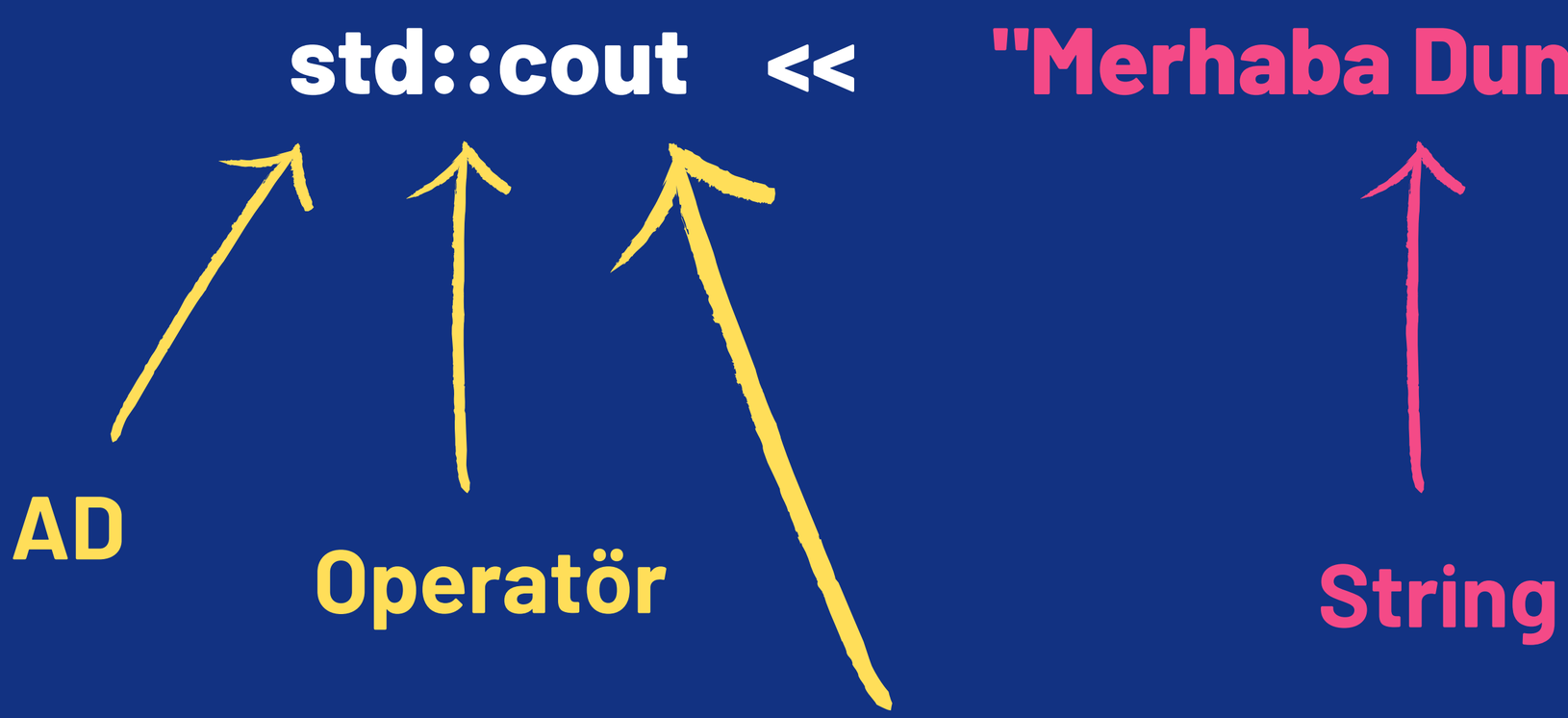
C++
KONU ANLATIMI
Creos

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 önişlemci 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.



cout, std ad alanında tanımlanmıştır.

İşlem Operatörleri

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

Temel Kullanım İçin Veri Tipleri (sadece aklında bulunsun)

- int (integer) = tam sayılar
  * Ör: int32: 32 bit integer, 4 bytes
- float (floating point) = ondalıklı sayılar
  * Ör: 32 bit, 4 bytes
- double (double floating point) = ondalıklı sayılar
- bool = doğru veya yanlış
  * Ör: 1 bit, 1 byte
- string = chars + '\0' = yazılar
  * char her karakter 1 byte
- void (valueless) = Tipi belli olamyan değişken
- wchar_t (wide chracter) = karakter dizisi

Bunların yanında

- signed
- unsigned
- short
- long

Kullanarak boyutunda oynamalar (küçültme-büyütme) yapabiliriz.

Bunların boyutlarına bakmak için

```
std::cout << "Boyutu:  " << sizeof(DataTipi) << endl;
```

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.

```cpp
#include <iostream>

int main()
{
    std::cout << "\n Odaya girmek icin dogru sayiyi bulmalisin.";
    std::cout << std::endl;
    std::cout << "Doğru sayiyi giriniz ve enterlayiniz.";


    const int Sayi1 = 5;
    const int Sayi2 = 3;
    const int Sayi3 = 2;

    const int Topla = Sayi1 + Sayi2 + Sayi3;
    const int Carp = Sayi1 * Sayi2 * Sayi3;

    std::cout << std::endl;
    std::cout << Topla << std::endl;
    std::cout << Carp << std::endl;

    return 0;
}

```

Önişlemci Direktifi

Main fonksiyonu

İfade Kısmı

Tanımlama Kısmı

İfade (Açıklama) Kısmı

Return Kısmı

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.

