir. İki tane değer taşır. Birisi anahtar diğeri anahtarın değeridir. Anahtar ve anahtarın değerinin veri tipleri farklı olabilir.

***VeriTipi = {1 =>”bir” , 2 =>”iki”}*** 🡪 ***VeriTipi*** isimli değişkene ***{1 =>”bir” , 2 =>”iki”}*** değerleri atandı. Artık bu değişken Hash veri tipli. Bu tanımlamada anahtar 1 integer tipinde 1’in değeri “bir” ise string tipindedir.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 {1 =>”bir” , 2 =>”iki”}

**Tuple Veri Tipi**

Tuple veri tipi kümeler diye de adlandırılır. Array veri tipine benzerdir. Array veri tipinden farkı ise Array veri tipi içerisinde tek bir veri tipinden elemanlar tutarken Tuple veri tipinde içindeki elemanların veri tipinin bir önemi yoktur. Yani her veri tipinden elemanları içinde barındırabilir.

***VeriTipi = {“Crystal”,10,’a’,true,100}*** 🡪 ***VeriTipi*** isimli değişkene ***{“Crystal”,10,’a’,true,100}*** değerleri atandı. Artık bu değişken Tuple veri tipli. İçinde ise sırasıyla string, integer, char, bool, integer veriler tutmaktadır.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 ***{“Crystal”,10,’a’,true,100}***

**Range Veri Tipi**

Range veri tipi bir sayı aralığını gösterir ve tanımlanırken atanan iki sayı arasındaki değerleri tutar.

***VeriTipi = 1..5*** 🡪 ***VeriTipi*** isimli değişkene ***1ile beş arası*** değerleri atandı. Artık bu değişken Range veri tipli.

***puts VeriTipi*** 🡪 ***VeriTipi*** değişkenini yazdırıldı.

Ekran Çıktısı 🡪 ***234***

**TypeOf Söz Dizimi**

Typeof söz dizimi bir değişkenin veri tipini verir. Program içerisinde typeof(degiskenadi) şeklinde yazarak değişkenin veri tipini öğrenmiş oluruz.

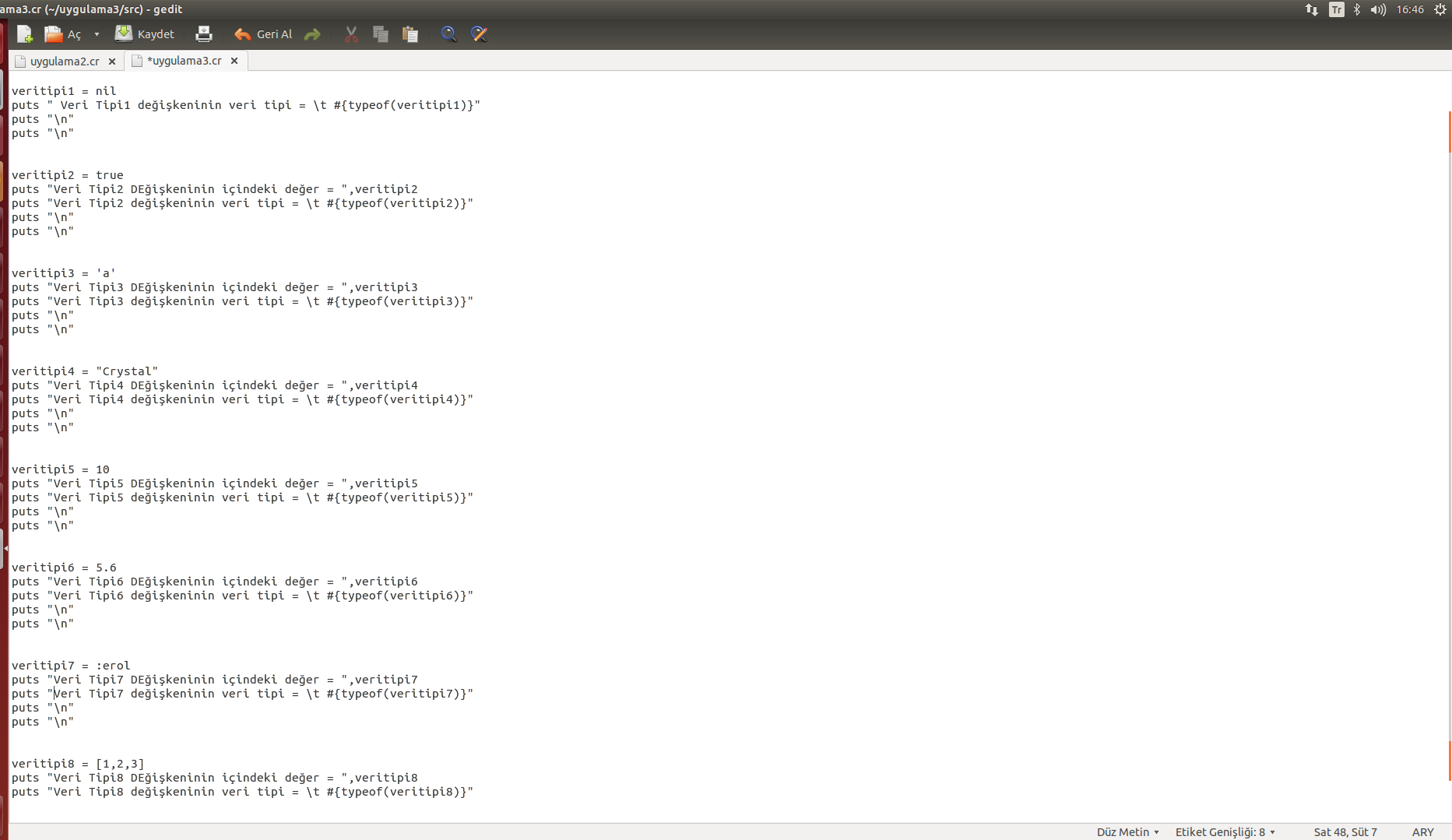
**Kullanımı**

***sayi = 5 🡪*** sayi değişkenine 5 değeri atıldı.

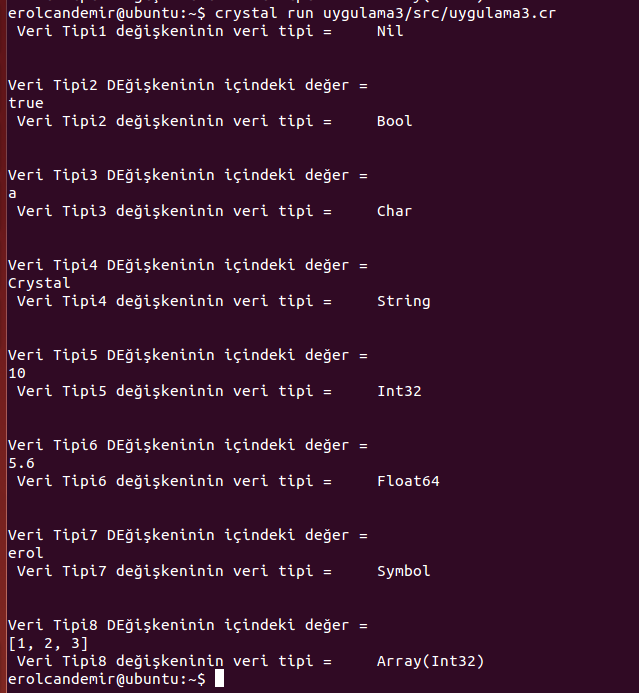
***puts “#{typeof(sayi)}” 🡪* *typeof*** ile sayi değişkeninin veri tipi yazdırıldı.

