scanf () Fonksiyonu

Klavyeden veri okumak için kullanılır. Yapı olarak printf () fonksiyonu aynıdır. Kullanım biçimi:

Genel yazım formatı;
scanf(“format_dizisi”,değer yada değişken listesi);

scanf("%d",&x);
Girilen karakterler format ile belirtilen şekle göre değişkenlere aktarılır. Değişkenler işaretçi tipinde olmalıdır. Yani parametre olarak değişkenin adresi gönderilmelidir.

```
scanf("%f %f %f ", &a, &b, &c);
```

scanf işlevinin değeri =0 ise hiçbir değişkene değer atanmamış

>0 ise başarılı bir şekilde değer atanan değişken sayısı

```
int a,b,c;
float m,n;
```

```
scanf("%d", &a);
```
Klavyeden tamsayı okur. Girilen değer a değişkenine aktarılır.

```
scanf("%d %d",&a,&b);
```
Klavyeden girilen ilk değer a değişkenine, ikinci değer b değişkenine aktarılır.

```
scanf("%f %d", &m, &a);
```
Klavyeden 1. float, ikincisi tamsayı olmak üzere iki değer okur.

İkinci dereceden denklem çözümünün yapıldığı örnekte katsayıları klavyeden okutmak istersek

```
scanf("%f %f %f ", &a, &b, &c);

printf(“Katsayıları sırasıyla giriniz (a  b  c) :” );
scanf("%f %f %f ", &a, &b, &c);

printf(“a katsayısını giriniz : “);  scanf(“%f”, &a);
printf(“b katsayısını giriniz : “);  scanf(“%f”, &b);
printf(“c katsayısını giriniz : “);  scanf(“%f”, &c);
```

cin Komutu (C++)

cin komutu ile klavyeden girilen değerler ismi geçen değişkene atanır. cin komutu ile >> karakterleri birlikte kullanılır.

Örnek:

```
1.  # include <iostream>
2.  # include <string.h>
3.  int main()
4.  {
5. char x;
6.      cin>>x;
6.      cout<<x;
7.  }
```

Ayrıca cout << komutu programda belirlenen ifadenin ekranda gösterilmesini sağlar.

```
#include <iostream>
#include <math.h>
using namespace std;

int main()  {
double a, b, c, delta, x1, x2;
cout << "Lütfen 2.Derece denklemin katsayılarını giriniz: " << endl;
cin >> a >> b >> c ;
delta = pow(b,2) - 4 * a * c ;
if(delta < 0)
cout << "Reel kök yoktur.!!" << endl;
else if (delta == 0) {
    x1 = -b / (2 * a);
    cout<<"Denklemin Kökü=" << x1;
    cout << endl ;
}
else {
x1 = (-b + sqrt(delta)) / (2 * a);
x2 = (-b - sqrt(delta)) / (2 * a);
cout<<"Denklemin Kökleri:"<<x1<<"and"<<x2;
cout << endl;
}
}
```

puts() Fonksiyonu (Karakter dizisi yazdırma)

```
#include <stdio.h>
puts( katar );
```

katar olarak belirtilen karakter topluluğunu ekrana yazdıktan sonra, imleci alt satıra geçirir. Örneğin:

```
puts("puts() fonksiyonunun gösterimi!");
```

şekinde kullanılırsa çıkış şöyle olacaktır.

```
puts() fonksiyonunun gösterimi!
```

puts() fonksiyonu ESC karakterleri ile kullanılabilir.

```
puts("Bu birinci satır...\nBu ikinci satır...");
```

```
Bu birinci satır...
Bu ikinci satır...
```

gets() Fonksiyonu

Klavyeden bir karakter topluluğu, katar, okumak için kullanılır.

Okuma işlemi yeni satır karakteriyle karşılaşılıncaya kadar sürer.
puts()-gets() arsındaki ilişki, printf()- scanf() arasındaki gibidir.
puts() ile ekrana bir katar yazdırılırken, gets() ile okunur.

Örneğin:

```
...
char ktr[10];
puts("Bir şeyler yazın:");
gets(ktr);
...
```

Yukarıdaki program parçası, klavyeden girilen karakterlerin, gets() fonksiyonu ile ktr katarına aktarmak için kullanılır. ktr[10] şeklindeki kullanım girilen katarın içerisinden ilk 10 karakteri değerlendir manasındadır. Bu kullanım daha sonra açıklanacaktır.

getchar() Fonksiyonu

Standart girişten bir karakter okur. Programı istenen bir yerde durdurup, bir karakter girinceye kadar bekletir.Örneğin:

```
...
 {
 getchar();
 printf("%d\n",i);
}
...
```

Yukarıdaki program parçası 0-9 arası sayıları sırasıyla ekranda göstermek için kullanılır. Fakat her rakamı yazdırılmadan önce klavyeden herhangi bir karakter girip ENTER tuşuna basılması beklenir. Bu bekleme getchar() fonksiyonu ile gerçekleştirilir.

