

2. PROGRAMLAMANIN TEMELLERİ

Giriş

Çok basit bir program yazmak için bile, o dile ait bazı esasları bilmek gerekir. Bu bölümde C++ dilinin temel özelliklerinden bazıları tanıtılacaktır. Tanıtılacak esaslar, Temel program yapısı, değişkenler, giriş çıkış, açıklamalar, aritmetik operatörler, arttırma operatörü, veri dönüşümleri ve kütüphane fonksiyonlarıdır.

Bu bölümde ele alınacak konular zor değildir. Ancak, öğrenciler arasında C ve C++ dillerinin zor olduğu konusunda yaygın bir kanaat vardır. C++'ı tanıdıkça bu dil size daha az rahatsız edici görünecektir.

Bu ve bundan sonraki bölümlerde, programlama kavramları ele alınırken verilecek örnekleri çalıştırmak için DEV C++ derleyicisini kullanabilirsiniz. Derleyiciler, sizin yazacağınız kaynak kodları çalıştırılabilir bir dosya haline getirir. C++ 'da kaynak dosyalar, .cpp uzantılı metin dosyaları, çalıştırılabilir dosyalar ise, .exe uzantılıdır.

2.1. DEV C++ Kurulumu ve Kullanılması

Bu ve bundan sonraki bölümlerde, programlama kavramları ele alınırken verilecek örnekleri çalıştırmak için DEV C++ derleyicisini kullanabilirsiniz. Derleyiciler, sizin yazacağınız kaynak kodları çalıştırılabilir bir dosya haline getirir. C++ 'da kaynak dosyalar, .cpp uzantılı metin dosyaları, çalıştırılabilir dosyalar ise, .exe uzantılıdır. Aynı zamanda Dev C++ size kaynak dosyalarınızı oluşturabilmeniz için bir editör, hata ayıklayıcı ve benzeri birçok özellik sunar. Program geliştiricilere bu şekilde birçok kolaylık sağlayan yazılımlara **Tümleşik Geliştirme Ortamı**(*Integrated Development Environment- IDE*) denir.

C++ programlarını geliştirmek için piyasada ücretli veya ücretsiz birçok tümleşik geliştirme ortamı vardır. Örneğin Visual Studio, Eclipse, NetBeans, Dev C++ bunlardan birkaçıdır. Biz bu derste örnek uygulamalarımızı Dev C++ ortamında geliştireceğiz. Bu IDE 'yi seçmemizin nedeni, bu geliştirme ortamının sadeliği ve kolay olmasıdır.

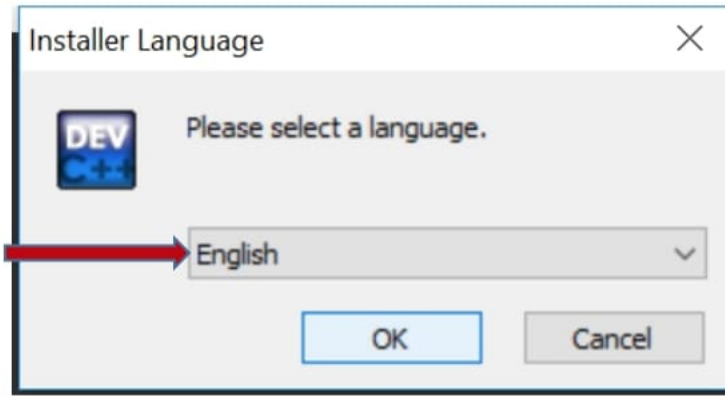
2.1.1. DEV C++'ın Kurulumu

İlk geliştirilmesi Bloodshed firması tarafından yapılan DEV C++, 2011 yılından sonra Orwell firmasına geçmiş ve bundan sonraki geliştirmeler bu firma tarafından yapılmıştır. İnternet üzerinden indirip bilgisayarımıza kurabileceğiniz DEV C++'ı <https://sourceforge.net/projects/orwelldevcpp/> adresinden indirebilirsiniz.

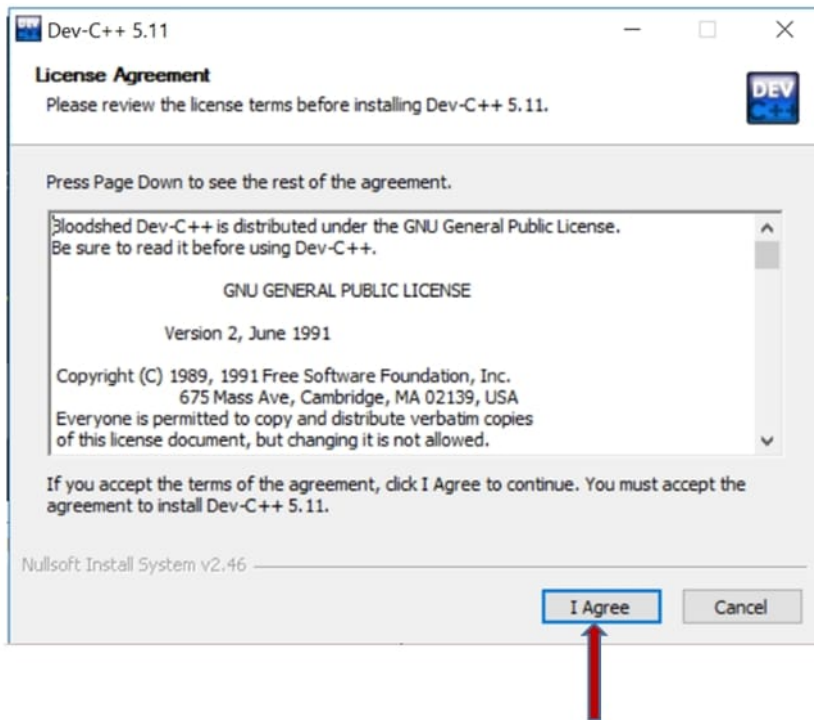
1. önce DEV C++'ı <https://sourceforge.net/projects/orwelldevcpp/> adresinden bilgisayarınıza indiriniz.