Ekran Çıktısı 🡪 ***Int32***

**Örnek – Uygulama3**



**Ekran Çıktısı:**

******

---------------------------------------------------

**3.BÖLÜM**

---------------------------------------------------

**KARAR YAPILARI**

---------------------------------------------------

**3.1.İf-Else Karar Yapısı**

İf-else karar yapısı şartlı ifadeleri belirtir. İf ilk şart doğruysa if’in içindeki işlemler eğer if şartı yanlışsa else’nin içindeki işlemler yapılır.

**Kullanımı:**

***if \*\*\*şartlar\*\*\****

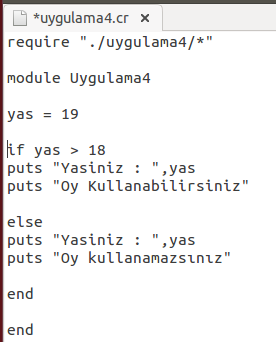
***\*\*\* Yapılacak işlemler\*\*\****

***else***

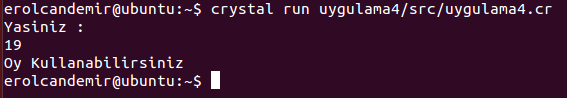
***\*\*\* Yapılacak işlemler\*\*\****

***end***

**Örnek – Uygulama4**



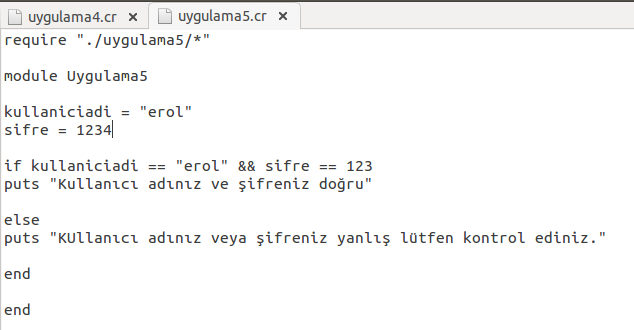
**Ekran Çıktısı:**



Örnek – Uygulama4 örneğinde bir yas değişkeni tanımlanıyor ve eğer yaş 18’den küçükse yaşını ve “Yaşınız 18’den küçük oy kullanamazsınız yazdırıyoruz.”. Else kısmında ise yaşın 18’den küçük olmadığı her durum ele alınır. Bu örnekte yaşın 18 olma durumuyla ilgili bir şey belirtilmemiştir. Herhangi bir belirtme olmadığı için program burada yaşın 18’den büyük olmadığı her durumda yani yaşın 18’e eşit olma durumunda da else bloğunu çalıştıracaktır.

İf bloğunun içerisinde birden fazla şart koşulabilir. Bu şartları birbirine bağlamak için ise **&&(ve)** ya da **||(veya**) operatörleri kullanılır. Ve operatörü sunulan iki şartında doğru olmasını, veya operatörü şartlardan sadece birinin doğru olmasını arar.

**Örnek – Uygulama5**

****

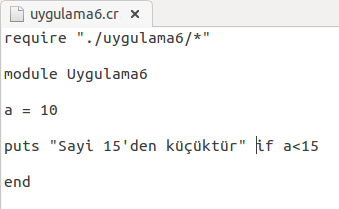
**Ekran Çıktısı:**

**C:\Users\Asus\AppData\Local\Temp\vmware-Asus\VMwareDnD\d8ec7851\Ekran Görüntüsü - 2017-11-17 22^%07^%37.png**

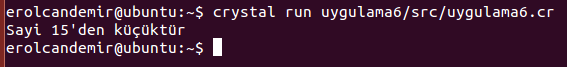
Örnek – Uygulama5 örneğinde bir kullanıcı adı ve şifre tanımlanmış. İf bloğunda iki şart sunulmuş. Eğer kullanıcı adı “erol” a eşitse ve şifre 123 e eşitse “kullanıcı adınız ve şifreniz doğru yazdırıldı. Eğer bu iki şartta sağlanmıyorsa else bloğunda kullanıcı adı veya şifreniz yanlış lütfen kontrol ediniz yazdırıldı. Programın else bloğunu çalıştırmasının sebebi ise tanımlanan şifre değeri 1234 iken if bloğun da şifre şartının 123 olması istenmiş. Şifre değeri yanlış olduğu için program if bloğunu değil else bloğunu çalıştırdı.

İf karar yapısının başka bir kullanım şekli de tek satırda kullanımdır. Önce yapmasını istediğimiz işlemi yanına da if ile şart gerçekleşirse şeklinde kullanılabilir. Yukarda anlatılan kullanımıyla hiçbir farkı yoktur. Sadece programda satır sayısı bakımından avantaj sağlar.

**Örnek – Uygulama6**



**Ekran Çıktısı:**

****

**3.2.Çok Şartlı İfadeler(elsif)**

Çok şartlı ifadeler birden fazla if bloğu isteyen ifadelerdir. Bunun için elsif komutu kullanılır. İf bloğunda olduğu gibi elsif deyiminin yanına da istenilen şartlar yazılarak kullanılır. Bir şartlı ifade de bir tane if ve bir tane else bloğu olur fakat programa kullanıcının şart sayısına göre istediği sayıda elsif bloğu eklenebilir.

**Kullanımı:**

***if \*\*\*şartlar\*\*\****

***\*\*\*Yapılacak işlemler\*\*\****

***elsif \*\*\*Şartlar\*\*\****

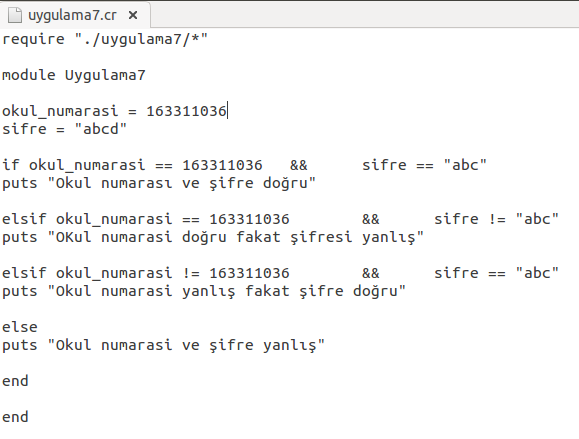
***\*\*\*Yapılacak işlemler\*\*\****

***else***

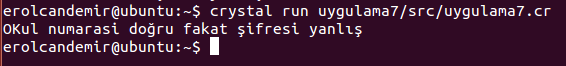
***\*\*\*Yapılacak işlemler\*\*\****

***end***

**Örnek – Uygulama7**

****

**Ekran Çıktısı:**

****

Örnek-Uygulama7 programında okul numarası ve şifresinin her yönden kontrolü yapılmıştır. Eğer okul numarası ve şifresi belirtilen değerlere eşitse if bloğu, okul numarası doğru şifre yanlışsa programda olduğu gibi birinci elsif bloğu, okul numarası yanlış şifre doğruysa ikinci elsif bloğu, okul numarası ve şifre yanlışsa else bloğu çalışacaktır. Programda verilmiş değerlere göre okul numarası doğru fakat şifre yanlıştır bu yüzden program birinci elsif bloğu ile çalışmıştır.

