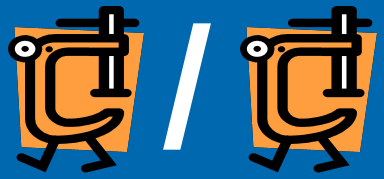
C / C++ Eğitimi

Eğitmen: **Bülent Çobanoğlu** (Bülend Hoca)
e-mail: cobanolgubulent@gmail.com



/ ++ Eğitimi-1

–: Yazılım, Derleyici - Yorumlayıcı Kavramları

–: C / C++ Dillerin Tarihi

–: C/C++ Temel Veri Türleri

--: Otomatik Tür Dönüşümü

--: Veri Tiplerinin Değer Aralığı

--: Derleyici

--: Derleyici -Yorumlayıcı Farkı

--: Derleme Zamanı Hataları

--: C-C++ Derleyicileri

--: C/C++ Program Yapısı

--: Sayı Sistemleri ve Dönüşümleri

–: Çıktı komutlarının (printf() , cout,...) kullanımı

--: Niçin C/C++?

--: C-C++ Dillerinin Farkı

--: Programlama Nasıl Öğrenilir?

–: Kaynak Kodun Derlenme Aşamaları

--: UNIX/Linux Sistemlerinde C Programlarının Komut Satırından Derlenerek Çalıştırılması

--: Derlenen Dosyanın Komut Satırından Çalıştırılması

--: Yorum Satırları

--: sizeof Operatörü

--: Bilinçli Tip Dönüşümü

–: Tip Dönüşümlerinde Veri Kaybı

–: Çıktıların Biçimlendirilmesi

Software(Yazılım) Nedir?

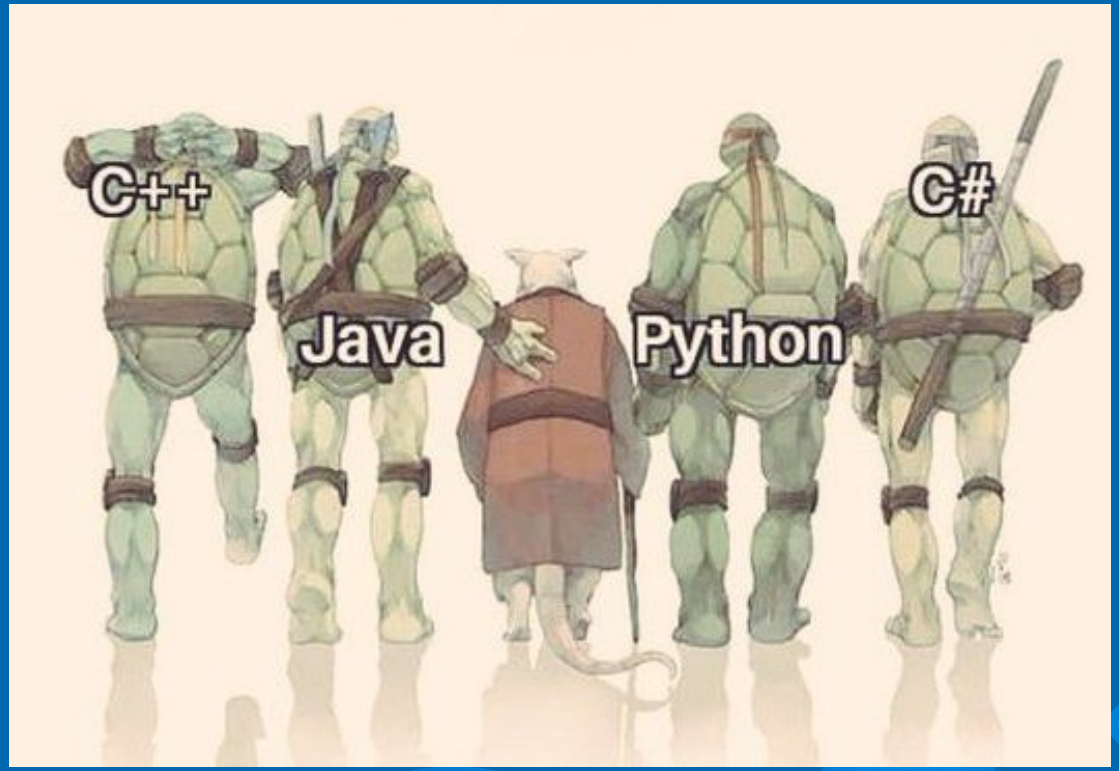
Bütün yazılımlar bir programlama dili ile yazıldığına göre programlama dilleri nasıl yazılmıştır?

Niçin C/C++?

- Büyük projelerde tercih edilen bir dil
- Sistem dili,
- Embedded sistem,
- Simülasyon yazılımları,
- Oyun programlama,

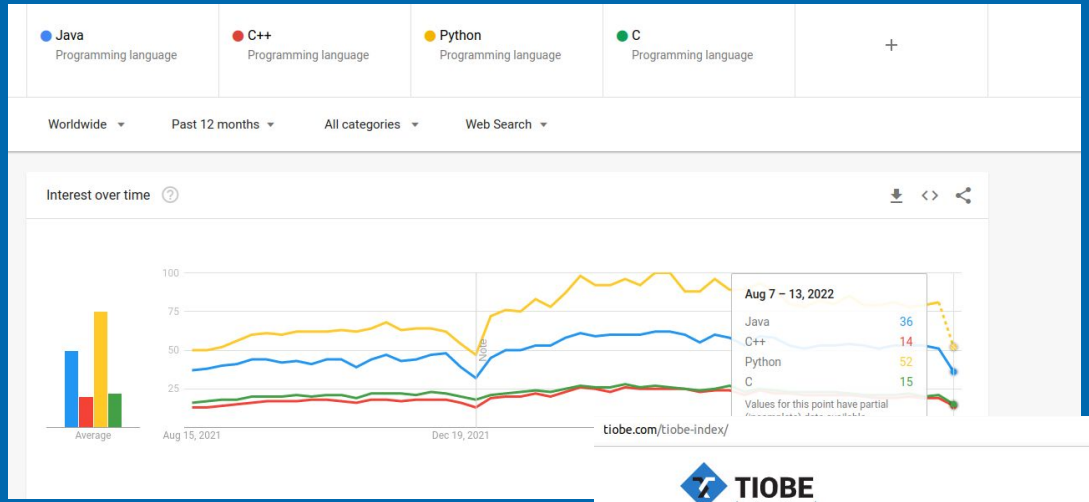
……

Niçin C / C++ ?



