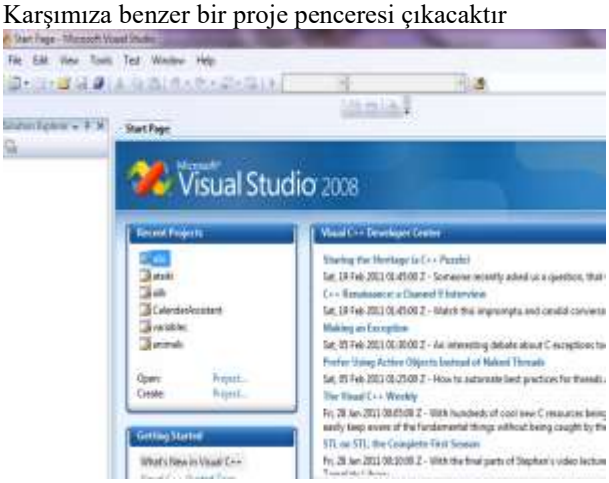
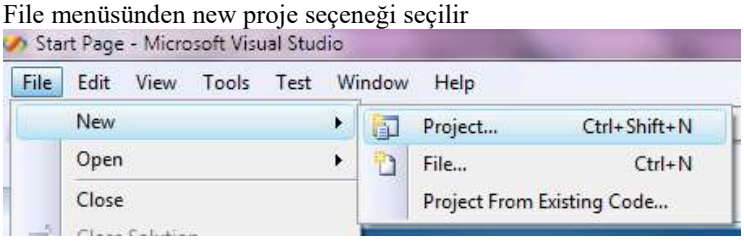
2017 YILI İSTATİSTİK BÖLÜMÜ 2. SINIF VİSUAL C, C++ C+ //aliatalay.net

Visual 2008 yükleme
Projeyi oluşturmak için visual studio simgesine tıklanır

Karşımıza benzer bir proje penceresi çıkacaktır

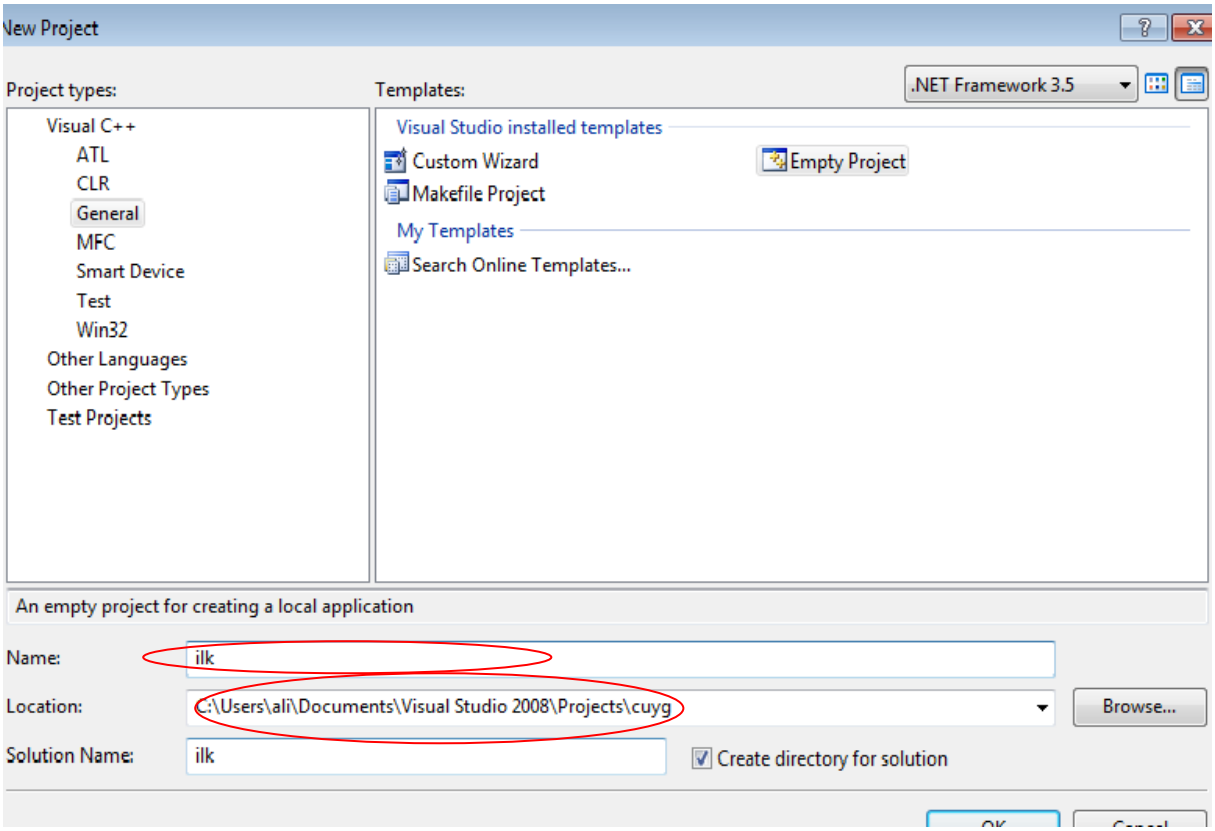


File menüsünden new proje seçeneği seçilir

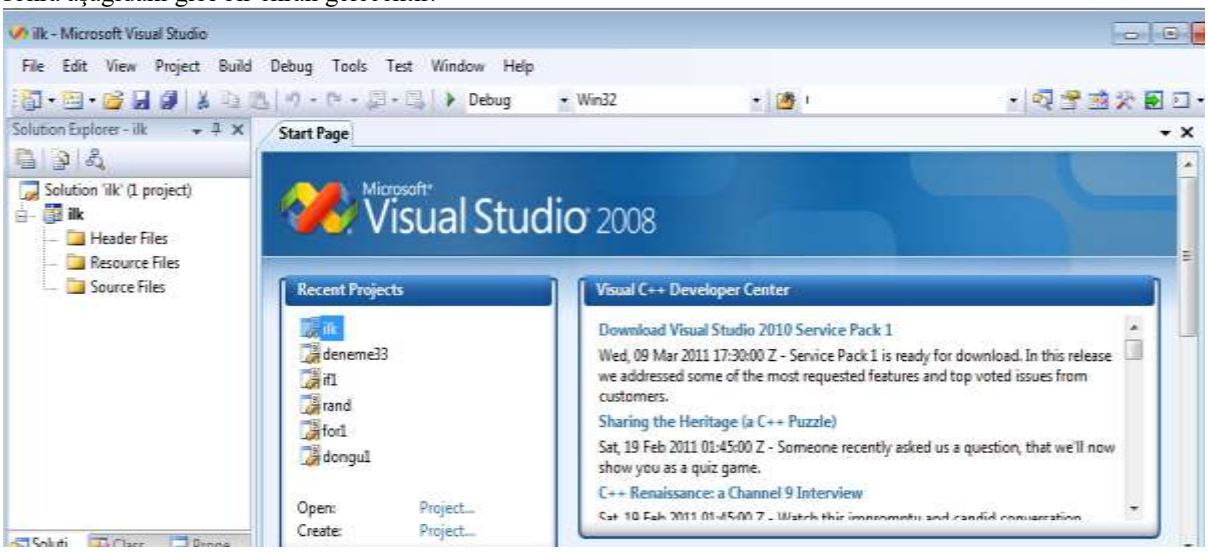


Karşımıza visual proje dosyası oluşturma ekranı gelir. Eğer istediğimiz programlama dili c görünmüyor ise other languages sekmesine tıklanarak buradan, visual c++ menüsüne tıklanır.

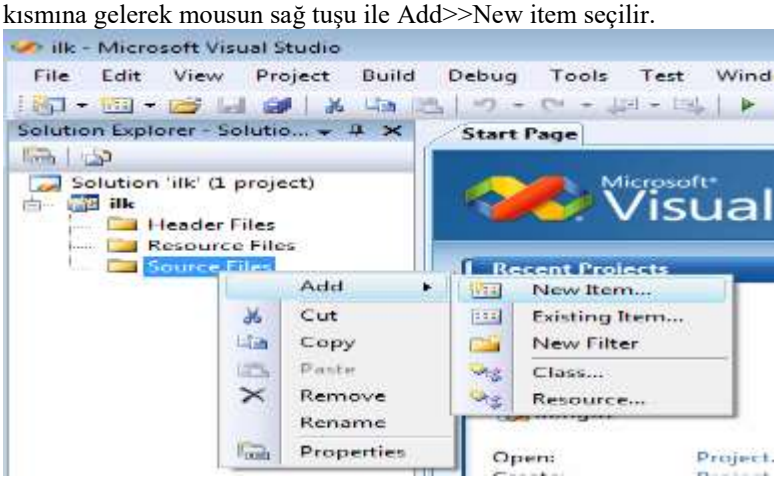
Karşımıza Aşağıdaki gibi visual C++ programının Empty Project kısmı seçilir. Burada önemli olan Location kısmının ve name kısmının kendimiz tarafından düzenlenebilmesidir.



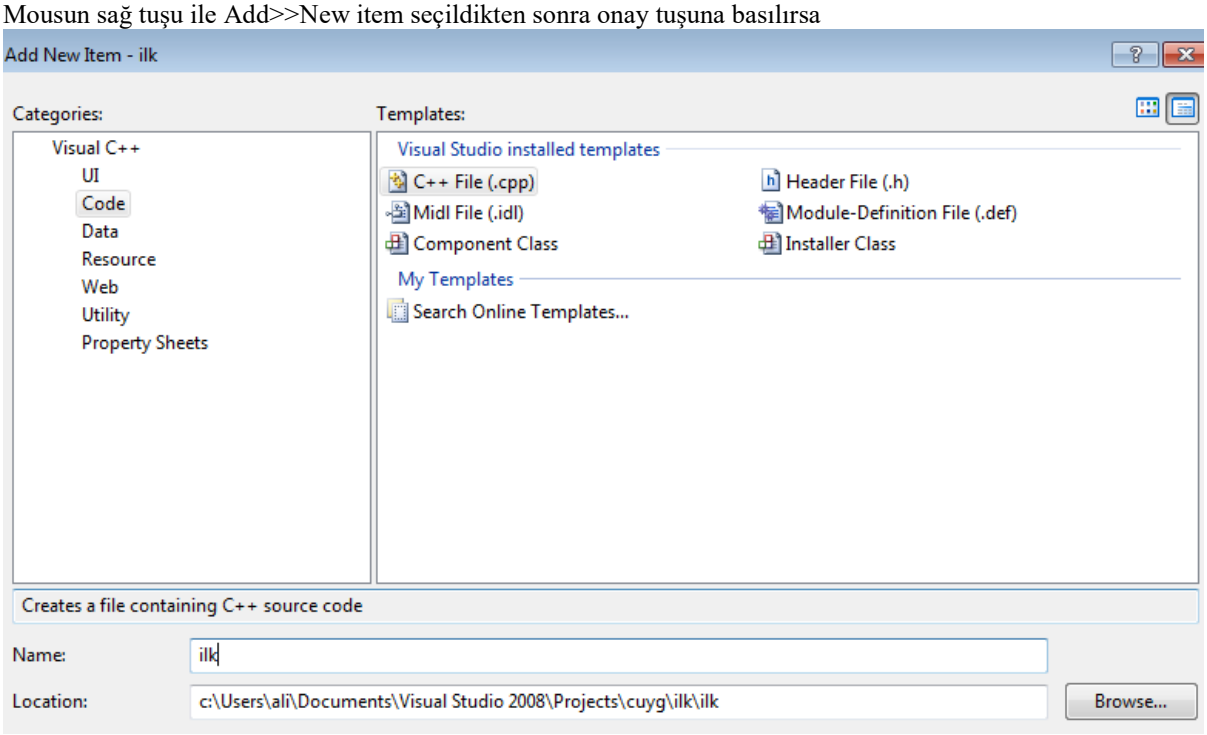
Ben kendi lokasyonumu cuyg olarak değiştiriyorum , ayrıca dosya isminede ilk proje ismini veriyorum. OK tuşuna bastıktan sonra aşağıdaki gibi bir ekran gelecektir.



Şimdi ise proje içinde çalışacak dosyamızı oluşturmaya (source) geldi. Bunun için soldaki source yada resource files kısmına gelerek mousun sağ tuşu ile Add>>New item seçilir.



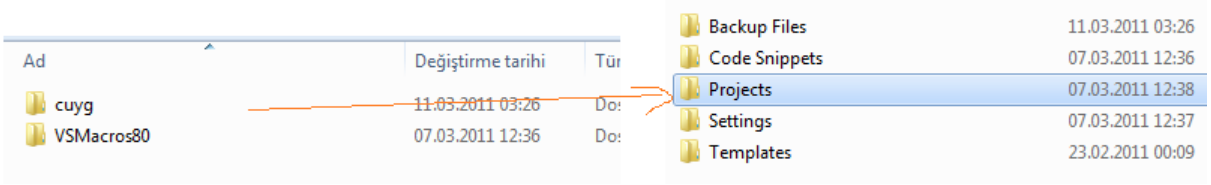
Mousun sağ tuşu ile Add>>New item seçildikten sonra onay tuşuna basılırsa



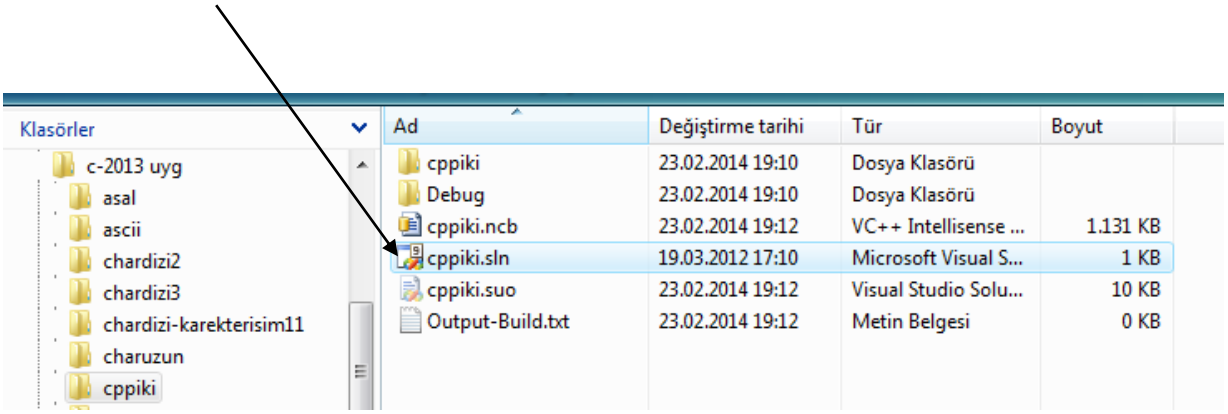
Categori sekmesinden Code seçilir. Templates sekmesinden ise C++ File(.cpp) seçilerek Name kısmına istenilen c dosya ismi yazılır.

Eğer aynı isimde projemiz varsa kullandığımız program bizi uyaracaktır(aynı isimde proje var şeklinde).

Eğer kayıtlı projemiz var ise Project klasörünün içindeki projelerden *.sln dosyalarına basılarak kayıtlı dosyalarımızı geri getirebiliriz.



Buradan proje içindeki klasöre girilir.



Sln uzantılı dosyası tıklandığında daha önce kaydettiğimiz proje c dosyamız karşımıza gelecektir.

Benzer dosya karşımıza gelir.

Şimdi C++ programının özelliklerini inceleyelim. Neden C++
C++ Programlama Dili'ni popüler kılan önemli nedenler aşağıda listelenmiştir:
C++, güçlü ve esnek bir dildir. C++ ile işletim sistemi veya derleyici yazabilir, kelime işlemciler oluşturabilir veya grafik çizebilirsiniz.

C++, iyi bir yazılım geliştirme ortamına sahiptir.

C++, özel komut ve veri tipi tanımlamasına izin verir.

C++, taşınabilir bir dildir.

C++, gelişimini tamamlamış ve standardı oluşmuş bir dildir.

C++, yapısal bir dildir. C++ kodları *fonksiyon* olarak adlandırılan alt programlardan oluşmuştur.

C++, Java, JavaScript, JavaApplet, PHP, C#, .Android.. gibi diller C dilinden esinlenmiştir.

C++ dilinde bir program yazılırken, başlık dosyası (header file) olarak adlandırılan bir takım dosyalar `#include` önişlemcisi kullanılarak program içine dahil edilir. #include <iostream> ve using namespace std;kütüphanesinde bulunan birçok fonksiyon, başlık dosyaları içindeki bazı bildirimleri kullanır.

C++ programları veya kaynak kodları (source code) uzantısı `.c` olan dosyalarda saklanır. Kaynak kod, bir C++ derleyicisi (C++ compiler) ile nesne koduna (object code) daha sonra uygun bir bağlayıcı (linker) programı ile işletim sistemininde çalıştırılabilen (executable) bir koda dönüştürülür.

C++ Kodlarının Temel Özellikleri

Bir C++programı aşağıda verilen özellikleri mutlaka taşımalıdır.

- Yazılımda kullanılacak olan her fonksiyon için ilgili başlık dosyası programın başına ilave edilmelidir.
- Her C++ programı `main()` fonksiyonunu içermelidir.
- Program içinde kullanılacak olan değişkenler ve sabitler mutlaka tanımlanmalıdır.
- Satırın sonuna `;` işareti konmalıdır.
- Her bloğun ve fonksiyonun başlangıcı ve bitişi sırasıyla `{` ve `}` sembolleridir.
- C++ dilinde yazılan kodlarda küçük-büyük harf ayrımı vardır (case sensitive). Örneğin `A` ile `a` derleyici tarafından farklı değerlendirilir.
- Açıklama operatörü `/* */` sembolleridir. Yada // kullanılır.

Tanımlayıcılar (Identifiers)

•Birveyabirdenfazlakarakteresahipolabilirlervetanımlayıcınınboyuüzerine kısıtlama yoktur.