**3.3.Unless Karar Yapısı**

Unless karar yapısı if ile ters bir mantığa sahiptir. Unless false değer döndürür. Unless karar yapısı yanına yazılan şartın değilini sağlıyorsa bloğun içine girerek işlemleri yapar.

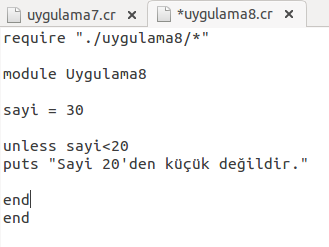
**Kullanımı:**

***unless \*\*şartlar\*\****

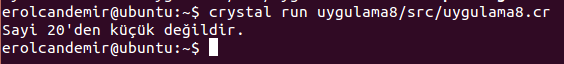
***\*\*Yapılacak işlemler\*\****

***end***

**Örnek – Uygulama8**

****

**Ekran Çıktısı:**

****

---------------------------------------------------

**4.BÖLÜM**

---------------------------------------------------

**DÖNGÜLER**

---------------------------------------------------

Döngüler programda bir şart ya da bir ifade doğru veya yanlış olana kadar programın o işlemi yapmasını sağlar. Bazı durumlarda uzun kod satırlarıyla yapacağımız işlemleri kısa zamanda pratik olarak yapma imkanı sunarlar.

**4.1.While Döngüsü**

While döngüsü programlamada genel kullanım alanına sahip bir döngüdür. Neredeyse bütün programlama dillerinde kendine yer bulmuştur. While döngüsü bir şart doğru olduğu sürece döngü içerisinde belirtilen işlemlerin yapılmasını sağlar.

**Kullanımı:**

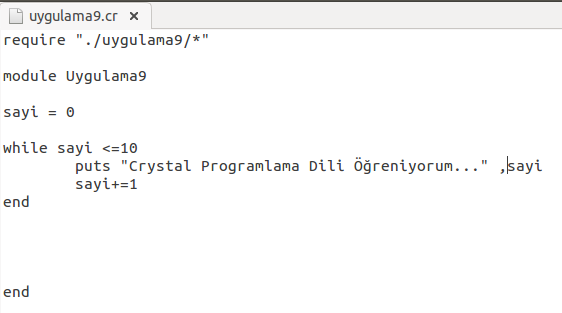
***while \*\*\*şart\*\*\****

***\*\*\*Yapılacak işlemler\*\*\****

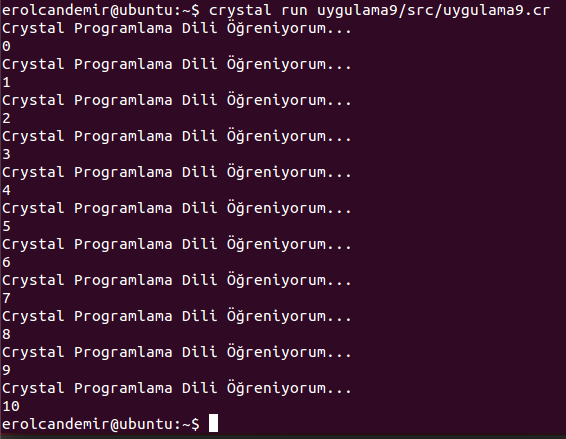
***\*\*\*Arttırma veya azaltma işlemi\*\*\****

***end***

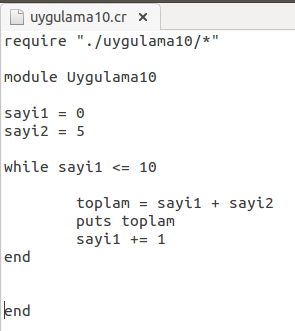
**Örnek – Uygulama9**



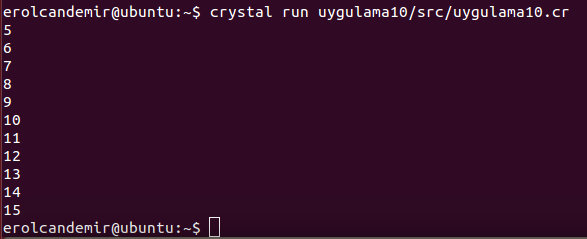
**Ekran Çıktısı:**



**Örnek – Uygulama10**



**Ekran Çıktısı:**

****

Uygulama10 programında birinci sayıyı while döngüsü ile 0 dan 10 a kadar arttırıldı. İkinci sayıya ise sabit olarak 5 değeri atandı. Birinci sayının döngü içerisinde aldığı her değer ikinci sayı ile toplanarak ekrana yazdırıldı.

**4.2.Until Döngüsü**

Until döngüsü bir şart olana kadar içerisinde istenen işlemin gerçekleştirilmesini sağlar. While döngüsünde true değer döndüğü sürece döngü içerisindeki işlemler yapılırken until döngüsünde şart sağlanana kadar yani false değer döndüğü sürece döngü içerisindeki işlemler gerçekleşir ve true değer döndüğü zaman döngüden çıkar.

**Kullanımı:**

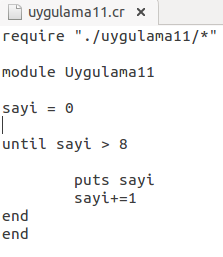
***Until \*\*\* şart ifadesi\*\*\****

***\*\*\*Yapılacak işlemler\*\*\****

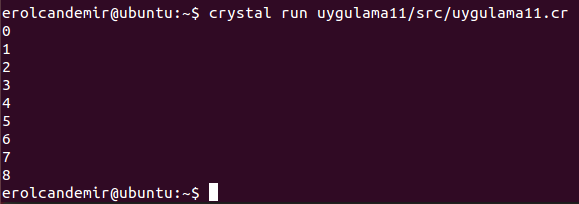
***\*\*\* Arttırma veya azaltma işlemleri\*\*\****

***end***

**Örnek – Uygulama11**



**Ekran Çıktısı:**



**4.3.Upto-Downto Döngüleri**

Upto – Downto döngüleri Crystal programlama diline Ruby dilinden geçmiş döngü çeşitleridir. Diğer döngüler gibi belirtilen şarta kadar içinde belirtilen işlemleri yapar. Fakat döngüyü gerçekleştirmek için döngü yanında do içerisinde bir literatür tanımlama zorunluluğu verir. Döngü içerisinde yapılan işlemler bu literatür arttırarak veya azaltılarak yapılır.

**4.3.1.Upto Döngüsü**

Upto döngüsü bir sayıdan bir sayıyı kadar şartı sunar. Yani Upto döngüsü belirlenen literatürü arttırarak döngü içerisindeki işlemleri gerçekleştirmeyi sağlar.

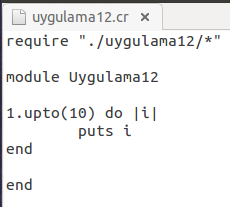
**Kullanımı:**

***Hangi sayıdan . upto ( Hangi sayıya kadar ) do |literatür|***

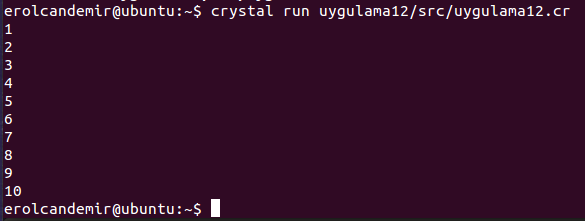
***\*\*\* Yapılacak işlemler \*\*\****

***end***

**Örnek – Uygulama12**



**Ekran Çıktısı:**

****

**4.3.2.Downto Döngüsü**

Downto döngüsü de Upto döngüsü gibi Crystal programlama diline Ruby dilinden geçmiştir. Downto döngüsü upto döngüsünün tersidir. Downto da bir sayıdan bir sayıya kadar işlem yapar fakat sonran başlayarak ve azaltarak işlemleri sıralar.

