2. C++ programlama dili

C++ Dilinin Tarihsel Gelişimi

C Programlama dili genel amaçlı orta seviyeli ve yapısal bir programlama dilidir. 1972 yılında Dennis Ritchie tarafından Bell Laboratuarında Unix işletim sistemi ile kullanılmak için tasarlanmıştır. C, özellikle sistem programlamada sembolik makine dili (Asembly) göre tercih edilmektedir. İşletim sistemleri, derleyiciler ve debug gibi aşağı seviyeli sistem programlarının yazılımında yoğun olarak C programlama dili kullanılır.

C++ dili C diline bazı özelliklerin eklenmesi ile Bjarne Stroustroup tarafından BELL laboratuarlarında gelistirilmistir. 1983 ilk ***C***++ derleyicisi sunulmuştur. 1998 ISO ***C***++ dili standartlari kabul edilmiştir.

Neden C++
-Daha iyi bir C dir. C dilinin bir üst kümesi olduğundan bu dilin hemen hemen bütün özelliklerini destekler
-Veri Soyutlamasını destekler
-Nesne yönelimli programlamayı destekler
-Geliştirilebilir programlamayı destekler[1]
-Çok sayıda kaynak dokümana sahiptir

.

C++ İle Programlamaya Giriş

C++ programlama dilini öğrenmeye basit bir program yazarak başlayalım. Aşağıda "Hello.cpp" olarak bilgisayara kaydedilen bir program görülmektedir.Bu program ekrana "Hello World" Türkçesi "Merhaba Dünya" anlamına gelen kelimeleri yazmaktadır.

```
// Hello.cpp
// Ekrana " Hello World" yazan program
#include <iostream.h>

int main()
{
cout << " Hello World\n";

return 0;
}
```

Şekil 2.1 Basit bir C++ programı

Programın ekran çıktısı şu şekildedir.

```
Hello World
```

Programın açıklaması

C dilinde yazılan programları uzantısı " .c" ve C++ dilinde yazılan programların uzantısı ".cpp" dir.
"//" işareti açıklama(yorum) satırlarını ifade etmektedir. C++ derleyicisi bu işaretle başlayan satırları dikkate almaz. Birden fazla satırı iptal etmek veya açıklama satırı olarak göstermek için bunları da "/*" ve "*/" işaretleri arasına almak gerekir.

#include <iostream.h> satırı C++ da bulunan ve ismi belirtilerek istenilen kütüphane dosyasını programa dahil etmek için kullanılmıştır. Burada <iostream.h> kütüphane dosyası programa dahil edilmektedir. Bu dosya içinde cout ve cin komutları bulunmaktadır. Cout ve cin komutlarını kullanabilmek için bu kutuphane dosyası programa dahil edilmek zorundadır. ".h" ile biten dosyalar kütüphane dosyalarıdır. C++ da ki hazır kütüphanelerde bir çok hazır fonksiyonlar vardır.

Her C++ programında mutlaka bir main() isimli ana fonksiyon yer almak zorundadır. Program main() fonksiyonundan itibaren çaşlıştırılmaya başlanır. Main() fonksiyonunun önünde yer alan "int" ifadesi main() fonksiyonunun dönüş tipini göstermektedir. Yani bu fonksiyon geriye int(tamsayı) bir değer döndermektedir. Dolayısı ile "return 0" ifadesi ile geriye 0 değeri dönderilmektedir. Cout komutu ekrana bir değeri veya cümleyi yazdırmak için kullanılan bir komut(fonkiyon)dur.
C++ da ana fonksiyondan hemen sonra yer alan komutlar " { } " parantezleri arasına yazılır.

Önemli bir nokta ise C++ dilinde her komut " ; " ile bitmek zorundadır.