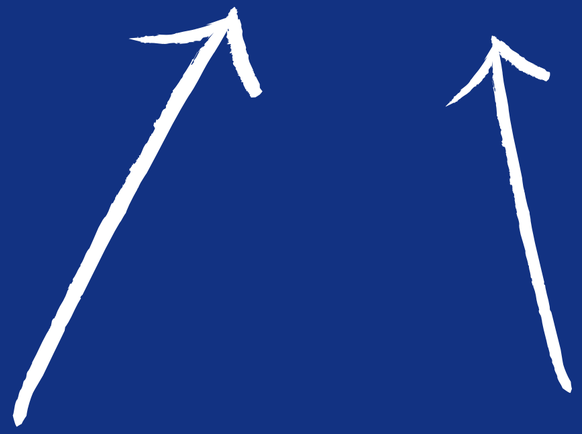
Kelimeler

Örnek

auto bool break case

const int friend return

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



Character Output

Yerleştirme (İçeri aktarma) Operatörü

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

Character Input

Çıkarma(Veri alma) Operatörü

Çıkan veriyi al ve GirilenYazi değişkenine ata

Kullanıcı Girişi Alma

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

typedef Kullanımı

veri türlerine özel isim vermek için kullanırız bu sayede okunması ve kullanılması daha kolay olur.

```cpp
#include <iostream>

typedef int Sayilar;
using namespace std; // adı tanımladığımız için her seferinde std yazmamıza gerek kalmıyor.

int main()
{
    Sayilar sayi1 = 5;
    Sayilar sayi2 = 10;

    cout << "islem Sonucu: " << sayi1 + sayi2;
}
```

enum Kullanımı

değişkenlere özel isim vermek için kullanırız bu sayede okunması ve kullanılması daha kolay olur.

```cpp
#include <iostream>

enum Sayilar {
    sayi1,
    sayi2,
    sayi3,
    sayi4 = 10,
    sayi5 = 20,
};

int main()
{
    std::cout << "Sayi1 = " << sayi1 << std::endl;
    std::cout << "Sayi2 = " << sayi2 << std::endl;
    std::cout << "Sayi3 = " << sayi3 << std::endl;
    std::cout << "Sayi4 = " << sayi4 << std::endl;
    std::cout << "Sayi5 = " << sayi5 << std::endl;
}

```

```
Sayi1 = 0
Sayi2 = 1
Sayi3 = 2
Sayi4 = 10
Sayi5 = 20
```

#define Kullanımı

Ön işlemci komutudur. Yazdığımız kodlar derlenmeden önce **#define** ile yazdıklarımızı inceler.

```
4   #define Dsayi1 10
5   #define Dsayi2 20
6
7   int main()
8   {
9       std::cout << "Define ile sayi carpimi = " << Dsayi1 * Dsayi2 << std::endl;
10      std::cout << "Define ile sayi toplama = " << Dsayi1 + Dsayi2 << std::endl;
11  }
12
```

volatile Kullanımı

Bir türün değişken olduğunu yani kod içerisinde zamanla değerinin değişeceğini bildirir. (const zıttı diyebiliriz.)

```
    volatile int a = 5;
    std::cout << a << std::endl;

    a = 10;
    std::cout << a << std::endl;
}
```

C++ if ve else kullanımı

- Eşitlik (denklik) operatörü: ==
- Ve operatörü: &&
- if(sorgu)

  * Koşul sağlanırsa kod bloğu çalışır.
- else

  * Koşul sağlanmazsa kod bloğu çalışır.

Karşılaştırma Operatörleri

Eşitlik Operatörü	==
Eşit olmama Operatörü	!=
Büyüklük Operatörü	>
Küçüklük Operatörü	<
Büyük ve eşit olma Operatörü	>=
Küçük ve eşit olma Operatörü	<=

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";
}
```

```
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 arttırır.

```
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 hiçbiri sağlanmıyorsa ayarlanan işlemi yapar

class Kullanımı

```
class ClassIsmi
{
    Erişim Tipi:

    Değişkenler;

    Değişkenler için fonksiyon() {}
}
```

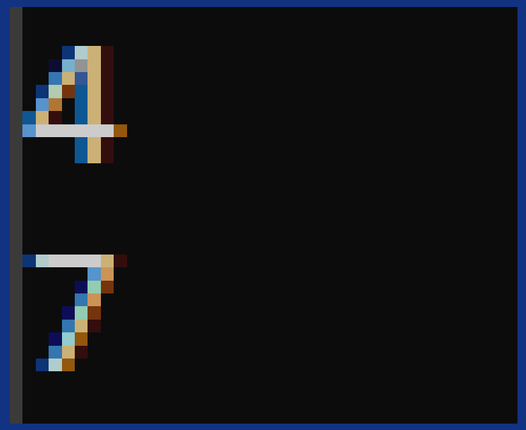
public veya private olabilir.

Kullanılacak değişkenler

class Kullanımı

Hem değişkenleri hem de fonksiyonları daha sonra erişmek için kullanabileceğimiz yapı. Public veya private ile korumaya alabiliriz.

```
4   class Toplama {
5       int toplanacak1, toplanacak2;
6   public:
7       Toplama(int, int);
8       int topla() {
9           return (toplanacak1 + toplanacak2);
10      }
11  };
12
13  Toplama::Toplama(int a, int b) {
14      toplanacak1 = a;
15      toplanacak2 = b;
16  }
17
18  int main()
19  {
20      Toplama sayilar1(2, 2);
21      Toplama sayilar2(5, 2);
22
23      std::cout << sayilar1.topla() << std::endl;
24      std::cout << sayilar2.topla() << std::endl;
25  }
26
```



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 !!

- <value> % <modül>
- <value> değerini bir sayı aralığında eşler.
- Aralık 0 ile <modül> -1 arasındadır.

Aralığımızı Ekleme

- rand( ) % <modül>

  * 0 aralığındaki rastgele sayıyı ve modül değerini (1 çıkarılır) eşler.

Olması Gereken	rand() % O.G (aralık)	rand() % O.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.