Niçin C/C++?

trends.google.com/



tiobe.com/tiobe-index/

TIOBE

About us | Join TIOBE | News | Coding Standards | TIOBE Index | Contact

Products | Quality Models | Markets | Request a demo

Programming Language	2022	2017	2012	2007	2002	1997	1992	1987
Python	1	5	8	7	12	28	-	-
C	2	2	2	2	2	1	1	1
Java	3	1	1	1	1	16	-	-
C++	4	3	3	3	3	2	2	6
C#	5	4	4	8	15	-	-	-
Visual Basic	6	14	-	-	-	-	-	-
JavaScript	7	8	10	9	9	23	-	-
Assembly language	8	10	-	-	-	-	-	-
SQL	9	-	-	-	7	-	-	-
PHP	10	7	6	5	6	-	-	-
Prolog	24	33	33	27	17	21	12	3
Lisp	33	31	13	16	13	10	5	2
Pascal	269	112	15	20	99	9	3	5
(Visual) Basic	-	-	7	4	4	3	6	4

Programlama Nasıl Öğrenilir?

> “Programming is learned by writing programs.”
>
> —Brian Kernighan

1- Problemi (öncelikle) kendiniz çözmeye çalışın,

2- Kodu kağıda yazın,

3- Kağıda yazdığınız kodu test edin,

4- Kağıttaki kodunuzu olduğu gibi bilgisayara geçirip, kodlayınız.

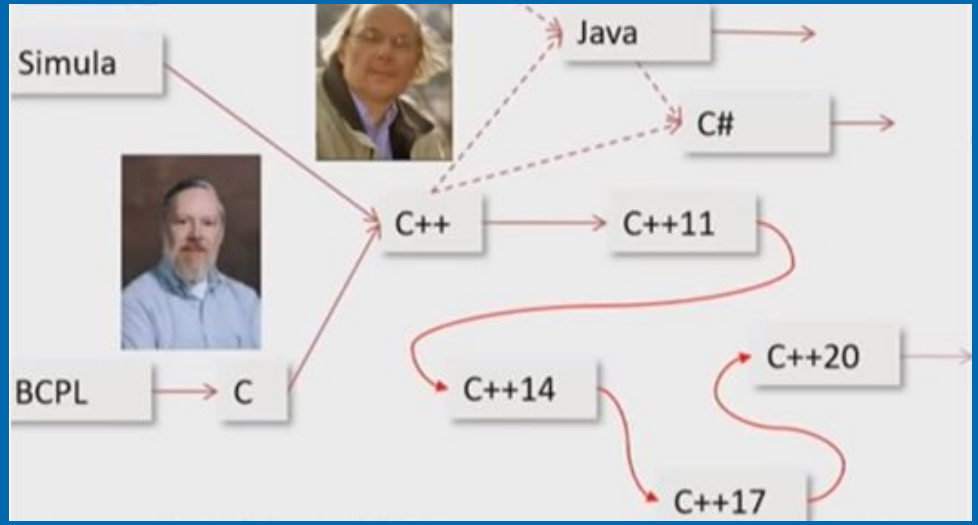
--- Muhtemelen bir sürü hata ile karşılacaksınız, hataların bir listesini yapınız, giderdiğinizde nasıl giderdiğinizi not ediniz. ----

C /C++ Dillerinin Gelişimi

1970	**B (BCPL) dili** geliştirildi (*Ken Thompson* tarafından)	1979	**C with Classes** ismi ile C++ geliştirilmeye başlandı (***Bjarne Stroustrup*** tarafından)
1972	**C dili** geliştirildi (***Dennis Ritchie*** tarafından)	1983	**C++** ismini aldı.
1989	**ANSI C** geliştirildi. (ANSI Komitesi tarafından) *ANSI: American National Standards Institute*	1998	**C++98** sürümü geliştirildi *(ISO/IEC tarafından)*
1999	**C99** sürümü geliştirildi (ISO/IEC tarafından) *ISO:International Organization for Standardization* *IEC: International Electrotechnical Commission*	2003	**C++03** sürümü geliştirildi *(ISO/IEC tarafından)*
2011	**C11** sürümü geliştirildi (ISO/IEC tarafından)	2011	**C++11** sürümü geliştirildi (ISO/IEC tarafından)
2017	C17 ve **C18** sürümleri geliştirildi (ISO/IEC tarafından)	2017	**C++17** sürümü geliştirildi (ISO/IEC tarafından)
2023	***C23 sürümü geliştirilmektedir.***	2020	**C++20** sürümü geliştirildi (ISO/IEC tarafından)

https://en.cppreference.com/w/c/language/history

C++ Dilinin Gelişim Süreci



C: 1970'lerin başında Ken Thompson ve Dennis Ritchie tarafından UNIX İşletim Sistemi için geliştirilmiş bir programlama dilidir. En çok kullanılan sistem (işletim sistemi) geliştirme dilidir.

