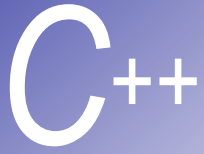


C++

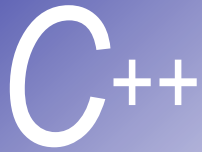
Ders 11

C' den Kalma Program Başlıkları



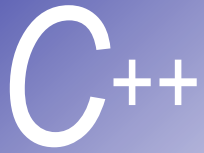
Giriş

- Bu bölümde ileri düzey birkaç başlık var
- Birçok özellik işletim sistemine bağlı olarak değişmesi (UNIX ve/veya DOS.)
- C den kalma kod yapısının C++ programcılarına faydası



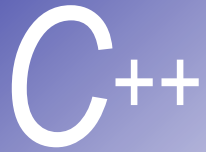
UNIX ve DOS' da Input/Output' u Tekrar Yönlendirme

- Standard I/O – klavye ve ekran
 - input ve output' u tekrar yönlendirebiliriz
- Tekrar yönlendirme sembolü (<)
 - İşletim sisteminin özelliği, C++ özelliği anlamına gelmez
 - UNIX ve DOS
 - \$ veya % komut satırını temsil eder
 - örneğin: `$ myProgram < input`
 - input dosyasını klavyeden girmektense dosyadan okutma yapılabilir
- Boru komutu (|)
 - Bir programın çıkış verileri başka bir programın giriş verileri olabilir
 - `$ firstProgram | secondProgram`
 - `firstProgram` in çıkış verileri `secondProgram`' na giriş verileri oluyor



UNIX ve DOS' da Input/Output' u Tekrar Yönlendirme (II)

- Redirect output (>)(Tekrar yönlendirilen çıkış dosyası)
 - Bir programın çıkış verilerinin nereye gideceğini belirler
 - `$ myProgram > myFile`
 - Çıkış verileri `myFile` dosyasına gider(mecut içerik silinir)
- İlave output (>>)
 - Dosyanın sonuna programın çıkış verilerini ekler(önceki içerik kaybolmaz bitimine eklenir)
 - `$ myOtherProgram >> myFile`
 - Program çıktısı `myFile` dosyasının sonuna eklenir

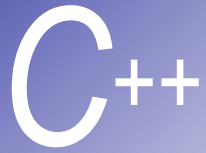


Değişken –Argüman Uzunluğu Listesi

- C++ da fonksiyon overloading kullanırız
 - Değişken argüman uzunluğu listesi C den kalma kod özeliğini kullanan programcılar içindir
- Argüman sayısı özel olamayan fonksiyonlar
 - `<cstdarg>`' yi yükle
 - Parametre listesinin bitiminde `(...)` 'si kullan
 - Tanımlanmış en az bir tane parametreye ihtiyaç vardır

```
double myfunction (int i, ...);
```

- Değişken argüman uzunluğu listesinin protipi



Değişken –Argüman Uzunluğu Listesi (II)

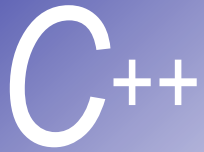
- Fonksiyon tanımlamasında makrolar ve tanımlamalar

`va_list` – tür belirleyici, (`va_list arguments;`)
gerekir

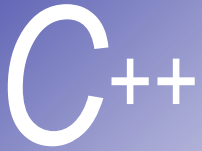
`va_start(arguments, diğer değişkenler)`
- parametrelerin inişilay edilmesi kullanılmadan önce yapılmalıdır

`va_arg(arguments, tür)` – `va_arg` her çağrıldığında bir parametre ile döner. Otomatik olarak bir sonraki parametreyi gösterir

`va_end(arguments)` – Fonksiyonlara bir geri dönüş değerleri olması için yardım eder



```
1  // Fig. 18.2: fig18_02.cpp
2  // Using variable-length argument lists
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7  using std::ios;
8
9  #include <iomanip>
10
11 using std::setw;
12 using std::setprecision;
13 using std::setiosflags;
14
15 #include <cstdarg>
16
17 double average( int, ... );
18
19 int main()
20 {
21     double w = 37.5, x = 22.5, y = 1.7, z = 10.2;
22
23     cout << setiosflags( ios::fixed | ios::showpoint )
24           << setprecision( 1 ) << "w = " << w << "\nx = " << x
```



```

25         << "\nv = " << v << "\nz = " << z << endl;
26     cout << setprecision( 3 ) << "\nThe average of w and x is "
27         << average( 2, w, x )
28         << "\nThe average of w, x, and v is "
29         << average( 3, w, x, v )
30         << "\nThe average of w, x, v, and z is "
31         << average( 4, w, x, v, z ) << endl;
32     return 0;
33 }
34
35 double average( int i, ... )
36 {
37     double total = 0;
38     va_list ap;
39
40     va_start( ap, i );
41
42     for ( int i = 1; i <= i; i++ )
43         total += va_arg( ap, double );
44
45     va_end( ap );
46
47     return total / i;
48 }

