



C++

KONU ANLATIMI

Creos

int main() & White Space

- `int main()` her c++ programı için gereklidir.
- `int main()` olmadan c++ programı çalışmaz.
- Bunun sonunda `return 0` ile başarılı bir şekilde çalıştığının sinyalini göndeririz.
- Derleyici boşlukları önemsemez.
- Kod yazımı okunabilir olmalıdır.

Merhaba Dünya !!

- **#** bir önışlemci yönergesini temsil eder.
- **<iostream>** dosyanın baş kısmıdır.
- **cout, std** ad alanında tanımlanmıştır.
- "strings (yazı) formatında olanlar tırnak içine alınırlar."
- **std::cout << std::endl;** yeni satır için.

std::cout <<

"Merhaba Dunya";

AD

Operatör

String

**cout, std ad alanında
tanımlanmıştır.**

İşlem Operatörleri

- Ekleme operatörü = +
- Çıkarma operatörü = -
- Çarpma operatörü = *
- Bölme operatörü = /

Const

(Short of constant)

- **const** kelimesi ile niyetimizi herkeze bildiririz:
- "Bu değişkeni kodun herhangi bir yerinde değiştirmeyeceğim!"
- Derleyici = yazdığımız bu kodu korur.

Const & değerler atamak

(Short of constant)

- Atama operatörü =
- Değişkene değer atayabiliriz örneğin:
`a = 4;`
- Bir değişken tanımlanmadan önce atanamaz!
- `const` kelimesi değişkeninizi `constant` olarak tanımlar.

```
1 #include <iostream>
```

Önişlemci Direktifi

```
2  
3 int main()
```

Main fonksiyonu

```
4 {  
5     std::cout << "\n Odaya girmek icin dogru sayiyi bulmalisin.";  
6     std::cout << std::endl;  
7     std::cout << "Doğru sayiyi giriniz ve enterlayiniz.";
```

İfade Kısmı

```
8  
9  
10     const int Sayi1 = 5;  
11     const int Sayi2 = 3;  
12     const int Sayi3 = 2;  
  
13  
14     const int Topla = Sayi1 + Sayi2 + Sayi3;  
15     const int Carp = Sayi1 * Sayi2 * Sayi3;
```

Tanımlama Kısmı

```
16  
17     std::cout << std::endl;  
18     std::cout << Topla << std::endl;  
19     std::cout << Carp << std::endl;
```

İfade (Açıklama) Kısmı

```
20  
21     return 0;
```

Return Kısmı

```
22 }  
23
```


Açıklama Satırı

- Hem kendiniz için hem de başkalarının kodu okuyabilmesini kolaylaştırmak için açıklama satırı ekleyin.

// Bu bir açıklama satırıdır.

Değişken İsimlendirmesi

- Değişkenlerinize iyi isim vermeniz çok önemlidir.
- Bir harf veya alt çizgi ile başlamalıdır.
 - * Sayı ile başlayamaz.
- Kullanılmış bir kelimeyi kullanamazsınız.

Kelimeler

Örnek

auto

bool

break

case

const

int

friend

return

```
std::cout << "Merhaba Dünya!";
```



Character
Output



Yerleştirme (İçeri aktarma)
Operatörü

```
std::cin >> GirilenYazi;
```

Çıkarma(Veri alma)
Operatörü

Çıkan veriyi al ve
GirilenYazi
değişkenine ata

Character
Input

Kullanıcı Girişi Alma

- **cout = Character Output (kullanıcı çıkışı)**
 - * **Insertion Operatör = <<**
- **cin = Character Input (kullanıcı girişi)**
 - * **Extraction Operatör = >>**

Eğer Sorgusu

if (sorgu)

```
{  
std::cout << "Bildin !!";  
}
```

Birleşik İfadeler

(Kod bloğu olarak da bilinir.)