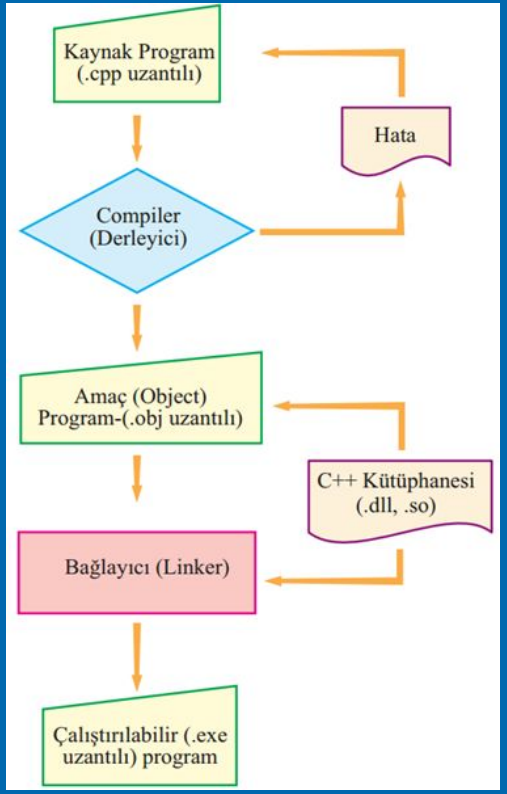
C++ Dili:1980'lerin başında Bjarne Stroustrup tarafından geliştirilen C++ dili; nesne yönelimli bir programlama dilidir.

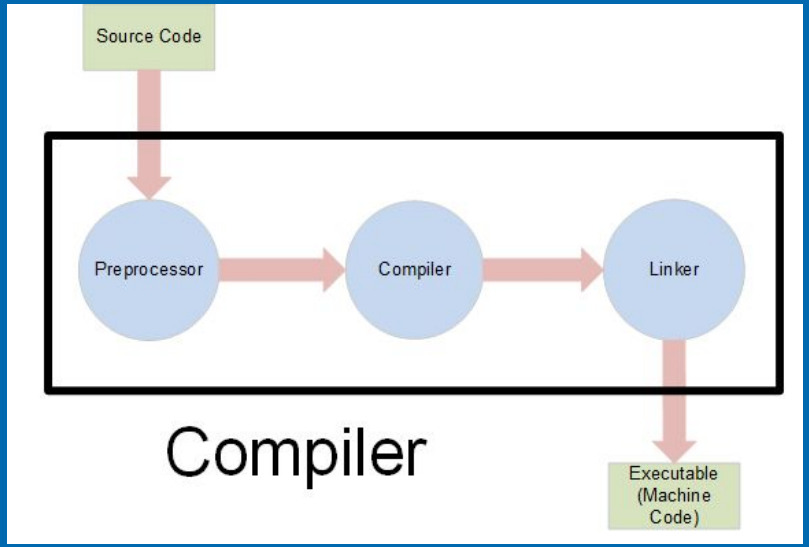
Bjarne Stroustrup's homepage

C ve C++ Dillerinin Farkı

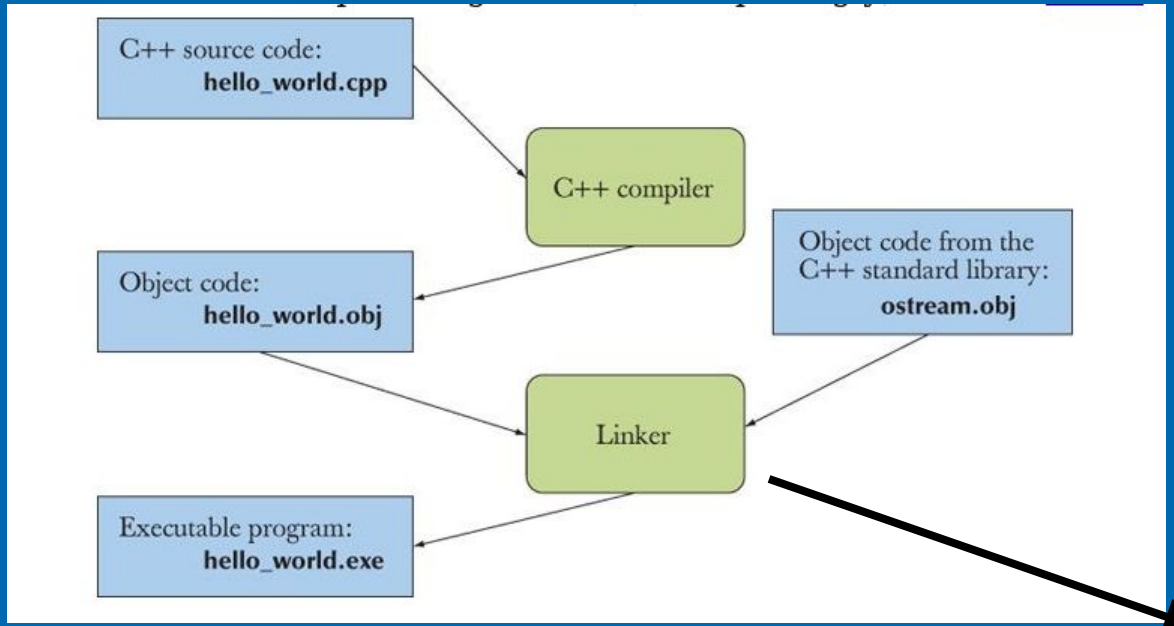
C	C++
Uzantısı **.c**	Uzantısı **.cpp**
19**70**'lerin başında Ken Thompson ve Dennis Ritchie tarafından geliştirilmiştir.	19**80**'lerin başında Bjarne Stroustrup tarafından geliştirilmiştir.
Fonksiyonel bir dil,	Hem fonksiyonel hem de nesne yönelimli bir dildir.
32 adet keyword'e sahip	C Keywords + C++ Keywords (63) *(Her yeni sürümde keywords sayısı artmaktadır..)*
Her C programı aynı zamanda bir C++ programıdır.	Tersi doğru değildir.
Başlık dosyaları <limits.h>	Başlık dosyaları <climits>
…	…