C++'da Okuma ve yazma komutları

C++ derleyicilerine C dilinden gelen başlık dosyaları da yeni standarda uydurulmuş ve uzantıları kaldırılmıştır. Bu dosyaların C'den geldiğini belirtmek için isimlerinin başına '**c**' harfi eklenmiştir.

C'de	**C++'da**
include<stdio.h>	#include<cstdio>
include<stdlib.h>	#include<cstdlib>

C++'da C'den farklı olarak giriş/çıkış fonksiyonlarının yerine giriş/çıkış nesneleri kullanılır.

cin : Standart giriş birimi. Çoğunlukla tuş takımı işlemlerinde kullanılır.

cout : Standart çıkış birimi. Çoğunlukla ekran işlemlerinde kullanılır.

cout nesnesi üzerinde ekrana bir veri yazmak için gönderme operatörü '**<<**' kullanılır.

Giriş işlemlerinde ise cin nesnesi ve giriş operatörü '**>>**' kullanılmaktadır.

Örnek program

- Kullanıcıya yarıçapını sorduğu dairenin alan ve çevresini hesaplayan program.

Örnek program

- **Adım 1:** Başlangıçta içi boş programı yazalım:
- Eğer cin, cout kullanılacaksa dev c++ 5 ve üstü versiyonlarda using namespace std; komutu header tanımının altında eklenmelidir

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

 main() {
            ......
 system("pause");
}
```

Örnek program

- **Adım 2: yapılacakları sırayla sözel olarak yazalım**:

```
#include <iostream.h>
using namespace std;
void main() {
   // kullanıcıdan yarıçapı oku
   // alanı hesapla
   // çevreyi hesapla
   // sonuçları göster
}
```

Örnek program

Adım 3: Hangi değişken ve formüllerin kullanılacağına karar verelim :

```
#include <iostream.h>
using namespace std;
 int main() {
     const double pi = 3.1415;  // yarıçap ,için r sayısına ihtiyacımız var
     double r;                  // kullanıcıdan bu sayıyı okuyoruz
     cin >> r;                  // alanı hesaplıyoruz. A = pi * r^2
     double A = pi * r^2;       // çevreyi hesaplıyoruz
     double C = 2 * pi * r;     // sonuçları ekrana yazıyoruz
     cout << A;
     cout << C;
     system("pause");
     }
```

- Şimdi eğer C++ kuralı hatası yaptıysak derleyiciyi çalıştırıp öğreniyoruz.
- Derleyici r^2 "unknown“ mesajı verdi, r^2 yi tanımadığını söylüyor bu satırı değiştiriyoruz:
  - A = pi * r*r;

Adım 4: Programımızın son durumu:

```
//#include <iostream>
#include <iostream.h>
#include <stdlib.h>
using namespace std;
 main() {
 const double pi=3.1415;
 double r,A,C;
    cin >> r;
    A = pi * r*r;
    C = 2 * pi * r;
    cout << A;
    cout << C;
    system("pause"); }
```

Örnek program

- Yeniden derleyici çalıştırıldığında hata vermeyecektir.
- Programı çalıştırıyoruz. Bağlayıcı (Linker) da hata vermezse çalıştırılabilir dosya (executable file) üretilir ve çalıştırılır.
- Sonuç doğru ama ekran görüntüsü düzgün değilse giriş-çıkış komutlarını yeniden gözden geçiririz.
- Ancak sonuç hatalı ise mantık hatası yapmışızdır. Programı adım adım yürüterek hatayı bulabiliriz.

Örnek 1

```
// cout nesnesinin kullanımı

#include<iostream> // Başlık dosyası ekleniyor

int main()
{
    int i=5;
    float f=4.6;

std::cout<<"Tamsayı i="<<i<<"Kayan noktalı sayı f="<<f;

return 0;
}
```

using namespace std;
Eklenmediği takdirde bu şekilde de gösterilebilir

Örnek 2

```
// cin ve cout nesnelerinin kullanımı
#include<iostream>
using namespace std;  // Her erişimde std ismi kullanılmayacak