Eğer Sorgusu

İfadeden doğru ya da yanlış olarak dönüş alınır.



```
if (KullaniciGirisi == KodCikisi)
{
    std::cout << "Bildin !!";
}
```



```
if (KullaniciGirisi == KodCikisi && KullanGirisi2 == KodCikisi2)
{
std::cout << "Bildin !!";
}
else
{
std::cout << "Bilemedin";
}
```

C++ if ve else kullanımı

- Eşitlik (denklik) operatörü: **==**
- Ve operatörü: **&&**
- **if(sorgu)**
 - * Koşul sağlanırsa kod bloğu çalışır.
- **else**
 - * Koşul sağlanmazsa kod bloğu çalışır.

```
std::cout << "Dedi ki"Merhaba Dünya";
```

```
std::cout << "Dedi ki \"Merhaba Dünya\"";
```

Çift tırnak kaçış
işareti

```
std::cout << "Merhaba Dünya" << std::endl;
```

```
std::cout << "Merhaba Dünya\n";
```

Yeni sayır
kaçış işareti

While Döngüsü

```
while(sorgu)  
{  
    Döngüyü başlat;  
}
```

**Kod bloğu, koşul karşılanmayana
kadar tekrar tekrar yürütülür.**



std::cin.clear(); // Tüm hataları temizler.

std::cin.ignore(); // Ara belleği atar.

Fonksiyonlardan veri döndürme

- **while** Kod bloğu, koşul karşılanmayana kadar tekrar tekrar yürütülür.
- CTRL + C programdan çıkar.
- **void** tipinde olmayan fonksiyonlar mutlaka return ifadesi içermelidir.
- **bool** = doğru ya da yanlış değer

For Döngüsü

Kod bloğu her çalıştığında i değerini bir artırır.

```
for(int i = 1; i<=10;i++)  
{  
    std::cout << "Merhaba Dunya";  
}
```

Kod bloğu, i değeri 10 değerine eşit olana kadar döngü devam eder.

switch-case Yapısı

```
switch(koşul)
{
    case durum:
        cout << işlem;
        break;

    case durum2:
        cout << işlem;
        break;

    default:
        cout<< Hiçbiri sağlanmıyorsa işlem;
}
```

Kod bloğu, sadece durumun olup olmadığına bakar.

break: bizim istediğimiz durumlarda işlemi gerçekleştirir.

default: Durumların hiçbirisi sağlanmıyorsa ayarlanan işlemi yapar

Karşılaştırma Operatörleri

Eşitlik Operatörü

==

Eşit olmama Operatörü

!=

Büyüklik Operatörü

>

Küçüklük Operatörü

<

Büyük ve eşit olma Operatörü

>=

Küçük ve eşit olma Operatörü

<=

Rastgele Sayılarla Çalışma

- Değişkenlerimizi şu fonksiyon ile başlatacağız:

rand()

* Bu fonksiyon bize rastgele sayı döndürür.

Mod Operatörü

- %
- Bölme yapar fakat kalanı verir.
- $9 / 4 = 2 - 1$
- $9 \% 4 = 1$
- Bunu kendi aralımızda **rand()** ile kullanabiliriz.

Bir Pattern Oluştu !!

- $\langle \text{value} \rangle \% \langle \text{modül} \rangle$
- $\langle \text{value} \rangle$ değerini bir sayı aralığında eşler.
- Aralık 0 ile $\langle \text{modül} \rangle - 1$ arasındadır.

Aralığımızı Ekleme

- `rand() % <modül>`
 - * 0 aralığındaki rastgele sayıyı ve modül değerini (1 çıkarılır) eşler.

Olması Gereken	$\text{rand()} \% 0.G$ (aralık)	$\text{rand()} \% 0.G + 1$ (aralık)
1	0	1-1
2	0-1	1-2
3	0-2	1-3
4	0-3	1-4
5	0-4	1-5

Farklı Aralıklar ile rand()

- rand()'i farklı aralıklarlar başlatmamız gerekebilir.
 - * Bu daha fazla rastgele sonuçlar üretecek.
- Bunun en iyi yöntemi bilgisayar zamanını esas olarak almak.
- `#include <ctime>`
- Bunu kod satırının başına ekleyin.
 - * `srand(time(NULL));`
 - * Zamanı esas olarak alıp size random sayı verecektir.