```

```

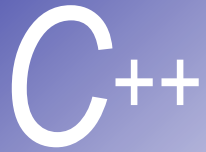
w = 37.5
x = 22.5
y = 1.7
z = 10.2

```

```

The average of w and x is 30.000
The average of w, x, and y is 20.567
The average of w, x, y, and z is 17.975

```

Komut satırı Argümanı Kullanımı

- DOS ve UNIX' de `main`' e argüman geçirilir

```
int main( int argc, char *argv[] )  
    int argc – Geçirilen argüman sayısı  
    char *argv[] – Argüman isimlerini tutan  
    string dizisi (argv[0] ilk argüman)
```

Örnek: `$ copy input output`

`argc: 3`

`argv[0]: "copy"`

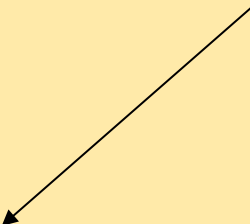
`argv[1]: "input"`

`argv[2]: "output"`

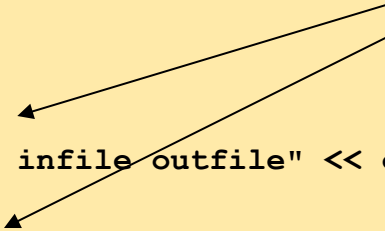
C++

```
1 // Fig. 18.3: fig18_03.cpp
2 // Using command-line arguments
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7 using std::ios;
8
9 #include <fstream>
10
11 using std::ifstream;
12 using std::ofstream;
13
14 int main( int argc, char *argv[] )
15 {
16     if ( argc != 3 )
17         cout << "Usage: copy infile outfile" << endl;
18     else {
19         ifstream inFile( argv[ 1 ], ios::in );
20
21         if ( !inFile ) {
```

**argc ve argv[] ,
main'de bulunur**



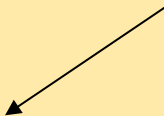
**argv[1] ikinci
argümandır**




C++

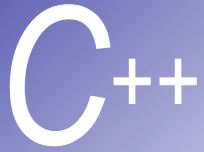
```
22         cout << argv[ 1 ] << " could not be opened" << endl;
23         return -1;
24     }
25
26     ofstream outFile( argv[ 2 ], ios::out );
27
28     if ( !outFile ) {
29         cout << argv[ 2 ] << " could not be opened" << endl;
30         inFile.close();
31         return -2;
32     }
33
34     while ( !inFile.eof() )
35         outFile.put( static_cast< char >( inFile.get() ) );
36
37     inFile.close();
38     outFile.close();
39 }
40
41 return 0;
42 }
```

argv[2] üçüncü argümandır



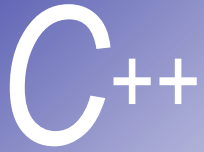
End Of File'e kadar
döngü devam eder.
inFile dan karakter alıp
outFile dosyasına yazar





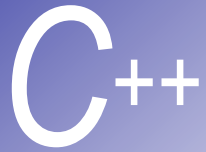
Birden fazla Kaynak dosyasından oluşan Programları Derlemek

- Birden fazla Kaynak kaynak dosyasından oluşan Programlar
 - Fonksiyon tanımlamaları bir dosyada yapılmalı
 - Global değişkenler bulundukları dosyadaki fonksiyonlara ulaşabilirler
 - Global değişkenler kullanıldıkları her dosyada tanımlanmalıdır
 - Örnek:
 - `myGlobal` integer olarak bir dosyada tanımlanmış, başka bir dosyada kullanımı için:
`extern int myGlobal;`
 - `extern` – bu değişken başka yerde yada yerlerde de tanımlanmış anlamına gelir



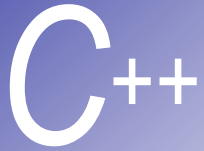
Birden fazla Kaynak dosyasından oluşan Programları Derlemek (II)