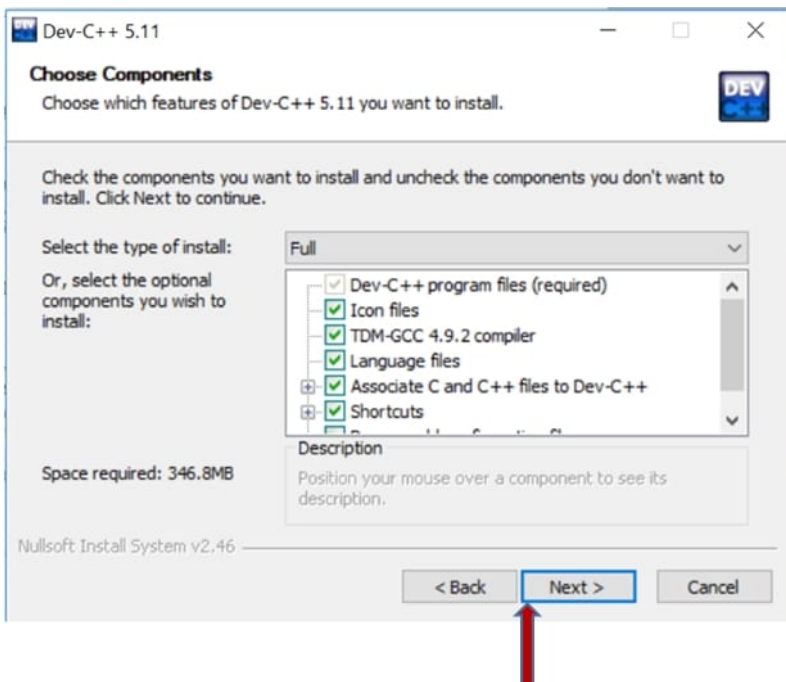
2. İndirilen dosyayı çalıştırın ve aşağıdaki pencereden dil seçimini yapınız.



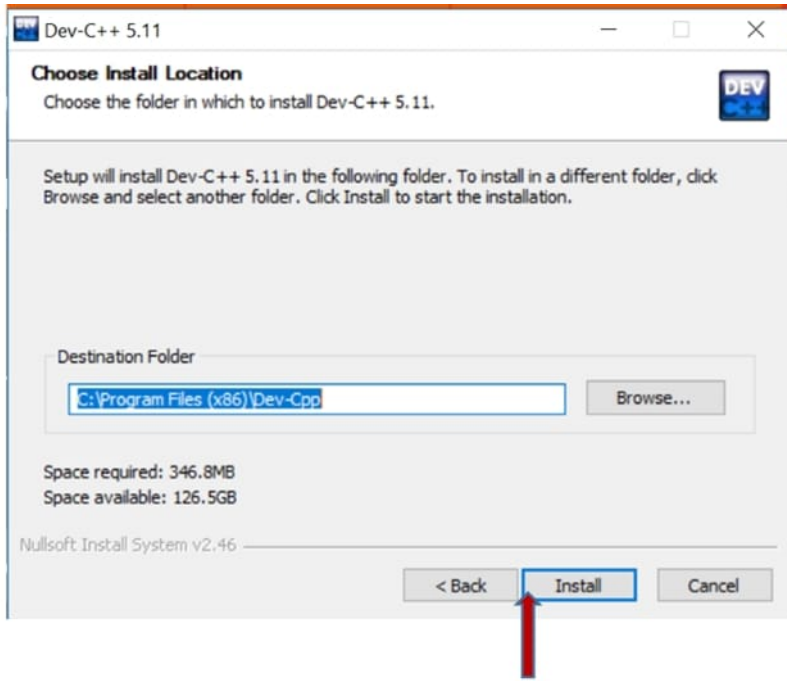
3. Sözleşmeyi okuyup onaylayınız.



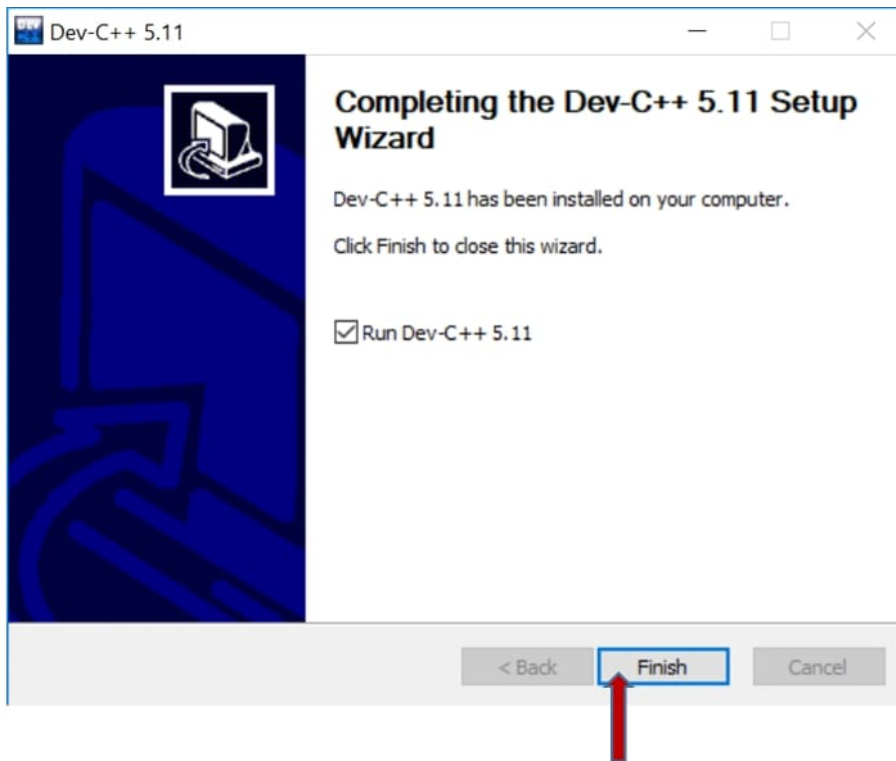
4. Kurulum tipini **Full** seçiniz ve **ileri** butonunu tıklayınız.



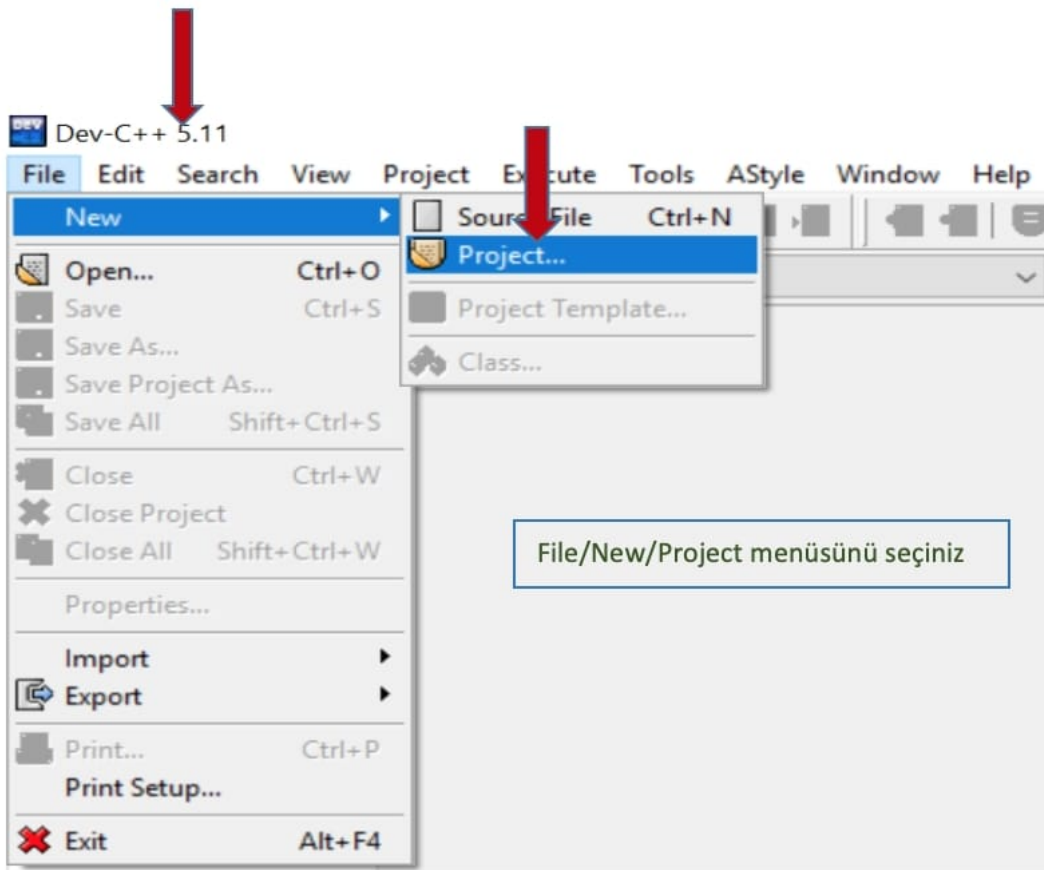
5. Hedef klasör seçiminde bir deęişiklik yapmadan **Install** butonuna tıklayınız.