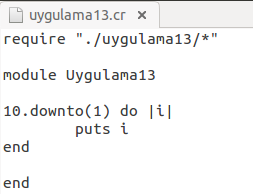
**Kullanımı:**

***Hangi sayıdan . downto ( Hangi sayıya kadar ) do |literatür|***

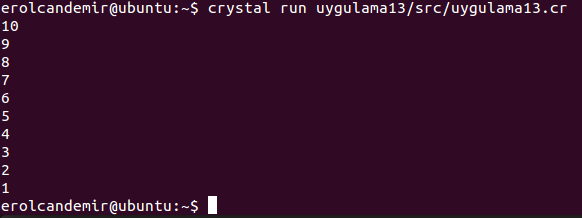
***\*\*\* Yapılacak işlemler \*\*\****

***end***

**Örnek – Uygulama13**



**Ekran Çıktısı:**



**4.4.Each Döngüsü**

Each döngüsünün amacı ve mantığı da diğer döngüler ile aynıdır. Kolay yoldan belirli bir sayıdan belirli bir sayıya kadar içinde belirtilen işlemleri yapar. Upto ve Downto döngüleri gibi Each döngüsü de bir literatüre ihtiyaç duyar ve arttırma azaltma işlemleri bu literatür üzerinden yapılır.

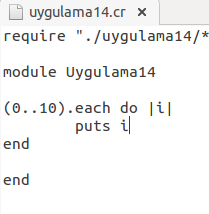
**Kullanımı:**

***(Hangi sayıdan .. Hangi sayıya ). each do |literatür|***

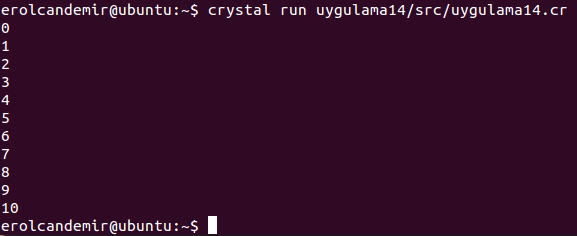
***\*\*\* Yapılacak işlemler \*\*\****

***end***

**Örnek- Uygulama14**

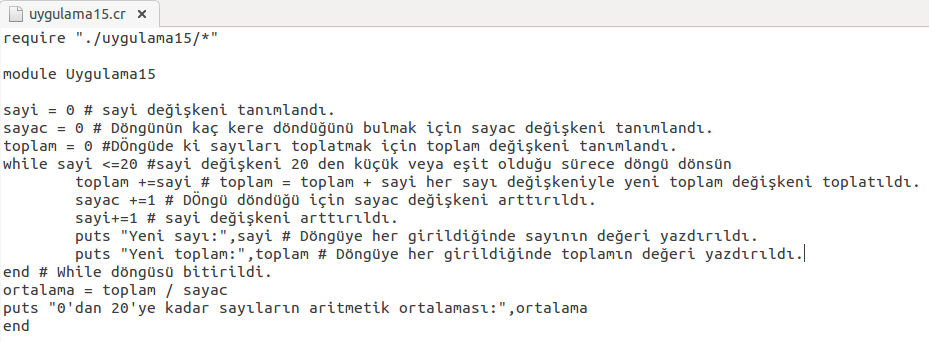


**Ekran Çıktısı:**

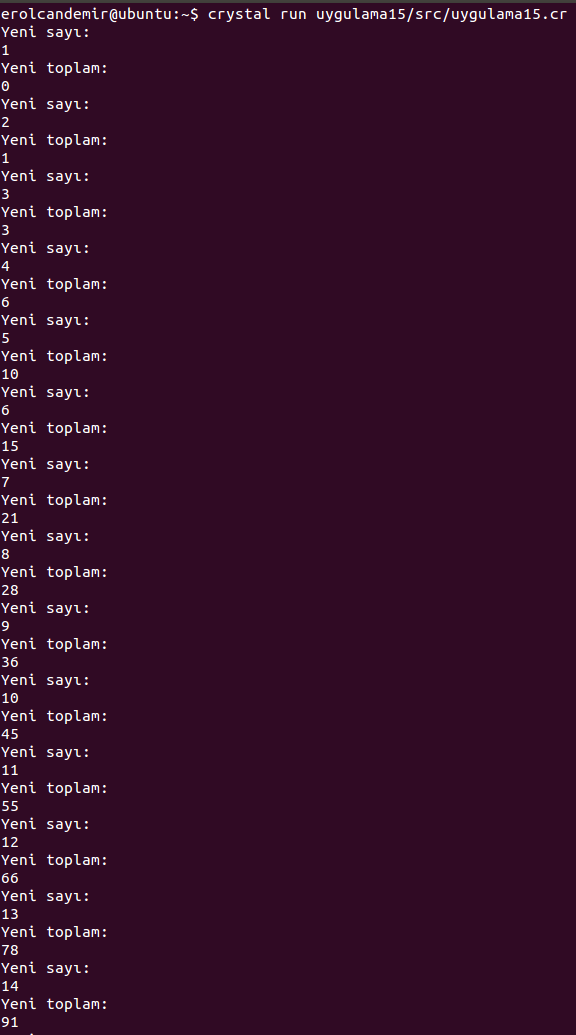
******

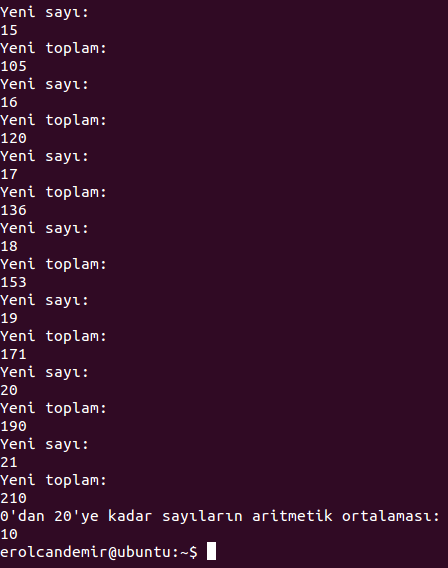
**Örnek – Uygulama15**

0’dan 20 ye kadar olan sayıların aritmetik ortalamasını hesaplayan program.

******

**Ekran Çıktısı:**

******

******

---------------------------------------------------

**5.BÖLÜM**

---------------------------------------------------

**METOTLAR**

---------------------------------------------------

**5.1.Metot Tanımlama**

Metotlar uzun ve yazımı karışık hale gelmiş kodları daha anlaşılır ve sade hale getirmek için kullanılır. Belirli bir metot oluşturulduktan sonra bu metodu programın istenilen yerinde istenilen şekilde kullanılabilir. Kod karmaşasını azaltır belirli işlevleri karmaşasız görmeyi sağlar.