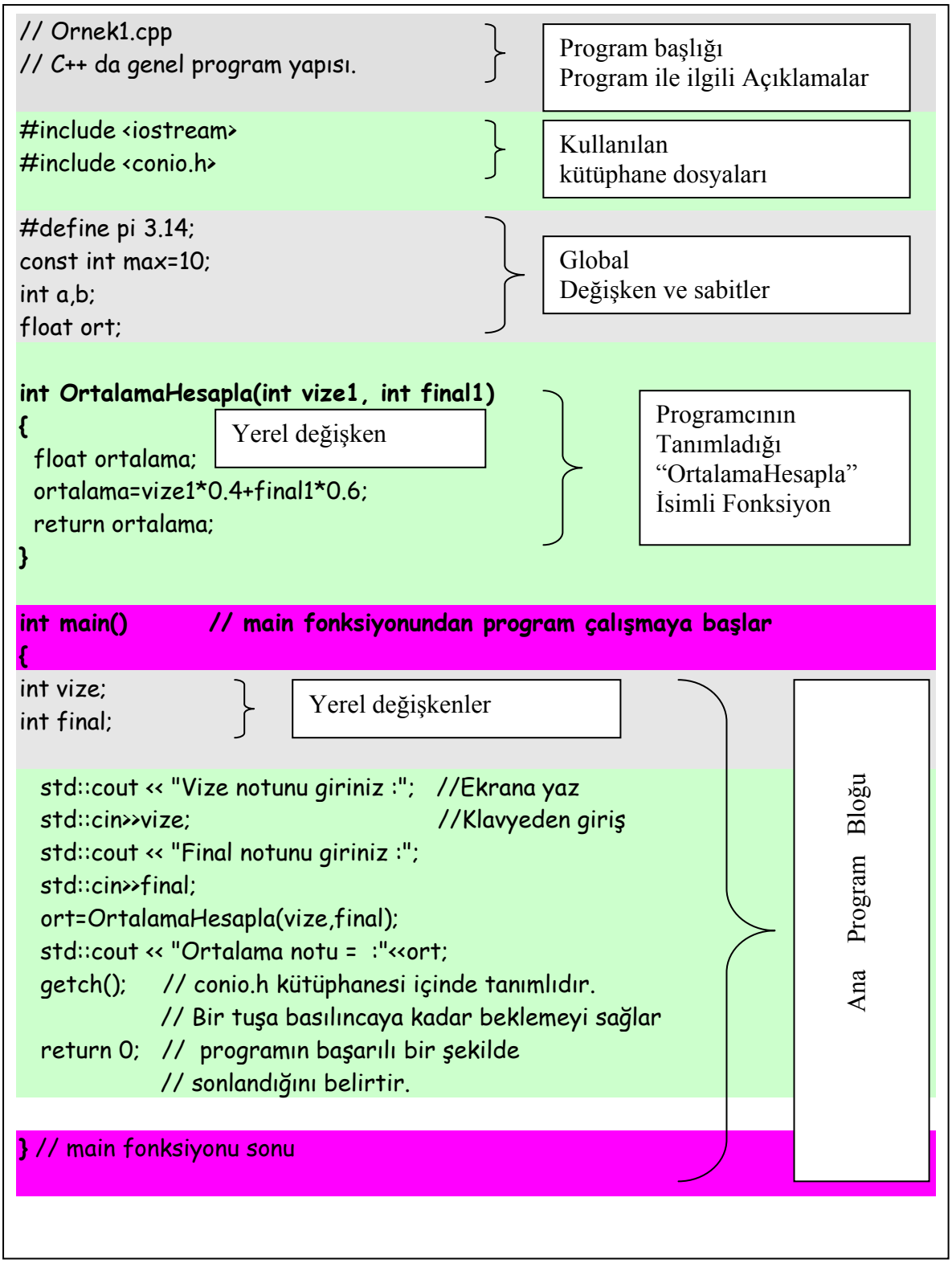
" Hello World\n" cümlesinin sonunda yer alan " \n " karakterleri imleci bir sonraki satıra konumlandırmak için kullanılmaktadır. Bu şekilde kullanılan karekter sabitleri ve ne için kullanıldıkları aşağıda görülmektedir.