- Fonksiyon prototipleri başka dosyalarda belirtilebilir, **extern** kullanmaya gerek yoktur
 - Her dosya kullandığınız fonksiyonun prototipini belirtiniz
- Örnek: başlık dosyasını yükleyerek
 - **#include <cstring>**
 - Fonksiyon prototipleri içerir
 - Fonksiyonu nerede tanımlandığını bilemeyiz
- **static** anahtar kelimesi
 - Değişkenler sadece tanımlandıkları dosyada kullanılabilirler
- Birden fazla Kaynak Kaynak dosyasından oluşan Programlarda
 - Her şeyi derlemek sıkıcıdır
 - Eğer küçük bir şey değiştirilmişse sadece o dosya tekrar derlenebilir
 - İşletim sistemine göre değişir
 - UNIX: **make** utility



exit ve atexit ile Program Sonlandırılması

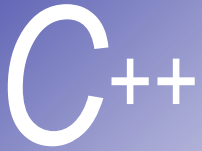
- **exit** fonksiyonu – bir programı sonlanmaya zorlar
 - Parametreler – sembolik sabit olan **EXIT_SUCCESS** ve **EXIT_FAILURE**
 - İmplemente edilen değerle geri döner
 - **exit(EXIT_SUCCESS) ;**
- **atexit** fonksiyonu
 - **atexit(functionToRun) ;** –Çalışan programın sonlanması için **functionToRun** register yapar
 - **atexit** tek başına programı sonlandıramaz
 - 32 fonksiyonu register yapar (multiple **atexit()** statements)
 - functions called in reverse register order
 - Çağrılan fonksiyonlar argüman almaz ve geri dönüş değeri olmaz



```

1  // Fig. 18.4: fig18_04.cpp
2  // Using the exit and atexit functions
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7  using std::cin;
8
9  #include <cstdlib>
10
11 void print( void );
12
13 int main()
14 {
15     atexit( print );           // register function print
16     cout << "Enter 1 to terminate program with function exit"
17           << "\nEnter 2 to terminate program normally\n";
18
19     int answer;
20     cin >> answer;
21
22     if ( answer == 1 ) {
23         cout << "\nTerminating program with function exit\n";
24         exit( EXIT_SUCCESS );
25     }
26
27     cout << "\nTerminating program by reaching the end of main"
28           << endl;
29
30     return 0;
31 }
32
33 void print( void )

```



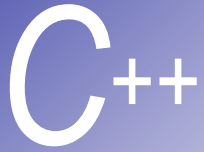
```
34 {  
35     cout << "Executing function print at program termination\n"  
36         << "Program terminated" << endl;  
37 }
```

```
Enter 1 to terminate program with function exit  
Enter 2 to terminate program normally  
: 1
```

```
Terminating program with function exit  
Executing function print at program termination  
Program terminated
```

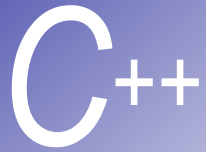
```
Enter 1 to terminate program with function exit  
Enter 2 to terminate program normally  
: 2
```

```
Terminating program by reaching the end of main  
Executing function print at program termination  
Program terminated
```

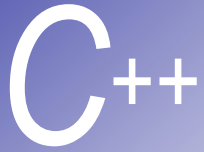
volatile Tip Qualifier

- **volatile** qualifier – değişken program dışından değiştirilebilir
 - Değişken program kontrolünde değil
 - Değişken optimizme edilemez



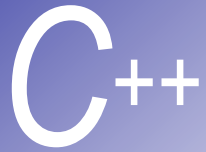
Integer ve Floating-Point Sabitleri için Sonek

- C++ sabitler için sonek sağlar
 - Integer - `u` veya `U` (`unsigned integer`)
 - `long integer` - `l` veya `L`
 - `unsigned long integer` - `ul` veya `UL`
 - `float` - `f` veya `F`
 - `long double` - `l` veya `L`
- Örnek:
 - `174u`
 - `467L`
 - `3451ul`
- defaults
 - integers: en kısa tür (`int`, `long int`, `unsigned long int`)
 - floating point sayıları: `double`



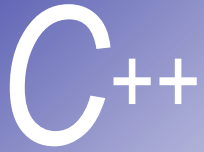
Sinyal İşleme

- **signal**
 - Beklenmedik bir olayda program sonlandırılabilir
 - (`<ctrl> c`) yarıda keser, illegal bilgiler, kesimleme hataları, sonlandırma komutu, floating-point exceptions (sıfıra bölme, büyük sayı ile çarpma)
- **signal** fonksiyonu beklenmedik olayları tespit eder
 - Başlık dosyası `<csignal>`
 - İki argüman: işaret sayısı, fonksiyon pointeri
- **raise** fonksiyonu
 - Integer işaret sayısı alır ve işaret oluşturur