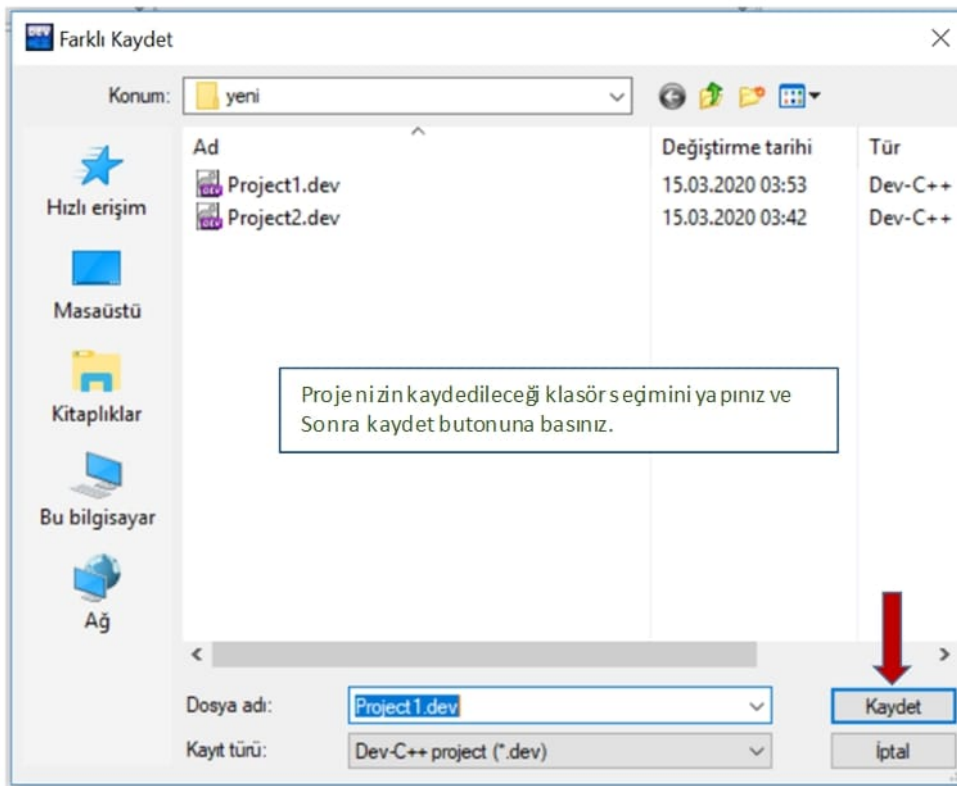
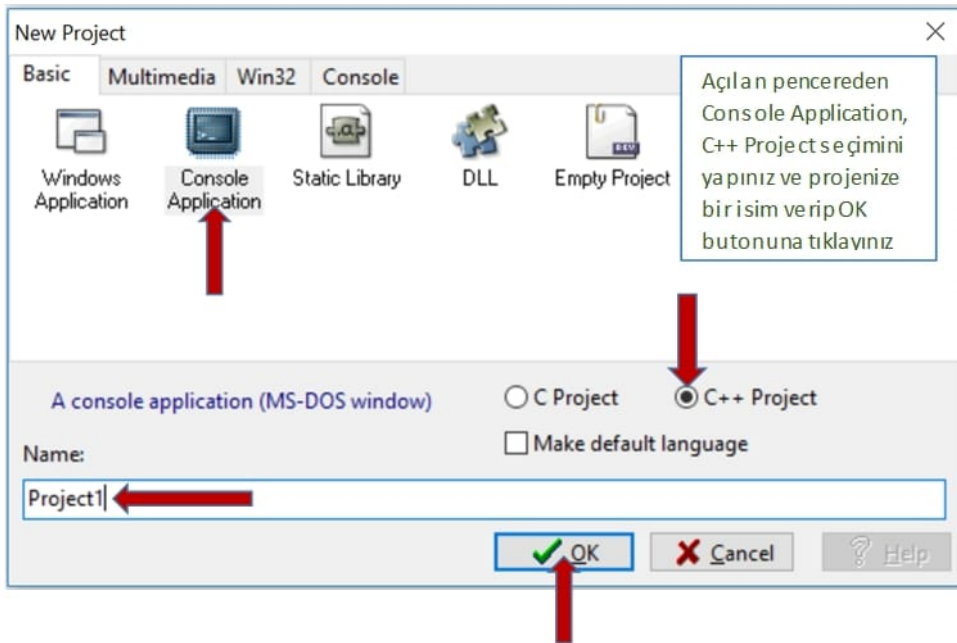


6. Kurulum tamamlandıktan sonra gelen pencereden **Finish** butonuna tıklayınız.



2.1.2. DEV C++ Kullanımı





File/New/Project menüsünü seçiniz

2.2. C++ Program Yapısı

Aşağıda verilen C++ programı, ekrana “*ilk C++ programı*” cümlesini yazar. Program basit olmasına karşılık, C++ programlama dili hakkında birçok bilgiyi içerir. Bu bölümde aşağıda verilen C++ programı ile ilgili ayrıntılar ele alınacaktır.

```
#include <iostream>

using namespace std;

int main()

{

    cout << "ilk C++ Programı
    \n";
```

2.2.1. Fonksiyonlar

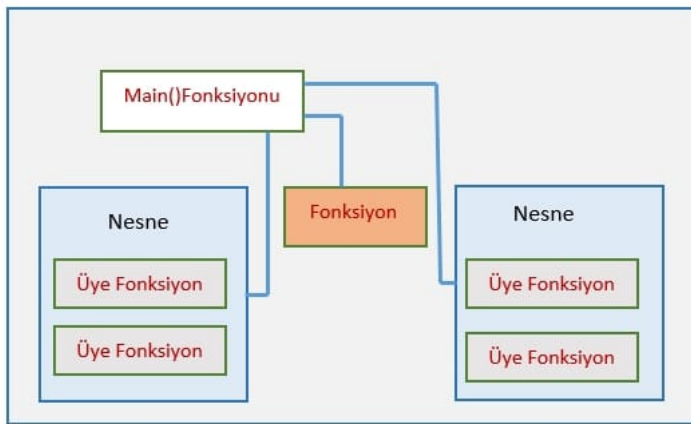
Fonksiyonlar C++'ın en temel bloklarından biridir. Örnek programımız hemen hemen main fonksiyonundan oluşmaktadır.