•Sadece alfabetik harfler, rakamlar ve(_) geçerlidir. (ASCII karakterseti)

•Tanımlayıcınınilk karakteri alfabetikolmakzorundadır.Bir tanımlayıcı hiç bir zaman “rakam” ile baĢlayamaz !

•Tanımlayıcılarküçük/büyükharffarklarınaduyarlıdırC/C++’da küçük ve BÜYÜK harfler farklı yorumlanır.

•C/C++ anahtarsözcükleritanımlayıcıolarakkullanılamaz.

C , C ++ genellikle değişken yada sabit atamaları main bloğundan sonra beklenir. Böylelikle değişkenlerimizin ve sabitlerimizin yerel olduğunu anlarız.

- **source -----> compiler -----> object -----> link**
- **kaynak derleyeci amaç bağlama**
- **kaynak kod : C ++dili ile yazılmış olan program.**
- **derleyeci : Kaynak kodu makina koduna çevirir**
- **amaç kodu : Kaynak kodun makina dilindeki karşılığı**
- **bağlama : Birden fazla amaç kodu dosyasının tek dosyada birleştirilmesi**

DEĞİŞKENLER
Değişkenler bilgisayarın geçici belleğinde bilginin saklandığı gözlere verilen sembolik adlardır. Bir C programında, bir değişken tanımlandığında bu değişken için bellekte bir yer ayrılır.
C++ programlama dilinde, değişkenler ve sabitler programın başında bulunmalıdır. Bazı uygulamalarda değişkenin bir başlangıç değerinin olması istenir. Böyle durumlarda değişken bildirilirken başlangıç değeri verilebilir

```
#include <iostream>
using namespace std;
void main()
{ float b;
int c;  float a;}
```

Değişken isimleri verirken bazı kurallara uymak zorunludur.

- Değişken adları en fazla 32 karakterden oluşabilir. 32 karakterden uzun değişken adları ilk 32 karakteri değerlendirilir.
- Değişken adları ingiliz alfabesinde bulunan karakterler (A-Z) veya (a-z) yada rakamlar (0-9) ile yazılmalıdır. Türkçe karakterler, özel karakter veya boşluk karakteri kullanılamaz.
- Değişken adları herhangi bir rakam ile başlayamaz. İlk harfi zorunlu olarak harf olmalıdır. Sonrakiler rakamlardan oluşabilir.
- Değişken isimleri C++ için ayrılan komutlardan verilemez.

C++ temelde bize şu yenilikleri getirmektedir:
Sınıf kavramı;
Genişletilmiş fonksiyonlar yaratabilme ortamı;
Dinamik belleği hazır kullanma;
Operatorleri yeniden yapılandırma;
Template Fonksiyonlar;
Hata denetimli yazılımı kolaylaştıran araçlar;

DEĞİŞKEN TÜRLERİ

Değişkenler türlere ayrılır. Her türün kendine özel bir adı ve alabileceği değerleri vardır. Örneğimizde “int” kullandık ama daha bir çok tür var. Bazı önemli değişken türleri:
short : -32768 ile 32767 arasında değer alan tam sayılar. Ram da 2 byte yer tutarlar.
int : -2147483648 ile 2147483648 arasında değer alan tam sayılar. Ram da 4 byte yer tutarlar.
unsigned : tek başına değişken türü değildir. Diğer türler ile beraber kullanılır.(unsigned int, unsigned short, unsigned float …) Sıfırdan büyük değerler alabilir. Bu yüzden limiti iki kat artar. Negatif değerler yerine daha çok pozitif değer tutmak için kullanılır.
float : Virgüllü sayılar Ram da 4 byte yer tutarlar
double : Büyük virgüllü sayılar Ram da 8 byte yer tutarlar
char : karakter(a,b,c,d,e gibi.) Tek karakter tutar. (abc olmaz.) Atama yapmak için karakteri tek tırnak içine alın.(char x = ‘a’) Ram da 1 byte yer tutarlar.
bool : Doğru ya da Yanlış anlamına gelen “true” ya da “false” değerlerinden birini tutarlar. Ram da 1 byte yer tutarlar.

Tablo halinde veri tipleri

Tip	Sınırlar
Char	-128..127
Unsignedchar	0..255
Signedchar	-128..127
Int	-32768..32767
Unsignedint	0..65535
Signedint	-32768..32767
Shortint	-32768..32767
Unsignedshortint	0..65535
Signedshortint	-32768..32767
Longint	-2147483648..2147483649
Signedlongint	-2147483648..2147483649
Unsignedlongint	0..429967296
Float	3.4E-308..3.4E308
Double	1.7E-308..1.7E308
Longdouble	3.4E-493..3.4E493

Sabitler

Sabit bildirimi, başlangıç değeri verilen değişken bildirimi gibi yapılır. Ancak, veri tipinin önüne `const` anahtar sözcüğü konmalıdır.

Const
int Veri=12;
float Vergi=5,67;

Değişken Bildirim Yerleri ve Türleri

Yerel (local) Bildirim

Yerel değişkenler kullanıldığı fonksiyon içerisinde bildirilir. Yalnızca bildirildiği fonksiyon içerisinde tanınır ve kullanılabilir.

```
int topla(int xx,int yy)
{    /* yerel (local) değişken c nin bildirimi */
 int p;     p= xx + yy;
 return p;   }
```

Genel (general) Bildirim

Genel değişkenler bütün fonksiyonların dışında bildirilir. Bir değişken program boyunca sürekli olarak kullanılıyorsa genel olarak bildirilmelidir.

```
#include <stdio.h>
void karesi();
/*  a ve b global tip değişkendir.
    Bu iki değişken tüm program boyunca kullanılmaktadır. */
int a,b;
main()
{a=7;
  karesi();
  cout<<a<<"nin karesi”<<b<<”dir”;   }
void karesi(){b = a*a;   }
```

C++ dilinde karekök almak için bir deyim yoktur. Örnekte bunu yerine getiren C++ diline eklenmiş olan sqrt() fonksiyonu kullanılmştır. Aşağıda buna benzer artık C++ derleyecilerinde standart olmuş bazı fonksiyonlar verilmiştir. Bu işlevler math.h başlık dosyasında tanımlıdır.

Fonksiyon	x ,y	Sonuç	
abs(x)	int	int	x'in mutlak değeri
fabs(x)	double	double	x'in mutlak değeri
pow(x, y)	double	double	x^y
sqrt(x)	double	double	x'in karekökü
exp(x)	double	double	e^x değeri
log(x)	double	double	ln(x) değeri
log10(x)	double	double	$\log_{10}(x)$ değeri
ceil(x)	double	double	x ten büyük ilk tamsayı
floor(x)	double	double	x ten küçük ilk tamsayı

Operatör Çeşitleri

1- Aritmetiksel Operatörler
2- Karşılaştırma Operatörleri
3- Mantıksal Operatörler

1- Aritmetiksel Operatörler

Operatör	Tanımı	Operatör	Tanımı
+	Toplama	%	Kalan
-	Çıkarma	+=	Ekleyerek Atama
*	Çarpma	-=	Çıkararak Atama
/	Bölme	*=	Çarparak Atama
++	Bir arttırma	/=	Bölerek Atama
--	Bir azaltma	%=	Kalanı Atama

Atama	***Karşılığı***
a+=b	***a=a+b***
a+=b+c	***a=a+b+c***
a-=b	***a=a-b***
a-=b+c	***a=a-(b+c)***
a*=a	***a=a*a***
a%=7	***a=a%7***
a/=8	***a=a/8***
-	-

X++	İşlemden sonra arttır
++X	İşlemden önce arttır
Y--	İşlemden sonra azalt
--Y	İşlemden önce azalt

//atama işlem örneği

```
int main()
{
int a, b;
        a = 10;
b =++a;
cout<< a<< b;}
```
Sonuç a=11 b=11

```
int main()
{
int a, b;
        a = 10;
b =a++;
cout<< a<< b;}
```
Sonuç a=11 b=10

```
int main()
{ int a, b;
        a = 10;
b =--a;
cout<< a<< b;}
```
Sonuç a=9 b=9

Mantıksal ifadeler

Sonucu Doğru veya Yanlış olan ifadelerdir. Sonuç sıfır ise yanlış aksi halde doğru kabul edilir.

İlişkisel işleçler(operatör) : iki değer arasındaki ilişkiyi test etmek için kullanılır

işleç	anlamı
>	büyük
>=	büyük - eşit
==	eşit
<	küçük
<=	küçük - eşit
!=	eşit değil

x=8, y=5 için

t=4, m=4

if(x > y) Doğru

if(x < y) Yanlış

if(x !=y) Doğru

if(t==m) doğru

Mantıksal işleçler : İki mantıksal ifade arasındaki ilişki üzerindeki ilişkide kullanılır.

! DEĞİL (NOT)

&& VE (AND)

|| VEYA (OR)

(X>0) && (X>Y)

(X>0) || (Y>0)

İfadelerde işleçlerin yürütülme sırası

işleç	Önceliği
( )	en yüksek (ilk yürütülür)
!	
*, /, %	
+, -	
<, <=, >=, >	

==, !=

&&, ||

= en düşük (son yürütülür)

= işleci sağdan sola, diğerleri soldan sağa doğru yürütülür.

Görüldüğü gibi ifadelerde matematiksel ve mantıksal işlemler bittikten sonra ilişki test edilir.

X=50, Y=80, Z=45 için

((X / 4 + Y / 4 + Z / 2) >= 50) && (Z >= 50)

C++ dilinde bir fonksiyon tanımlanmasında fonksiyonun geri dönüs değerinin türü olarak void anahtar sözcüğü yazılmamıssa, fonksiyon return anahtar sözcüğü kullanılarak mutlaka bir geri dönüs değeri üretmelidir. Fonksiyonun geri dönüs değeri üretmemesi durumunda derleme zamanında hata olusacaktır. Yani C dilinde olduğu gibi rasgele bir değer üretilmesi söz konusu değildir. Yine C++ dilinde geri dönüs değeri üretecek bir fonksiyonun tanımlanması içinde return anahtar sözcüğü yalın olarak kullanılamaz. return anahtar sözcüğünün yanında mutlaka bir ifade yer almalıdır.

C++ kullanılan başlık tanımlama komutları

#include komutu: Her C++ programının başında görürsünüz bu kodu. #include içeriye almak, bir takım kod tanımlarının olduğu dosyaları, programınıza dahil eder. Örneklerimizde kullanacağımız "cout" komutu, "iostream" dosyası içerisindedir. "cout" komutunu kullanabilmek için önce onun tanımlamasının yani ne iş yapacağının anlatıldığı dosyayı programımıza tanıtılması gerekir.
iostream : In Out Stream (Giriş, çıkış akışı) Uygulamamızın klavyeden, dosyadan vs. girdi; ekrana ya da bir dosyaya çıktı vermesini sağlar. Yalnızca C++ da kullanılabilir.
stdio.h : Standart In Out (Standart Giriş Çıkış) iostream 'ın C versiyonudur. C++ tarafından da desteklenir, ancak tavsiye edilmez.
cmath / math.h : Matematiksel işlemleri barındırır.(Karekök, üs, logaritma vs.) C++ ve C de kullanılabilir. (C versiyonu math.h, C++ versiyonu cmath)
clocale / locale.h : Dil dosyalarını barındırır. Eğer eklenmezse türkçe karakterler görünmez/bozuk görünür. C++ ve C de kullanılabilir. (C versiyonu locale.h, C++ versiyonu clocale)
stdlib.h : Standart Library (Standart Kütüphane) İçerisinde giriş/çıkış hariç bir çok gerekli işlem barındırır. C++ için olanı ise cstdlib dir. ancak stdlib de C++ da kullanılabilir. Her programa eklenmesi tavsiye edilir.
windows.h : Windows a ait işlemleri barındırır. DirectX uygulamalarımızın hepsinde ekleyeceğiz. C ve C++ da kullanılabilir.
NOT: Bir çok C kütüphanesi, C++ için yeniden yazılmış ve isimlerinin önüne "c" getirilip ".h" uzantısı kaldırılmıştır. C++, uzantısız kütüphane kullanımını tavsiye eder.. Ancak C, ".h" uzantılı olmasını tavsiye eder.
using namespace std; Kodların tanımlandığı bloklardır. Aslında "cout" kodu, std alanı içinde tanımlıdır. Ve kullanılırken, std::cout denilmesi gerekir. Ancak bu kod sayesinde std:: ön ekine gerek kalmaz.(Kodların kullanımı: namespace::kod; şeklindedir. Her kod, (;)(noktalı virgül) ile bitmelidir. VC++, kodun bittiğini ; işareti ile algılar.

INPUT-OUTPUT İÇİN STREAM KULLANILMASI

Cin komutu

- C++'ın standard input stream'i ise **cin** olarak adlandırılır. Aşağıdaki program klavyeden bir tamsayının nasıl girdi olarak alınacağını göstermektedir:
- Cin komutu >>okları yardımı ile değişkeni algılar. Cin>>a; yada cin>>a>>b; örneklerinde olduğu gibi.

```
#include <iostream>
using namespace std;

int main()
{  int miktar = 123;
cout << "Bir miktar girin..."<<endl;
```

```
            cin >> miktar;
    cout << "Girdiğiniz Miktar: " << miktar;
    return 0;}


            #include <iostream>
            using namespace std;

  int main()
            {  char isim[20];
     cout << "Bir isim girin...\n";
            cin >> isim;
     cout << "Girdiğiniz İsim: " << isim;}
```

cout (Character Out) Değişken ve sabitlerin ekrana bastırılması için kullanılır. "<<" ifadesi, bu fonksiyon için yeniden yazılmış ve işlev kazanmıştır. Cout fonksiyonu için birden fazla satıra gerek yoktur. Tek satırda da birden çok değişken için kullanıla bilinir.
Örnek :cout<<"Ahmet";
Cout<<x<<y<<z;

AÇIKLAMALAR (COMMENTS)

- C++, C'nin /* ... */ açıklama formatına ek olarak, tek-satır açıklamaları için '//' işaretinin kullanımına izin vermektedir.

Ayrıca // karekterleride kullanılabilir.

- Örnek:

```
            // C++'da Açıklama Yazılması
            #include <iostream>
            using namespace std;
            void main()
            {
      char isim[20];   // Karakter kelimeyi bildirimi
 cout << "Bir isim girin...\n";  //İsim giriş isteği
 cin >> isim;                    //İsmin okunması
cout << "Girdiğiniz İsim: " << isim;
          }
```

//C++ile ilk programım(//)işaretlerinin sağındaki hiç birşey derleyici tarafından gözönüne alınmaz

Örnekler

```
//Veri giriş örnekleri
#include <iostream>
using namespace std;
void main()
 {
int x=125;
int y=200;
int z=500;
//veri girişini enter yada boşlukla yapınız
cin>>x>>y>>z;
cout<<"x ="<<x<<" y ="<<y<<" z="<<z; }
```

```
#include <iostream>
using namespace std;
void main()
 { int a=10, b=121;
//a = 100;
```

```
//atama yoluyla toplama
#include <iostream>
//#include <stdio.h>
using namespace std;
void main()
 {
int a=100, b=121;
//a = 100;
a=b;
b=a;
//b =a++;
cout<<"a ="<<a<<"b ="<<b ;}
```

```
a=b;
//b=a;
b =a++;
cout<<"a ="<<a<<"b ="<<b ;}


//toplama işlemi
#include <iostream>
using namespace std;
void main()
{
int a,b,c;
cin>>a>>b;
c=a+b;
cout<<"c="<<c;}
```

Döngü ve Koşul Deyimleri

Programlar (algoritmalar) üç temel blok kullanılarak gerçekleştirilebilirler. Bunlar; art arda, bir koşula bağlı olarak ve sonlu sayıda yineleme (döngü) dir.

Koşul Deyimleri

Birkaç seçenekten birini seçmek veya bir deyimin bir koşula bağlı olarak işlemek için kullanılır.

if--else Deyimi

if (<mantıksal ifade>**)**
    blok_doğru;
else
    blok_yanlış;

Mantıksal ifade doğru ise blok_doğru, yanlış ise else sözcüğünden sonraki blok_yanlış yürütülür. else kısmı seçimlidir, gerekmiyorsa kullanılmayabilir.

Buradaki şart bir değişken ,ifade veya sabit olabilir.Şart mutlaka parantez içerisinde yazılır.Eğer şarta bağlı olarak çalıştırılması istenen komutlar birden fazla ise blok içine alınırlar. Bloklu komutlar birbirinden noktalı virgül ile birbirlerinden ayrılırlar. İf satırından sonra iki satır var ise blok kullanılmadığı taktirde yazım hatasından dolayı program hata verir. Not İf satırında eşitliğin = = iki adet olduğunu unutmayalım.

```
//ifelse.cpp
#include <iostream>             // cin,cout,endl
using namespace std;
void main(){
int not;
cout<< "Notu giriniz:";
cin >> not;
if (not >= 50)
cout << "Geçti!";
else
cout << "Kaldı!";}
```

if kısmından anlatmaya başlayacağım. Eğer notumuz 50 ye eşit veya 50 den büyük ise geçiyoruz aksi halde kalıyoruz.Bir de bir if-else in altında bir tane daha if-else kullanalım.

```
#include <iostream>             // cin,cout,endl
using namespace std;
void main(){
int not;
cout<< "Not`u giriniz:";
cin >> not;
if (not >= 50)
cout << "Geçtiniz!";
else
{cout <<"Bütten alınan not:";
cin >>not;
if( not>=60 )
```

```
cout << "Geçtiniz!";
else
cout <<"Kaldınız!";}}
```

Burada da şunu inceledik: Diyelim ki sınava girdik ve notumuzu öğrendik, notu giriyoruz 50 nin altındaysa kalıyoruz ve bütünleme sınavına giriyoruz. Bütünlemede de geçer not en az 60. Sanırım bu basit örneklerle olayı iyice kavramışızdır. if-else i de burada bitiriyoruz.