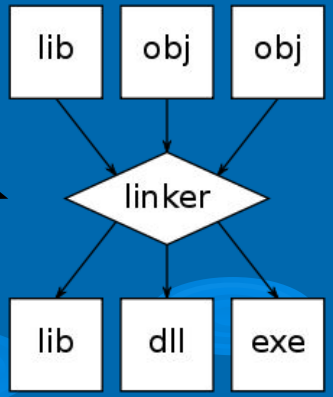
Kaynak kodun Derlenme Süreci



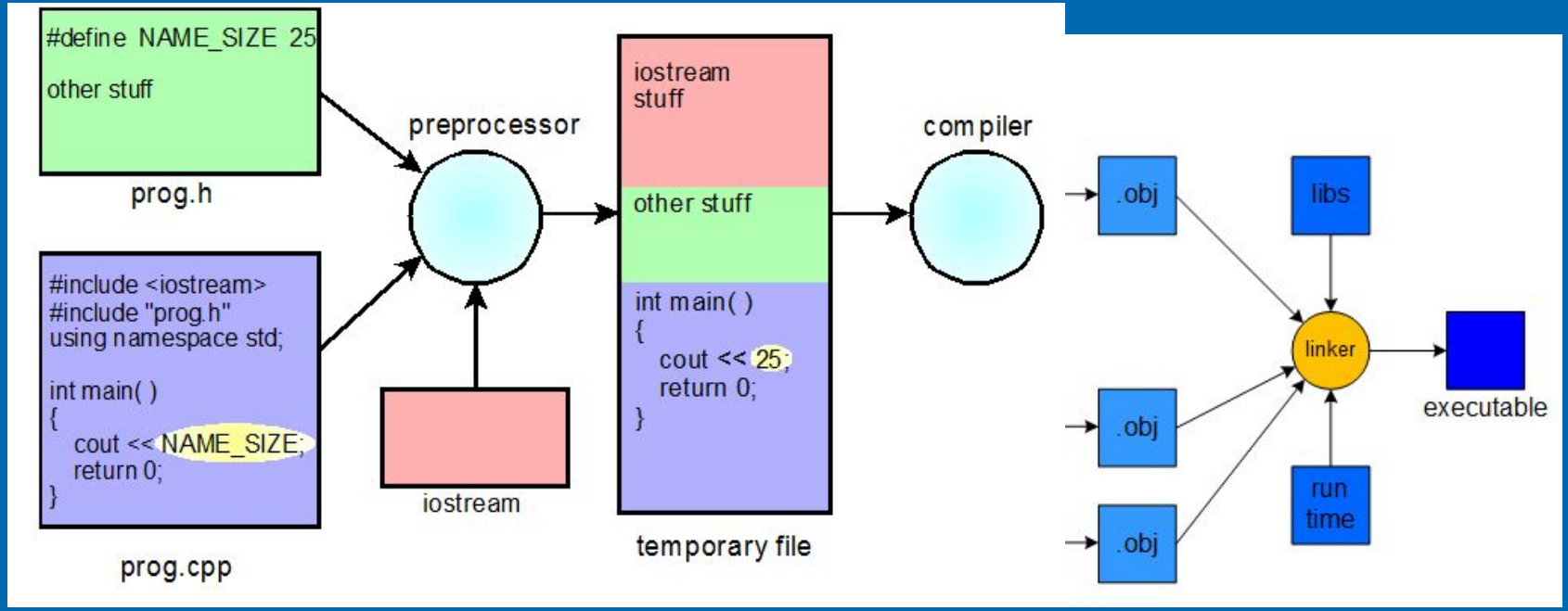


Derleyici (Compiler)