Her C++ programı, çalıştırıldığında main fonksiyonu içerisindeki ifadeler çalıştırılır. Eğer programda main isimli bir fonksiyon yoksa program çalıştırıldığında hata verir. main() kelimesinin peşinden gelen parantezler bir fonksiyonun ayırt edici özelliğidir. Fonksiyonda parantezlerin kullanılmaması, derleyicinin **main** ifadesini bir değişken veya başka bir program ögesi olarak yorumlamasına neden olur.

Fonksiyon isminden önce gelen *int* kelimesi, bu fonksiyonun *int* tipinde bir değer döndüreceğine işaret etmektedir.

2.2.2. Küme Parantezleri ve Fonksiyonun Gövdesi

C++'ta fonksiyon gövdesi küme parantezleri içine alınır. Bu parantezler diğer dillerdeki BEGIN ve END anahtar sözcüklerine karşılık gelir. Küme parantezleri program ifadelerinden oluşan bir bloğu çevreler ya da sınırlandırır. Her fonksiyonun gövdesi etrafında bu parantez çiftlerinden biri olmak zorundadır. Yukarıdaki fonksiyon gövdesinde yalnızca iki ifade bulunduğuna dikkat ediniz. C++ 'ta main fonksiyonu çeşitli nesnelerin fonksiyonlarını veya bağımsız fonksiyonları çağırmak için kullanılır. Bu durum aşağıda şekil 2.1'de gösterilmiştir.



Şekil 2.1 C++ 'ta main fonksiyonunun yapısı

2.2.3. C++ Program İfadeleri

Program İfadeleri C++ programlarının en temel birimleridir. Aşağıda iki tane ifade gösterilmiştir.

```
cout << "ilk C++ Programı \n";
```

```
return 0;
```

İfadeler bilgisayara ne yapması gerektiğini bildirir ve C++ kullanılan ifadelerin büyük çoğunluğu C'deki ifadelerle aynıdır.

Noktalı virgül(;), ifadenin bittiğini gösterir. Eğer ifadenin sonuna noktalı virgül konulması unutulursa derleme sırasında hata mesajı alınır.

Fonksiyon gövdesindeki en son ifade `return 0;` ifadesidir. Bu ifade `main()` fonksiyonuna, kendisini çağırana 0 değerini döndürmesini bildirir. C++'ın eski sürümlerinde, `main` fonksiyonunun dönüş tipi `void` olarak belirleniyordu ve `return` ifadesine gerek kalmıyordu. Ancak standart C++'ta bu doğru değildir.

2.2.4. C++ Programı İçerisinde Boşluklar

C++ derleyicisi bazı özel karakterleri dikkate almaz; boşluk, satır başı, satır atlama, sekmeler vb. Yukarıda verilen programı aşağıdaki şekilde yazıp tekrar çalıştırırsanız aynı sonucu elde edersiniz. Ancak, bu yazım şekli programın okunabilirliğini zorlaştırmaktadır. Bu nedenle aşağıdaki şekilde program yazılması tavsiye edilmez.

```
#include <iostream>

using namespace std;

int main ()

{ cout <<

    "ilk C++ Programı \n"

    ; return

0; }
```

Boşlukların derleyici açısından önemli olmadığı kuralının birkaç istisnası vardır. `#include` ile başlayan programın ilk ifadesi bir önışlemci direktifi olduğu için tek bir satırda yazılmak zorundadır. Bundan başka `"ilk C++ Programı \n"` gibi karakter katarı sabitleri de ayrı satıra bölünerek yazılmaz. Eğer uzun bir karakter katarı sabitine ihtiyacınız varsa, satır sonuna ters bölü işareti (`\`) ekleyebilir veya karakter katarını, tırnakla çevrilmiş iki ayrı karakter katarına bölebilirsiniz.

2.2.5. Açıklamalar

Hangi program olursa olsun açıklamalar programın önemli bir parçasıdır. Açıklamalar programı yazana kolaylık sağladığı gibi kaynak dosyanın okunabilirliğini artırır. Okunabilir kaynak program dosyalarına daha sonra bakım yapmanın maliyeti düşük olur. Açıklamaları derleyici dikkate almaz. Açıklamaların dosya büyüklüğüne etkisi yoktur ve çalışma hızını etkilemezler.

Aşağıda C++ programlarında açıklamaların nasıl yapılacağı gösterilmiştir.

Tek satırda yapılacak açıklama:

..

Birden fazla satırda yapılacak açıklama:

```
/* .....
..... */
```

```
#include <iostream> //ön işlemci direktifi
using namespace std; //using direktifi
/* Bu program ekrana
   İlk C++ Programı” yazar */
int main()
{
    cout << “ilk C++ Programı \n|”;
    return 0;
}
```

Program

içerisinde

Açıklama satırı

kullanılmasına

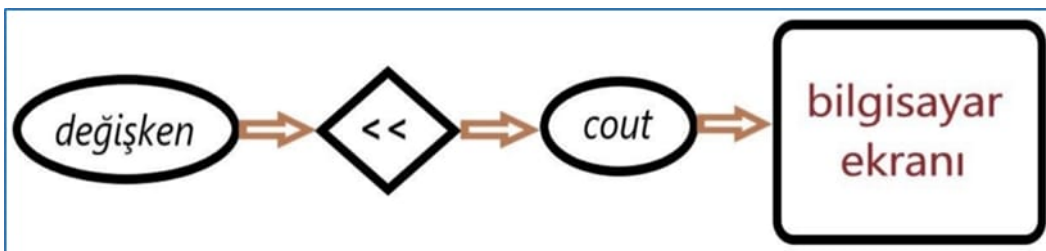
Örnek vermek için

2.2.6. C++ Programlama Dilinde *cout* Kullanarak Ekran Çıktısı Almak

Cout, aslında bir nesnedir. C++’ta önceden tanımlıdır ve standart çıktı akışına (standart output stream) karşılık gelir.

<< operatörüne ekleme operatörü denir.

<< operatörü, sağ taraftaki değişkenin değerini sol taraftaki nesneye yönlendirir.