Örnek Girilen sayının tek/çift olduğunu yazan program

```
#include <iostream>
using namespace std;
void main(){
  int i;
cin>>i;
  if ( i % 2 == 1)
cout<<"Tek";
  else
cout<<"Cift"; }
```

```
//girilen sayının negatif yada pozitif özelliklerini
#include <iostream>
using namespace std;
void main(){ //ana blok bas
int a;
cout<<"bir sayı giriniz=";
cin>>a;
if (a>0)   //if parentez içinde olacak
{ //1.blok bas yoksa hata verir
cout<<"girdiğiniz sayi : "<<a;
cout<<"bu sayı pozitifdir.\n";
}//1.blok sonu
else
{ //2.blok bas blok olmasada sorun yaratmaz
cout<<"girdiğiniz sayi : \n"<<a;
cout<<"bu sayı negatifdir.\n";
} //2.blok sonu } //ana blok sonu
```

```
// rasgele iki sayının karşılastırılması
#include <iostream>
#include <time.h>
using namespace std;
void main()
{int a=0,b=0,c=0;
b=time(NULL);
srand(b);
a=rand() % 100 + 1; //1--100
c=rand() % 100 + 1;
cout<<"a="<<a<<" c="<<c<<endl;
if (a>c) cout<< a<<">"<<c;
else
if (a<c)
cout<< a<<"<"<<c;
else
 cout<<a<<" ="<< c;// elsesiz olmaz
}
```

```
//if yapısı ile rasgele 0- 9 arasında sayı üretmek
#include <iostream>
using namespace std;
#include <time.h>
void main()
{
int a,b;
srand ( time (NULL) );
a=(rand() %10);
b=(rand()%10);
cout<<"a="<<a<<" b="<<b<<endl;
if (a>b) cout<< a<<">"<<b;
else
if (a<b)
cout<< a<<"<"<<b;
else
 cout<<a<<" ="<< b;// elsesiz olmaz
}
```

switch Deyimi

switch(<seçici>**) {**
case seçenek1 : Deyim;
case seçenek2 : Deyim;
.
default : Deyim;
}

Seçicinin aldığı değere eşit seçeneğin olup olmadığına bakar. Var ise o noktadan sonraki deyimler yürütülür. **switch** deyiminin sonuna gelindiğinde veya **break** deyimi ile karşılaşıldığında yürütme işlemi durur ve programın akışı switch deyimini izleyen deyim ile devam eder.

```
switch(i) {
  case 1 : cout<<"Bir";
  case 2 : cout<<"İki";
  default : cout<<"Hiçbiri";}
```

i=1 ise çıkış BirİkiHiçbiri
i=2 ise çıkış İkiHiçbiri
Sorunu ortadan kaldırma için her durum için break deyimi eklenmeli.

Seçici **Ordinal** tiplerden biri olmalıdır (Ordinal tip: tüm değerleri listelenebilinen veri tipleri - integer, char).
. Seçici ile seçenekler aynı tipte olmalıdır.
. default kısmı seçimliktir. Seçeneklerin hiçbiri uygun değil ise yürütülür.

```
//case örneği veri girişinde önce karekter sonra iki adet sayı girilecektir
#include <iostream>
using namespace std;
void main(){
  char islem;
  int s1, s2, s3;
  cout<<"Önce işlemi sonra sayıları girin ";
  cin>>islem>>s1>>s2;
  switch (islem) {
    case '+' : s3 = s1 + s2; break;
    case '-' : s3 = s1 - s2; break;
    case '*' : s3 = s1 * s2; break;
    case '/' : s3 = s1 / s2; break;
    default : cout<<"Hatalı işlem";      }
  cout<<"\nSonuç = "<<s3;}
```

Döngü Deyimleri (Yineli)

Bir ya da birden fazla deyimin tekrar edilemesini sağlarlar. C++ dilinde while, for ve do-while deyimleri döngü işlevini saşlar. Tekrar edilen deyimlere döngü gövdesi denir.

while Deyimi

while <mantıksal ifade>
Deyim

Mantıksal ifade doğru olduğu sürece Deyim yürütülür. Eğer yanlış ise kontrol bir sonraki deyime geçer.

```
//rasgele sayı üretme örneği
#include <iostream>
#include <time.h>
using namespace std;
void main()
 {
        int a=0, k=0;
        srand(time(NULL)); //RANDOMİZE ANLAMINDADIR
        while  (a<32767){
                k=k+1;
                a=rand();
                                cout<<a<<endl;}}
```

Örnek 1'den 100'e kadar olan sayıların toplamı.

1. i =1
2. j = 0
3. i < 101 olduğu sürece
   3.1 j = j + i
   3.2 i = i + 1
4. Toplam j ' yi yaz

```
#include <iostream>
using namespace std;
void main()     {
        int i, j;
        i =0;
        j = 0;
        while (i<101) {
            j =j+i;
            i =i+1;
        }
         Cout<<"Toplam ="<<j;          }
```

Örnek: Girilen sayının faktoriyelini hesaplayan programı yazınız. //farklı bir faktör örneği

```
#include <iostream>
using namespace std;
void main() {
           int i=0, j=0,k,faktor=1;
           cin>>k;
             while (k>i) { //k=4 ,
i=1,2,3,4
                        i=i+1;
                faktor=faktor*i;//4
                                  }
             Cout<<"faktor="<<faktor;}
```

```
#include <iostream>
using namespace std;
void main(){
 int n;
 long f;
cout<<"sayıyı giriniz "); cin>>n;
 f = 1;
 while (n > 1) {
  f = f * n;
  n = n - 1;
 }
 Cout<<"Sonuç = "<< f;}
```

```
// 1 den 5 e kadar sayıların toplamları
#include <iostream>
using namespace std;
void main()
{          int i, j=0,k=0;
            while(k++<5) //dikkat edelim
            {               //k++;
                  j=j+k;
                  cout<<"j="<<j;}}
```

Örnek : Klavyeden girilen sayıları oku. Sayıların toplamı 21'den büyük veya eşit olduğu zaman dur.

```
#include<iostream>
using namespace std;
void main()
{
      int a=0,b,c=0,d;
      while(a<21)
      {     cout<<"sayi gir=";cin>>b;
      a=a+b;
      c=c+1;
      }
      Cout<<"toplam ="<<a<<" "<<c<<"adet sayi girdiniz";}
```

****

Başka bir örnek yaz okulu için ücret hesaplama yapalım.

Dikkat edilirse `#include<conio.h>` kütüphanesi kullanılmıştır. Bunun nedeni konsol üstünde Getche() fonksiyonu yada Getch() fonksiyonu klavyeden karekter okuması için kullanılmaktadır. getch fonksiyonunu kullanmak için ayrıca cev=='e' satırınada dikkat edelim.

Ayrıca **char cev='e';başlangıçta niçin verilmiştir.**

// **örnek yaz okulu için ücret hesaplama**

```
#include<iostream>
#include<conio.h>
#include<locale.h>
using namespace std;
void main()
{       int a=0,b,c=0,d;
        char cev='e';
        char soy[20];
        int miktar;
        float ucreti=0;
        int katsayi=1.32;
        cout<<"soyadiniz=";cin>>soy;
        while(cev=='e')
        {cout<<"alacaginiz ders saati=";cin>>miktar;
        ucreti=miktar*katsayi*7*2;
        cout<<"devam etmek istermisiniz? (e/h)\n";
        cev=getch();    }
        cout<<"sevgili "<<soy<<" ";
        cout<<"toplam = "<<miktar<<" "<<"saat icin"<<ucreti<<" tl ödeyiniz ";}
```

For Deyimi

```
for (ifade1 ; ifade2 ; ifade3 )
   ifade;
```

ifade2 doğru (veya farklı 0) olduğu sürece ifade yürütülür (bitiş koşulu).
Döngünün ilk adımından önce ifade1 yürütülür (başlangıç adımı).
Döngünün her adımında ifade3 yürütülür (artış miktarı).

```
for (i = 1; i < 5; i++)
   cout<<i;
```

ifade1, ifade2 ve ifade3 seçimliktir. ifade2 belirtilmez ise herzaman doğru olduğu (== 1) kabul edilir. Yani sonsuz döngü oluşur.

```
for (i = 1; ; i++)
        cout<<i;
```

Örnek 1'den 100'e kadar olan sayıların toplamı.

```
j =0;
for (i=1; i<=100; i++)
      j =j+i;
cout<<"Toplam ="<<,j;
```

Örnek Girilen sayının faktöriyelini bulunuz.

```
#include<iostream>
using namespace  std;
void main()
{int j=0,i=0;
float fact;
  fact =1;
cout<<"sayi girin";
cin>>i;
            for (j=1; j<=i; j++)
                  fact =fact*j;
            cout<<"Faktöriyel  ="<<fact;        }
```

//for dongüsü ile enbüyük enkucuk bulma

```
#include <iostream>
using namespace std;
```

```
void main()
{
int i,a,ek,eb,top;
float  b,ort,t;
eb=0;ek=100;top=0;
for (i=1;i<=5;i=i+1){
bas:cout<<i<<". sayisini gir:=";
        cin>>a;
        if(a<=ek)ek=a;
if(a>=eb)eb=a;
top=top+a;}
cout<<"ortalama="<<top/5;
cout<<"enb="<<eb<<" enk="<<ek;}
```

```
//enb enk ortalama sirasi ile bulma
#include <iostream>
using namespace std;
void main()
 {int i,a,ek,eb,top,sirak,sirab;
float  b,ort,t;
eb=0;ek=100;top=0;
for (i=1;i<=5;i=i+1){
bas:cout<<i<<". sayisini gir:=";
        cin>>a;
        if(a<0||a>100) goto bas;
if(a<=ek)
{ek=a;sirak=i;}
if(a>=eb)
 {eb=a; sirab=i;}
top=top+a;}
cout<<"ortalama="<<top/5;
cout<<"enb="<<eb<<" sira="<<sirab;
cout<<"enk="<<ek<<" sira="<<sirak;}
```

```
Örnek 1- ile 30 arasındaki sayıların içinden 3 e bölünenleri bulma
#include <iostream>
using namespace std;
void main()
 {int i,a;
float  b,ort,t;
eb=0;ek=100;top=0;
for (i=1;i<=30;++i){
if (i%3==0)
cout<<i;}}
```

do-while Deyimi

Bir koşul doğru olana kadar döngü yürütülür.

```
do
   Deyim
while (<mantıksal ifade>)
```

Mantıksal ifade doğru olduğu sürece döngü tekrar edilir. Yanlış olduğunda while sözcüğünden sonraki deyim yürütülür.

Örnek :5 sayısı girilene kadar oku

```
do
   cin>>i; -→ bu yapıda program 5 i bulamayacağı için sonsuz döngü oluşur
while (i!=5);
```

```
i =1;
do {
   cout<<i*i;
   i =i+1;
} while (i<=10);
```

```
do//while ile 1 ile 10 arsındaki tek sayilarin toplamini bulalım.
#include<iostream>
using namespace std;
void main()
{
      int a=0,b,c=0,d;
      do
      {
      c=c+1;
cout<<c<<endl;
      a=a+c;
      c=c+1;
      }
while(c<10);
cout<<"toplam =<<a;}
```

Karşılaştırma

while : Koşul başlangıçta test ediliyor. Döngü sıfır veya daha fazla yürütülüyor.

do-while : Koşul sonda test ediliyor. Döngüye en az bir defa kesin giriliyor.

Ayrıca do while döngüsünde son satırda bulunan while satır sonunda noktalı virgül kullanıldığını unutmayalım

rand () fonksiyonu

Sıfır ile bir arasında rasgele sayı üretilmesini sağlar. Rand() % 12 ----→ 0 ile 11 arasında rasgele sayı üretir. Ayrıca üretilen sayıların farlı olması için **srand(time(null)) yada srand(time(0))** fonksiyonu zorunlu olarak kullanılır. Zamana bağlı sayı üretildiği için programın en başında mutlaka **#include<time.h> tanıtılma zorunluluğu vardır.**

// do /while örneği 10 rakamını bulunca program dursun

```
#include <iostream>
#include <time.h>
using namespace std;
void main()
{ int a,b,i=0;
srand(time(0));
do
{     i=i+1;
a=(rand() % 10 + 1);
cout<<i<<a;
}
while (a!=10);}cout<<i;}
```

//iki sayının karşılaştırılması

//if fonksiyonu ile rand fonksiyonu kullanım örneği

```
#include <time.h> //mutlaka olacak
#include <iostream>
using namespace std;
void main()
```

```
{int s1,s2,i;
//srand(time(0)); //randomize
srand(time(NULL)); //randomize
s1=(rand() % 10)+1;//1,2,3,4,5,6,7,8,9,10
s2=(rand() % 10)+1;
cout<<"uretilen sayilar"<<s1<<s2;
if (s1>s2)
cout<<s1<<">"<<s2;
else
if(s2>s1)
cout<<s1<<"<"<<s2;
else
//if(s2==s1)
cout<<s1<<"="<<s2;}
```

// 1 ile 10 arasında rasgele sayı üretmek ve rasgele üretilen 10 değerinin(rakamının)
//kaçıncı denemede bulunduğunu bildirme

```
#include <iostream>
#include <time.h>
using namespace std;
void main()
{ int a=0,i=0;
srand(time(0));
do
{ i=i+1;
a=(rand() % 10 + 1);
cout<<i<<" "<<a<<endl;}
while (a!=10);
cout<<i;}
```

Aynı örneği while döngüsü , break komutu ile kullanalım
// 1 ile 10 arasında rasgele sayı üretmek ve rasgele üretilen 10 değerinin(rakamının)
//kaçıncı denemede bulunduğunu bildirme UNUTMAYINIZ

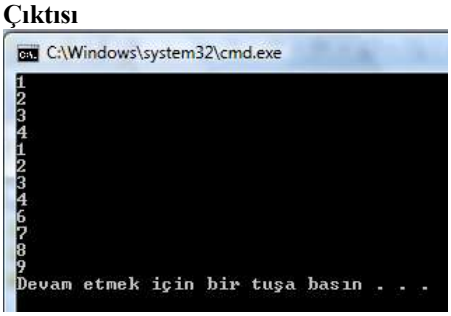
```
#include <iostream>
#include <time.h> //
using namespace std;
void main()
{ int a,b,t=1,say=0,c=0,i=0,sor,kac;
srand(time(NULL));
cout<<"lütfen 1 ile 10 arasında sayi gir";
cin>>sor;
cout<<"istenilen tekrar sayısını girin";
cin>>kac;
do
{a=(rand() % 10+1 );
i=i+1;
cout<<a;
if (a==sor) say=say+1;  }
while (say!=kac); //
cout<<i<<". seferde";}
```

Continue deyimi Döngünün herhangi bir satırında iken, bir sonraki döngü adımına geçilmesini sağlar.

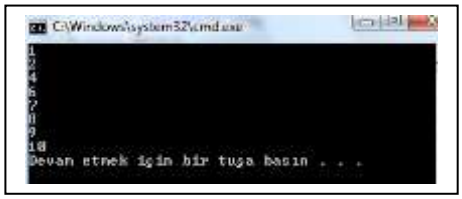
```
//break ve continue örneği
#include <iostream>
using namespace std;
void main()
{int i;
   for(i = 1 ; i < 10 ; i++)
   {       if (i == 5)
          break;
      cout<< i;    }

   for(i = 1 ; i < 10 ; i++)
   {       if (i == 5)
          continue;
      cout<< i;  }}
```

```
// continue 3 ve 5 dışındaki değerleri
//engellemek için
#include <iostream>
using namespace std;
void main()
{
int a=0,b=0,c=0,i=0;
while(i<10)
{
i=i+1;
 if(i==3 || i==5) continue;
cout<<i;
}   }
```

Çıktısı

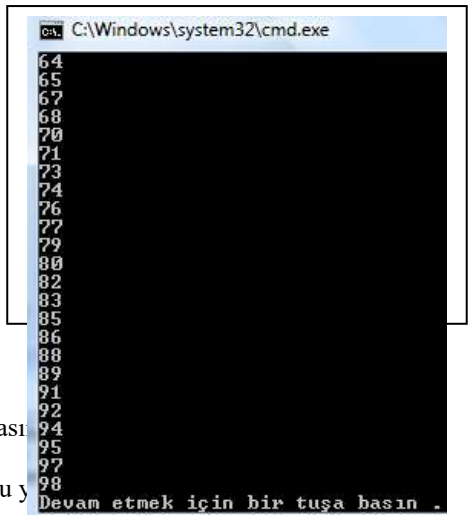


çıktısı



//continue için örnek

```
//3 katlarını basmayalım
#include <iostream>
using namespace std;
void main()
{int i, k;
char ch;
   for (i = 0; i < 100; ++i) {
            if (i % 3 == 0)
                        continue;
            cout<< i;}}
```



goto : Programın herhangi bir yerinden başka bir yere atlanması
Gidilecek yerin tanımlanması gerekir. Kulanım şekli;

goto son programın akışı son: etiketinin alduğu y
...
son :
return;

hatırlatma ++ / -- İşleçleri

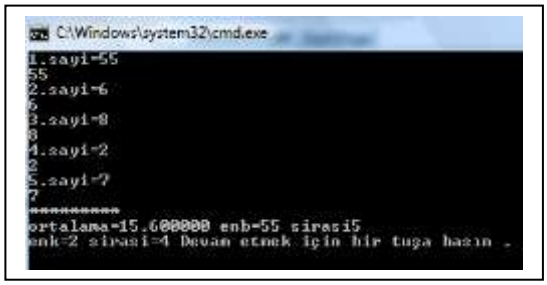
İfade içerisinde değişkenin
ardında ise değişkenin değeri ifadede kullanılır ve değeri bir arttırılır
önünde ise değişkenin değeri bir arttırılır ve yeni değeri ifadede kullanılır.