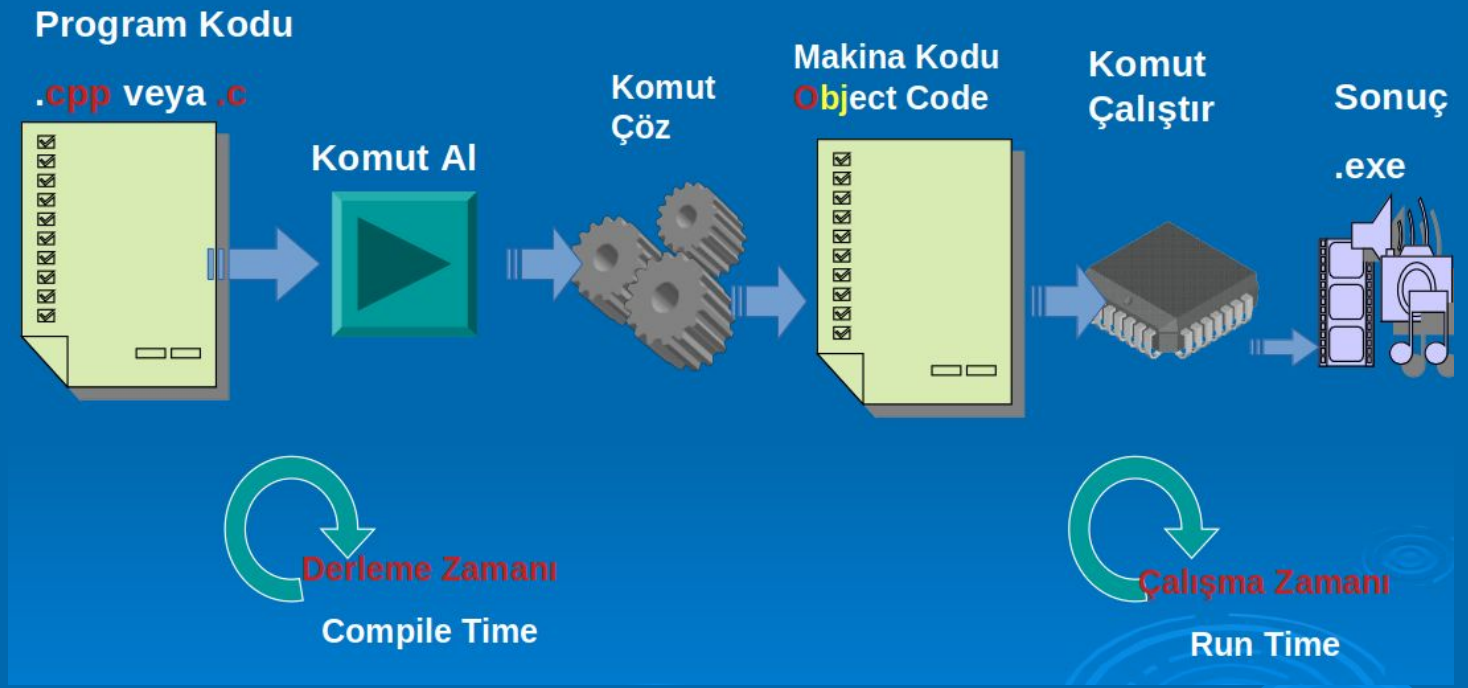




Derleyici (Compiler)



Bir Program Nasıl Derleniyor?



UNIX/Linux Sistemlerinde C Programlarının Komut Satırından Derlenerek Çalıştırılması

Bir program bir metin editörde yazılıp saklandıktan sonra derleme işlemi komut satırından şöyle yapılmaktadır.

```
gcc -o <çalıştırılabilen dosya ismi> <kaynak dosya ismi>

ya da :

clang -o <çalıştırılabilen dosya ismi> <kaynak dosya ismi>

Örneğin:

gcc -o sample sample.c

ya da örneğin:

clang -o sample sample.c
```

Derlenen Dosyanın Komut Satırından Çalıştırılması

UNIX/Linux sistemlerinde bulunulan dizindeki bir programı komut satırından çalıştırabilmek için yalnızca dosyanın ismi yazılmaz. Onun dizini de belirtilmelidir. Tipik çalıştırma şöyle yapılır.

```
./sample
```

"." karakterinin "bulunulan dizini temsil ettiğini anımsayınız.

Örnek;

```
bull@bull-end:~/Downloads$ ./Selam2
C s fun
```

Olası Derleme Zamanı Hata Mesajları?

Error:

```
error: expected ';' before '}' token
```

```
example.cpp:1:20: fatal error: iostraem: No such file or directory
 #include <iostraem>
                    ^
```

```
warning: division by zero [-Wdiv-by-zero]
     div = n/0;
```

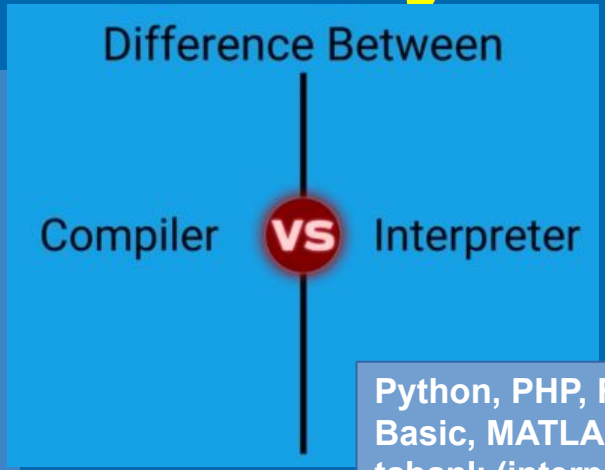
Task :

C/C++ dili Çalışma Zamanı (Runtime) hatalarına örnekler verebilir misin ?

Derleyici – Yorumlayıcı Farkı?

- Pascal, C, C++, C#, dilleri derleyici tabanlı (compiler-based).
- Compiler, kaynak kodu amaç (object) koda dönüştürürler.
- Tüm kodu birden çevirerek, standart bir amaç kod üretirler.
- Derleyici tün kaynak kodu derler

Ve varsa tüm hataların bir listesini verir.



Python, PHP, Ruby, Perl, JavaScript, Basic, MATLAB dilleri yorumlayıcı tabanlı (interpreter-based).

-Yorumlayıcılar, kaynak kodu satır satır doğrudan çalıştırırlar.

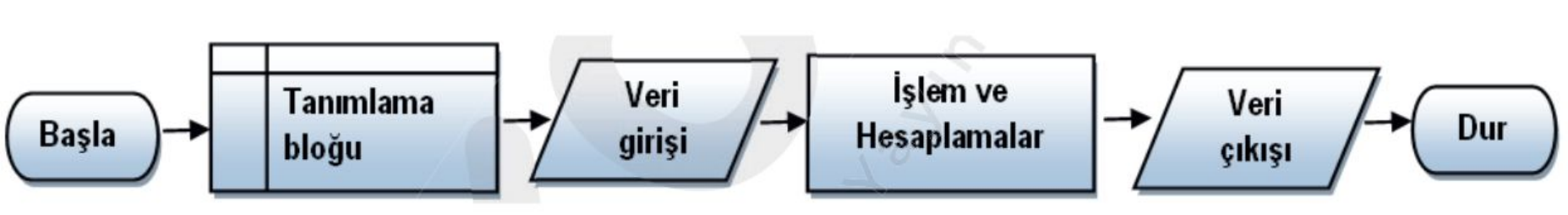
- Yorumlayıcı ilk hatalı satırda durur.