	Tanım
'\0'	NULL karakter
'\a'	çan sesi (alert)
'\b'	geri boşluk (back space)
'\t'	tab karakteri (tab)
'\n'	aşağı satır (new line)
'\v'	düşey tab (vertical tab)
'\f'	sayfa ileri (form feed)
'\r'	satır başı (carriage return)
'\"'	çift tırnak (double quote)
'\\'	ters bölü (back slash)

" \n " karekter sabiti kullanmadan aşağıdaki komut da aynı işlevi görmektedir. "endl" komutu bir alt satıra imleci konumlandırır.

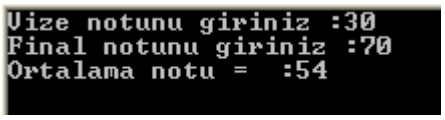
```
cout << " Hello World"<<endl;
```

C++ dilinin genel yapısı aşağıda verilen örnek programla daha iyi anlaşılacaktır.



Şekil 2.2 Bir C++ programının yapısı

Programın ekran çıktısı



Bir C++ programı sırası ile aşağıdaki bölümlerden oluşur.

Program başlığı, program ile ilgili açıklamalar

Kullanılan kütüphane dosyaları

Global değişken tanımları

Programcının tanımladığı fonksiyonlar

Main fonksiyonu ana program bloğu

Şimdi bu bölümleri kısaca açıklayalım.

Program başlığı, program ile ilgili açıklamalar

Bu kısımda yazılan program ile ilgili açıklamalar, programı kimin yazdığı, yazım tarihi, amacı gibi açıklamalar yazılır. Yazımı zorunlu olmamakla birlikte tavsiye edilmektedir.

Kullanılan kütüphane dosyaları

Bu bölümde programda kullanılacak kütüphane dosyalarının isimleri yazılır.

Global değişken tanımları

Değişkenler
Programın içerisinde farklı değerler alabilen, değeri değiştirilebilen semboler değişken olarak tanımlanır. Bu semboller aslında bilgisayar belleğinde verinin saklandığı yeri ifade ederler. Dolayısı ile bu alana yerleştirilecek veri de sürekli olarak değişebilir. Değişkenleri global ve yerel değişkenler olarak iki gruba ayırmak mümkündür.

Global değişkenler main() fonksiyonu dışında programın baş kısmında tanımlanan değişkenlerdir. Bu değişkenler programcının tanımladığı tüm fonksiyonlar ve ana program bloğu içinden erişilip kullanılabilir. Başka bir ifade ile programın her yerinde faaliyet gösterebilen, geçerli olan değişkenlerdir. Birden fazla fonksiyon aynı isimli global değişkene erişip kullanabilir.Bu durum program hatalarına neden olabilir.

Yerel değişkenler Bir fonksiyon içinde ya da bir program bloğu içinde tanımlanan değişkenlerdir. Main() fonksiyonu içinde tanımlanan değişkenler de yerel değişkenlerdir.

Değişkenler aşağıdaki şekilde tanımlanır.

[Veri Tipi] [Değişken Adı];
Örnek:
char c; int sayi; float f; double d; unsigned int k; string adsoy;

Veri Tipleri

Sayısal veri tipleri

Karakter veri tipi

Mantıksal veri tipi

String (Karakter dizisi) veri tipi

Sayısal veri tipleri(Tamsayı,Gerçek)

Tamsayi veri tipi (char, int, long)

Tamsayı değerler alırlar. Aşağıdaki tabloda tamsayı veri türleri görülmektedir.

TÜR İSMİ	UZUNLUK(byte)	SINIR DEĞERLERİ	
char	1	-128	127
unsigned char	1	0	255
int	2	-32.768	32.767
unsigned int	2	0	65.535
long	4	-2.147.483.648	2.147.483.647
unsigned long	4	0	4.294.967.296

1. char tipi

Tamsayı değerler alır. 8 bit uzunluğundadır. Bellekte 1 bayt yer kaplar. -128 ile +128 arasında değer alır. Unsigned : 0 ile 255 arasında değer alır.