i = 2;
k = 5*i++; --> k = 10
k = 5*++i; --> k = 15

//enbüyük enkucuk bulma ve yerlerini bulma

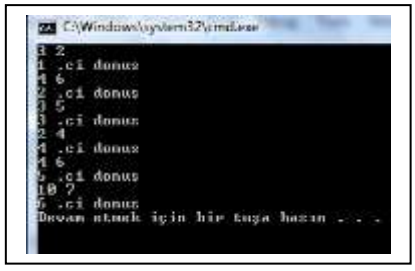
```
#include <iostream>
using namespace std;
```

```
void main()
{ int i,a,top=0,eb=0,ek=100,sr=0;
float  ort,t;
for (i=1;i<=5;i++){
      cout<<i;
      cin>>a;
      if(a<ek){
            ek=a;
sr=i;             }
if(a>eb) eb=a;
top=top+a;
cout<<a;
}cout<<"*********"<<endl;
cout<<"ortalama="<<top/5.0<<endl;
cout<<"enb= "<<eb<<endl;
cout<<"sirasi="<<i-1<<endl;
cout<<"enk= "<<ek<<endl;
cout<<"sirasi= "<<sr<<endl;}
```



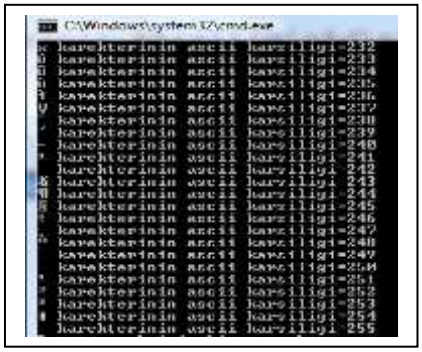
// (and kullanımı) üretilen sayılardan bir tanesi 10 çıktığında program duracak

```
#include <time.h>
#include <iostream>
using namespace std;
void main()
{
int a=0,b=0,c=0,i=0;
srand(time(0));
do{
a=rand() % 10 + 1;
c=rand() % 10 + 1;
cout<<a <<" "<<c;
i=i+1;
cout<<i".ci donus"<<endl; }
while (a<10 && c<10);
}
```



//karekter uygulaması

```
#include <iostream>
#include <locale.h>
using namespace std;
void main()
        {
        int i=0,a;
unsigned char karekter;
        cout<<"Ascii kodlari tamamı"<<endl;
        while (i<255)
        {               i=i+1;
             karekter=i;
cout<<karekter<<"karekterinin ascii karsiligi=" <<i<<endl; }}
```



//10 sayısının 2 kez tekrar bulunması

```
#include <iostream>
#include <time.h>
using namespace std;
void main()
      { int a,b,t=1,say=0,c=0,i=0;
srand(time(NULL));
do {
i=i+1;
a=(rand() % 10+1 );
cout<<a<<endl;;
if (a==10) say=say+1;}
while (say!=2);
cout<<i<<" . seferde"<<endl;}
```

//tek çift sayı üzerinden istenilen miktarda tek çift işlemini bulma

```
#include<iostream>
#include<time.h>
using namespace std;
void main()
//10 adet sayı üret kac tek cift
{int a,b=0,c,d,f,g,h,j,k,t=0;
srand(time(0));
while(b+t<10){
c=rand()%101;
if (c %2 ==0)
{b++;cout<<"cift="<<c<<endl;}
else
{t++;cout<<"tek="<<c<<endl;}}
cout<<" cift="<<b<<" tek="<<t<<endl;}
```

//Üç basamaklı rakamları birbirinden farklı tüm sayıları ekranda gösteren ve bu kurala uygun kaç tane sayı olduğunu söyleyen C programını yazınız.
Örnek çıktı : 102 103 104 105 106 107 108 109 120 123 . . . . . 980 981 982 983 984 985 986 987
Bu kurala uygun 648 sayı vardır.

```
#include<conio.h>
#include<iostream>
using namespace std;
void main()
{
int i, a, b, c, sayac=0;
for(i=100;i<=999;i++)
{
a = i/100; // yüzler basamağı
b = (i%100)/10; // onlar basamağı
c = i%10; // birler basamağı
if(a!=b && a!=c && b!=c) {
cout<< " "<<i;
sayac++;
}}
cout<<"\n\nBu kurala uygun sayi adedi=" <<sayac<<endl;
getche();
}
```

```
//Fibinochi açılımının ilk 20 satırı için hazırlayalım.
#include<conio.h>
#include<iostream>
using namespace std;
void main()
{int a,b,c=0,d=0,f,g,h,t=0;
a=1;b=1;
cout<<a<<" "<<b<<endl;
for(d=3;d<=20;d++)
{c=a+b;
        a=b;
b=c;
cout<<" "<<c<<endl;
if(c % 3==0) d=d+c;}
system("pause");}
```

```
//Eskişehir  elektrik dağıtım firması EDAŞ bir abonenin ödeyeceği elektrik ücretini elektrik kullanım yeri
//(Mesken:0, İşyeri:1) türüne göre aşağıdaki gibi hesaplatmak istiyor;
//□ İşyerlerinde;
//o her kilowat-saat (kwh) için 2 TL,
//□ Meskenlerde ise
//o İlk 50 kwh‘ e kadar her bir kwh 1 TL,
//o Sonra 100 kwh’e kadar her bir kwh 2 TL,
//o Fazlasına (100 kwh’den fazla) her bir kwh için ise 3 TL, ücretlendirme yapmaktadır.

//Buna göre klavyeden ilk ve son sayaç okuma değerleri ve kullanım yeri türü girildikten sonra bir abonenin
//ödeyeceği elektrik faturasını hesaplayan program
#include<iostream>
#include<cmath>
using namespace std;
void main()
{
int kt, io, so, syc;
double Ucret;
cout<<"Gir sayacin ilk okuma degerini..:";
cin >>io;
cout<<"\nGir sayacin son okuma degerini...:";
cin >>so;
syc=so-io;
cout<<"\nKullanim turunu gir..:";
cin >>kt;
if (kt==1)
Ucret=syc*2;
else if (syc <= 50)
Ucret=syc*1;
else if (syc > 50 && syc <=100)
Ucret=50 + (syc-50)*2;
else
Ucret=50 + 100 + (syc-100)*3;
cout<<"\nUcret..:"<< Ucret<< "TL"<<endl;
system("pause");
}
```

```
//Klavyeden girilen bir tamsayının hanelerindeki en büyük sayıyı bulan C kodunu yazınız.
//Örnek çıktı lar :   Bir sayı giriniz:   1905       Bir sayı giriniz:   327045
//   En büyük hanenin değeri = 9      En büyük hanenin değeri = 7
#include<conio.h>
#include<iostream>
using namespace std;
void main()
 { int sayi, b, enb=0;
    cout<<"Bir sayi girin : ";
       cin>>sayi;
     while(sayi>0) {
       b = sayi % 10;
        if(b>enb)  enb = b;
       sayi /= 10;
    }
     cout<<"En buyuk hanenin degeri :"<<enb;
    getche();
 }


// Bir öğrenci, bir romanın her gün bir önceki gün okuduğu sayfadan 5 sayfa fazlasını okumaktadır.  İlk gün   10
//sayfa okuyarak başlayan öğrencinin 1.000 sayfalık bir romanı kaç günde bitireceğini bulan programı C   dilinde
//kodlayınız.
#include<conio.h>
#include<iostream>
using namespace std;
void main()
 {  int sayfa=10, okunansayfa=0, gun = 0;
 while(okunansayfa < 1000)   {
     okunansayfa += sayfa;
     sayfa += 5;
     gun ++; }
cout<<gun<<". gun icinde kitap bitmis olur";getche(); }

//Y=x2+5x*sin(x) fonksiyonunun, Xin 0 ile 17 aralığındaki 0,5 lik artışlarla Y sonucunu veren
//C++ programını yazınız.
#include<iostream>
#include<cmath>
using namespace std;
void main()
{
   double Y;
   for(double x=0; x<=17; x=x+0.5)
   {        Y=pow(x,2)+5*x*sin(x);
      cout<<"x = "<<x<<" icin Y = "<<Y<<endl<<endl;
   }
  system("pause");}
```

Hatırlatma tip uyumluluğu c programlama dilleri için aranan özelliklerdendir

```
//iki sayının bölümünü düşünürsek
#include <iostream>
#include <math.h>
using namespace std;
 void main()
 {        int c=3;
float j=2.0;
float k=c/j;
         cout<<"k="<<k<<endl;
         }Tip farklı tanımlandığından sonuç 1.5
```

```
//iki sayının bölümünü düşünürsek
#include <iostream>
#include <math.h>
using namespace std;
 void main()
 {        int c=3;
int j=2.0;
int k=c/j;
         cout<<"k="<<k<<endl;
                 }
```

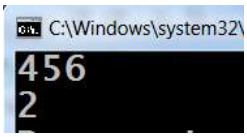
sonuç 1 olurdu.

GLOBAL DEĞİŞKEN KULLANIMI

- C++, bu tür durumlarda, global değişkene de erişime izin verir. Değişkenin önüne kapsam çözümleyici operatör olan **::** işaretinin konulması halinde kullanılanın yerel değil global değişken olduğu anlaşılır.
- Örnek:

```
      //global işlemi
#include <iostream>
      using namespace std;
int miktar1 = 2;  // Global Degisken
void main()
      {           int miktar1 = 456; // Yerel Degisken
    cout << miktar1;
      cout << endl;
    cout << ::miktar1<<endl;  }
```

Çıktısı



- İÇ İÇE GÖMÜLMÜŞ KAPSAM ALANLARINDA :: BİR ÜST KAPSAM ALANINDAKİ DEĞİŞKENLERE DEĞİL GLOBAL DEĞİŞKENLERE ERİŞİM SAĞLAR.

```
//cpp için for döngüsü ile faktor hesabı
#include <iostream>
using namespace std;
void main()
{int fac, sayi;
cout<<"Sayıyı giriniz: ";
cin>>sayi;
fac=1;
for (int j=1; j<=sayi; j++)
{fac=fac*j;
}
cout<<"Sonuc: \a"<< fac;}
```

```
//klavyeden girilen üç sayının en büyüğünü bulma
#include <iostream>
using namespace std;
void main() {
int a,b,c ;
bas:cout<<(" uc adet sayi giriniz")<<endl;
cin>>a>>b>>c;//
   if ((a==b) || (a==c) || (b==c))
   { cout<<"esit sayi girildi\n";
   goto bas;  }
if (a > b)
            if (a > c)
            cout << "enbuyuk a="<<a<<endl;
       else
           cout << "enbuyuk c="<<c<<endl;

        else if (b > c)
       cout << "enbuyuk b="<<b<<endl;

 else
      cout << "enbuyuk c="<<c<<endl;}
```

```
//tekcift.cpp
//adet toplamları
#include <iostream>
using namespace std;
void main() {
int sayi,s=0;
int tek=0,cift=0;
while (s<4)
{
cout<< "Bir sayı giriniz:";
cin >> sayi;
s=s+1; //while döngüsünü döndürmek icin
if (sayi %2==1 ){
cout <<s<<"=tek"<<endl;
tek=tek+1;}
else
{
cout <<s<< "=çift"<< endl;
cift=cift+1;}
}
cout<<"teksayi adedi="<<tek<<endl;
cout<<"cift sayi adedi="<<cift<<endl;
}
```

?: Komutu

Bu komut, yalnızca C++ a özgüdür. Kendine özgü bir yazılımı ve mantıksal kullanımı vardır ve if-else komutunun kısaltılmışıdır. Fakat, tüm if-else komutları yerine kullanılmaz. Yalnızca karşılaştırma sonrası, tek komut kullanılan uygulamalar için geçerlidir. Bu söylediklerimizi örnek üstünde gösterelim:

//tekcift.cpp yi bu sefer ?: ile yapıyoruz

```
#include <iostream>              // cin,cout,endl
using namespace std;
void main(){
int sayi;
cout<< "Bir sayı giriniz:"; //cout
cin >> sayi;
cout << (sayi %2==1 ? "tek" : "çift") ;}
```

Kullanış biçimi: cout << (sayi %2==1 ? "tek" : "çift") ;

Şimdi ise ? komutu ile ne gatif pozitif sayıların ayrımını yapalım.

```
#include <iostream>             // cin,cout,endl
using namespace std;
void main(){
  int sayi;
  cout << "Bir sayı girin :";
  cin >> sayi;
  cout<<( sayi > 0 ? "pozitif sayi girdiniz" : " 0 yada negatif sayı girdiniz") << endl;}
```

Havaya atılan bir paranın yazı tura olasılığına bakalım.

```
//istenilen miktarda havaya atılan paranın yazı tura gelme olsalığı
#include<iostream>
#include<time.h>
using namespace std;
void main()
{    int count, i;
    int number;
int heads = 0, tails = 0;
    cout << "Kac kez atilacak "; cin >> count;
    srand(time(0)); //randomize
    for (i = 1; i <= count; i=i+1)
        {            number =  rand()%2;
            if (number < 1)
            {            cout << number  ;
                  cout << "Yazi" << endl;
           ++tails;  }
        else    {  cout << number  ;
                  cout << "Tura" << endl;
            heads++;    }    }
      cout << "  Yazi sayisi: " <<
tails<< ", Yuzdesi: %" << 100.0 * tails / count << endl;
   cout << "  Tura sayisi: " << heads << ", Yuzdesi: %"
<< 100.0 * heads / count << endl;    }
```

//Asal sayı örneği klavyeden girilen sayının asal olup olmadığının bulunması

```
#include <time.h>
#include<iostream>
using namespace std;
void main()
{     int  n=0 ;
      int i=0,y=0;
            cout<<"Bir sayi giriniz: "<<endl;
      cin>>n;
      for (i=2 ; i<n ; i++)
                  if ( n%i ==0 )break;
      if ( i== n)
            cout<<n<<" sayisi asaldir."<<endl;
      else
            cout<<i<<" sayisi asal degildir."<<endl;}
```

Farklı yapıda asal bulma

Asal sayılar, 1 ve kendinden başka sayılara kalansız bölünemeyen sayılardır (3,5,7,11,13,15,... gibi). O halde, ekrandan okunan sayının, 2'den başlayarak n-1'e kadar bütün sayılarla kalansız bölünüp bölünmediği test edilmelidir.

Örnek kodlarda, yukarıdaki bilgilerle if-goto yapısı birleştiriliyor. s1: i++; satırında i'nin değeri artırılıyor ve if (i<n) goto s1; komutuyla bu satıra i, n'den küçük olduğu sürece yönlendirme yapılıyor. Ayrıca, if (n%i == 0) goto s2; satırıyla, kalansız bölme olması durumunda sayının asal olmadığı anlaşılıyor ve çevrim kesilip s2: etiketli satıra yönlendirme yapılıyor. Bu tekrarlar, i, n'e eşitlenene kadar sürerse if (i<n-1) goto s1; komutu çalışmayıp alt satıra inecektir ve hala kalansız bölme gerçekleşmediğinden, alt satırdaki cout<< " \n\n Asal sayı ";

goto s3; satırları çalışacaktır.

```
#include <iostream>
#include <locale.h>
using namespace std;
void main()
{         int n,i=1;
  clrscr( );
 cout<< "Sayıyı giriniz= ";
 cin>> n;
 s1: i++;
if ( n % i == 0 ) goto s2 ;
 if (i <n-1) goto s1 ;

cout<< "\n\n Asal sayı ";
goto s3 ;

s2: cout<< " \n\n Asal değil ";
s3:cout<<"işlem bitti";}
```

//Obeb girilen iki sayının ortak bölenlerinin en büyüğünü bulma
Örnekte, okunan iki sayının ortak katlarının en büyüğü (OBEB) bulunuyor. OBEB en fazla, sayılardan büyük olanın değeri kadar olabilir. Bundan dolayı ilk çevrim sonrası if-else yapısıyla büyük olan sayı bulunarak i'ye eşitleniyor. i ikinci çevrimde de başlangıç değerini oluşturacak ve çevrim bu sayıdan sıfıra doğru küçülecektir. OBEB değeri, her iki sayıya da tam bölünebilir. Bu sayı if cümlesiyle kontrol ediliyor ve ilk sağlandığı değer OBEB olarak cout<< komutuyla ilan ediliyor. break; komutu, ikinci çevrimin de dışına çıkılmasını sağlıyor.

```
//iki sayının ortak bolenlerinin en büyüğünü bulma
#include<iostream>
using namespace std;
void main()
{
   int sayi1, sayi2;
   int a, b, r = 1,x=0;
bas:sayi1=0:sayi2=0;
          cout<<"\nSayilari yaziniz: ";
   cin>>sayi1>>sayi2;
if(sayi1>sayi2)a=sayi1;else a=sayi2;
if(sayi1>sayi2)b=sayi2;else b=sayi1;
   while (b > 0) {  //8 ve 6    r=8/6 için 1. donus r=2 a=6 b=2
      r = a % b;             //  r=6/2  2. donus r=0 a=2 b=0
      a = b;
      b = r;
   }
cout<< sayi1<<"ve"<<sayi2<<" sayilarin en buyuk ortak boleni:"<< a<<endl;}
```

Bilindiği gibi, ikinci derece bir denklem,

$$ax^2 + bx + c$$

yapısında olup, kökleri,

$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$$

denklemiyle bulunur. Burada delta,

$$\Delta = b^2 - 4ac$$

eşitliğinden hesaplanır.