C/C++ DERLEYİCİLERİ

Hemen hemen her işletim sistemi için ayrı bir C/C++ derleyicisi geliştirilmiştir. Yaygın olarak kullanılan bazı C/C++ derleyicileri ve kullanıldığı sistemler;

- **GCC (GNU Compiler Collection) : Linux, Windows (Mingw, Cygwin)**, Android, iOS
  - **MinGW (Minimalist GNU for Windows)**: MinGW ise **Windows** işletim sistemi ortamında çalışan bir GCC derleyicisidir.
- **MSVC (Microsoft Visual C++): Windows**
- **Intel C++: Windows, Linux, MacOS**
- **Clang / Apple clang : X86 ve ARM işlemcili sistemler (Windows, Linux, MacOS)**
- **Embarcadero C++ Builder : Windows**

Genel bir programın yapısı



C/C++ Program Yapısı

	C Program Yapısı: Selam.c	C++ Program Yapısı: Selam.cpp
Başlık (kütüphane) dosyaları	`#include <stdio.h>` `#include <stdlib.h>`	`#include <iostream>` `#include <cstdlib>`
Ana (main) fonksiyon	`int main()`	`using namespace std;` `int main()`
{ Başla	`{`	`{`
Tanımlama bloğu	`//Değişkenler tanımlanır` `int a = 7;`	`//Değişkenler tanımlanır` `int a = 7;`
Program gövdesi (işlem ve hesaplamalar)	`printf ("Selam!");` `printf ("%d", a);`	`cout << "Selam!" << endl;` `cout << a << endl;`
} Dur	`return 0;` `}`	`return 0;` `}`

C/C++ Program Yapısı

```
#include <stdio.h>
#include <stdlib.h>
```

kütüphane dosyalarının yazılması

```
#const int N = 100;
#define pi 3.14
```

Sabit tanımlanması

```
char kar;
char dizi[N]; int
sayi=5;
```

Global değişkenlerin tanımlanması

```
void function_adi() {
....}
```

Alt programların (fonksiyonların) yazılması

```
int main(){
int x;
float y;

//Komutlar
.....
return 0;
}
```

Yerel değişkenler

Ana programın yazılması

C/C++ Program Yapısı

```
{

....

}
```

Yeni bir kod bloğu

```
int x;

float y;
```

Her kod satırı ; ile sonlandırılır.

Yorum (Açıklama) Satırları

```
//  ..... .
```

Tek satırlık açıklamalar

```
/*
     ... .

*/
```

Birden fazla satırlık açıklama satırları

Yorum satırları kimin için yazılır?

Programcı

User

Yorum (Açıklama) Satırları

Programınızı açıklama/yorum satırları ile süslemenin olası faydaları şunlardır;

- O kod satırının işlevini açıklar.
- Kodun anlaşılırlığını artırır.
- Programı belgelendirir.

C/C++ Temel Veri Türleri- Tamsayılar

Tamsayılar:	Byte miktarı
int (unsigned int)	Sistem(Derleyici) Bağımlı
short (unsigned short)	Sistem(Derleyici) Bağımlı
char (unsigned char)	1-Bayt
long (unsigned long)	Sistem(Derleyici) Bağımlı

char<short<=int<=long int<=....

C/C++ Veri Türleri- Reel

Reel Sayılar	Byte miktarı
float	Derleyici bağımlı
double	Derleyici bağımlı

Not. Reel(gerçel) sayı türlerinin işaretli ve işaretsiz biçimleri yoktur. Default işaretlidir.

sizeof Operatörü

Bir nesnenin veya veri tipinin bellekte byte cinsinden kapladığı alanın belirlenmesini sağlayan bir operatördür.

Kullanım şekilleri:

```
sizeof <ifade>
sizeof(<veri tipi>)
sizeof <dizi ismi>
```

```
1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)
```

sizeof operatörü

```
char boyutu..: 1
short boyutu..: 2
int boyutu..: 4
long boyutu..: 8
float boyutu..: 4
double boyutu..: 8
long double boyutu..: 16
```

```c
#include <stdio.h>
int main()
{
    printf("char boyutu..: %ld\n", sizeof(char));
    printf("short boyutu..: %ld\n", sizeof(short));
    printf("int boyutu..: %ld\n", sizeof(int));
    printf("long boyutu..: %ld\n", sizeof(long));

    printf("float boyutu..: %ld\n", sizeof(float));
    printf("double boyutu..: %ld\n", sizeof(double));
    printf("long double boyutu..: %ld\n",sizeof(long double));
    return 0;
}
```

sizeof Operatörü Önceliği

( )	Soldan-Sağa
+ - ++ -- ! sizeof	Sağdan-Sola
* / %	Soldan-Sağa
+ -	Soldan-Sağa
...	...

```c
char a = 'a';
short b = 99;
int dizi[5] = {3, 4, 5, 6, 7};
printf("lale değişkeninin boyutu.:%ld\n",sizeof("lale"));        // 5
printf("a değişkeninin boyutu.:%ld\n",sizeof(a));                // 1
printf("Dizinin kapladigi alan.:%ld\n",sizeof(dizi));            // 20
printf("b değişkeninin boyutu.:%ld\n",sizeof(b));                // 2
printf("%ld \n",sizeof(a+b)); //a + b nin sizeof'unu verir     // ?
printf("%ld \n", sizeof a+b); //a'nın sizeof'u b ile toplanır // ?
```

sizeof operatörü

Aşağıdaki programın ekran çıktısı ne olur? (2018-Tübitak Bilgisayar Olimpiyatları Sorusu)

```
int main() {
 double d[] = {1.0, 2.0, 3.5, 4.5, 5.5, 6.0};
 printf("%d\n", sizeof(d)/sizeof(d[1]));
return 0;
}
```

Veri Tiplerinin Değer Aralığı

Veri tiplerine **_MIN** ve **_MAX** eklentilerini (CHAR_MIN, INT_MAX gibi) getirerek değer aralıklarını öğrenebiliriz.

_MIN ve _MAX eklentileri <limits.h> kütüphanesine ihtiyaç duyar.

Not. Gerçel/Reel veri tiplerinin değer aralığını elde etmek için #include <float.h> kütüphanesini de program başına eklemek gerekir.