**Kullanımı:**

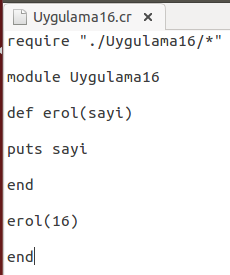
***def metod\_adi (argüman\_adi)***

***\*\*\*Yapılacak İşlemler\*\*\****

***end***

Crystal programlama dilinde metotlar “def” ile tanımlanır “end” ile sonlandırılır. Bir metotta def’ten sonra metodun adı daha sonrada alacağı parametreler yazılarak metot tanımlamayı sonlandırmış oluruz. Daha sonra bu metodun içine uygun ve istenilen işleri yapmasını sağlayacak programımızı yazabiliriz.

**Örnek – Uygulama16**

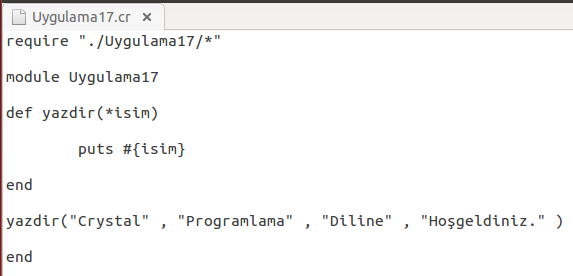


**Ekran Çıktısı:**

**C:\Users\Asus\AppData\Local\Temp\vmware-Asus\VMwareDnD\8804c145\Screenshot from 2017-12-12 11^%52^%49.png**

Metotların içine istenilen sayıda parametre alabiliriz. Bunun için metodun içinde ki parametrenin başına \* işaretini ekleriz.

**Örnek-Uygulama17**



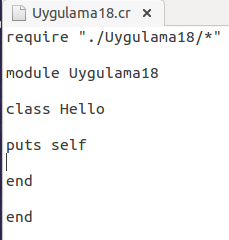
Metodu çağırmak için ise yazdığımız metodun dışında yani ana bölümde metodun adını varsa yollanacak parametreleri girerek metodu çağırmış ve çalıştırmış oluruz.

**5.2.Ana(Main) Metot**

Crystal programlama dilinde bir ana metot yani main metodu yoktur. İstenilen metot oluşturulur ve bu metot end ile bitirilir. Bitirildikten sonra dışarıda yazılan her şey ana metot içinde yazılmış gibidir ayrı bir ana metot tanımlamaya gerek yoktur.

Ayrıca main kodlarında self paremetresi ile ait olduğu sınıf ismini yazdırabiliriz. Yani bu işlem uzun satırlı programlarda yazdığımız programın hangi sınıfa ait olduğunu bize göstererek kolaylık sağlar.

**Örnek-Uygulama18**

****

**Ekran Çıktısı:**

**C:\Users\Asus\AppData\Local\Temp\vmware-Asus\VMwareDnD\41cb950e\Screenshot from 2017-12-16 05^%36^%36.png**

---------------------------------------------------

**6.BÖLÜM**

---------------------------------------------------

**SINIF(CLASS) YAPISI**

---------------------------------------------------

Crystal programlama dili de nesneye yönelik bir programlama dilidir. Bu yüzünden içerisinde sınıf ve nesne kavramlarını da barındırır.

**6.1.Sınıf Tanımlama**

Sınıflar içerisinde metotlar ve kod satırları barındırır. Her sınıf kendi içinde alakalı işlerin görüldüğü alandır.

**Kullanımı:**

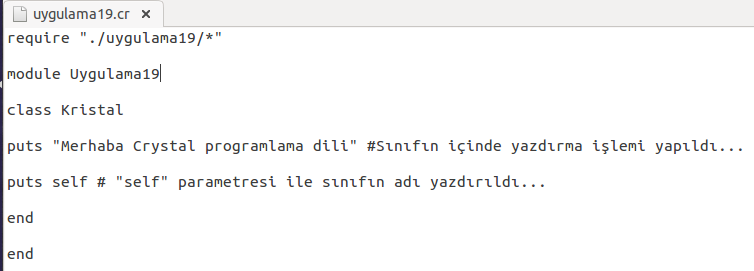
***class sinif\_adi***

***\*\*\*Programlar\*\*\****

***end***

Crystal programlama dilinde her türlü sınıf ve metot isimleri büyük harfle başlamak zorundadır.

**Örnek-Uygulama19**



**Ekran Çıktısı:**

C:\Users\Asus\AppData\Local\Temp\vmware-Asus\VMwareDnD\8dbe8ad0\Screenshot from 2017-12-16 05^%55^%59.png

**6.2.Sınıflardan Nesne Tanımlama**

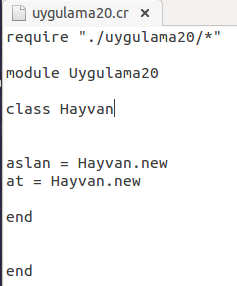
Her sınıfın kendine ait nesneleri vardır. Bu nesneler ait olması istenen sınıftan türetilir ve o sınıf içinden veya başka sınıftan verilen değerler ile kullanılır.

**Kullanımı:**

***Nesne\_adi = sinif\_adi.new***

Nesneleri program içinde her farklı yerlerde çağırarak kullanabiliriz. Nesneler sınıflara ait veri tipleri gibi de düşünülebilir.

**Örnek-Uygulama20**

****

**6.3.Constructor Metodlar**

Comstructor metotlar metot içindeki değişkenlere başlangıç değeri atamak için kullanılır. Bu metod içinde değişkenlere verilen değerler sınıf içerisinde her yerde o değişkenin başlangıç değerini göstermektedir. Sınıf içerisinde herhangi bir yerde bu değişken yazdırıldığında bu başlangıç değeri yazdırılmış olacaktır.

Crystal’de constructor metotların adi her zaman initialize olmalıdır. Daha sonra parantez içerisinde başlangıç değeri atanacak değişkenlerin veri tipleri aktarılabilir.

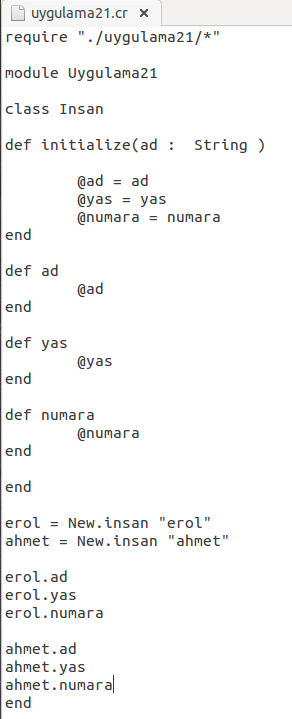
**Kullanımı:**

***def initialize (degisken\_adi : veri\_tipi)***

***\*\*\*Değişkenler ve başlangıç değerleri\*\*\****

***end***

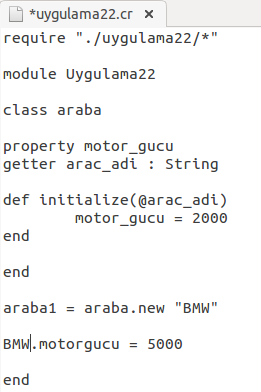
**Örnek-Uygulama21**



**6.4.Getters ve Setters Metodları**

Getter ve Setter metodları globalde erişilemez olarak tanımlanmış bir değişkene erişmek, bu değişkenin değerini değiştirmek, başlangıç değeri atamak gibi işlemlerde kullanılırlar. Nesneye yönelik programlama dillerinde sıkça kullanılan bir yöntemdir.