Şekil 2.2. *cout* nesnesinin çalışma şekli

2.2.7. Karakter Katarı Sabitleri

Tırnak içerisinde yer alan “ilk C++ Programı \n “ deyimi karakter dizini sabitlere bir örnektir. Bilindiği üzere değişkenlerden farklı olarak sabitler, program çalıştığında yeni bir değer almazlar. Sabit değeri program yazılırken belirlenir ve program çalıştığı sürece sabitler değerlerini korur.

Karakter dizininin sonundaki “\n” karakterleri, bir satır atlatılır. Burada, “\n” ekran çıktısının bazı derleyicilerin program sonunda verdiği mesajlarla karışmaması için konulmuştur.

2.3. Direktifler

Yukarıdaki programın başında yer alan ilk iki satır birer direktiftir. Bunlardan ilki ön işlemci direktifi ikincisi ise using direktifidir. Direktifler temel C++ programının bir parçası değildir ama gereklidirler.

2.3.1. Ön İşlemci Direktifleri

Programdaki `#include <iostream>` bir ön işlemci direktifidir.

Ön işlemci, derleyiciye verilen bir komuttur. Derleyicinin ön işlemci denilen bir parçası gerçek derleme işlemi başlamadan önce bu direktiflerle ilgilenir.

Ön işlemci direktifi olan `#include` derleyiciye, kaynak dosyası içine başka bir dosya eklemesi gerektiğini bildirir.

Programın derleme işlemi tamamlandığında `#include` direktifinin yerine belirtilen dosya içeriği yerleştirilir.

`#include`, pek çok ön işlemci direktifinden yalnızca bir tanesidir. Ön işlemci direktiflerinin kullanımı C++'da C kadar yaygın değildir.

2.3.2. using Direktifi

Bir C++ programı değişik isim uzaylarına(namespace) bölünebilir. İsim uzayı, programdaki belirli isimlerin derleyici tarafından tanınmasını sağlayan program parçasıdır. İsim uzayının dışında bu isimler tanınmaz.

```
using namespace std;
```

Bu direktifi takip eden tüm program ifadelerinin, `std` isim uzayında olduğunu belirtilir.

`cout` bu isim uzayında tanımlıdır. Eğer `using` direktifi kullanılmamış olsaydı her `cout` kullanıldığında `cout` 'un başına `std` kelimesini eklemek gerekecekti.

```
std::cout<<"ilk C++ Programı \n .";
```

Defalarca `std::` yerine `using` direktifini bir defa yazmak bir çok açıdan daha yararlıdır.

2.3.3. #define Direktifi

C++ programlama dili de kullanımı tavsiye edilmemekle birlikte sabit tanımları `#define` ile tanımlanabilir. `const`'an farklı olarak `#define` ile yapılan sabit tanımlarında tip belirtilmez.

```
#define pi=3.141618
```

2.3.4. const Niteleyici

Sabit tanımlamak için kullanılır ve değişkenin değerinin program sonuna kadar değişmeyeceğini bildirir. Bu tür değişkenlerin içeriğini değiştirme girişiminde bulunulduğunda hata mesajı ile karşılaşılır.

```
const float pi=3.141618;
```

2.3.5. Başlık Dosyaları

`#include` ön işlemci direktifi derleyiciye programı derlemeden önce *iostream* dosyasını programa eklemesini bildirir. *iostream*, en temel giriş/çıkış işlemlerinden sorumludur. *cout* ve `<<` operatörü için gerekli tanımları içerir. Bu tanımlar olmadan derleyici, *cout* 'u tanıyamaz. Buna benzer birçok başlık dosyası vardır. Güncel C++ başlık dosyalarından dosya uzantısı kaldırılmıştır. Ancak, C günlerinden kalma bazı eski başlık dosyaları.h uzantısı içerir. *iostream* içerisinde ne olduğunu merak ediyorsanız derleyicinin *include* dizininde bu kaynak dosyayı *notepad* ile açıp inceleyebilirsiniz.

2.3.6. iomanip Başlık Dosyası

endl manipülatörü dışındaki diğer manipülatörlerin tanımları, *iostream* başlık dosyasında değil, *iomanip* isimli başka bir başlık dosyasındadır. Bu durumda, *setw* manipülatörünü kullandığımızda *iomanip* başlık dosyasını da programımıza eklemek zorundayız. Yukarıdaki programda *setw* manipülatörünü kullanmak için *iomanip* programa ilave edilmiştir.

2.3.7 setw Manipülatörü

Ekrana basılacak karakterlerin belirli bir formatta n karakterlik bir alanda yazılmasını sağlar *setw(n)*, Burada n *setw(n)* fonksiyonunun argümanıdır.

setw(10) kendisinden sonra gelen değişkeni 10 karakterlik alanda yazdırır. Sayılar sağa, metinler sola dayalıdır.

```
#include <iostream>

#include <iomanip> // setw için

using namespace std;

int main()
{
    int not1=50, not2=67,not3=80;

    cout<<setw(8)<<"Adi"<<setw(8)<<setw(8)<<"notu"<<endl;

    cout<<setw(8)<<"===== "<
<endl;

    cout<<setw(8)<<"Aysegül"<<setw(8)<<not1<<endl;

    cout<<setw(8)<<"Hasan"<<setw(8)<<not2<<endl;

    cout<<setw(8)<<"Abdullah"<<setw(8)<<not3<<endl;

    cout<<setw(8)<<"===== "<
<endl<<endl;

    return 0;

}
```

ö	ğ	r	e	n	c	i		p	u	a	n				
A	y	ş	e	g	ü	L								5	0
H	a	s	a	n										6	7
A	b	d	u	l	l	a	h							8	0

setw

manipülâtörünün çıktıya etkisi

2.4. Değişkenler

Değişkenler, programlama dillerinin en temel parçalarıdır. Her değişkenin sembolik bir ismi vardır. Değişkenler bilgisayarın belleğinde belirli bir bölgede yerleşiktir. Bir değişkene bir değer verildiğinde bu değer gerçekte, bellekte o değişken için ayrılmış olan yere atanır. Programlama dilleri hemen aynı değişken tiplerini kullanır. Örneğin tamsayılar, kayan noktalı sayılar, karakterler vb.

2.4.1. Tamsayı Değişkenler

Tamsayıların kesirli kısmı bulunmaz. Beş(5) kavramını tam sayı kullanarak ifade edebiliriz, fakat beş buçuk(5.5) kavramını tamsayı tipinde değişken kullanarak ifade edemeyiz. Bu tür sayıları ifade edebilmek için farklı veri tipleri vardır.