Karekök içindeki ifadenin sıfırdan küçük, eşit veya büyük olmasına göre değişik sonuç deklarasyonu yapılmalıdır. Kodlar
incelendiğinde, önce katsayıların okunduğu, sonra deltanın (karekök içi) hesaplandığı ve onları takip eden üç duruma ait if-else blokları görülüyor. Eğer bir şart cümlesi sadece bir komutu kontrol ediyorsa, { } küme parantezli bloklama yapmak gerekli değildir. Ancak birden fazla işlem yapılacaksa, mutlaka kullanılmalıdırlar. Nitekim, ilk if sorgusuna bağlı gerçek kök olmadığını deklare eden cout komutu, bir blok içinde değildir. Ama, diğer iki şart ifadesini bloklar takip etmiştir.

```
//  ikinci derece denklem köklerini bulan program *

#include <iostream.h>
#include <conio.h>
#include <math.h>

float delta, a, b, c, kok1, kok2;

void main( )
{ clrscr( );
 cout<< x2'nin katsayısı= "; cin>>a;
 cout<< "\n x'in katsayısı= " ; cin>>b;
 cout<< "\n Sabit değer= "; cin>>c;
 delta = b * b – 4 * a * c;
 // ilk durum; delta sıfırdan küçük. Gerçek kök yok.
 if (delta < 0) cout<<"\n\n Gerçek kök yok! \a\a" ;
// kinci durum; delta sıfıra eşit. Tek gerçek kök var
 else if (delta == 0)
 { kok1 = (-1 * b) / (2 * a);
 cout<< "  \n\n Tek kök var="<< kok1; }
// Üçüncü durum, iki gerçek kök var.
 else if (delta > 0)
 { kok1 = (-1 * b + sqrt (delta) ) / (2 * a);
 kok2 = (-1 * b - sqrt (delta) ) / (2 * a);
 cout<< " \n\n 1. kök ="<< kok1;
 cout<< "  \n 2. kök = " << kok2; }

}
```

DİZİLER

Bellekte ardışık bir biçimde bulunan ve aynı türden nesnelerin oluşturduğu veri yapısına dizi (array) denir. Dizilerin kullanılmasını gerektiren iki önemli özellikleri vardır:

1. Dizi elemanları bellekte ardışıl (contiguous) olarak bulunurlar.
2. Dizi elemanları aynı türden nesnelerdir.

Diziler bileşik nesnelerdir. Yani bir dizinin tanımlanmasıyla birden fazla sayıda nesne birlikte tanımlanabilir. (Bileşik sözcüğü ingilizce "aggregate" sözcüğünün karşılığı olarak kullanılmıştır.)
10 elemanlık bir dizi tanımlamak yerine, şüphesiz farklı isimlere sahip 10 ayrı nesne de tanımlanabilir. Ama 10 ayrı nesne tanımlandığında bu nesnelerin bellekte ardışık olarak yerleşmeleri garanti altına alınmış bir özellik değildir. Oysa dizi tanımlamasında, dizinin elemanı olan bütün nesnelerin bellekte ardışık olarak yer almaları garanti altına alınmış bir özelliktir.

Dizilerde bir veri türü olduğuna göre dizilerin de kullanılmalarından önce tanımlanmaları gerekir.

Dizilerin Tanımlanması

Dizi tanımlamalarının genel biçimi:
<tür> <dizi ismi> [<eleman sayısı>];
Yukaridaki gösterimde köşeli parantez eleman sayısının seçimlik olduğunu değil, eleman sayısı bilgisinin köşeli parantez içine yazılması gerektiğini göstermektedir.

tür : Dizi elemanlarının türünü gösteren anahtar sözcüktür.
dizi ismi : İsimlendirme kurallarına uygun olarak verilecek herhangi bir isimdir.
eleman sayısı : Dizinin kaç elemana sahip olduğunu gösterir.

Örnek Dizi Bildirimleri:

```
double a[20];              /* a, 20 elemanlı ve elemanları double türden olan bir dizidir*/
float ave[10];             /* ave 10 elemanlı ve her elemanı float türden olan bir dizidir. */
unsigned long total[100];  /* total 100 elemanlı ve her elemanı unsigned long türden olan bir dizidir
char path[80];             /* path 80 elemanlı ve her elemanı char türden olan bir dizidir. */
```

Tanımlamada yer alan eleman sayısının mutlaka tamsayı türlerinden birinden sabit ifadesi olması zorunludur. (Sabit ifadesi [constant expression] tanımını hatırlayalım; değişken ve fonksiyon çağırımı içermeyen, yani yalnızca sabitlerden oluşan ifadelere, sabit ifadesi denir.)

```
int dizi[x];      /* x dizisinin bildirimi derleme zamanında hata olusturur .*/
int dizi[5.];     /* gerçek sayı türünden sabit ifadesi olduğu için derleme zamanında hata olusturur . */

int sample[10 * 20]        /* sample dizisinin bildirimi geçerlidir. Eleman sayısını gösteren ifade sabit ifadesidir. */
```

Dizi bildirimlerinde eleman sayısı yerine sıklıkla sembolik sabitler kullanılır:

```
#define     MAXSIZE     100
...
int dizi[MAXSIZE];         /* geçerli bir bildirimdir */
...
```

Diğer değişken bildirimlerinde olduğu gibi, virgül ayıracıyla ayrılarak, birden fazla dizi tek bir tür belirten anahtar sözcükle tanımlanabilir.

```
int x[100], y[50], z[10];
```

x, y ve z elemanları **int** türden olan dizilerdir.

Dizi tanımlamaları diğer değişken tanımlamaları ile kombine edilebilir.

```
int a[10], b, c;
```

a **int** türden 10 elemanlı bir dizi, b ve c **int** türden nesnelerdir.

Dizi elemanlarının her biri ayrı birer nesnedir. Dizi elemanlarına index operatörüyle [] ulaşılabilir. Index operatörü bir gösterici operatörüdür. Göstericiler konusunda ayrıntılı bir şekilde ele alınacaktır.

İndex operatörünün operandı dizi ismidir. (Aslında bu bir adres bilgisidir, çünkü dizi isimleri adres bilgisi belirtirler.) Köşeli parantez içinde dizinin kaçıncı indisli elemanına ulaşacağımızı gösteren bir tamsayı ifadesi olmalıdır.

C dilinde dizilerin ilk elemanı sıfırıncı indisli elemandır.

a[n] gibi bir dizinin ilk elemanı a[0] son elemanı ise a[n - 1] dur.

Örnekler:

dizi[20] /* a dizisinin 20. indisli yani 21. sıradaki elemanı. */
ave[0] /* ave dizisinin 0. indisli yani birinci sıradaki elemanı */
total[j] /* total dizisinin j indisli elemanı */

Görüldüğü gibi bir dizinin n. elemanı ve bir dizinin n indisli elemanı terimleri dizinin farklı elemanlarına işaret eder. Bir dizinin n indisli elemanı o dizinin n + 1 . elemanıdır.

```
int a[20];
int k = 10;
int i =5;

a[k++] = 100; /* a dizisinin 10 indisli elemanına yani 11. elemanına 100 değeri atanıyor. */
a[--i] = 200;          /* a dizisinin 4 indisli elemanına yani 5. elemanına 200 değeri atanıyor. */
```

İndex operatörünün kullanılmasıyla artık dizinin herhangi bir elemanı diğer değişkenler gibi kullanılabilir.

Örnekler:

```
a[0] = 1;/* a dizisinin ilk elemanına 0 değeri atanıyor. */
cout<< sample[5];                    /* sample dizisinin 6. elemanı yazdırılıyor. */
++val[3];                                    /* val dizisinin 4. elemanı 1 artırılıyor. */
val[2] = sample[2];          /* val dizisinin 3. elemanına sample dizisinin 3. elemanı atanıyor. */
```

Dizilere ilk değer verilmesi (array initialization)

Dizilere ilk değer verme işleminin genel şekli aşağıdaki gibidir :

<tür> <dizi ismi>[[uzunluk]] = {d1, d2, d3.....};

Örnekler :

```
double sample[5] = {1.3, 2.5, 3.5, 5.8, 6.0};
char str[4] = {'d', 'i', 'z', 'i'};
unsigned[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Dizilere yukarıdaki gibi ilk değer verildiğinde, verilen değerler dizinin ilk elemanından başlayarak dizi elemanlarına sırayla atanmış olur.

Dizilerin tüm elemanlarına ilk değer verme zorunluluğu yoktur. Dizinin eleman sayısından daha az sayıda elemana ilk değer verilmesi durumunda kalan elemanlara otomatik olarak 0 değeri atanmış olur. Bu kural hem yerel hem de global diziler için geçerlidir.

Bu durumda bütün dizi elemanlarına 0 değeri atanmak isteniyorsa bunun en kısa yolu :
int a[20] = {0};
yalnızca dizinin ilk elemanına 0 değeri vermektir. Bu durumda derleyici dizinin kalan elemanlarına otomatik olarak 0 değeri atayacaktır.

Dizi elemanlarına ilk değer verilmesinde kullanılan ifadeler, sabit ifadeleri olmalıdır.

```
int a[10] = {b, b + 1, b + 2};
```

gibi bir ilk değer verme işlemi derleme zamanı hatası oluşturur.

Bir diziye ilk değer verme işleminde dizi eleman sayısından daha fazla sayıda ilk değer vermek derleme zamanında hata oluşumuna neden olur :

int b[5] = {1, 2, 3, 4, 5, 6}; /* error */
yukarıdaki örnekte b dizisi 5 elemanlı olmasına karşın, ilk değer verme ifadesinde 6 değer kullanılmıştır.

dizi elemanlarına ilk değer verme işleminde dizi uzunluğu belirtilmeyebilir, bu durumda derleyici dizi uzunluğunu verilen ilk değerleri sayarak kendi hesaplar ve dizinin o uzunlukta açıldığını kabul eder. Örneğin :

```
int sample[] = {1, 2, 3, 4, 5};
```

derleyici yukarıdaki ifadeyi gördüğünde sample dizisinin 5 elemanlı olduğunu kabul edecektir. Bu durumda yukarıdaki gibi bir bildirimle aşağıdaki gibi bir bildirim eşdeğer olacaktır:

int sample[5] = {1, 2, 3, 4, 5};

başka örnekler :

char name[ ] = {'N', 'e', 'c', 'a', 't', 'i' ' ', 'E', 'r', 'g', 'i', 'n', '\0'};
unsigned short count[ ] = {1, 4, 5, 7, 8, 9, 12, 15, 13, 21};
derleyici name dizisinin uzunluğunu 13, count dizisinin uzunluğunu ise 10 olarak varsayacaktır.

#define komutu

Alıntıdır.

Adres: http://www.yazilimogren.com/2009/03/c-programlama-define-onislemci-komutu/

#define komutu, adından da anlaşılabileceği gibi tanımlama işlemleri için kullanılır. Tanımlama komutunun kullanım mantığı çok basittir. Bütün yapmamız gereken, neyin yerine neyi yazacağımıza karar vermektir. Bunun için #define yazıp bir boşluk bıraktıkan sonra, önce kullanacağımız bir isim verilir, ardından da yerine geçeceği değer.

Altta ki program, PI sembolü olan her yere 3.14 koyacak ve işlemleri buna göre yapacaktır:

```
/* Çember alanını hesaplar */

#include<stdio.h>
#define PI 3.14
int main( void )
{
        int yaricap;
        float alan;
        cout<< "Çemberin yarı çapını giriniz> ";
        cin>>yaricap ;
        alan = PI * yaricap * yaricap;
        cout<< "Çember alanı: ", alan ;

}
```

Gördüğünüz gibi, PI bir değişken olarak tanımlanmamıştır. Ancak #define komutu sayesinde, PI'nin aslında 3.14 olduğu derleyici (compiler) tarafından kabul edilmiştir. Sadece #define komutunu kullanarak başka şeylerde yapmak mümkündür. Örneğin, daha önce dediğimizi yapalım ve cout yerine, ekrana_yazdir; cin yerine de, deger_al isimlerini kullanalım:

```
/* Yarıçapa göre daire alanı hesaplar */

#include<stdio.h>
#define PI              3.14
#define ekrana_yazdir
#define deger_al
int main( void )
{
        int yaricap;
        float alan;
        ekrana_yazdir( "Çemberin yarı çapını giriniz> " );
        deger_al( "%d", &yaricap );
        alan = PI * yaricap * yaricap;
        ekrana_yazdir( "Çember alanı: %.2f\n", alan );

}
```

#define komutunun başka marifetleri de vardır. İlerki konularımızda göreceğimiz fonksiyon yapısına benzer şekilde kullanımı mümkündür. Elbette ki, fonksiyonlar çok daha gelişmiştir ve sağladıkları esnekliği, #define tutamaz. Bu yüzden

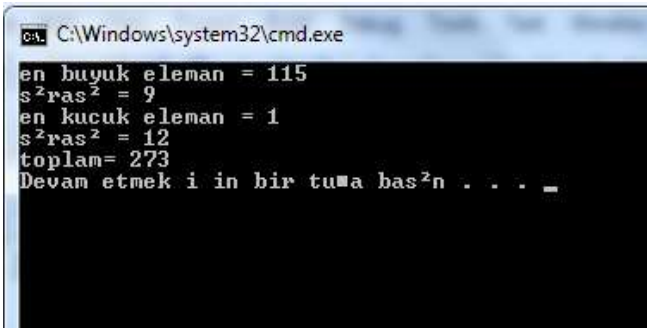
#define ile yapılacakların sınırını çok zorlamamak en iyisi. Ancak yine de bilginiz olması açısından aşağıda ki örneğe göz atabilirsiniz:

```
/* Istenen sayida, "Merhaba" yazar */

#include<stdio.h>
#define merhaba_yazdir( x ) int i; for ( i = 0; i < (x); i++ ) cout "Merhaba\n" ;
int main( void )
{
        int yazdirma_adedi;
        cout "Kaç defa yazdırılsın> ";
        cin>>yazdirma_adedi ;
        merhaba_yazdir( yazdirma_adedi );
        }
```

dizilerle ilgili uygulama örnekleri:
//Dizinin en küçük elemanını,en buyuk elemanını ve yerlerini bulmak

```
#include <iostream>
#include <time.h>
//#define SIZE 15
using namespace std;
void main()
{
int dizi[15] = {100, 2, 3, 4, 5,6,24, 2, 115,8,3,1}; //anlat
int min=100, k,indis,Toplam=0,b,max=0,mindis;
//min = dizi[0]; //anlat
for (k = 0; k < 12; ++k)
{Toplam=Toplam+dizi[k];
if ( dizi[k]<=min ){
min = dizi[k];
indis = k+1;} }//anlat
//buyuk islemi
for (b = 0; b < 12; ++b)
{
if ( dizi[b]>max ){
max = dizi[b];
mindis = b+1;} }//anlat
cout<<"en buyuk eleman = \n", max);
cout<<"sırası = \n", mindis);
cout<<"en kucuk eleman = \n", min);
cout<<"sırası = \n", indis);
cout<<"toplam= \n", Toplam);
}
```

çıktısı

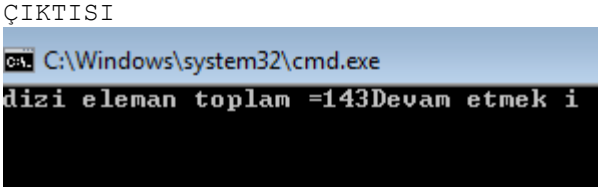


****

//dizinin elemanlarının toplamını bulma

```
// bir ismin kendisini bir par‡as□n□ ve bir karakterini g”sterir.
#include <iostream>
#include <stdio.h>
using namespace std;
#define SIZE 10
void main()
{ int dizi[SIZE] = {12, 9, -34, 45,00,
 89, 24, -2};
int toplam=0 ; //hatırlat
//int k;
for (int k = 0; k < SIZE; ++k)
toplam += dizi[k]; //toplam=toplam+dizi[k];
//cout<<"dizi eleman toplam ="<< toplam;
Cout<<"dizi eleman toplam="<<toplam; }
```

ÇIKTISI

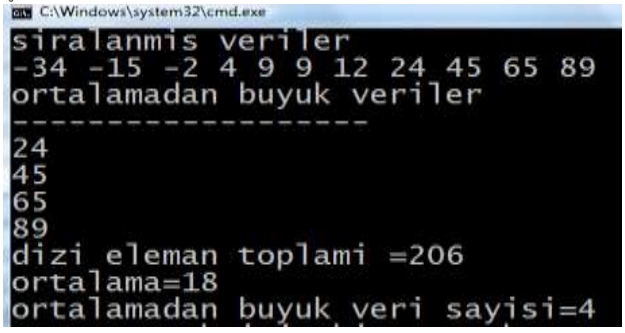


****

//Dizinin elemanlarını küçükten büyüğe sıralama (22-04-2013 tarihinde güncellendi)

```
#include <iostream>
using namespace std;
#define SIZE 11 //hatırlat
void main()
{ int dizi[SIZE] = {12, 9, -34, 45, 65,89, 24, -2, -15,9,4.5};
int toplam=0 ,tut,u,i,k,say=0; //hatırlat
for ( i = 0; i < SIZE; i++)
for ( k = 0; k < SIZE; k++)
if (dizi[i] < dizi[k]) {
tut = dizi[i];
dizi[i] = dizi[k];
dizi[k] = tut;}
for ( i = 0; i <SIZE; i++){
toplam=toplam+dizi[i];//dizi toplamı
{cout<<dizi[i]<<" "; }}
//ortalamadan büyükleri küçükten büyüğe sıralaam
cout<<"ortalamadan buyuk veriler"<<endl;
cout<<"-------------------"<<endl;
for ( u = 0; u <SIZE; u++)
if(dizi[u]>toplam/11){cout<<dizi[u]<<endl;say++;}
cout<<"dizi eleman toplami ="<< toplam<<endl;
cout<<"ortalama="<<toplam/11<<endl;
cout<<"ortalamadan buyuk veri sayisi="<<say<<endl;}
```

Çıktısı

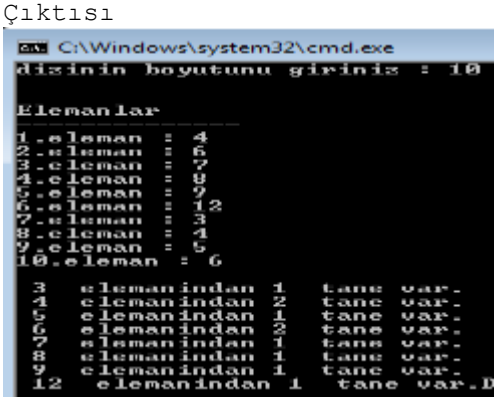


//Farklı bir sıralama