**Örnek-Uygulama22**



---------------------------------------------------

**7.BÖLÜM**

---------------------------------------------------

**TÜR KESMESİ**

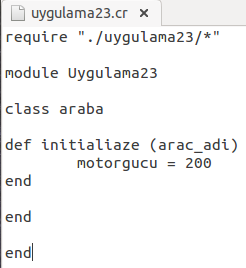
---------------------------------------------------

Crystal programlama dili bizlere mümkün oldukça az tür açıklama imkanı sunmaktadır. Fakat bazı durumlarda türleri tam anlamıyla açıklamamız gerekmektedir. Bunun için tür kesme yöntemi kullanılır.

Eğer bir sınıfta türü belirlenmemiş bir değişken varsa bu derleyicide sorunlar yaratacaktır. Crystal özünde bu değişkenin türünü anlayacak fakat derleyici bu konuda sorunlar yaşayacak istenilen gibi bir çıktı veremeyecektir.

2 tane tür kesme yöntemi vardır. Öncelikle bir sınıf oluşturalım ve bu sınıftaki constructor metodu yazalım. Metotta türü belirlenmemiş bir değişken kullanalım. Daha sonra tür kesme yöntemiyle bu değişkenin durumunu nasıl değiştirebileceğimizi görelim.

**Örnek-Uygulama23**

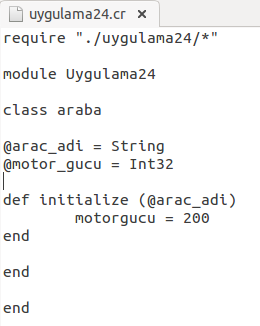
****

Bu örnekte arac\_adi değişkeninin türü belli değildir. Tür kesme yöntemleriyle nasıl derleyicinin hata vermesi engellenir bunları sırasıyla inceleyelim.

**7.1. Tür Açıklama Yöntemi**

En basit yöntem budur. Fakat programcıyı yorar ve kod kalabalığını arttırır.

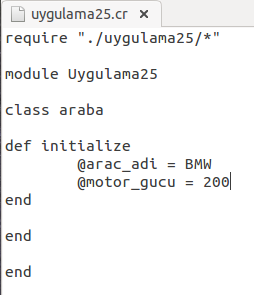
**Örnek-Uygulama24**

****

**7.2. Ebedi Değer Atama Yöntemi**

Bu yöntem ile değişkene ebedi bir değer atayarak bu değişkenin türünün en başından belli olması sağlanabilir.

**Örnek-Uygulama25**

****

---------------------------------------------------

**8.BÖLÜM**

---------------------------------------------------

**MODÜLLER**

---------------------------------------------------

Modüller Crystal programlama dilinde bir sınıfı ve metodları içinde bulunduran alandır. Genel amacı sınıfları metotları ve programları içinde bulunduran bir ad vermek tanımlamak içindir. Modül kullanımının 2 türü vardır;

1. Metot, değişken ve sınıfları barındıran bir ad gibi
2. Karışabilen kısmi bir tür gibi

Modüller Java programlama dilinde de vardır fakat Java programlama dilinde ki pacgake ismi Crystal programlama dilinde modül olarak işlenmiştir.

**8.1. Bir Ad Gibi Modül Kullanımı**

Modüller bir ad gibi kullanıldığında sadece içinde bulundurdukları sınıf metot ve değişkenleri kapsamak, onlara kolayca ulaşmak ve anımsatmak işlerini görmektedirler.

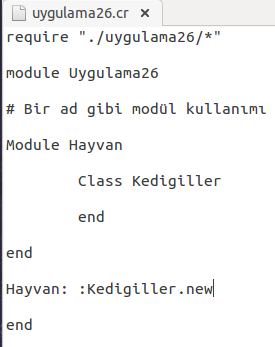
**Kullanımı:**

***Module modul\_adi***

***\*\*\*Sınıflar,değişkenler,metotlar\*\*\****

***end***

**Örnek-Uygulama26**

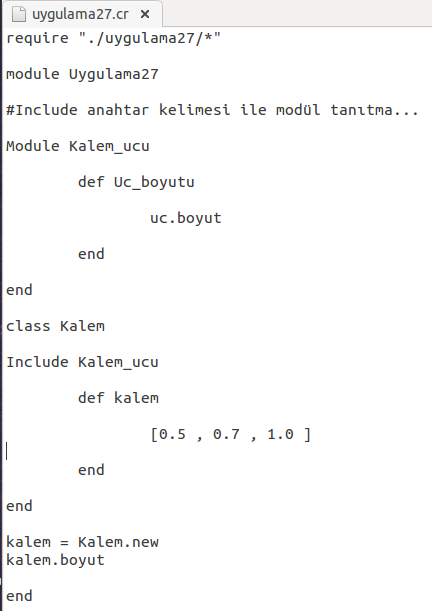
****

Modülleri bir ad gibi kullanma işlemi genelde programlama dillerine kütüphane yazarlarının kullandığı bir alandır.

Modülü oluşturduktan sonra modülü farklı sınıfların veya metotların içerisinde kullanmak için bu modülü o alan içerisine tanıtmamız gerekecektir. Tanıtma işlemi Crystal programlama dilinde 2 anahtar kelime ile gerçekleşir. Bunlardan birisi “include” diğeri “extend” dir.

**8.1.1.Include ile Modül Tanıtma**

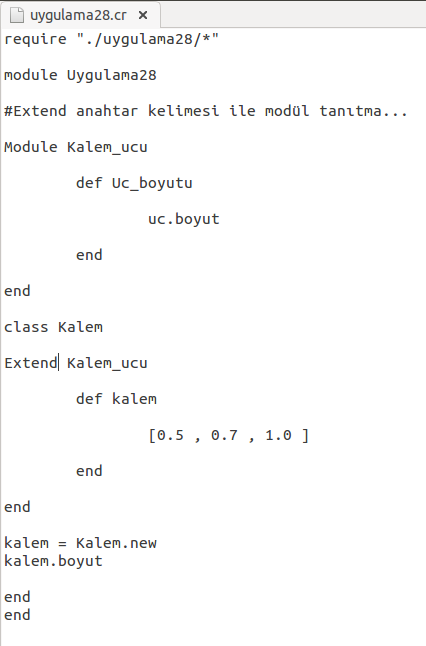
**Örnek-Uygulama27**



“Include” ile modülü farklı bir sınıfta tanıtarak modül özelliklerini o sınıfta kullandık. Bu da modüllerin birer paket gibi özellik gösterdiklerini göstermektedir.

**8.1.2.Extend ile Modül Tanıtma**

**Örnek-Uygulama28**



Örnek-Uygulama27’ de ki modülü include yöntemi kullanarak tanıtma yapılmıştır. Bu örnekte ise aynı uygulamada bulunan modülü extend yöntemi kullanarak tanıtma yapılmıştır.

**8.2.Karışabilen Kısmi Bir Tür Gibi Modül Kullanımı**

Crystal programlama dilinde bu tarz modül kullanımına çokça rastlamamaktayız. Modüller genellikle kütüphane yazmakta ve bir işlemi sıkça tekrar etmek için kullanıldıkları için bu yöntem pek kullanılmamaktadır.

