Temel C/C++ kuralları

Hazırlayan: Selçuk TERZİOĞLU

C++ Nedir?

- C++ dili birçok platformda kullanılan geniş ve çok güçlü bir programlama dilidir.
- Kullanımı dikkat isteyen bir dildir. Çünkü C++ imla kurallarına sıkı sıkıya bağlı bir dildir. Eğer kod yazarken imla hatası yaparsanız derlenme işlemini keser ve hata verir. Bundan gözünüz korkmasın kodlama tecrübeniz arttıkça daha kolay hale gelecektir.
- Bizim kullanacağımız C++ ise Arduino için özelleştirilmiş versiyonudur.
- Şimdi sırayla imla kurallarını inceleyelim.

(;) Noktalı Virgül

- Noktalı virgül C++ dilinde komut satırının bittiğini gösterir.
- Bazı istisnalar hariç hemen hemen her satırın sonuna konulur. (;) konulmayan yerleri yeri geldikçe açıklıcam.
- (;) koymayı unutursanız derleme işlemi kesilir ve bir sonraki satır işaretlenerek hata mesajı verilir.
- Aşağıda noktalı virgül kullanım örneklerine bakalım

```
pinMode(3, OUTPUT);
digitalWrite(3, HIGH);
delay(500);
```

{ } Küme parantezleri

- Küme parantezleri fonksiyonların veya kod bloklarının başladığı ve bittiği yerleri gösterir.
  - { parantez işareti ile fonksiyon veya kod bloğu başlar.
  - } parantez işareti ile fonksiyon veya kod bloğu biter.
- Küme parantezlerinin açıldığı ve kapatıldığı yerlerde (;) kullanılmaz!
- İç içe kod blokları olabilir. Bu sebeple açılan parantez kadar parantez kapatılmalıdır. Yoksa derleme esnasında hata ile karşılaşırsınız.

{ } Küme parantezleri

setup fonksiyonu bu { parantez ile başlar.

```
void setup( ) {
    //setup kodunu buraya yazınız

}
```

setup fonksiyonu bu } parantez ile biter.

setup kodu { } parantezleri arasına yazılmalıdır

{ } Küme parantezleri

```
if (sayac < 999 ) {
     sayac = sayac + 1;
     digitalWrite( 3, LOW);
}
```

Yukarıda ise { } parantezleri if karşılaştırma bloğunun başını ve sonunu belli eder. Yani if { parantezi ile başlar } parantezi ile biter. Dolayısıyla bu bloklar içine yazılan kod if bloğunun kodudur.

Not: if karar yapısına takılmayın ileride görücez.

{ } Küme parantezleri

```
void setup( )
{
        if (sayac < 999 )
        {
                digitalWrite( 3, LOW);
        }
}
```

- Program içerisinde iç içe bloklar açılması gayet doğal ve sıkça kullanılan bir durumdur. Dikkat edilmesi gereken açılan parantezlerin kapatılmasıdır.
- Kırmızı ile işaretlenen setup fonksiyonunun bloğudur. Yeşil ile işaretlenen ise if bloğudur. Yani if bloğu loop bloğunun içindedir.

( ) Parantez

- ( ) parantezlerinin 3 farklı kullanımı vardır. Kullanım alanlarına geçmeden önce hatırlanması gereken nokta açılan parantez kadar kapatılması zorunludur!!!

1. Fonksiyonlarda parantez kullanımı zorunludur. Hem tanımlarken hem de kullanırken parantez kullanımı zorunludur.
2. Kontrol ve döngü yapılarında şartların girilmesi için parantez kullanımı zorunludur.
3. Matematik işlemlerinde işlem önceliğini değiştirmek için kullanılır

( ) Parantez – Fonksiyonlarda

- C++ dili fonksiyon tabanlı bir dildir. Fonksiyonlarda parantezler hem tanımlarken, hem de kullanırken zorunludur. Eğer fonksiyona parametre aktaracaksak yine parantezlerin yardımı ile bu işi yapıyoruz.
- Aşağıdaki örneklerde bu durumu görebiliriz.

```
void setup( ) {
    pinMode(3, OUTPUT);
    digitalWrite(3, HIGH);
    delay(500);
}
```