Veri Tiplerinin Değer Aralığı

```c
#include<stdio.h>
#include<limits.h>
int main(){
    printf("int deger araligi:%d ... +%d\n", INT_MIN, INT_MAX);
    return 0;
}
```

UTPUT DEBUG CONSOLE TERMINAL JUPYTER: VARIABLES

TERMINAL

```
int deger araligi:-2147483648 ... +2147483647
```

Veri Tiplerinin Değer Aralığı

https://en.cppreference.com/w/cpp/types/climits

https://en.cppreference.com/w/cpp/language/types

Fixed width integer types (since C++11) - cppreference.com

Karaktersel değişkenler	`char`	1 byte	CHAR_MIN - CHAR_MAX veya SCHAR_MIN - SCHAR_MAX
Tam sayı değişkenleri	`unsigned char`	1 byte	0 - UCHAR_MAX
	`short`	2 byte	-2^{15} … $+2^{15}-1$=-32768 … +32767 (SHRT_MIN - SHRT_MAX)
	`short int`	2 byte	-2^{15} … +215-1=-32768 … +32767 ~32bin
	`unsigned short int`	2 byte	0 … $2^{16}-1$ = +65535 (0 - USHRT_MAX)
	`unsigned int`	4 byte	0 … 2^{32} -1 = +4294967295 ~ (0-UINT_MAX)
	`int`	4 byte	-2147483648 … +2147483647 (INT_MIN … INT_MAX)
	`long`	4 byte	-2^{31} = -2147483648 … $+2^{31}-1$ = +2147483647 (LONG_MIN … LONG_MAX)
	`long int`	4 byte	-2^{31} = -2147483648 ~ -2 trilyon, $+2^{31}-1$ = +2147483647 ~ +2 trilyon
	`long long`	8 byte	LLONG_MIN - LLONG_MAX
***Gerçek (Kesirli) sayı değişkenleri**	`float`	4 byte	FLT_MIN – FLT_MAX
	`double`	8 byte	DBL_MIN – DBL_MAX
	`long double`	12 byte	LDBL_MIN – LDBL_MAX

Tip Dönüşümü

C/C++ dillerinde tüm aritmetik türler birbirlerine atanabilir. Ancak atama işleminde bilgi kaybı söz konusu olabilir.

Küçük veri tiplerinin büyük veri tiplerine dönüşümünde bir veri kaybı yaşanmaz

Otomatik Tip Dönüşümü: char --> short, int --> long, float - -> double

İşlem	Veri tipi dönüşümü
int + long	long
int + float	float
float + double	double
int + double	double

Otomatik Tip Dönüşümü

1) Eğer bölme işleminde iki operand da tamsayı türündense sonuç tamsayı türünden çıkar. Bu durumda bölme yapılır, sayının noktadan sonraki kısmı atılır.

```c
#include <stdio.h>

int main()
{
    double a;

    a = 10 / 4;

    printf("%f\n", a);      /* 2.0 */

    return 0;
}
```

Otomatik Tip Dönüşümü

2). Eğer operandlardan her ikisi de int türünden küçükse (örneğin char-char, char-short, short-short gibi) Bu durumda önce her iki operand da bağımsız olarak int türüne dönüştürülür, sonuç int türünden çıkar. Bu kurala "int türüne yükseltme kuralı (integral promotion)" denilmektedir. Örneğin:

```c
#include <stdio.h>

int main()
{
    int a;
    short b = 30000;
    short c = 30000;

    a = b + c;  /* 60000, sonuç int türden */
    printf("%d\n", a);

    return 0;
}
```

Not. Eğer ilgili sistemde short ile int aynı uzunluktaysa ve operandlardan biri unsigned short türündense dönüştürme int türüne doğru değil unsigned int türüne doğru yapılır.

Otomatik Tip Dönüşümü

3) Operandlardan biri tamsayı türünden, diğeri gerçel (reel) sayı türünden ise dönüştürme her zaman reel sayı türüne doğru yapılır. Örneğin **long ile float** işleme sokulursa sonuç **float** türden çıkar.

4) Operandlar aynı tamsayı türünün işaretli ve işaretsiz biçimine ilişkinse dönüştürme her zaman işaretsiz türe doğru yapılır (örneğin **int ile unsigned int** işleme sokulsa sonuç unsigned int türünden çıkar.)

Bilinçli Tip Dönüşümü

Tip dönüşümü otomatik olarak değil de programcı isteği doğrultusunda da gerçekleşebilir. Bu durumda tip dönüşümüne uğrayacak ifade, dönüştürülecek veri tipi sarmalına alınır. Örnek;

```
(<tür>) ifade
```

Buradaki parantez öncelik parantezi değildir, operatör görevindedir.

Örnek tip dönüşüm örnekleri;

```
(int) 4.9;                      // Tam kısım alınır(yuvarlanmaz!). Sonuç: 4
int sayi = 10;
double ort = sayi/ 4;           // Ort= 2.0 dır,2.5 değildir.
float x = (float) sayi/4;       // x = 2.5 olur.
```

Tip Dönüşümünde veri kaybı!