Tamsayı değişkenler farklı büyüklüklerde olabilir. En yaygın olarak kullanılan *int* tipidir. Tamsayıların bellekte kapladığı yer sisteme bağlı olarak değişmektedir. Örneğin 32 bit Windows sistemlerde bir *int* tipindeki değişken 4 bayt yer kaplar. Bu, *int* tipindeki bir değişken ile -2,147,483,648 - 2,147,483,647 Aralığındaki sayıları saklayabileceğimizi gösterir. *int* tipi sayılar Windows işletim sisteminin ilk sürümlerinde 2 bayt yer kaplamaktaydı. Aşağıda *int* tipinde birkaç değişken tanımlayan ve kullanan bir program verilmiştir.

```
#include <iostream>

using namespace std;

int main()

{

    int sayi1;

    int sayi2;

    sayi1=20;

    sayi2=Sayi1+10;

    cout << "sayi1 +10 =";

    cout << sayi2 << endl;

    return 0;

}
```

Tamsayı tipindeki değişkenlerin C++'ta kullanılması

```
İnt sayi1;

İnt sayi2;
```

Yukarıdaki ifadelerle ile *sayi1* ve *sayi2* adında iki tamsayı değişken tanımlanır. *int* anahtar kelimesi değişkenin tipini belirtir. Deklarasyonlar(declarations) olarak adlandırılan bu ifadeler, diğer ifadelerde olduğu gibi noktalı virgül ile sona ermelidir.

Bir değişken kullanılmadan önce deklare edilmelidir. Değişken deklarasyonunu programın herhangi bir yerinde yapabilirsiniz.

Deklarasyon, bir değişkenin ismini ve tipini programa tanıtır. Bir deklarasyon değişken için bellekte yer de ayırıyorsa aynı zamanda tanımdır. Biz daha çok hem deklarasyon hem de tanım olan ifadelerle ilgileneceğiz. Yukarıdaki verilen programda,

```
İnt sayi1;

İnt sayi2;
```

ifadeleri hem deklarasyon hem de tanımdır.

2.4.2. Değişken İsimleri

Bir C++ programında kullanılacak olan değişkenlerin programın başında, yürütülen deyimlerden önce tanımlanmaları gerekir.

Değişken tanımlanması,

a) bilgisayar belleğinde o değişkenler için yer ayrılmasını sağlar,

b) ilgili bellek gözlerinde sembolik isimler verilmesini sağlar. Değişken İsimleri:

- Harf, rakam ve alt çizgi '_' içerirler.
- Harf veya '_' ile başlamak zorundadır. '_' ile başlaması önerilmez.
- Büyük/küçük harf ayrımı vardır.
- C dilinin anahtar sözcükleri değişken ismi olarak kullanılamaz.
- Özel karakterler ve Türkçe 'ye özgü harfler (ş,ğ,ü,ı,ç) kullanılamaz.

Geçerli isimler:

a1, sayi, sayi_3, ilkDeger, SonSayi, a1c32, deneme_degeri

Geçersiz isimler:

1a1, değer, int

2.4.3. Değer Atama

```
Sayi1=10;
```

```
Sayi2=20;
```

Yukarıda verilen ifadeler değişkenlere değer atar. Eşit işareti(=) tahmin edilebileceği gibi, sağdaki değer soldaki değişkene atanmasını sağlar.

2.4.4. Tam Sayı Sabitler

Sabitler program boyunca değerleri değiştirilmeyen, yani sabit kalan değişkenler olarak ta düşünülebilir.

2.4.5. Çıktı Varyasyonları

```
cout<< "Sayi1 +10 eşittir ";
```

İfadesi, önceden de açıkladığımız gibi, bir karakter katarı sabitini ekrana yazdırır.

```
cout<< sayi2<<endl
```

İfadesi, sayi2 değişkeninin değerini ekrana yazdırır.

Dikkat edilecek olursa cout ve ekleme(<<) operatörü, tamsayı ve karakter katarı sabitlerini farklı şekilde ele alır. Bu operatörlere karakter katarı gönderilirse, karakter katarını olduğu gibi ekrana basar. Tamsayı gönderilirse, ekrana tamsayının değerini basar.

2.4.6. endl Manipülatörü

endl manipülatörü çıktıda satır atlmasını sağlar. Bu şekilde, bunu takip eden metin bir sonraki satırda görüntülenir. endl manipülatörü \n karakterleri ile aynı etkiye sahiptir ve bir manipülatör örneğidir. Manipülatörler çıktı akışını değişik yollardan değiştiren komutlardır.

2.4.7. Diğer Tamsayı Tipleri

int tipinin yanı sıra iki tane tam sayı tipi daha vardır. Bunlar long, short tipleridir. int tipinin büyüklüğünün sisteme bağlı olduğunu daha önce belirtmiştik. Ancak long ve short tamsayı tipleri hangi sistemde kullanılırsa kullanılsın aynı büyüklüğe sahiptirler. long her zaman bellekte 4 byte yer kaplar, short tamsayı tipi ise bellekte 2 Bayt yer kaplar.

2.4.8. Karakter Değişkenleri

char tipi -128 ile 127 arasındaki tamsayıları saklar. Bu tipteki değişkenler bellekte 1 byte (8bit) yer kaplar ve daha çok ASCII karakterleri saklamak için kullanılır. Karakter sabitlerinde bir karakter etrafında tek tırnak işareti kullanılır. Söz gelişi, 'a', 'b' gibi (Bunun çift tırnak kullanan karakter katarı sabitlerinden farklı olduğuna dikkat etmelisiniz) Aşağıdaki program bazı karakter sabitlerine ve karakter değişkenlerine örnekler içerir:

```
#include <iostream>

using namespace std;

int main()
{
    char karakter1 = 'A';

    char karakter2 = '\t';

    cout << karakter1;

    cout << karakter2;

    karakter1 = 'B';

    cout << karakter1;

    cout << '\n';

    return 0;

}
```

char tipindeki değişkenlere de tanımlandıkları anda ilk değer atanabilir. Bu programda char tipindeki iki değişkene ilk değer atanmıştır.

Kaçış Sekansı	Karakter
<code>\a</code>	<i>Zil</i>
<code>\b</code>	<i>Backspace</i>
<code>\f</code>	<i>Formfeed</i>
<code>\n</code>	<i>Newline</i>
<code>\r</code>	<i>return</i>
<code>\t</code>	<i>sekme</i>
<code>\\</code>	<i>Ters bölü</i>
<code>\'</code>	<i>Tek tırnak</i>
<code>\''</code>	<i>Çift Tırnak</i>

Kaçış Sekansları

2.4.9. Kayan Noktalı Temel Veri Tipleri

Kayan noktalı değişkenler, sayıları ondalık kısmıyla birlikte simgeler. Bu tür sayıların hem tam kısmı hem de ondalık kısmı vardır.