Bu yöntemin işlevi ise modül bir sanki bir değişken bir türmüş gibi kullanılır. Modüller diğer bazı programlama dillerinde ki paket yapısı gibi kullanılır. Modül içerisinde belirli özellikler tanımlanır daha sonra bir değişkenin başında bu özellikler o değişkene veya sabite tanıtılır. Böylece modül kısmi bir tür gibi kullanılma özelliği göstermiş olur.

---------------------------------------------------

**9.BÖLÜM**

---------------------------------------------------

**SABİTLER**

---------------------------------------------------

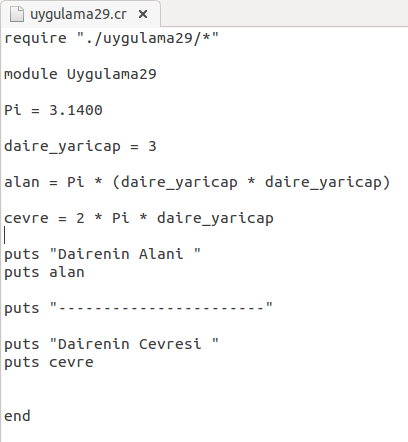
Sabitler bir programlama dilinde değeri hiçbir zaman değişmeyecek değişkenlerdir. Sabitler her programlama dilinin kendi özelliğine göre tanımlanır, tanımlandığı isim ve en başta belirlenmiş sabit değeri ile programın her noktasında o değerle çağırılıp işlem gerçekleştirebilir.

Crystal programlama dilinde sabitler tanımlanırken kesinlikle büyük harf ile başlanmalıdır. Sabit değişkeninin ilk harfi büyük olmalıdır.

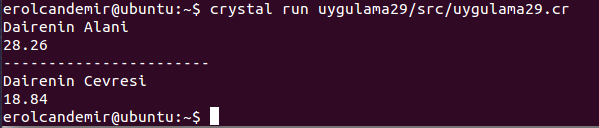
**Tanımlama:**

***Sabit\_Adi = Sabit\_degeri***

**Örnek-Uygulama29:**



**Ekran Çıktısı:**

****

---------------------------------------------------

**10.BÖLÜM**

---------------------------------------------------

**ENUM YAPISI**

---------------------------------------------------

Enum yapısı programlama dillerinde sabitler kümesi tanımlama işine yaramaktadır. Enum yapısında bulunan bütün değerlerin bir tam sayı değeri vardır. Ayrıca tanımlama içerisin de kendimiz de belirli değerler ve işlemler verebiliriz.

**Kullanımı:**

***Enum enum\_adi***

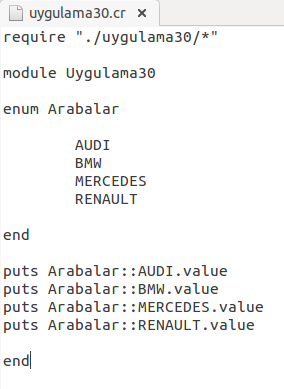
***\*\*\*Tanımlanacak değerler\*\*\****

***end***

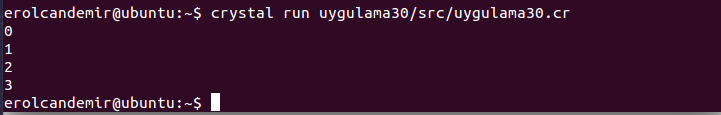
**10.1.Enum Tam Sayı Değeri Çağırma**

Eğer enum ile tanımladığımız değerin tam sayı değerini çağırmak için value anahtar kelimesi kullanılır.

**Örnek-Uygulama30:**

****

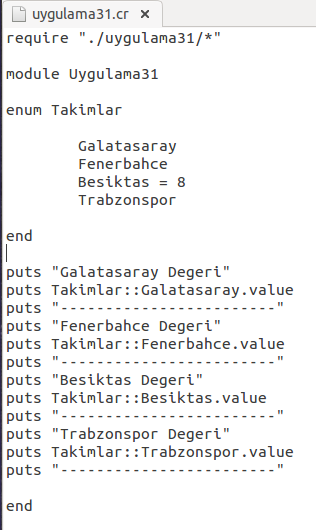
**Ekran Çıktısı:**

****

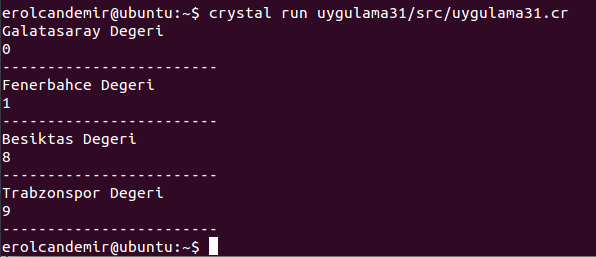
**10.2.Enum Tam Sayı Değeri Belirleme**

Enum içerisinde yazılan değerlerde ilk değerin değeri 0’dır. Sonraki değerler ise birer birer artarak devam eder. Eğer herhangi bir değere başlangıç değeri atarsak o değerden sonrakiler de verilen değerden birer artarak devam eder.

**Örnek-Uygulama31:**

****

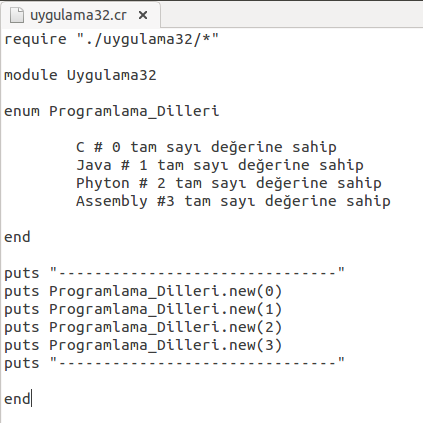
**Ekran Çıktısı:**



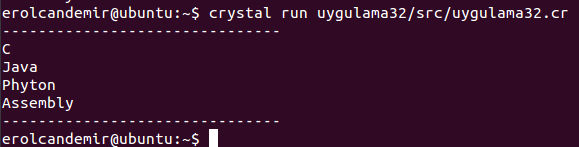
**10.3.Tam Sayı Değeri İle Enum Çağırma**

Enum içinde yazılmış her değerin bir tamsayı değeri vardır. Enum değerlerini sırasıyla çağırmak ve kullanmak için enum içerisindeki değer tam sayı sırasına göre o tam sayı değeri yazılarak çağırılabilir ve kullanılabilir.

**Örnek-Uygulama32:**



**Ekran Çıktısı:**



---------------------------------------------------

**11.BÖLÜM**

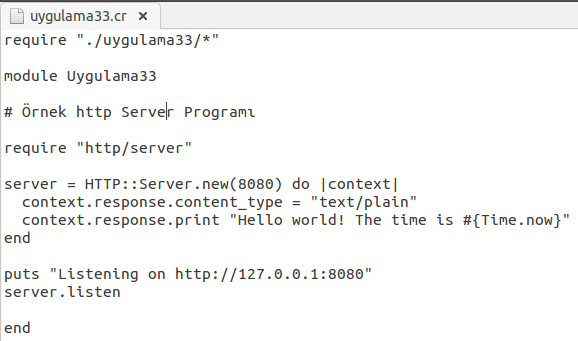
---------------------------------------------------

**ÖRNEK http SUNUCU PROGRAMI**

---------------------------------------------------

Crystal programlama dilinin en çok kullanıldığı ve en verimli olduğu alan web sunucusu programlama alanıdır.