2.int tipi

Tamsayı değerler alır. 16 bit uzunluğunda ifade edilirler. Bellekte 2 Bayt kaplarlar. -32768 ile +32767 arasinda değer alır. unsigned: 0 ile 65535 arasinda değer alır.

Örnek: int i; i değişkeni tam sayı türündendir. 3, -7 gibi değerler alabilir.

3. Long tip.
Tamsayı değerler alır. 32 bit uzunluğundadır. -2147483648 ile +2177483647 arasinda değer alır, unsigned: 0 ile 65535 arasinda değer alır.

Gerçek Tipler (Float, Double)

Gerçek (noktalı, küsüratlı) sayıları içerirler.
float : Bellekte 4 Byte yer tutar. 3.4x10-38 ile . 3.4x10+38 aralığında değer alır. Hassasiyet 7-8 basamaktır.
double : Bellekte 8 Byte ter tutar. . 1.7x10-308 ile . 1.7x10+308 aralığında değer alır. Hassasiyet 15-16 basamaktır.

Karakter tipi (char)

Char: Tek bir karakter değer alır. '0','1','2',... ' 'a', '8' , '%' , '+' , 'F' gibi değerler alır. Aynı zamanda 8 bit uzunluğunda tamsayı değerler de alabilir. Ancak bir harfi veya klavyeden basılacak bir tuşu değer olarak bir değişkene atamak istenirse char olarak tanımlanır.

Mantıksal veri tipi(Bool)
Bool: true(dogru) = 1 veya false(yanlis) = 0 değerini alır. Eski derleyiciler bu türü desteklemeyebilir. Yeni ANSI C++ standardında eklenmiştir.

String (Karakter dizisi) veri tipi

Bir ismi veya bir cümleyi bir değişkene atamak gerekirse bu değişken string olarak tanımlanır. Örnek: "ali", "bu gün hava güzel" gibi değerlerbu değişkene atanabilir. String veri tipini kullanabilmek için

#include <string>

kütüphane dosyasının programa eklenmesi gerekmektedir.

Değişkenlere ilk değerleri tanımlandıkları anda da verilebilir.

Örnekler:

int a=5;

float f=3.14;

string adsoy="Ali Demir";

bool onay=true;

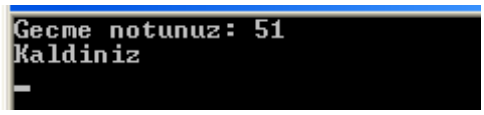
char ch='A';

Operatörler
Bir değişkene değer atamak, bir işlem yapmak veya iki değişkeni karşılaştırmak gibi işlemler yapmak için kullanılan '=', '+', '>' gibi sembollerdir.

C++ da kullanılan Operatörleri öncelikle türlerine göre şu şekildedir.
1) Aritmetiksel operatörler + , - , * , / , % , ++ , --
2) Karşılaştırma operatörleri < , > , <=, >= , ==, !=
3) Atama operatörleri = , += , -=, *= , /= , %= , <=, >>=, &=, != , ^=
4) Mantıksal Operatörler ! , || , &&
5) Bit bazında işlem yapan operatörler & , ! , ^ , ~

```
//Vize ve final notlarinindan geçme notunu hesaplama
#include <iostream.h>
#include <conio.h>
int main()
{
int vize, final, ort;
vize = 10;            //atama operatörü
final = 80;
ort = vize * 0.4 + final * 0.6;  //Çarpma ve toplama operatörü
cout<< "Gecme notunuz: " << ort;
if(ort>=60) cout<<"\nGectiniz \n";    //Karşılaştırma operatörü
if(ort<60) cout<<"\nKaldiniz\n";
getch();
return 0;
}
```

Şekil 2.3 Operatör kullanımı



Şekil 2.3 de operatörlerin kullanımı ile ilgili bir örnek program verilmiştir. Bu programda atama "=", çarpma "*" ve toplama"+" ile karşılaştırma ">=", "<" operatöreleri kullanılmıştır.