C++'da kayan noktalı değişkenler; float, double, long double şeklindedir.

float tipi: 3.4×10^{-38} ile 3.4×10^{38} arasındaki sayıları saklar. Ondalık kısmı noktadan sonra en fazla 7 basamak uzunluğunda olabilir. Bellekte 4 byte kaplar.

double ve long double : float' a göre bellekte daha fazla yer kaplarlar ve daha geniş değer aralığını kapsarlar. 8 byte yer kaplarlar ve noktadan sonra 15 basamak ondalık kısma sahiptirler.


```
#include<iostream>

using namespace std;

int main()

{

    float yaricap;

    const float PI=3.14159F;

    cout<<"Yari çap Giriniz...";

    cin>>yaricap;

    cout<<endl;

    float alan=PI*yaricap*yaricap;

    cout<<alan<<endl;

    return 0;

}
```

2.4.10. Çoklu Tanımlama

```
int not1=50, not2=67,not3=80;
```

Her üç değişken tek satırda bir tek int anahtar kelimesi ile tanımlanmıştır. Bu şekilde yapılan tanımlamalarda değişkenler birbirinden virgül ile ayrılır. Bu şekilde yapılan tanımlamalara çoklu tanımlama denir.

2.4.11. Değişken Tiplerinin Özeti

Temel C++ Değişken Tipleri

Anahtar Kelime	Alt Sınır	Üst Sınır	Maks. Ondalık kıs.	Bellek Alanı (byte)
bool	false	true	yok	1
char	-128	127	yok	1
short	-32,768	32,767	yok	2
int	-2,147,483,648	-2,147,483,647	yok	4
long	-2,147,483,648	-2,147,483,647	yok	4
float	3.4 x10 ⁻³⁸	3.4 x 10 ³⁸	7	4
double	1,7 x 10 ⁻³⁰⁸	1.7 x 10 ³⁰⁸	15	8

İşaretsiz (unsigned) Tamsayı Tipleri

Anahtar Kelime	Alt Sınır	Üst Sınır	Maks. Ondalık kıs.	Bellek Alanı (byte)
unsigned char	0	255	yok	1
unsigned short	0	65,535	yok	2
unsigned int	0	4,294,967,295	yok	4
unsigned long	0	4,294,967,295	yok	4

Aşağıda verilen ve işaretli-işaretsiz sayıların sınanması için yazılan programı inceleyiniz.

```
//işaretli ve işaretsiz tamsayıların sınanması

#include<iostream>

using namespace std;

int main()

{

    int sayi1=1500000000;

    unsigned int sayi2=1500000000;

    sayi1=(sayi1*2)/3;

    sayi2=(sayi2*2)/3;

    cout<<"işaretli sayi : "<<sayi1<<endl;

    cout<<"işaretsiz sayi : "<<sayi2<<endl;

    return 0;

}
```

2.4.12. Tip Dönüşümü

C++, C 'de olduğu gibi, birkaç veri tipi içeren deyimleri ele alırken diğer dilleri oranla esnek davranır. Aşağıdaki programı inceleyiniz:

```
//Tip dönüşümü-Karışık tipler

#include<iostream>

using namespace std;

int main()

{

    int sayac=7;

    float kutle=155.5F;

    double toplamkutle=sayac*kutle;

    cout<<"Toplam Kutle :"<<toplamkutle<<endl;


    return 0;

}
```

Programda int tipindeki değişken float tipindeki bir değişkenle çarpılıyor ve sonuç double bir değişkene atanıyor. Birçok programlama dili bu tür işlemlere sınır koyar. Ancak C++ bu satırı hatalı olarak işaretlemeyebilir ve program çalışır.

Bazı durumlarda tip dönüşümleri otomatik olarak gerçekleşir. Derleyici karışık tipli deyimlerle yüz yüze geldiğinde tipler, aşağıdaki tabloda gösterilen sıralamaya göre ele alınır.

Veri Tiplerinin Sıralanması

Veri Tipi	Sıralama
long double	En Yüksek
double	
float	
long	
int	
short	
char	En Düşük

+ ve * gibi aritmetik operatörler aynı tip operandlar üzerinde işlem yaparlar. Aynı deyim içerisinde farklı tipte iki operand karşı karşıya gelirse, düşük tipteki değişken yüksek tipteki değişkenin tipine dönüştürülür ve işlem sonra yapılır.

2.4.13. cin İle Giriş Almak

cin anahtar kelimesi standart giriş akışına karşılık gelen, C++ da önceden tanımlı bir nesnedir. >> operatörü, extraction(çıkarma) operatörüdür. Solundaki akış nesnesinden değeri alır, sağındaki değişkene yerleştirir. Aşağıdaki, programda cin nesnesi ve çıkarma operatörünün kullanımı gösterilmiştir.

```
#include <iostream>
```

```
using namespace std;

int main()

{

    int ftemp;

    cout << "fahrenheit sicaklik Giriniz : ";

    cin >> ftemp;

    int ctemp= (ftemp-32) * 5/9;

    cout << " derece : " << ctemp << endl;

    return 0;

}
```

2.4.14. Kullanım Anında Değişken Tanımlama

Yukarıdaki programda *ctemp* değişkeni programın en başında değil, kullanılacağı yerde tanımlanmıştır. Önceden de belirtildiği gibi C++ ta değişkenler programın başında tanımlanabildiği gibi kullanılacağı yerde de tanımlanabilir. Değişkenleri kullanıldığı yerde tanımlamak program listesini okumayı kolaylaştırır.

2.4.15. Basamaklanmış ekleme (<<) Operatörleri

Yukarıdaki programda ikinci *cout* 'a dikkat ederseniz bu satırda birden fazla << operatörü kullanılmıştır. Bu satır, birden fazla *cout* kullanılarak yapılabileceği gibi yukarıdaki gibi aynı ifade içerisinde birden fazla ekleme operatörü yazarak da yazılabilir. Bu şekilde yazılmış ifadeler Basamaklanmış Ekleme Operatörü denir.

2.4.16. Öncelikler

$(ftemp-32) * 5/9$

Deyimindeki parantezlere dikkat edin. Parantezler olmasaydı * operatörü – operatöründen daha yüksek önceliğe sahip olduğu için ilk önce çarpma işlemi yapılacaktı. Parantezler olduğu için ilk önce çıkarma sonra çarpma işlemi yapılacaktır. * ve / operatörleri arasındaki öncelik ise, * ve / aynı öncelik derecesine sahip olduğu için sol taraftaki işlem, yani önce çarpma işlemi yapılacaktır. İşlem öncelikleri ile ilgili kuralların matematikteki öncelik sırasına uygun olduğuna dikkate edin.

2.5. Operatörler

2.5.1. Aritmetik Operatörler

C++'ta dört tane normal aritmetik operatör kullanılır. Toplama için +, çıkarma için -, çarpma için * ve bölme için / kullanılır. Bu operatörler hem tamsayı hem de kayan noktalı sayı tiplerine uygulanır. Kullanımları hemen hemen diğer dillerdeki kullanımları ile aynıdır. Bununla birlikte kullanımları çok açık olmayan başka operatörler de vardır.