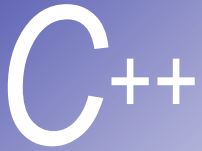
Sinyal İşleme (II)

Signal	Explanation
SIGABRT	Abnormal termination of the program (such as a call to abort).
SIGFPE	An erroneous arithmetic operation, such as a divide by zero or an operation resulting in overflow.
SIGILL	Detection of an illegal instruction.
SIGINT	Receipt of an interactive attention signal.
SIGSEGV	An invalid access to storage.
SIGTERM	A termination request sent to the program.



Sinyal İşleme (III)

- Dinamik bellek tahsisatı
 - Dinamik dizi yaratabilir
- `calloc(nmembers, size)`
 - `nmembers` – üye sayısı
 - `size` – her bir üyenin büyüklüğü
 - Dinamik dizinin pointeri ile döner
- `realloc(pointerToObject, newSize)`
 - `pointerToObject` – tekrar tahsisat yapılmış nesneyi gösterir
 - `newSize` – yeni nesnenin büyüklüğü
 - Tekrar tahsisat edilmiş belleği gösterir
 - Bellek tahsisatı yapılamazsa `NULL` ile döner
 - Eğer `newSize = 0`, nesne serbest bırakılmış olur
 - Eğer `pointerToObject = 0`, `malloc` gibi davranır



```

1 // Fig. 18.6: fig18_06.cpp
2 // Using signal handling
3 #include <iostream>
4
5 using std::cout;
6 using std::cin;
7 using std::endl;
8
9 #include <iomanip>
10
11 using std::setw;
12
13 #include <csignal>
14 #include <cstdlib>
15 #include <ctime>
16
17 void signal_handler( int );
18
19 int main()
20 {
21     signal( SIGINT, signal_handler );
22     srand( time( 0 ) );
23
24     for ( int i = 1; i < 101; i++ )
25     {
26         int x = 1 + rand() % 50;
27
28         if ( x == 2 )
29             raise( SIGINT );
30
31         cout << setw( 2 ) << i << " ";
32         if ( i % 10 == 0 )
33             cout << endl;
34     }

```

signal tipi SIGINT olduğu zaman,
signal signal_handler
fonksiyonun çağrılmasını sağlar.

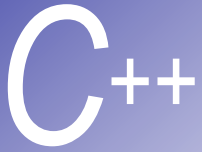
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Interrupt signal (4) received.

Do you wish to continue (1 = yes or 2 = no)? 1


89 90

91 92 93 94 95 96 97 98 99 100

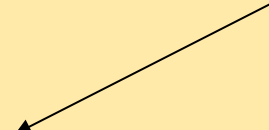


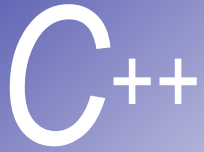
```
35
36     return 0;
37 }
38
39 void signal_handler( int signalValue )
40 {
41     cout << "\nInterrupt signal (" << signalValue
42         << ") received.\n"
43         << "Do you wish to continue (1 = yes or 2 = no)? ";
44
45     int response;
46     cin >> response;
47
48     while ( response != 1 && response != 2 ) {
49         cout << "(1 = yes or 2 = no)? ";
50         cin >> response;
51     }
52
53     if ( response == 1 )
54         signal( SIGINT, signal_handler );
55     else
56         exit( EXIT_SUCCESS );
57 }
```

Kullanıcı program
kapama seçimini verir

An arrow points from the text box to line 41 of the code, specifically to the first occurrence of 'signalValue' in the cout statement.

signal handler, **signal**
tekrar çağrılarak tekrar
initialized edilir

An arrow points from the text box to line 54 of the code, specifically to the 'signal' function call.



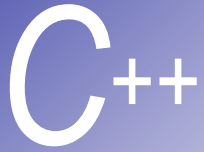
```
1  2  3  4  5  6  7  8  9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88
```

Interrupt signal (4) received.