( ) Parantez – Kontrol Yapıları

- Döngü ve Kontrol yapılarında da ( ) parantezleri şartların girilmesi için zorunludur.

```
if (sayac < 999) {
    sayac = sayac + 1;
    digitalWrite( 3, LOW);
}
```

- sayac değerinin 999'dan küçük olma şartı parantez içine yazılmıştır.
- İleriki konularda göreceğimiz diğer kontrol ve döngü yapılarında da parantezler kullanılacaktır.

( ) Parantez – Matematiksel İşlem Önceliği

- Parantezin bir başka kullanım alanı da matematikse işlem önceliğidir. Matematiksel işlemlerde parantezlerin önceliği vardır.
- Parantezleri kullanarak işlem sonucunu istediğimiz şekilde hesaplanmasını sağlayabiliriz.

Örn: `sonuc = ( 5 + 2 ) * 3 – 1`

- Yukarıda ki örnekte önce parantez işlemi yapılacaktır.

(Not: İşlem önceliği parantez, üs, çarpma veya bölme, toplama veya çıkarma …)

// Açıklama Satırı

- Açıklama satırları programcılar için büyük bir kolaylıktır.
- Açıklama satırları derleyici tarafından okunmayan satırlardır.
- Burada program veya programın ilgili bölümü hakkında açıklamalar yapılır.
- **//** iki adet slaş (bölü) işareti ile açıklama satırı yapılır.
- Açıklama satırları komutun sonuna(**;** den sonra) veya boş satırlara yazılır.
- Açıklama satırları kesinlikle komutun başına yazılmaz!
- **//** işareti ile 1 satır açıklama satırı haline getirilebilir.

// Açıklama Satırı

Örnek:

```
pinMode(3, OUTPUT);          //3 nolu pin çıkış
//Bu bir açıklama satırıdır.
digitalWrite(3, HIGH);       //3 nolu pin lojik1
delay(500);                  //500msn bekle
```

/* */ Açıklama Bloğu

- Eğer ki birden fazla satırı açıklama satırı yapacaksanız ve buralarda kod yazılmayacaksa açıklama bloğunu da kullanabilirsiniz.
- **/*** işareti ile açıklama bloğunu başlatabilirsiniz.
- ***/** işareti ile açıklama bloğunu bitirebilirsiniz.
- Eğer açıklama bloğunu açtıysanız kapatmak zorundasınız. Aksi takdire **/*** işaretinden sonraki tüm program açıklama olarak algılanacaktır.
- Bir programda istediğiniz kadar açıklama satırı veya bloğu kullanabilirsiniz.

/* */ Açıklama Bloğu

Örnek:

```
/* Burası bir açıklama bloğudur ve kapatılana
    kadar  da açıklama bloğudur. */
pinMode(3, OUTPUT);          //3 nolu pin çıkış
//Bu bir açıklama satırıdır.
digitalWrite(3, HIGH);       //3 nolu pin lojik1
delay(500);                  //500msn bekle
```

Küçük Harf Büyük Harf Duyarlılığı

- C++ dili küçük harf büyük harf duyarlılığına sahiptir. Yani bir değişkeni, sabiti, fonksiyonu aynen tanımlandığı gibi kullanmak zorundayız!!!

Örnek: delay(500);

Doğru

Delay(500);

Yanlış

Boşluklar

- C++ dilinde boşlukların bir önemi yoktur. Komut kelimesini bölmeden istediğiniz kadar boşluk bırakabilirsiniz.

Örnek: Aşağıdaki yazımların hepsi aynıdır aralarında fark yoktur.

✓
```
pinMode(3, OUTPUT);
```
✓
```
pinMode(3,              OUTPUT);
```
✓
```
pinMode     (3, OUTPUT)      ;
```

Aşağıdaki yazım ise yanlıştır. Çünkü komut kelimesi bölünmüştür.

✗
```
pin  Mode(3, OUTPUT);
```