```
#include <iostream>
using namespace std;
void main(){
```

```
  signed int n=15;
      int i,d[100];
  int e[1000]={0};
cout<<"dizinin boyutunu giriniz : ";
cin>>n;
  cout<<"\n\nElemanlar\n--------------\n";
     // bilgi girisi
  for( i=0; i<n; i++ ){
  cout<<i+1<<".eleman : ";
    cin>>d[i];  }   //  en buyuk en kucuk elemanlari
  int enbuyuk, enkucuk;
  enbuyuk=d[0];
  enkucuk=d[0];
  for( i=0; i<n; i++ ){
    if( d[i] > enbuyuk )
   enbuyuk = d[i];   }
  for( i=0; i<n; i++ )
    { if( d[i] < enkucuk )
     enkucuk = d[i]; }
       for( i=0; i<n; i++ ){
       e[d[i]]++;  }
      // ekrana basalim
    for( int j=enkucuk; j<=enbuyuk; j++ ){
     if( e[j] > 0 ) { //basılmayan sayıları yoksaydık
   cout<<"\n"<<" "<<j<<" " <<" elemanindan"<<" "<< e[j]<<"tane var.";}}}
```

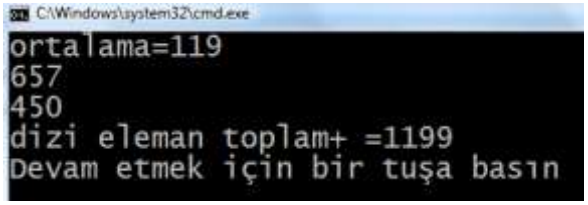
Çıktısı



//Dizinin elemanlarını küçükten büyüğe sıralanarak ortalamadan büyükleri göstermek(güncellendi)

```
#include <iostream>
using namespace std;
#define SIZE 10 //hatırlat
void main()
{ int dizi[SIZE] = {12, 9, -34, 450, 657, 89, 24, -2, -15,9};
int toplam=0 ,tut; //hatırlat
for (int i = 0; i < SIZE; i++)
for (int k = 0; k < SIZE; k++)
if (dizi[i] > dizi[k]) {
tut = dizi[i];
dizi[i] = dizi[k];
dizi[k] = tut;}
cout<<"ortalama="<<toplam/10<<endl;
for (int i = 0; i <SIZE; i++){
      toplam=toplam+dizi[i];  }
//ortalamadan büyükleri küçükten büyüğe sıralaam
for (int u = 0; u <SIZE; u++)
if(dizi[u]>toplam/10)cout<<dizi[u]<<endl;
cout<<"dizi eleman toplam+ ="<< toplam<<endl;}}
```

ÇIKTISI



KARAKTER DİZİLERİ

Karakter dizileri char türden dizilerdir. Karakter dizilerinin bazı ilave özellikler dışında diğer dizi türlerinden bir farkı yoktur. Char türden diziler daha çok içlerinde yazı tutmak için tanımlanırlar.

char s[100];

yukarıdaki tanımlamada s dizisi bütün elemanları char türden olan 100 elemanlı bir dizidir.
char türden bir dizi içinde bir yazı tutmak demek, dizinin her bir elemanına sırayla yazının bir karakterini atamak anlamına gelecektir. (Bu arada char türden bir dizinin içinde bir yazı tutmanın zorunlu olmadığını, böyle bir dizinin pekala küçük tamsayıları tutmak amacıyla da kullanılabileceğini hatırlatalım)

Yukarıda tanımlanan dizi içinde "Ali" yazısını tutmak isteyelim :

s[0] = 'A';
s[1] = 'l';
s[2] = 'i';

Şimdi şöyle bir problem ortaya çıkıyor. Dizi 100 karakterlik olmasına karşın biz dizi içinde 100 karakter uzunluğundan daha kısa olan yazılar da tutabiliriz. Peki biz yazıya tekrar ulaşmak istediğimizde bunu nasıl yapacağız? Yazının uzunluk bilgisini bilmiyoruz ki! Örneğin yazıyı ekrana yazdırmak istediğimizde, int türden dizilerde kullandığımız şekilde aşağıdaki gibi bir döngü kullanırsak :

```
for (k = 0; k < 100; ++k)
        putchar(s[k]);
```

bu döngü ile yalnızca ALi yazısı ekrana yazdırılmayacak dizinin diğer 97 elemanının da görüntüleri (rasgele değerler) ekrana yazdırılacak.

```
....
for (k = 1; k <= s[0]; ++k)
        putchar(s[k]);
```

C++ dilinde bu yaklaşım kullanılmaz.

C++ dilinde karakterler üzerinde işlemlerin hızlı ve etkin bir biçimde yapılabilmesi için “sonlandırıcı karakter” (NULL KARAKTER) kavramından faydalanılmaktadır.
Sonlandırıcı karakter ASCII tablosunun (ya da sistemde kullanılan karakter setinin) sıfır numaralı (‘\x0’ ya da ‘\0’) karakteridir. Dolayısıyla sayısal değer olarak 0 sayısına eşittir. Görüntüsü yoktur. Bu karakter DOS ve UNIX sistemlerinde sonlandırıcı karakter olarak kullanılmaktadır. stdio.h içerisinde NULL sembolik sabiti 0 olarak tanımlandığı için sonlandırıcı karaktere NULL karakter de denir. NULL karakter ‘0’ karakteri ile karıştırılmamalıdır.

‘0’ karakterinin ASCII sıra numarası 48’dir. Dolayısıyla tamsayı değeri olarak 48 değerine sahiptir.
‘\0’ karakterinin ASCII sıra numarası 0’dır. Dolayısıyla tamsayı değeri olarak 0 değerine sahiptir.

Hatırlat

```
#include<iostream>
using namespace std;
void main()
{    int i=0;
     char k;
     do
     {         i=i+1;
cout<<i<<" karsiligi= " <<char(i)<<endl;}
          while(i<255);
cout<<i;  }
```

char türden dizilere ilk değer verilmesi

char türden dizilere ilk değer verme işlemi (initializing) aşağıdaki biçimlerde yapılabilir .

Diğer türden dizilerde olduğu gibi virgüllerle ayrılan ilk değerler küme parantezi içinde yer alır :

char name[12] = {'A', 'T', 'A', 'L', 'A', 'Y'};

Diğer dizilerde olduğu gibi dizi eleman sayısından daha fazla sayıda elemana ilk değer vermek derleme zamanında hata oluşumuna neden olur.

char name[5] = {'A', 'T', 'A', 'L', 'A', 'Y','A'}; /* error */

Dizi eleman sayısı kadar elemana ya da dizi eleman sayısından daha az sayıda elemana ilk değer verilebilir. Daha az sayıda elemana ilk değer verilmesi durumunda ilk değer verilmemiş elemanlar diğer dizilerde olduğu gibi 0 değeriyle başlatılacaklardır. 0 değerinin NULL karakter olduğunu hatırlayalım :

char name [10] = {'A', 'l', 'i'};

...	...
'A'	name[0]
'l'	name[1]
'i'	name[2]
'\0'	name[3]
'\0'	name[4]
'\0'	name[5]
'\0'	name[6]
'\0'	name[7]
'\0'	name[8]
'\0'	name[9]

Dizi elemanlarına ilk değer verilirken dizi boyutu belirtilmeyebilir. Bu durumda derleyici dizi boyutunu verilen ilk değerleri sayarak saptar ve diziyi bu boyutta açılmış varsayar.
char name[ ] = {'A', 'l', 'i'};
yukarıdaki tanımlama deyimiyle derleyici name dizisinin 3 elemanlı olarak açıldığını varsayacaktır.
Boyut bilgisi vermeden, char türden dizilere tek tek ilk değer verilmesi durumunda, ve boyut bilgisi verilerek dizinin toplam eleman sayısı kadar elemana tek tek ilk değer verilmesi durumunda, derleyici NULL karakteri dizinin sonuna otomatik olarak yerleştirmeyecektir. Bu durumda eğer sonlandırıcı karakter özelliğinden faydalanılmak isteniyorsa, NULL karakter de verilen diğer ilk değerler gibi programcı tarafından verilmelidir. Örnek :

char name[ ] = {'A', 'l', 'i', '\0'};
char isim[7] = {'N', 'e', 'c', 'a', 't', 'i', '\0'};

Bu şekilde ilk değer vermek zahmetli olduğundan, ilk değer vermede ikinci bir biçim oluşturulmuştur. Karakter dizilerine ilk değerler çift tırnak içinde de verilebilir :

char name[ ] = "Ali";

bu biçimin diğerinden farkı derleyicinin sonuna otomatik olarak NULL karakteri yerleştirmesidir.

...	...
'A'	name[0]
'l'	name[1]
'i'	name[2]
'\0'	name[3]

yukarıdaki örnekte derleyici diziyi 4 elemanlı olarak açılmış varsayacaktır. Dizi eleman sayısından daha fazla sayıda elemana ilk değer vermeye çalışmak, bu biçimin kullanılması durumunda da derleme zamanında hata oluşumuna neden olacaktır.

char city[5] = "İstanbul"; /* error */

Bu durumun bir istisnası vardır. Eger tam olarak dizi eleman sayısı kadar dizi elemanına çift tırnak içinde ilk değer verilirse bu durum ERROR oluşturmaz. Derleyici bu durumda NULL karakteri dizinin sonuna yerleştirmez. Bu durum C++ dili

standartlarına yöneltilen eleştirilerden biridir. (C++ dilinde son kabul edilen standartlara göre bu durum da error oluşturmaktadır.)

char name[3] =”Ali”; /* C'de hata değil, C++'da hata */

...	...
‘A’	name[0]
‘l’	name[1]
‘i’	name[2]
...	buraya bu durumda null karakter yerleştirilmiyor.

'\0' karakteri karakter dizileri üzerinde yapılan işlemleri hızlandırmak için kullanılır. Örneğin int türden bir dizi kullanıldığında dizi elemanları üzerinde döngüleri kullanarak işlem yaparken dizi uzunluğunun mutlaka bilinmesi gerekmektedir. Ama char türden diziler söz konusu olduğunda artık dizi uzunluğunu bilmemiz gerekmez, çünkü yazının sonunda '\0' karakter bulunacağından, kontrol ifadelerinde bu durum test edilerek yazının sonuna gelinip gelinmediği anlaşılır. Ancak '\0' karakterin, karakter dizilerinde yazıların son elemanı olarak kullanılmasının dezavantajı da diziye fazladan bir karakter yani '\0' karakteri eklemek zorunluluğudur. Bu nedenle SIZE elemanlı bir diziye en fazla SIZE – 1 tane karakter girilmelidir.

klavyeden karakter dizisi alan ve ekrana karakter dizisi yazan standart C++ fonksiyonları

Daha önce gördüğümüz getchar, getch, ve getche fonksiyonları klavyeden tek bir karakter alıyorlardı. Yine putchar fonksiyonu ise tek bir karakteri ekrana yazıyordu. C++ dilinde birden fazla karakteri (bir stringi) klavyeden alan ya da birden fazla karakteri ekrana yazan standart fonksiyonlar bulunmaktadır.

Gets Fonksiyonu
gets fonksiyonu klavyeden karakter dizisi almakta kullanılan standart bir C++ fonksiyonudur. Kullanıcı karakterleri girdikten sonra enter tuşuna basmalıdır. Klavyeden girilecek karakterlerin yerleştirileceği dizinin ismini parametre olarak alır. daha önce de belirtildiği gibi dizi isimleri aslında bir adres bilgisi belirtmektedir. gets fonksiyonunun da parametresi aslında char türden bir adrestir. Ancak göstericilerle ilgili temel kavramları henüz öğrenmediğimiz için şimdilik gets fonksiyonunun char türden bir dizi içine klavyeden girilen karakterleri yerleştirdiğini kabul edeceğiz. Örneğin :
char name[20];

gets(name);

ile klavyeden enter tuşuna basılana kadar girilmiş olan tüm karakterler name dizisi içine sırayla yerleştirilirler. Klavyeden Necati yazısının girildiğini kabul edelim.

...	...
‘N’	name[0]
‘e’	name[1]
‘c’	name[2]
‘a’	name[3]
‘t’	name[4]
‘i’	name[5]
‘\0’	name[6]
...	...

gets fonksiyonu klavyeden girilen karakterleri diziye yerleştirdikten sonra dizinin sonuna NULL karakter (sonlandırıcı karakter) diye isimlendirilen 0 numaralı ASCII karakterini (ya da kullanılan karakter setinin 0 numaralı karakterini) koyar.
gets fonksiyonu dizi için hiç bir şekilde sınır kontrolu yapmaz. gets fonksiyonu ile dizi eleman sayısından fazla karakter girilirse, d,z, taşacağı için beklenmeyen sonuçlarla karşılaşılabilir. Bu tür durumlar göstericiler konusunda (gösterici hataları) detaylı olarak incelenecektir.
gets fonksiyonu ‘\0’ karakterini dizinin sonuna eklediği için SIZE uzunluğunda bir dizi için gets fonksiyonuyla alınacak karakter sayısı en fazla SIZE – 1 olmalıdır.

PUTS FONKSİYONU
puts fonksiyonu bir karakter dizisinin içeriğini ekrana yazdırmak için kullanılır. İçeriği yazdırılacak olan karakter dizisinin ismini parametre olarak alır. puts fonksiyonu karakter dizisini ekrana yazdıktan sonra imleci sonraki satırın başına geçirir.

```
char name[20];
gets(name);
puts(name);
```

yukarıdaki örnekte gets fonksiyonu ile klavyeden alınan karakter dizisi puts fonksiyonu ile ekrana yazdırılmaktadır.
karakter dizilerini ekrana yazdırmak için cout fonksiyonu da kullanılabilir.

Cout<< name;
ile
puts(name);
aynı işi yapmaktadır. Ancak cout fonksiyonu dizi içeriğini ekrana yazdırdıktan sonra imleci alt satıra taşımaz.
puts(name);
deyimi yerine aşağıdaki kod parçasını da yazabilirdik.

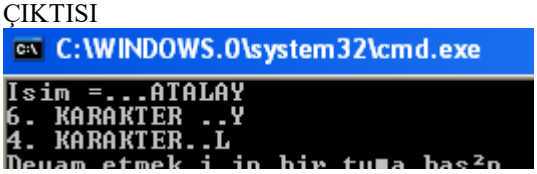
```
for (i = 0; name[i] != '\0'; ++i)
    putchar(s[i]);
putchar('\n');
```

puts fonksiyonu ve %s format karakteriyle kullanıldığında cout fonksiyonu, sonlandırıcı karakter görene kadar bütün karakterleri ekrana yazar. Bu durumda, herhangi bir şekilde NULL karakter ezilirse her iki fonksiyon da ilk sonlandırıcı karakteri görene kadar yazma işlemine devam edecektir.

Örneğin :
char city[ ] = "Ankara";
city[6] = '!';
deyimi ile sonlandırıcı karakter ortadan kaldırılırsa (ezilirse) :
puts(name);
şeklinde puts fonksiyonu çağırıldığında ekrana Ankara!yazıldıktan sonra tesadüfen ilk NULL karakter görülene kadar ekrana yazmaya devam edilir. puts ve cout fonksiyonları karakter dizilerini yazarken yalnızca NULL karakteri dikkate alırlar. karakter dizilerinin uzunluklarıyla ilgilenmezler.

```
// bir ismin kendisini bir parcasını ve bir karakterini g"sterir.
#include <iostream>
#include <stdio.h>
using namespace std;
void main()
{char isim[7];
system("cls");
   isim[0] = 'A';
   isim[1] = 'T';
   isim[2] = 'A';
   isim[3] = 'L';
   isim[4] = 'A';
   isim[5] = 'Y';
   //isim[6] = 0;     /* Bos karakter - kelimeyin sonu  */
   cout<<"Isim =..." <<isim<<endl;
     cout<<"6. KARAKTER .."<<isim[5]<<endl;
       cout<<"4. KARAKTER"<<isim[3]<<endl;
                          }
```

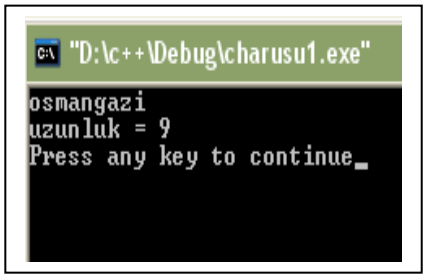
ÇIKTISI



Birinci öğretim 15.04.2013
karakter dizileriyle ilgili bazı küçük uygulama örnekleri

```
**********************************************************
#include <iostream>
#include <stdio.h>
using namespace std;
#define SIZE 10
void main()
{
      int k = 0;
      char s[50];
      gets(s);
      // while döngüsü yerine for döngüsü kullanılabilirdi
      while (s[k] != '\0')
            ++k;
      cout<<"uzunluk ="<< k;}
```

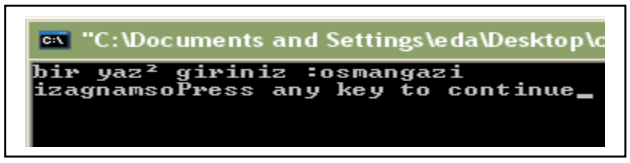
çıktısı



//yazının tersini yazdırma

```
#include <iostream>
#include <stdio.h>
using namespace std;
#define SIZE 10
void main()
{     char s[SIZE];
      int k;
      gets(s);
            for (k = 0; s[k] != '\0';)
            ++k;//sonsuz döngüyü engellemk için
      for (s[k] ; k >= 0; )
      {--k;
            putchar(s[k]);} }
```

çıktısı



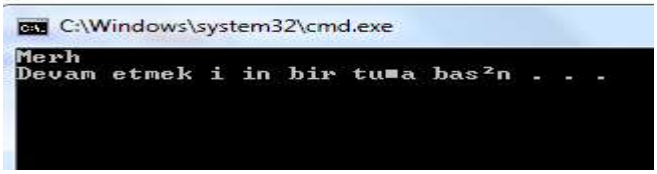
*** strcpy( ) ve strncpy( ) ile kelimey kopyalama**

Bir kelimeyi, bir başka kelimeye kopyalamak için *strcpy( )* fonksiyonunu kullanırız. **KELİMELER AYNI BOYUTTA OLMAK ZORUNDA DEĞİLDİR. Ancak kopya olacak kelime, kendisine gelecek kelimeyi alacak boyuta sahip olmalıdır.**

strcpy( ) fonksiyonunu bir örnekle görelim:

```
#include <iostream>
using namespace std;
void main(  )
{     char kaynak[5]="Merh";
      char kopya[9] = "12345423";
      strcpy( kopya, kaynak );
      cout<< kopya ; }
```

Çıktısı

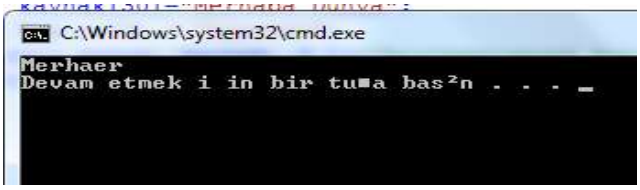


strncpy( ) fonksiyonu, yine kopyalamak içindir. Fakat emsalinden farklı olarak, kaç karakterin kopyalanacağı belirtilir. Protopi aşağıda verilmiştir:

Yukardaki örneği strncpy( ) fonksiyonuyla tekrar edelim:

```
#include <iostream>
using namespace std;
void main(  )
{     char kaynak[30]="Merhaba Dünya";
      char kopya[30] = "ayseler";
      strncpy( kopya, kaynak, 5 );//birleştirerek kopyalar
      cout<< kopya ; }
```

çıktısı



Yukardaki programı çalıştırırsanız, kopya isimli kelimeya sadece 5 karakterin aktarıldığını ve ekrana yazdırılan yazının "*Merhaer*" olduğunu görebilirsiniz.

*** strcmp( ) ve strncmp( ) ile kelime karşılaştırma**

strcmp( ) fonksiyonu, kendisine verilen iki kelimeyi birbiriyle karşılaştırır. Kelimeler birbirine eşitse, geriye 0 döner. Eğer ilk kelime alfabetik olarak ikinciden büyükse, geriye pozitif değer döndürür. Şayet alfabetik sırada ikinci kelime birinciden büyükse, geriye negatif değer dönmektedir. Bu dediklerimizi, daha iyi anlaşılması için bir tabloya dönüştürelim:

Dönen Değer	Açıklama
< 0	Kelime1, Kelime2'den küçüktür.
0	Kelime1 ve Kelime2 birbirine eşittir.
> 0	Kelime1, Kelime2'den büyüktür.

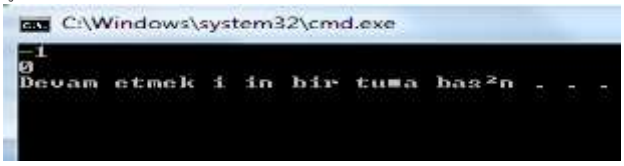
strncmp( ) için de aynı kurallar geçerlidir. Tek fark, karşılatırılacak karakter sayısını girmemizdir. *strcmp( )* fonksiyonunda iki kelimey, *null* karakter işareti çıkana kadar karşılaştırılır. Fakat *strncmp( )* fonksiyonunda, başlangıçtan itibaren kaç karakterin karşılaştırılacağına siz karar verirsiniz.

```
//rakamlar 48-57 arası karekterler 65-90 arası
```

Her iki fonksiyonu da kapsayan aşağıdaki örneği inceleyelim:

```
#include <iostream>
using namespace std;
void main(  )
{     int sonuc;
      char ilk_kelimey[40]="ahmet";
      char ikinci_kelimey[40]="ahsen";
      sonuc = strcmp( ilk_kelimey, ikinci_kelimey );
      cout<< sonuc;
      sonuc = strncmp( ilk_kelimey, ikinci_kelimey, 2 );
      cout<< sonuc;}
```

çıktısı

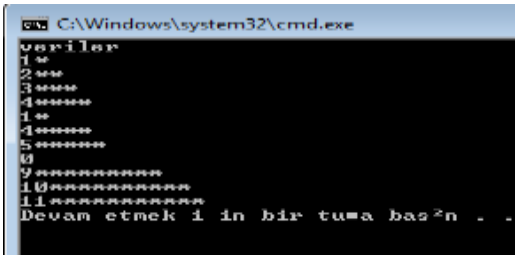


İlk önce çağrılan *strcmp( )*, null karakterini görene kadar bütün karakterleri karşılaştıracak ve geriye negatif bir değer döndürecektir. Çünkü "*ahmet*" kelimesi alfabede "*ahsen*" kelimesinden önce gelir; dolayısıyla küçüktür. Fakat ikinci olarak çağırdığımız *strncmp( )* geriye 0 değeri verecektir. Her iki kelimeyin ilk üç harfi aynıdır ve fonksiyonda sadece ilk üç harfin karşılaştırılmasını istediğimizi belirttik. Dolayısıyla karşılaştırmanın sonucunda 0 döndürülmesi normaldir.

****

//sıklıkları yıldız simgesi ile göstermek

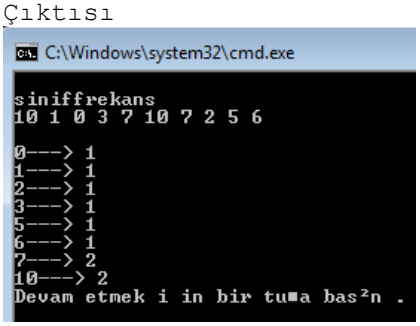
```
#define SIZE 10
#define lim 35  //yazılabilecek maksimum değer
#define boy 11 //sıklık miktarı
void main()
{     int cevap,sinif,j,i;
      int a[]={1,2,3,4,1,4,5,0,9,10,11};
      int frek[boy]={0};
      cout<<"veriler \n";
            for (i=0;i<=boy-1;i++){
cout<<a[i];
      for (j=1;j<=a[i];j++)
            cout<<'*';
            putchar('\n');} }
```



//Frekans hesabı ile ilgili örnek

```
#include <time.h>
#include <iostream>
using namespace std;
 void main()
{     int c,sinif,j,i;
            int p;
      int frek[100]={0};
      srand(time(NULL));
      cout<<"\n"<<"sinif"<<"frekans"<<endl;
      for (c=0;c<10;c++){
      p=rand() %101;
      cout<<p<<" ";
      ++frek[p];  }
      cout<<"\n";
      cout<<"\n";
      for (p=0;p<=99;p++)
      if (frek[p]>0)cout<<p << "---> "<<frek[p]<<endl;}
```

Çıktısı

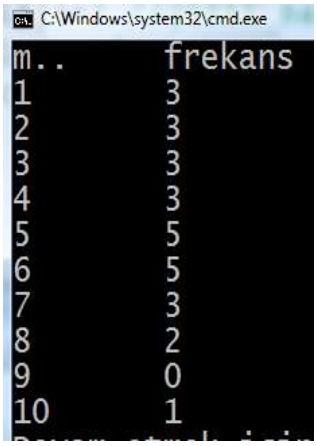


EK FREKANS İŞLEMİ (22-04-2013 TARİHİNDE EKLENDİ)

//datadan girilen sayıların frekansını bulma

```
#include <iostream>
using namespace std;
#define lim 28
#define boy 11
void main()
{ int cevap,m,j,i;
int v[]={1,2,3,4,5,6,5,6,4,5,6,7,10,3,2,3,5,6,7,1,8,8,6,5,4,1,7,2};
int frek[lim]={0}; //unutma
cout<<"m.."<<"\t"<<"frekans"<<endl;
for (i=0;i<lim;i++)
frek[v[i]]++; //gizli sayaç
for (m=1;m<=10;m++)
cout<<m<<"\t"<<frek[m]<<endl;}
```

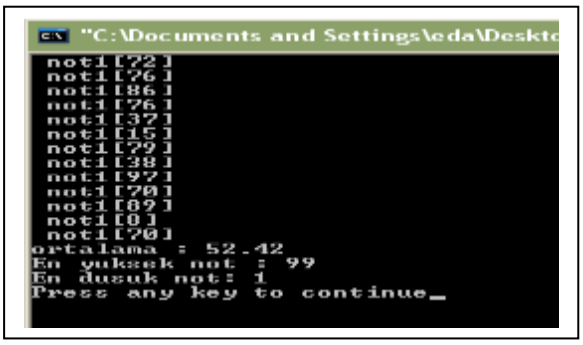
çıktısı



//60 kişilik bir sınıfta notları rasgele üretim ,otalama,enbüyük ve enküçük notu bulan programı yazınız.

```
#include <time.h>
#include <iostream>
#include <stdio.h>
using namespace std;
void main()
{ int not1[60];
int i, enBuyuk=not1[0];
int enKucuk=not1[0];
float ortalama = 0.0;
srand(time(NULL));
for (i=0;i<60;i++)
{not1[i] = rand()%101;
ortalama += not1[i];
cout<<"not1"<<"["<<i<<"]"<<"="<< not1[i]<<endl;}
cout<<"\nortalama="<< ortalama/60;
enBuyuk = enKucuk = not1[0];
for (i=0;i<60;i++)
{if (not1[i]>enBuyuk) enBuyuk = not1[i];
if (not1[i]<enKucuk) enKucuk = not1[i];}
cout<<"En yuksek not ="<<enBuyuk<<"En dusuk not: ="<<enKucuk;}
```

Çıktısı




```
//istenilen miktarda öğrencinin notunu klavyeden girip ,varyansını vb
#include <time.h>
#include <iostream> // cin,cout,endl
#include <math.h> // fabs,sqrt
using namespace std;
#define topsayi 100
void main()
{int score[topsayi];
int sayi = 0;
float ortalama, varyans, stdsapma, mutsapma;
```

```
float toplam = 0.0, sqr_toplam = 0.0, abs_toplam = 0.0;
int i = 0;
srand(time(0));
cout << "Kac ogrenci var? ";
cin >> sayi;
for (i = 0; i < sayi; i++)
 {score[i]= (rand()%100);
      cout << i + 1 << ". ogrencinin notu: "<<score[i]<<" ";
toplam = toplam + score[i];}
ortalama = toplam / sayi;
for (i = 0; i < sayi; i++)
 {sqr_toplam = sqr_toplam + (score[i] - ortalama) *
(score[i] - ortalama);
abs_toplam = abs_toplam + fabs(score[i] - ortalama);}
varyans = sqr_toplam / (sayi - 1);
stdsapma = sqrt(varyans);
mutsapma = abs_toplam / sayi;
cout << "Ortalama: " << ortalama << endl;
cout << "Varyans: " << varyans << endl;
cout << "Standart sapma: " << stdsapma << endl;
cout << "Mutlak sapma: " << mutsapma << endl;}
```

ÇOK BOYUTLU DİZİLER

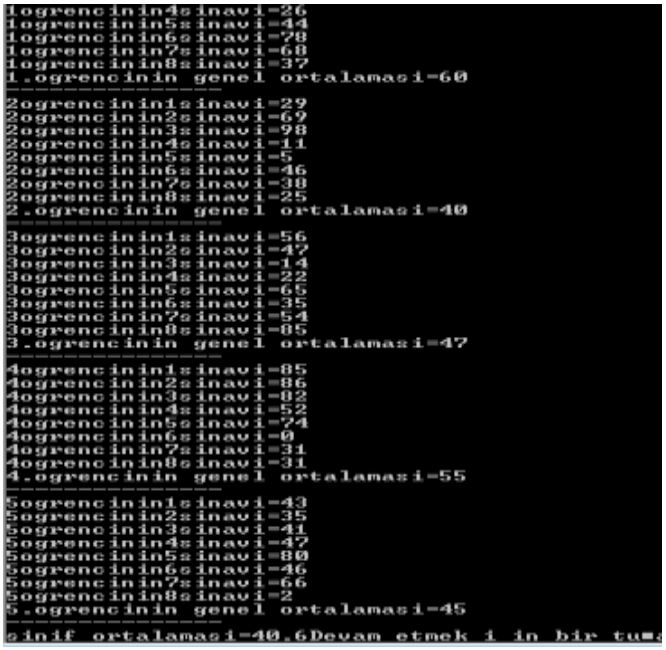
Önceki derslerimizde dizileri görmüstük. Kısaca özetleyecek olursak, belirlediğimiz sayıda değiskeni bir sıra içinde tutmamız, diziler sayesinde gerçeklesiyordu. Bu dersimizde, çok boyutlu dizileri inceleyip, ardından dinamik bellek konularına gireceğiz.

Simdiye kadar gördüğümüz diziler, tek boyutluydu. Bütün elemanları tek boyutlu bir yapıda saklıyorduk. Ancak dizilerin tek boyutlu olması gerekmez; istediğiniz boyutta tanımlayabilirsiniz. Örneğin 3x4 bir matris için 2 boyutlu bir dizi kullanırız. Ya da üç boyutlu Öklid uzayındaki x, y, z noktalarını saklamak için 3 boyutlu bir diziyi tercih ederiz.

Hemen bir baska örnek verelim. 5 kisilik bir öğrenci grubu için 8 adet test uygulansın. Bunların sonuçlarını saklamak için 2 boyutlu bir dizi kullanalım:

```
#include <iostream>           // cout,cin
#include <time.h>
using namespace std;
void main()
{     // 5 adet ogrenci icin 8 adet sinavi
      // temsil etmesi icin bir ogrenci tablosu
      // olusturuyoruz. Bunun icin 5x8 bir matris
      // yaratilmasi gerekiyor.
      int ogrenci_tablosu[5][8]={0};
      int i, j,k,m,top=0;
      float ort,sort=0;
      srand(time(0));
      for( i = 0; i < 5; i++ ) {
                        top=0; //anlat
            sort=0; //anlat
            for( j = 0; j < 8; j++ ) {
            cout<<( i + 1 )<<"ogrencinin"<<( j + 1 )<<"sinavi=";
                  ogrenci_tablosu[ i][j]=(rand()%101);
      top=top+ogrenci_tablosu[i][j];
                  cout<<ogrenci_tablosu[i][j];cout<<" - "<<endl;
      ort=top/8;
            sort=sort+ort;
            }
      cout<<i+1<<".ogrencinin genel ortalamasi="<<ort<<endl;
      cout<<"---------------"<<endl;           }
cout<<"sinif ortalamasi="<<sort/5;}
```

çıktısı



Bu programı çalıstırıp, öğrencilere çesitli değerler atadığımızı düsünelim. Bunu görsel bir sekle sokarsak, asağıdaki gibi bir çizelge olusur:
Tabloya bakarsak, 1.öğrenci sınavlardan, 80, 76, 58, 90, 27, 60, 85 ve 95 puan almıs gözüküyor. Ya da 5.öğrencinin, 6.sınavından 67 aldığını anlıyoruz. Benzer sekilde diğer hücrelere gerekli değerler atanıp, ilgili öğrencinin sınav notları hafızada tutuluyor.

Çok Boyutlu Dizilere Değer Atama

Çok boyutlu bir diziyi tanımlarken, eleman değerlerini atamak mümkündür. Asağıdaki örneği inceleyelim:

```
int tablo[3][4] = { 8, 16, 9, 52, 3, 15, 27, 6, 14, 25, 2, 10};
```

Diziyi tanımlarken, yukardaki gibi bir ilk değer atama yaparsanız, elemanların değeri asağıdaki gibi olur:

Satır 0 : 8 16 9 52
Satır 1 : 3 15 27 6
Satır 2 : 14 25 2 10

Çok boyutlu dizilerde ilk değer atama, tek boyutlu dizilerdekiyle aynıdır. Girdiğiniz değerler sırasıyla hücrelere atanır. Bunun nedeni de basittir. Bilgisayar, çok boyutlu dizileri sizin gibi düsünmez; dizi elemanlarını hafızada arka arkaya gelen bellek hücreleri olarak değerlendirir.
Çok boyutlu dizilerde ilk değer ataması yapacaksanız, değerleri kümelendirmek iyi bir yöntemdir; karmasıklığı önler. Örneğin yukarıda yazmıs olduğumuz ilk değer atama kodunu, asağıdaki gibi de yazabiliriz:

```
int tablo[3][4] = { 8, 16, 9, 52, 3, 15, 27, 6, 14, 25, 2, 10};
```

Farkedeceğiniz gibi elemanları dörderli üç gruba ayırdık. Bilgisayar açısından bir sey değismemis olsa da, kodu okuyacak kisi açısından daha yararlı oldu. Peki ya dört adet olması gereken grubun elemanlarını, üç adet yazsaydık ya da bir-iki grubu hiç yazmasaydık n'olurdu? Deneyelim...