Artırma ve Azaltma operatörleri

"++" artırma operatörüdür. "--" azaltma operatörüdür. Önünde veya sonunda bulunduğu değişkenin değerini bir artırmak veya azaltmak için kullanılır.

```
// artırazalt.cpp
// x i bir arttırıp y yi bir azaltarak çarpar
#include <iostream.h>

main()
{
int x = 4;
int y = 8;
cout << "x = " <<x << endl;
cout << "y = " << y << endl;
cout <<"++x * --y = " << ++x * --y ;
return 0;
}
```

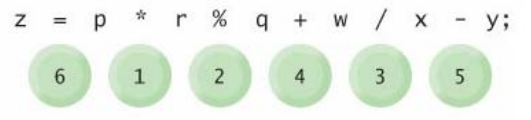
Programın ekran çıktısı

" endl " komutu satır sonunu belirterek yeni satıra geçmemizi sağlar, bir nevi "\n " sembolü gibi bir işleve sahiptir.

Seviye	Operatör	Tanım	Öncelik Yönü (associativity)
1	: :	Kapsam(scope)	
	( )	öncelik kazandırma ve fonksiyon çağırma	soldan sağa
	[ ]	indis operatörü (subscript)	
	.	yapı elemanına ulaşım (structure access)	
	->	yapı elemanına gösterici ile ulaşım	
2	+	işaret operatörü (unary)	sağdan sola
	-	işaret operatörü (unary)	
	++	1 artırma (increment)	
	--	1 eksiltme (decrement)	
	~	bitsel değil (bitwise not)	
	!	mantıksal değil (logical not)	
	*	içerik operatörü (indirection)	
	&	adres operatörü (address of)	
	sizeof	sizeof operatörü	
3	*	çarpma (multiplication)	soldan sağa
	/	bölme (division)	
	%	modulus (bölümden kalan)	
4	+	toplama (addition)	soldan sağa
	-	çıkarma (suntraction)	
5	<<	bitsel sola kaydırma (bitwise shift left)	soldan sağa
	>>	bitsel saga kaydırma (bitwise shift right)	
6	<	küçüktür (less than)	soldan sağa
	>	büyüktür (greater than)	
	<=	küçük eşittir (less than or equal)	
	>=	büyük eşittir (greater than or equal)	
7	= =	eşittir (equal)	soldan sağa
	!=	eşit değildir (not equal to)	
8	&	bitsel VE (bitwise AND)	soldan sağa
9	^	bitsel EXOR (bitwise EXOR)	soldan sağa
10	|	bitsel VEYA (bitwise OR)	soldan sağa
11	&&	mantıksal VE (logical AND)	soldan sağa
12	||	mantıksal VEYA (logical OR)	soldan sağa
13	?:	koşul operatörü (conditional operator)	sağdan sola
14	=	atama (assignement)	sağdan sola
	+=	işlemli atama (assignment addition)	
	-=	işlemli atama (assignment subtraction)	
	*=	işlemli atama (assignment multiplication)	

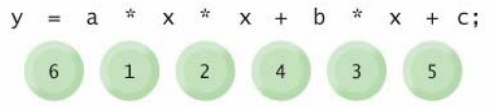
	/=	işlemli atama (assignment division)	
	%=	işlemli atama (assignment modulus)	
	<<=	işlemli atama (assignment shift left)	
	>>=	işlemli atama (assignment shift right)	
	&=	işlemli atama (assignment bitwise AND)	
	|=	işlemli atama (assignment bitwise OR)	
	^=	işlemli atama (assignment bitwise EXOR)	
15	,	virgül operatörü (comma)	

Operatörlerin öncelik sıralarına göre aşağıda verilen işlemi inceleyelim.