**Örnek-Uygulama33:**

****

Server programının adım adım incelenmesi;

**require “http/server**” 🡪 Server’ın yazıldığı dosyanın adını temsil eder.

**Server = http::Server.new(8080) do |context|** 🡪 Server’ı tanımlaman ve yerel değişkenleri atama işi yapılır.

Programın geriye kalan kısmında ise Server’ın mesaj göndermesi ve yapması istenen işlemlerin tanımlandığı ve uygulandığı nesneler bulunmaktadır.

---------------------------------------------------

**12.BÖLÜM**

---------------------------------------------------

**ÖZET**

---------------------------------------------------

**12.1.Giriş**

Crystal programlama dili halen gelişmekte olan nesneye yönelik bir programlama dilidir. 2014 yılında piyasaya sürülmüştür. Açık kaynak kodlu bir programlama dilidir. İlk sürümü “Crystal 0.18.0” ve şuan en gelişmiş son sürümü “Crystal 0.23.1”dir.

Aynı işi yapabildiği diğer programlama dillerine göre hızıyla öne çıkar ve söz dizimi ve benzeri özelliklerinde diğer programlama dillerinden alıntılar bulundurur.

**12.2.Değişkenler**

Crystal kullanmanın birçok yolu vardır. Windows üzerinde kullanmak için VMWare programı ile sanal ubuntu işletim sistemi kurarak kullanılabilir. Crystal paketleri vesayre kurulduktan sonra terminal üzerinden işlemler yapılır. Ubuntu terminalinde;

Proje oluşturma 🡪 Crystal init app proje\_adi

Proje çalıştırma 🡪 Crystal run proje\_yolu/proje\_adi.cr

**12.3.Karar Yapıları**

Karar yapıları bir şart veya koşula göre işlem yapan birimlerdir. İf-else karar yapısı neredeyse bütün programlama dillerinde bulunmaktadırlar. Şartlar sağlanırsa ve şartlar sağlanmazsa yapılacak işlemler ayrı ayrı yazılır.

Unless karar yapısı if-else ile ters mantığa sahiptir. Şart yanlışsa unless yapısı çalışır ve işlem yapılır.

**12.4.Döngüler**

While döngüsü döngüde bir şart sağlanana kadar o işlemi yapar. Bloğun içinde yapılacak işlemlerin dışında arttırma, azaltma işlemlerini de belirtmek zorundayız. While ile şarta true değer göndererek basitçe bir sonsuz döngü oluşturulmuş olur.

Until döngüsü de while ile aynı işlevi görür, aynı işi yapar. Fakat tek farkı şart false olduğunda döngü biter. While da ise şart true olduğunda biter.

Upto döngüsü bir sayıdan o sayıya kadar bir döngü oluşturur. Sayılar artarak devam eder. Downto döngüsü de upto ile aynı işi yapar fakat sayılar azalarak devam eder.

Each döngüsü de upto downto döngüleri gibi bir sayıdan diğer sayıya kadar arttırma azaltma işlemi yaparak döngüyü oluşturur.

**12.5.Metotlar**

Metotlar def anahtar kelimesi ile tanımlanır. Daha sonrası diğer gelişmiş programlama dilleri ile aynı olup end anahtar kelimesi ile metot bitirilir.

**12.6.Sınıf(Class) Yapısı**

Sınıflar class sinif\_adi şeklinde tanımlanır. İçerisinde metotlar, değişkenler ve benzeri bulundurur.

Sınıf adı ile o sınıfa ait nesneler tanımlanabilir. Nesne\_adi = sinif\_adi.new şeklinde o sınıfa ait bir nesne oluşturulmuş olur.

Constructor metotlar değişkenlere başlangıç değeri vermek için kullanılır. Crystal programlama dilince Constructor metotlar initailize anahtar kelimesi ile tanımlanır.

Getters ve Setters metotları globalde tanımlanmış ve erişilemez olan bir değişkene erişmek için kullanılır.

**12.7.Tür Kesmesi**

Crystal programlama dilinde mümkün oldukça az tür açıklaması aktiftir. Fakat bazı durumlarda türleri açıklamamız ve ona göre işlem yaptırmamız gerekmektedir. Türü normal açıklama ve ebedi değer atama şeklinde iki adet tür açıklama yöntemi vardır.

**12.8.Modüller**

Modüller bazı nesneye yönelik programlama dillerindeki paket kavramı gibi işlev görmektedir. İçerisinde sınıf, metot ve değişkenleri bulundurur. Modülleri bir ad gibi tanımlayarak kullanabiliriz. Bu tanımladığımız ve işlemlerini yaptığımız modülü programın başka bir kısmına include veya extend anahtar kelimeleri ile istenilen noktada tanıtabiliriz. Ayrıca modülleri kısmi bir tür gibi de kullanabiliriz.

**12.9.Sabitler**

Sabitler program program boyunca değeri hiçbir zaman değişmeyecek değişkenlerdir. Sabit\_adi = degeri şeklinde tanımlanır. Sabit adının ilk harfi mutlaka büyük olmalıdır.

**12.10.Enum Yapısı**

Enum yapısı sabit kümesi oluşturmakta kullanılır denilebilir. Enumların her birinin sıfırdan başlayarak artarak devam eden bir tam sayı değeri vardır.

Enumun tam sayı değeri el ile de girilebilir. Belirlediğimiz değerden sonraki enumlar verilen değerden artarak devam eden değerler alır.

Programın herhangi bir yerinde tam sayı değerini girerek o enum değişkenini çağırıp yazdırabiliriz.

**KAYNAKÇA**

1. [**https://hackernoon.com/an-introduction-to-the-crystal-programming-language-b9e0222b5b5e**](https://hackernoon.com/an-introduction-to-the-crystal-programming-language-b9e0222b5b5e)
2. [**http://www.emrealadag.com/crystal-programlama-dili.html**](http://www.emrealadag.com/crystal-programlama-dili.html)
3. [**http://www.gpedia.com/en/gpedia/Crystal\_(programming\_language)**](http://www.gpedia.com/en/gpedia/Crystal_(programming_language))
4. [**http://slides.com/sdogruyol/crystal-ve-kemal#/9**](http://slides.com/sdogruyol/crystal-ve-kemal#/9)
5. [**http://www.nenedirvikipedi.com/sozluk/algoritma-nedir-algoritma-ne-demektir-anlami-884.html**](http://www.nenedirvikipedi.com/sozluk/algoritma-nedir-algoritma-ne-demektir-anlami-884.html)
6. [**https://www.youtube.com/playlist?list=PL5VP\_EtUfhSLXFlbR-IFnAaJKN3LFhYnR**](https://www.youtube.com/playlist?list=PL5VP_EtUfhSLXFlbR-IFnAaJKN3LFhYnR)
7. [**http://www.wikiwand.com/en/Crystal\_(programming\_language)**](http://www.wikiwand.com/en/Crystal_(programming_language))
8. [**http://docs.w3cub.com/crystal/docs/**](http://docs.w3cub.com/crystal/docs/)
9. [**http://motion-express.com/blog/why-use-crystal-lang**](http://motion-express.com/blog/why-use-crystal-lang)
10. [**http://nithinbekal.com/notes/crystal/**](http://nithinbekal.com/notes/crystal/)
11. [**http://en.bmstu.wiki/Crystal\_(programming\_language)**](http://en.bmstu.wiki/Crystal_(programming_language))