```
int tablo[3][4] = { {8, 16}, {3, 15, 27} };
```

Tek boyutlu dizilerde ilk değer ataması yaparken, eleman sayısından az değer girerseniz, kalan değerler 0 olarak kabul edilir. Aynı sey çok boyutlu diziler için de geçerlidir; olması gerektiği sayıda eleman ya da grup girilmezse, bu değerlerin hepsi 0 olarak kabul edilir. Yani üstte yazdığımız kodun yaratacağı sonuç, söyle olacaktır:

Satır 0 : 8 16 0 0
Satır 1 : 3 15 27 0
Satır 2 : 0 0 0 0

Belirtmediğimiz bütün elemanlar 0 değerini almıstır. Satır 2'ninse bütün elemanları direkt 0 olmustur; çünkü grup tanımı hiç yapılmamıstır.

Fonksiyonlara 2 Boyutlu Dizileri Veri Aktarmak

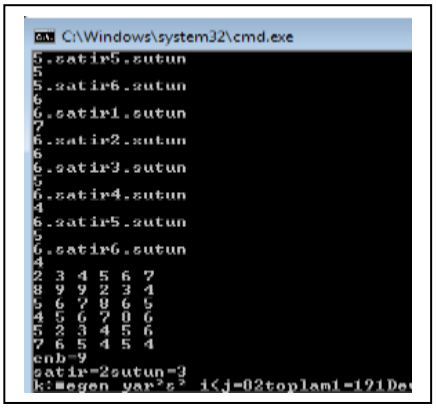
İki boyutlu bir diziyi fonksiyona parametre göndermek, tek boyutlu diziyi göndermekten farklı sayılmaz. Tek farkı dizinin iki boyutlu olduğunu belirtmemiz ve ikinci boyutun elemanını mutlaka yazmamızdır. Basit bir örnek yapalım; kendisine gönderilen iki boyutlu bir diziyi matris seklinde yazan bir fonksiyon olusturalım:

```
//atama yoluyla matrisi oluşturma
#include <iostream>           // cout,cin
#include <time.h>
using namespace std;

void main()
{
      int enb=0;
int matris[][ 4 ] = { {10, 15, 20, 25},{30, 35, 40, 45},{50, 55, 60, 65} };
for(int i = 0; i < 3; i++ ) {
for(int j = 0; j < 4; j++ ) {
cout<< matris[ i ][ j ]<<" ";
if (matris[ i ][ j ]> enb ) enb=matris[ i ][ j ];
}
cout<< "\n"<<"\n";}
cout<<"enbuyuk="<<enb<<endl;}
```

Kod içersinde bulunan yorumlar, iki boyutlu dizilerin fonksiyonlara nasıl aktarıldığını göstermeye yetecektir. Yine de bir kez daha tekrar edelim... Fonksiyonu tanımlarken, çok boyutlu dizinin ilk boyutunu yazmak zorunda değilsiniz. Bizim örneğimizde *int dizi[ ][4]* seklinde belirtmemiz bundan kaynaklanıyor. Sayet 7 x 6 x 4 boyutlarında dizilerin kullanılacağı bir fonksiyon yazsaydık tanımlamamızı *int dizi[ ][6][4]* olarak değistirmemiz gerekirdi. Kısacası fonksiyonu tanımlarken dizi boyutlarına dair ilk değeri yazmamakta serbestsiniz; ancak diğer boyutların yazılması zorunlu! Bunun yararını merak ederseniz, sütun sayısı 4 olan her türlü matrisi bu fonksiyona gönderebileceğinizi hatırlatmak isterim. Yani fonksiyon her boyutta matrisi alabilir, tabii sütun sayısı 4 olduğu sürece...

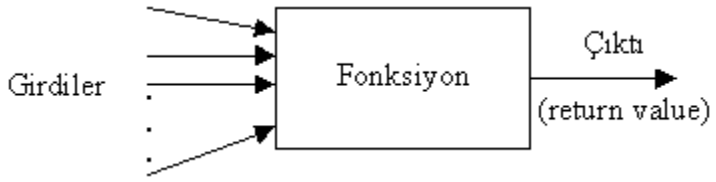
```
// klavyeden girilen 6*6 lık matrisin köşegenlerini ve köşegen üstü //toplamlarını
bulma
#include <time.h>
#include <iostream>
using namespace std;
void main()
{     int i, j,enb=0,sat=0,sut=0,top=0,top1=0;
      int matris[6][6] ={0};
      srand(time(0));
      for( i = 0; i < 6; i++ ) {
            for( j = 0; j < 6; j++ ) {
                  // matris[ i ][ j ]  = rand()%10;
cout<<i+1 <<".satir"<<j+1<<".sutun"<<endl;
                 cin>>matris[i][j];       } }
                              for( i = 0; i < 6; i++ ) {
            for( j = 0; j < 6; j++ ) {
                  // matris[ i ][ j ]  = rand()%10;
                              top1=top1+matris[i][j];
                                    if (matris[i][j]>=enb){
                        enb=matris[i][j];
                        sat=i;sut=j ;}
                                    cout<<matris[i][j]<<" ";
                                   }
                             cout<<endl; }
//hesap
      for( i = 0; i < 6; i++ ) {
            for( j = 0; j < 6; j++ ) {
            if (i<j) top=top+matris[i][j];
            //if (i==j) matris[i][j]=\;
                        }     }
cout<<"enb="<<enb<<endl;
cout<<"satir="<<sat+1<<"sutun="<<sut+1<<endl;
cout<<"köşegen yarısı i<j="<<top;
cout<<"toplam1="<<top1;
}
```

Çıktısı



FONKSİYONLAR (Alt Programlar)

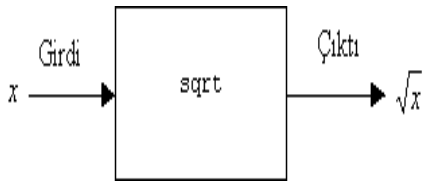
C programlama dili, fonksiyon olarak adlandırılan alt programların birleştirilmesi kavramına dayanır. Bir C programı bir ya da daha çok fonksiyonun bir araya gelmesi ile oluşur. Fonksiyon, belirli sayıda veriyi kullanarak bunları işleyen ve bir sonuç üreten komut grubudur. Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren argümanları (bağımsız değişkenleri) vardır. Genel olarak bir fonksiyon Şekil 1'deki gibi bir kutu ile temsil edilir:



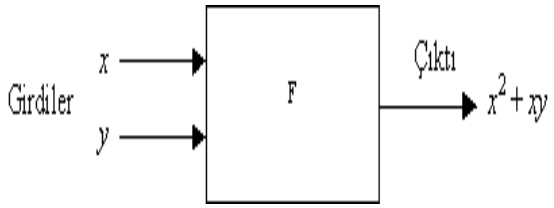
Şekil 1 . Bir fonksiyonun kutu gösterimi

Fonksiyonların girdilerine parametreler yada argümanlar denir. Bir fonksiyon bu parametreleri alıp bir işleme tabi tutar ve bir değer hesaplar. Bu değer çıktı, geri dönüş değeri (return value) olarak adlandırılır. Bir fonksiyonun kaç girişi olursa olsun sadece bir çıkışı vardır. Aşağıda bir ve iki girişli iki fonksiyonun kutu gösterimi verilmiştir.

Tek parametreli sqrt(x) fonksiyonu:



İki parametreli F(x,y) = x*x + x*y fonksiyonu:



C programlama dili, kullanıcısına bu türden fonksiyon yazmasına izin verir. C dilinde hazırlanan bir fonksiyonun genel yapısı şöyledir:

```
FonksiyonTipi FonksiyonAdı(argüman listesi)
argumanların tip bildirimleri
{
  Yerel değişkenlerin bildirimi
  ...
  fonksiyon içindeki deyimler veya diğer fonksiyonlar
  ...
  return geri dönüş değeri;
}
```

Örneğin iki sayının toplamını hesaplayacak bir fonksiyon şöyle tanımlanabilir:

```
int topla(x,y)          /* klasik biçim */
int x,y
{
  int sonuc;
  sonuc = x + y;
  return sonuc;
}
```

veya

```
int topla(int x,int y)   /* modern biçim */
{
  int sonuc;
  sonuc = x + y;
```

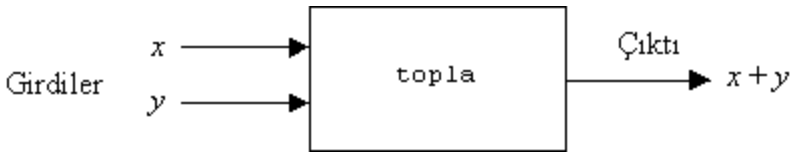
```
  return sonuc;
}
```

veya

```
int topla(int x,int y)   /* post modern biçim */
{
  return (x+y);
}
```

Bu örnekte, fonksiyonun *kimlik kartı* ve kutu gösterimi şöyledir:

- Fonksiyon tipi: int
- Fonksiyon adı : topla
- parametreler : x ve y
- geri dönüş değeri: x+y



Her üç program parçasında da return (geri dönüş) deyimi kullanılmaktadır. Bu deyim C programlama dilinin anahtar sözcüklerinden biridir ve fonksiyon içerisinde sonucu, kendisini çağıran yere göndermek için kullanılır. Yani topla fonksiyonu herhangi bir programın içerisinde kullanıldığında, fonksiyonun üreteceği sonuç return deyiminden sonra belirtilen değişken veya işlem olacaktır. Örneğin fonksiyon:

```
...
int t;
...
t = topla(9,6);
...
```

şeklinde kullanılırsa, t değişkenine 9+6=15 değeri atanır.

Fonksiyon Bildirimi

Bir fonksiyonun bildirimi iki türlü yapılır:

Ana programdan önce:

```
...
int topla(int x,int y)      /* fonksiyon */
{
...
}
...
main()
{
 ...
}
```

Ana programdan sonra:

Bu durumda fonksiyon örneği (function prototype) ana programdan önce bildirilmelidir.

```
...
int topla(int x,int y);  /* fonksiyon prototipi */
...
main()
{
 ...
}
...
int topla(int x,int y)      /* fonksiyon */
{
...
}
```

Bir C programı içinde, yazılmış olan fonksiyonlar genellikle bu iki tipte kullanılır. İkinci kullanımda fonksiyon prototipi mutlaka bildirilmelidir. Aksi halde bir hata mesajı ile karşılaşılır. Fonksiyon

prototipinde arguman isimlerinin yazılması zorunlu değildir. Sadece argüman tiplerini belirtmek de yeterlidir. Yukarıdaki topla fonksiyona ait prototip:

```
int topla(int x,int y);    şeklinde yazılabileceği gibi
int topla(int,int);        şeklinde de yazılabilir.
```

Örnek 1: İki sayıyı toplayan ve sonucu ekranda gösteren C++ programı:

```
#include <iostream>
using namespace std;
int topla( int x,int y );  /*** fonksiyon prototipi ***/
void main()
{
    int toplam,a,b;
   cout<<"Iki sayi girin : "; //ilk burası okunur
    cin>>a>>b; //ikinci burası
    toplam = topla(a,b); //önce topla devreye girer
   cout<<"toplami  = \n dir"<< a<<b<<toplam;
   //**********************************
}
int topla( int x, int y ) //dördüncü devreye girer
{
          int sonuc;
     sonuc = x + y; //ne aktarılacağı önemlidir
        return sonuc
```

ÇIKTI

```
Iki sayi girin : 5 12
5 ve 12 nin toplami  17 dir.
```

topla fonksiyonun prototipi ve kendisi satırlarda bildirilmiştir. Klavyeden okunan a ve b değişkenleri fonksiyona parametre olarak aktarılmıştır. Bu değişkenlerin isimleri ile topla fonksiyonunda kullanılan değişkenlerin (x ve y) isimlerinin aynı olması zorunlu değildir. Burada a ve b değişkenleri sırasıyla x ve y değişkenlerinin yerine konulmuştur. Toplam adlı tamsayı değişkenine topla fonksiyonunun dönüş değeri (a + b değeri) atanmıştır.

Geri Dönüş Değerleri

return anahtar sözcüğünün iki önemli işlevi vardır:

1. fonksiyonun geri dönüş değerini oluşturur
2. fonksiyonu sonlandırır

Bu deyimden sonra bir değişken, işlem, sabit veya başka bir fonksiyon yazılabilir. Örneğin:

```
return (a+b/c);      /* parantez kullanmak zorunlu değildir */
return 10;           /* değişken kullanmak mecbur değil */
return topla(a,b)/2.0; /* önce topla fonksiyonu çalışır */
```

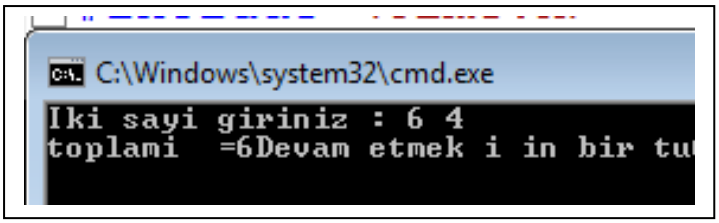
Bir fonksiyonda birden çok geri dönüş değeri kullanılabilir. Fakat, ilk karşılaşılan return deyiminden sonra fonksiyon sonlanır, ve çağrılan yere bu değer gönderilir.

Bir fonksiyonun her zaman geri dönüş değerinin olması gerekmez. Bu durumda return deyimi kullanılmaz. Eğer bu anahtar kelime yok ise, fonksiyon ana bloğu bitince kendiliğinden sonlanır. Böyle fonksiyonların tipi void (boş, hükümsüz) olarak belirtilmelidir. Bu tip fonksiyonlar başka bir yerde kullanılırken, herhangi bir değişkene atanması söz konusu değildir. Çünkü geri dönüş değeri yoktur. Ancak, void fonksiyonlara parametre aktarımı yapmak mümkündür.

```
#include <time.h>
#include <iostream>
using namespace std;
float topla( float x,float y );  //fonksiyon
float topla( float x, float y ) //tanım
{     float sonuc;
      sonuc = x + y;
      cout<<12;

return (x);
}
void main() //ilk burası şüpheli yaklaşma
{     float toplam,a,b,toplami;
    cout<<"Iki sayi giriniz : ";
     cin>>a;
       cin>>b;
     toplam = topla(a,b);
    cout<<"toplami  ="<<toplami;}
```

Çıktısı

Girilen ilk değeri basmasının nedeni return sonuç aktarmıyor,return deyimi x değerini dolayısıla girilen ilk değeri aktarıyor.



soru İstenen para miktarını 20, 10 ve 5'lik birimlere bölen ve sonucu ekrana gösteren C++ programı
(Bankamatik Simülasyonu)

```
#include <iostream>           // cout,cin
using namespace std;
 void bankamatik(int para)
{     int a,yirmilik,onluk,beslik;
    a = para;
    if(a%5==0){
       yirmilik = a/20;
       //a -= yirmilik*20;
         a =a-yirmilik*20;
       onluk = a/10;
       a -= onluk*10;
       beslik = a/5;
       a -= beslik*5;
       cout<<"\nYirmilik ="<<yirmilik;
       cout<<"\nOnluk= "<<onluk;
       cout<<"\nBeslik ="<<beslik;
   }
   else
   cout<<"Girilen miktar 5 YTL ve katlari olmali!\a\n";
     }
void main()
{
   int miktar;
   cout<<"Cekilecek para miktari (TL) =.. ";
   cin>>miktar;
   bankamatik(miktar); }
```

ÇIKTI
Cekilecek para miktari = **135**

Yirmilik = 6
Onluk = 1
Beslik = 1

ÇIKTI
Cekilecek para miktari = 456
Girilen miktar 5 YTL ve katlari olmali!

main Fonksiyonu

Ana program anlamına gelen main de bir fonksiyondur. C programlarının başlangıcı ve sonu bu fonksiyonla belirlenir. Ancak main fonksiyonu da geri dönüş değeri kullanabilir. main fonksiyonunun geri dönüş değerinin görevi programın çalışması bittikten sonra sonucu işletim sistemine iletmektir.

Programcılar çoğunlukla main fonksiyonunun başına void yazmaz.

```
main()
{
  ...
}
```

Bu durumda geri dönüş değeri tamsayı (int) kabul edilir. Bu şekilde kullanımda, yeni tip derleyiciler uyarı (warning) mesajı verebilirler. Bu yüzden, programcılar main fonksiyonunun önüne aşağıdaki gibi void deyimi eklerler.

```
void main()
{
  ...
}
```

main fonksiyonuna parametre aktarmak ta mümkündür. Ana programa parametre aktarımı, derlenmiş bir program komut satırından (işletim sistemi ortamından) ilk çalıştırılacağı zaman yapılır. Eğer parametre aktarılmayacaksa:

```
main()
{
  ...
}
```

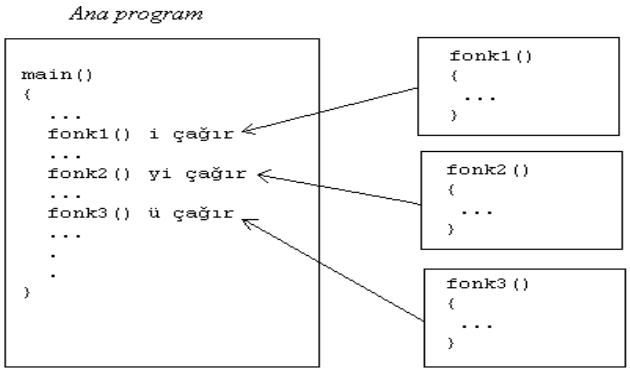
yada

```
void main(void)
{
  ...
}
```

şeklinde kullanılabilir.

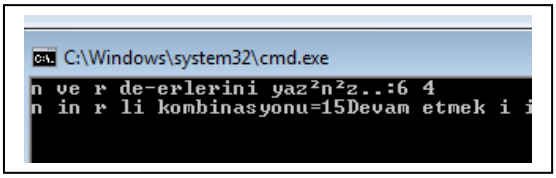
Fonksiyon Sayısı

Ana programdan, main fonksiyonundan, isteğe göre birden çok fonksiyon çağırmak mümkündür. Bir fonksiyonun çağrılması demek, o fonksiyonun geri dönüş değerinin ana programda kullanılması demektir. Birden çok fonksiyonun main tarafından nasıl çağrıldığını temsil eden blok diyagram Şekil 2'de gösterilmiştir.



Şekil 2. *Ana programdan alt programların (fonksiyonların) çağırılması.*
Fonksiyonu çağırmak için, fonksiyonun adını yazmak yeterlidir. Fonksiyonların sadece ana program tarafından çağrılması zorunlu değildir. Bir fonksiyon başka bir fonksiyon tarafından da çağrılabilir. Bu tür kullanıma dair bir örnek,.

```
//c(n,r)=n!/(r!*(n-r)!)  denkleminin alt program ile yazımı
#include <time.h>
#include <iostream>           // cout,cin
using namespace std;
int combin(int a, int b); //anlat
int fact(int x);
void main()
{    int n, r,comb;
    cout<<"n ve r değerlerini yazınız..:";
    cin>>n>>r;
comb=combin(n,r);
    cout<<"n in r li kombinasyonu="<<comb;
   // return 0;
}
int combin(int a, int b)
{    int f1, f2, f3,y;
    f1 = fact(a);
    f2 = fact(b);
    f3 = fact(a - b);
      y=f1 / (f2 * f3);
 return(y);
}
int fact(int x)
{     int fx = 1;
    int i;
    for (i = 2; i <= x; i++)
        fx = fx * i;
    return fx;}
```



Çıktısı