Do you wish to continue (1 = yes or 2 = no)? 1

```
89 90
```

```
91 92 93 94 95 96 97 98 99 100
```

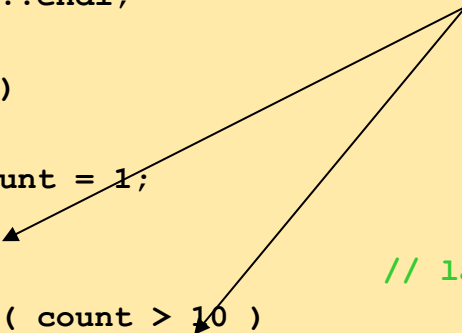
Şartsız Dallanma: goto

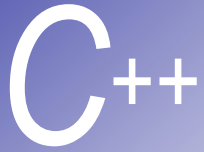
- Yapısal olmayan programlama
 - Performans önemli olduğu zaman kullanılır
 - **break** ile döngüden çıkılır, **false** durumunun oluşmasını beklemek yerine
- **goto** ifadesi
 - İlk ifadeden belirtilen ifadeye doğru program akışını yönlendirir
 - etiket: belirleyici ve iki nokta üst üste (i.e. **start:**)
 - Uzun döngülerden hızlı kaçılır
 - **goto start;**

C++

```
1 // Fig. 18.7: fig18_07.cpp
2 // Using goto
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 int main()
9 {
10     int count = 1;
11
12     start:                                // label
13         if ( count > 10 )
14             goto end;
15
16         cout << count << " ";
17         ++count;
18         goto start;
19
20     end:                                  // label
21         cout << endl;
22
23     return 0;
24 }
```

start: , end: ve goto
kullanımı

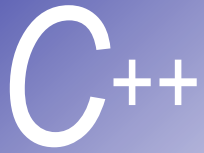




Birlikler (Union)

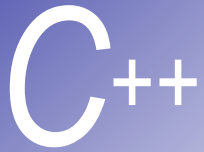
- **union** – farklı bir nesne içeren bellek
 - Her bir seferinde bir tane üyeyi içerir
 - Unin'ların üyeleri aynı belleği paylaşırlar
 - Bellek korurlar
 - Sadece son tanımlanan dataya ulaşılabilir
- **union** tanımlaması
 - **class** yada **struct** ile aynı

```
union Number {  
    int x;  
    float y;  
} ;  
Union myObject;
```



Birlikler (Union) (II)

- Geçerli **union** operatörleri
 - Aynı türden birliklere atama yapılır: `=`
 - Adres alır: `&`
 - Birlik üyelerine ulaşılabilir: `.`
 - Pointer kullanılarak birlik üyelerine ulaşılabilir: `->`
- İsimsiz **union**' lar
 - Tür ismi yoktur
 - Bir tür ismi yaratmazlar ama isimsiz bir nesne yaratırlar
 - **public** data üyeleri içerirler
 - Data üyeleri normal değişken gibi ulaşılabilir
 - İsim, no kullanılır `.` yada `->` gerekir
 - Global olarak tanımlanırsa, **static** olmak zorundadırlar



```

1 // Fig. 18.8: fig18_08.cpp
2 // An example of a union
3 #include <iostream>
4
5 using std::cout;
6 using std::endl;
7
8 union Number {
9     int x;
10    double y;
11 };
12
13 int main()
14 {
15     Number value;
16
17     value.x = 100;
18     cout << "Put a value in the integer member\n"
19          << "and print both members.\nint:   "
20          << value.x << "\ndouble: " << value.y << "\n\n";
21
22     value.y = 100.0;
23     cout << "Put a value in the floating member\n"
24          << "and print both members.\nint:   "
25          << value.x << "\ndouble: " << value.y << endl;
26     return 0;
27 }

```

Put a value in the integer member
and print both members.

int: 100
double: -9.25596e+061

Put a value in the floating member
and print both members.

int: 0
double: 100

C++

Put a value in the integer member
and print both members.

int: 100

double: -9.25596e+061

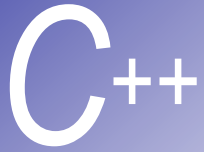
Integerlerin dahili temsilcisi ve
float fazlasıyla değişir

Put a value in the floating member
and print both members.

int: 0

double: 100

float set edildiğinde integer
değer korunmaz



Bağlantı Özellikleri

- Derlenmiş C fonksiyonları isim bilgisine sahip değildirler, fakat C++ fonksiyonları sahiptir
 - C kodunu ile C++ kodunu link etme (Bağlama) problemi
- C++ link kullanma izni verir
 - Derleyiciye C de hangi fonksiyon derlendiğini bildirir
 - Derleyici tarafından program içinde fonksiyon ismi encode edilemez
 - C fonksiyonları, C++ kodu ile linklenebilir
- single fonksiyonlar için: `extern "C" fonksiyon prototipi`
- multiple fonksiyonlar için:

```
extern "C"
{
    function prototypes
}
```