Yukarıda sayılan 4 operatörün dışında sadece tamsayı değişkenler üzerinde işlem yapan beşinci bir operatör daha vardır. Buna kalan operatörü (Modül operatörü) denir ve % işareti ile simgelenir. Bölme işleminden kalanı hesaplar. Aşağıdaki programı inceleyiniz.

```
#include<iostream>

using namespace std;

int main()

{

    cout <<" "<< 6%8 << endl

    <<" "<< 7%8 << endl

    <<" "<< 8%8 << endl

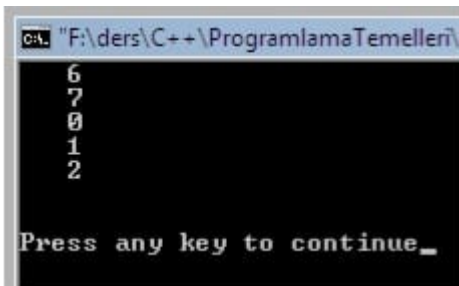
    <<" "<< 9%8 << endl

    <<" "<< 10%8 << endl <<endl << endl ;

    return 0;

}
```

Programın çalıştırılması sonucunda elde edilecek çıktı aşağıda verilmiştir.

A screenshot of a Windows command prompt window. The title bar shows the path "F:\ders\C++\ProgramlamaTemelleri\". The command prompt shows the output of a C++ program: the numbers 6, 7, 0, 1, and 2 are printed on separate lines. At the bottom, it says "Press any key to continue_" with a cursor.

2.5.2. Aritmetik Atama Operatörleri

C++ yazdığınız kodun daha kısa ve daha okunabilir yazabilmek için birçok yöntem sunar. Bu yöntemlerden birisi aritmetik atama operatörüdür. C++'ta beş adet aritmetik atama operatörü vardır.

```
toplam=toplam+sayi;
```

İfadesini atama operatörü kullanarak aşağıdaki gibi yazabiliriz.

```
toplam +=sayi;
```

Aşağıdaki programı ve çıktısını inceleyiniz.

```
#include<iostream>

using namespace std;

int main()

{

    int sayi=30;

    sayi +=10;

    cout << sayi<<endl;

    sayi-=5;

    cout << sayi<<endl;

    sayi*=2;

    cout << sayi<<endl;

    sayi/=3;

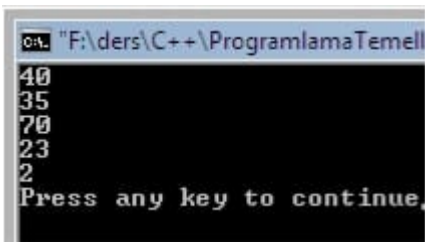
    cout << sayi<<endl;

    sayi%=3;

    cout << sayi<<endl;

    return 0;

}
```



```
40
35
70
23
2
Press any key to continue.
```

2.5.3. Arttırma ve Azaltma Operatörleri

Yukarıda bir değişkenin değerini bir arttırmak için çeşitli yollar açıkladık. Fakat bu işlemi daha kısa yoldan yapmanın yolu arttırma operatörünü kullanmaktır. Arttırma operatörü yanında bulunduğu değişkenin değerini bir arttırır. Azaltma operatörü ise yanında bulunduğu değişkenin değerini bir azaltır. Aşağıdaki örneği dikkatlice inceleyiniz. Arttırma veya eksiltme operatörünün, değişkenin önünde veya sonunda olmasının sonuca nasıl etki ettiğini tartışınız.

```
#include <iostream>
using namespace std;
int main() {
    int a,b,c,d=10,e=10,f=10,g=10;
    a=++d; //önce a'nın değeri bir artar sonra atama işlemi yapılır.
    cout<<" a=++d > "<<a<<endl;
    a=e++; //Önce atama işlemi yapılır. Sonar a'nın değeri bir artar.
    cout<<" a=e++ > "<<a<<endl<<endl;

    a=--f; //Önce f'nin değeri bir eksilir Sonra atama işlemi yapılır.
    cout<<" a=--f > "<<a<<endl;
    a=g--; //Önce atama işlemi yapılır. Sonra g'nin değeri bir azalır.
    cout<<" a=g-- > "<<a<<endl;
    return 0;
}
```

Yukarıdaki programın ekran çıktısı aşağıdaki gibi olur:

```
a=++d > 11    a=--f > 9
a=e++ > 10    a=g-- > 10
```

2.5.4. Karşılaştırma Operatörleri(İlişkisel Operatörler)

Karşılaştırma operatörleri genellikle karar yapılarında ve döngülerde kullanılır ve iki sayıyı ve iki karakteri karşılaştırır. C++'ta altı adet karşılaştırma operatörü(ilişkisel operatör) vardır.

Küçüktür Operatörü (<)

Büyüktür Operatörü (>)

Küçük Eşittir Operatörü (<=)

Büyük Eşittir Operatörü (>=)

Eşit Operatörü (==)

Eşit Değil Operatörü (!=)

2.5.5. Mantıksal Operatörler

İki veya daha fazla koşul karşılaştırılırken kullanılan operatörlerdir.

Operatör

Operatörün İşlevi

Ve Operatörü (&&) Karşılaştırılan ifadelerin tamamı doğru ise bir aksi halde sıfır döndürür

Veya Operatörü (||) Karşılaştırılan ifadelerden biri doğru ise bir aksi halde sıfır döndürür

Değil operatörü (!) Kendisinden sonra gelen ifadenin tersini alır. Bir ise sıfır, Sıfır ise bir döndürür.

Bölüm Özeti

Bu bölümde, dönem boyunca, bu ders kapsamında C++ programlama dili ile geliştireceğimiz programlar için ihtiyaç duyacağımız editör, derleyici, hata ayıklayıcı vb. özellikler sunan tümleşik geliştirme ortamı(IDE) sunan DEV C++ aracını nasıl temin edeceğimizi ve bilgisayarlarımıza nasıl kuracağımızı öğrendik.

Buna ek olarak, C++ programlarının büyük kısmının fonksiyonlardan oluştuğunu ve en temel fonksiyonun main() fonksiyonun olduğunu öğrendik ve basit bir C++ programı yazarak, main fonksiyonunun yapısını inceledik.

Bölümün devamında, program ifadelerini, cout kullanımını, ekleme operatörünü, cin kullanımını, çıkarma operatörünü ve direktifleri inceledik. Okunabilirliği arttırabilmek için, program içerisine açıklama satırları yazılması gerektiğini öğrendik.

Daha sonra bir programlama dilinin en temel parçası olan değişkenleri, sabitleri, C++'ta kullanılan temel veri tiplerini ve operatörleri inceledik.

H. Burak Tungut, Algoritma ve Programlama Mantığı, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2019.

Duygu Arbatlı Yağcı, Nesne Yönelimli C++ Programlama Kılavuzu, Alfa Basım Yayın Dağıtım Ltd. Şti, 2016.

G. Murat Taşbaşı, C programlama, Altaş yayıncılık ve Elektronik Tic. Ltd. Şti. 2005.

Muhammed Master, Süha Eriş, C++, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2012.

Sabahat Karaman, Pratik C++ Programlama, Pusula Yayınevi,2004