*,/, % operatörleri aynı önceliğe sahip olmalarına karşın, öncelik sırası soldan sağa doğru olduğundan Öncelikle p*r işlemi, daha sonra şekle göre 2 numaralı işlem(%) daha sonra bölme işlemi şeklinde verilen sırada işlemler gerçekleştirilecektir.

Bir başka örnek işlem sırasını inceleyin



Program Kontrol ve Döngü Komutları

if Komutu

Belirli bir şarta bağlı olarak bir işlem veya işlemlerin yapılmasını sağlamak amacı ile kullanılan karşılalştırma komutudur.

```
if (şart)
{
Komut1;
Komut2;
...
}
```

```
if ( şart )
{
komut1;
komut2;
...
}
else
{
komut1;
komut2;
...
}
```

Şeklinde bir kullanıma sahiptir. Örnek olarak notu 50 den büyük ise ekrana Geçtiniz yazan if komutu aşağıda gösterildiği gibidir.

```
if ( not >= 50 ) cout << "Geçtiniz!";
```

Notu 50 den buyuk ve 70 den kuçuk ise ekrana iyi değilse zayıf yazan if komutu aşağıdaki gibidir.

```
if ( not > 50  && not <70)
   cout << "iyi!";
else
   cout << "zayıf!";
```

```
//pozitif.cpp
main()
{
int x;
cout << "Bir sayı girin:" ;
cin>> x;
if ( x > 0 )
   cout << "\n Pozitif" ;
else
   cout<< "\n negatif"
return 0;
}
```

```
Bir sayi girin:2
Pozitif_
```

Switch-case Komutları

Switch Case deyimi bir çeşit if komutu gibidir. Kontrol değişkeninin aldığı değere göre istenen komutları işlemeye yarar. işlev bakımından if deyimine çok benzemektedir. Değişken değeri sabit1'e eşit ise komut1, . Değişken değeri sabit2'e

Eşit ise komut2 vb. İşlem görür. Hicbir komut işlem görmemiş ise default ifadesi altındaki komutlar işlem görecektir.

```
switch( Kontrol Değişkeni )
{
 case Sabit1 : komut1;  break;
 case Sabit2 : komut2;  break;
 .
 .
 .
  default : Komut;
}
```

switch örneği	if-else eşdeğeri

#include <iostream.h>
#include <conio.h>
int main()
{
int x;
cout<< " Bir sayı giriniz:";
cin>>x;
switch (x) {
  case 1:
  cout << "x değeri 1";
  break;
  case 2:
  cout << "x değeri 2";
  break;
  default:
  cout << "x 1 ve 2 den farkli ";
}
 	
getch();
return 0;
}
	.
	.
	.
	int x;
	cout<< " Bir sayı giriniz:";
	cin>>x;
	if (x == 1) {
	  cout << "x değeri 1";
	  }
	else if (x == 2) {
	  cout << "x değeri 2";
	  }
	else {
	  cout << "x 1 ve 2 den farklı ";
	  }
	.
	.
	.

```
Bir sayi giriniz:5
x 1 ve 2 den farkli _
```

Döngü Komutları

Bir ya da birden fazla komutun istenen sayıda veya belirli bir şart gerçekleşinceye kadar çalıştırılması istenirse döngü komutlarından yararlanılır.

For döngüsü,

while döngüsü,

do-while döngüsü

C++ da kullanılan döngülerdir.