Büyük veri tiplerinin küçük veri tiplerine dönüştürülmesinde (yüksek anlamlı byte değerinde) veri kaybı yaşanabilir.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char a; //küçük veri tipi
int b; //büyük veri tipi
b=0x1453;
a=b; // b, a'ya kopyalandı
printf( "%x\n",a);
return 0;
}
```

Ekran Çıktısı

```
53
```

SAYI SİSTEMLERİ

Sistem	Taban	C/C++ Öntakısı	Sayılar
İkili (Binary)	2	0b, 0B	0 1
Sekizli (Octal)	8	0	0 1 2 3 4 5 6 7
Onlu (Decimal)	10		0 1 2 3 4 5 6 7 8 9
Onaltılı (Hexadecimal)	16	0x, 0X	0 1 2 3 4 5 6 7 8 9 A B C D E F

SAYI SİSTEMLERİ

Buna göre onluk sistemde verilen 17 sayısı, aşağıdaki gibi dört farklı şekilde ifade edilebilir;

- `int desimalInt = 17`
- `int binaryInt = 0b10001`
- `int octalInt = 021`
- `int hexInt = 0x11`

SAYI SİSTEMLERİNİN DÖNÜŞÜMÜ

C/C++ dillerinde bir sayı sistemini diğer sayı sistemlerine (onlu, sekizli ve onlatılı) dönüştürüp ekranda göstermek için printf ile birlikte ;
‘**%d’ (onlu), ‘%x’(onaltılı), ‘%o’ (sekizli)** çıkış formatı karakterleri kullanılırken cout<< ile birlikte ‘**dec’ (onlu), ‘hex’ (analtılı), ‘oct’ (sekizli)** deyimleri kullanılır.

Fakat sadece ekrana yazdırmayıp, sonucu bir değişkende tutmak istiyorsanız Sayısaldan Sözele (stringe) veya tam tersi dönüşümü gerçekleştiren fonksiyonları **‘itoa(), atoi()** gibi’ kullanmak gerekir.

SAYI SİSTEMLERİNİN DÖNÜŞÜMÜ

- İki adet ikili binary) sayıyı toplayıp toplam sonucu farklı sayı tabanlarında (8'li, 10'lu ve 16'lı) ekranda gösteren programı kodlayınız?

  int A = **0b**1001;

  int B = **0b**0110;

NEGATİF SAYILARIN İKİLİ SAYI SİSTEMİNDE GÖSTERİLMESİ

- İkilik (binary) sayı sisteminde negatif sayıları nasıl gösteririz?

printf() fonksiyonu kullanımı

Sayısal değerler yazdırmak için:

```
printf(“a değişkeni=%d”,a);   // tamsayı
printf(“b değişkeni=%f”,b);    // reel
```

Karakter ve metin yazdırmak için:

```
printf(“c değişkeni=%c”,c);   //karakter
printf(“cümle=%s”,cumle);    //metin
```

printf() içerisinde kullanılan özel karakterlerin anlamı: %

%d, %i : Tamsayı yazdırma
%c : Karakter yazdırma
%s : Metin yazdırma
%f : Reel sayı yazdırma
(float, double)
%o : Sekizlik sayı yazdırma
%x, %X : Hex sayı yazdırma
%u : İşaretsiz tamsayı
%hd : Short tamsayı
%e, %E : Üslü reel sayı yazdırma
%g, %G : Üslü veya reel
%p :Adres (pointer) (4 veya 8 rakam)

%ld, %li, %lu, %lx, %lo : Long tamsayı
%Lf, %Le :Long reel

printf() içerisinde kullanılan format karakterlerin anlamı: %

Format Karakterleri	Anlamı
%d	int, short ve char türlerini 10'luk sistemde yazdırır
%ld	long int türünü 10'luk sistemde yazdırır
%x, %X	int, short ve char türlerini hex sistemde yazdırır
%lx, %lX	long int ve unsigned long int türlerini hex sistemde yazdırır
%u	unsigned int, unsigned short ve unsigned char türlerini 10'luk sistemde yazdırır
%lu	unsigned long int türünü 10'luk sistemde yazdırır
%f	float ve double türlerini 10'luk sistemde yazdırır
%c	char, short ve int türlerini karakter görüntüsaü olarak yazdırır
%o	char, short ve int türlerini octal sistemde yazdırır
%lo	long ve unsigned long türlerini octal sistemde yazdırır

cout\printf() içerisinde kullanılan özel karakterlerin anlamı: \

\a :bip
\b :geri silme
\f :ileri sarma (yazıcı)
\n :Sonraki satır başına git
\r :Geçerli satır başına git
\t :Yatay sekme
\v :Dikey sekme

\\ :Ters bölü karakteri
\? :Soru işareti karakteri
\’ :Tek tırnak karakteri
\” : Çift tırnak karakteri
%% : Yüzde karakteri

Çıktıların Biçimlendirilmesi

Tanımlama:

```
float reel=1.23456789;
```

Virgülden sonra 2 basamak yazdırılması:

```
printf("%.2f",reel);          //C, virgülden sonra iki basamağı al

cout.precision(3);            //C++, kesirli sayının soldan 3 basamağını al
cout<<reel;
```

Not. printf %f formatında default olarak noktadan sonra 6 basamak yazdırır. Yazdırılacak değer noktadan sonra 6 basamaktan fazlaysa yuvarlama yapılır. eğer printf ile noktadan sonra istediğimiz kadar basamak yazdırmak istiyorsak %.nf formatı kullanılmalıdır (burada n yerine bir sayı olmalıdır. Örneğin %.10f gibi).

C++ dili çıktıların biçimlendirilmesi

setprecision() fonksiyonu: setprecision fonksiyonu bir kesirli sayının kaç basamağının gösterileceğini belirler. Kullanım şekli;

```
setprecision(basamak_sayisi);
```

setw: Görüntülenecek alan genişliğini ayarlar.

setfill: Sağa yaslanmış bir ekranda boşlukları doldurmak için kullanılacak karakteri ayarlar.

setw(), setfill(), setprecision() fonksiyonları; <iomanip> kütüphanesine ihtiyaç duyar.

C++ dili çıktıların biçimlendirilmesi

```cpp
#include <iomanip>

using namespace std;
int main(){
int n = 5;
const double   d1 = 1.23456789;

cout << "d1 = " << setw(n) << d1 << endl;
cout <<  setprecision(n) << d1<< endl;
cout <<  setfill ('0') << setw (10)  << d1 << endl;
return 0;
}
```

```
OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
TERMINAL

d1 = 1.23457
1.2346
00001.2346
```