For döngüsü

İşlemin kaç defa tekrar edeceği önceden belli ise for döngüsü kullanılır.

for (baslangıc; bitiş şartı; artım) Komutlar;

örnek: 1 den 100 e kadar olan for dongusu şu şekilde yazılmalıdır.

```
for (j=0; j<100; j++) Komutlar;
```

For döngüsü için aşağıdaki örnekleri inceleyiniz.

```
//toplam.cpp
//for döngüsüyle 1 den 100 e kadar
// sayıların toplamını bulan program.
#include <iostream.h>

int main()
{
int top;

top=0;

for (int j=1; j<=100; j++)
{
top=top+j;
}
cout<<"toplam = "<< top;
return 0;

}
```

```
toplam = 5050_
```

```
//factoriyel.cpp
//for döngüsüyle faktöriyel hesap.
#include <iostream.h>
int main()
{
int fak, sayi;

cout<<"Sayıyı giriniz: ";
cin>>sayi;
fak=1;
for (int j=1; j<=sayi; j++)
{
fak=fak*j;
}
cout<<"Sonuc: "<< fak;
return 0;
}
```

```
Sayiyi giriniz: 6
Sonuc: 720
```

While ve do-while döngüsü

While döngüsü, döngü sayısının belli olmadığı zamanlarda kullanılır.Aşağıa verildiği şekilde kullanılır.

```
while( Şart )
{
Komut
Komut
Komut
.
.
.
}
```

```
Do
{
Komut
Komut
Komut
.
.
.
}while( Şart );
```

While döngüsü, parantez içinde verilen şart doğru olduğu sürece altındaki komut veya komut bloğu yürütülür. Eğer yanlış ise kontrol bir sonraki komut veya komut bloğuna geçer. do-while döngüsünün while döngüsünden farkı döngünün devam edebilmesi için gerekli olan şart kontrolünün döngü bloğunun sonunda yapılmasıdır. Dolayısı ile döngü şart yanlış olsa bile bir defa icra edilecektir.

```
// whileornek.cpp
// Klavyeden girilen sayıya kadar olan
//sayıları toplar
#include <iostream.h>
main()
{
int x, y;
y= 0;
cout<< " Bir Sayı Giriniz : ";
cin>>x;
while (x>0)
{
y =y+x;
x--;
}
cout<< "Toplam= "<< y;
return 0;
}
```

```
// do-whileornek.cpp
// Klavyeden girilen sayıya kadar olan
//sayıları toplar
#include <iostream.h>
main()
{
int x, y;
y= 0;
cout<< " Bir Sayı Giriniz : ";
cin>>x;
do
{
y =y+x;
x--;
} while (x>0);
cout<< "Toplam= "<< y;
return 0;
}
```

Break ve continue Komutu

Döngülerden herhangi bir noktada çıkılmak istenirse break komutu kullanılır. Örneğin x değeri 5 olduğunda döngüden çıkmak istenirse İf(x==5) break; komutu kullanılabilir. Break komutu for döngüsü dahil bütün döngülerden çıkmayı sağlar. Bu durumda program akışı döngü bloğunun sonundan devam eder.

while(1) { }, for(;;) { } döngüleri sonsuz döngüyü ifade eder. Sonsuz döngüden break komutu ile çıkılır.

Continue komutu bir döngü içinde çalıştırılırsa, o döngü içinde bulunan tur sona erer ancak döngü devam eder. Diğer bir deyişle, gövdenin içinde bulunan continue komutundan sonra gelen cümleleri atlayarak, döngüyü devam ettirir.

Aşağıdaki örnekte ilk programda x=6 olduğunda döngüden tamamen çıkılmaktadır. İkinci programda x=3 olduğunda döngü bir sonraki adımdan devam etmektedir. x=3 olduğu andaki değeri ekrana yazılmayacaktır.

```
//break komutunun kullanımı
#include <iostream.h>
main()
{
  for (int x = 1; x <= 20; x++) {
  if (x == 6) break;
  cout<< x << " ";
  }
}
```

```
1 2 3 4 5
```

```
//continue komutunun kullanımı
#include <iostream.h>
main()
{
for(int x=0; x <= 9; x++)
{
  if(x ==3) continue;
  cout<< x<< " ";
}
}
```

```
0 1 2 4 5 6 7 8 9
```