int main() {
    int i,j;              // İki adet tamsayı değişken tanımlanıyor
    cout << "İki sayı giriniz \n ";
                    // Mesaj ekrana çıkıyor alt satıra geçiliyor
    cin >> i >> j;          // i ve j tuş takımından okunuyor
    cout << "Toplam= " << i + j << "\n";
                    // Sayıların toplamı ekrana yazılıyor
    return 0;
}
```

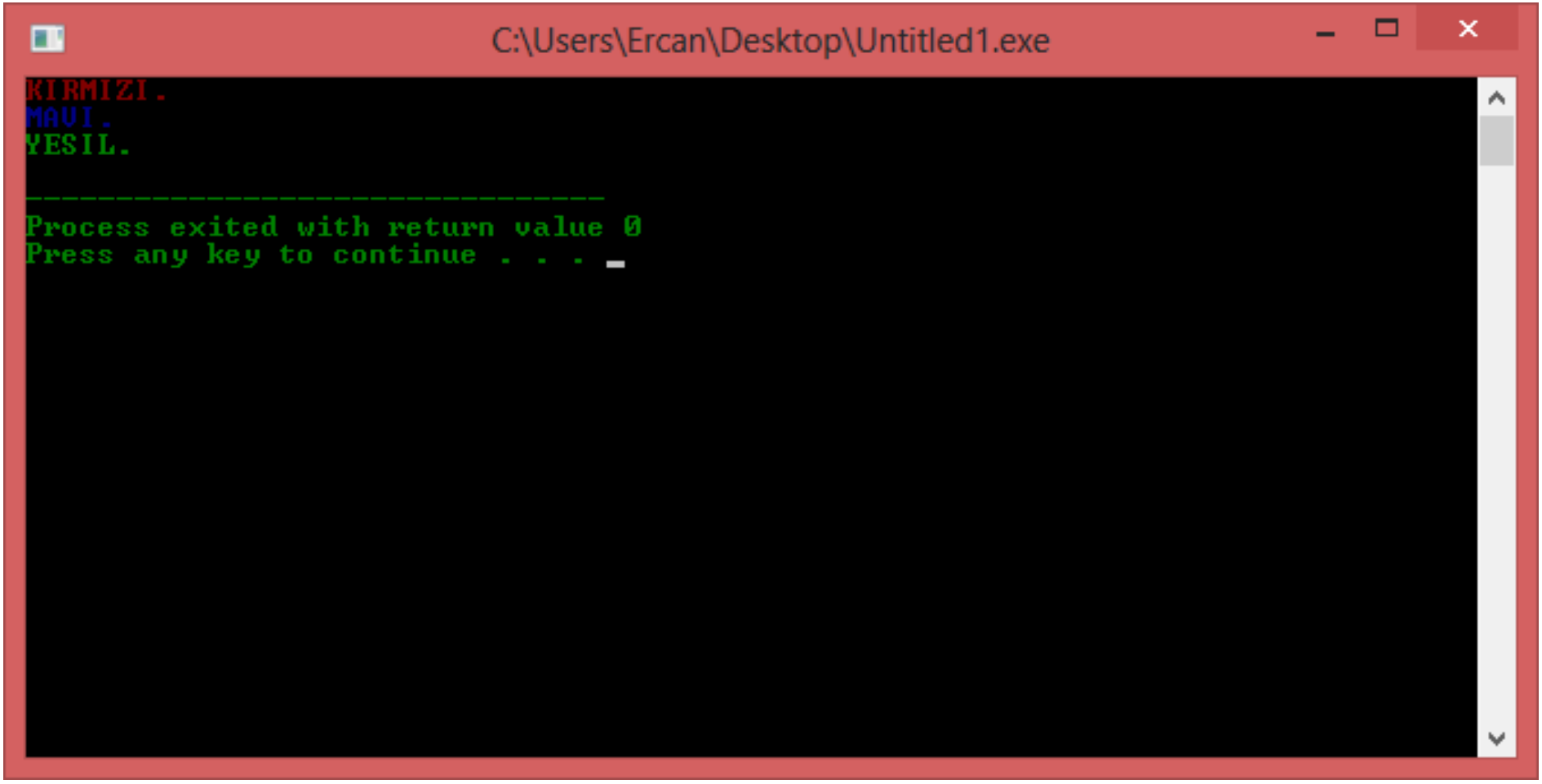
AÇIKLAMA İÇİN TÜRKÇE KARAKTER DESTEĞİ

```
#include "locale.h"

setlocale(LC_ALL,"turkish");
```

```
#include <iostream>
#include "locale.h"

using namespace std;

int main()
{
int a=0,b=0,c=0,d=0;
setlocale(LC_ALL,"turkish");
cout<<"------------"<<endl;
cout<<"Bir Sayı Giriniz: ";
cin>>a;
cout<<"Bir Sayı Daha Giriniz: ";
cin>>b;
cout<<"Bir Sayı Daha Giriniz: ";
cin>>c;
cout<<"Bir Sayı Daha Giriniz: ";
cin>>d;

cout<<endl<<"Girilen Sayılar: "<<a<<" "<<b<<" "<<c<<" "<<d;
cin.get();
return(0);
}
```

C Programlama Dilinde Çıktı Ekranının Renklendirilmesi

```c
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
main()
{
   HANDLE  hConsole;
   hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
   SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
   printf("KIRMIZI.\n");
   SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
   printf("MAVI.\n");
   SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
   printf("YESIL.\n");
}
```


C:\Users\Ercan\Desktop\Untitled1.exe
KIRMIZI.
MAVI.
YESIL.

Process exited with return value 0
Press any key to continue . . .

OPERATÖRLER

Bir işleme yol açan, işlem sonucunda belirli bir değer üretilmesini sağlayan atomlara operatör denir.

OPERATÖRLERİN SINIFLANDIRILMASI

1-İşlevlerine göre

a. Aritmetik operatörler(+,*,/...)
b. İlişkisel operatörler(<,>,<=,...)
c .Mantıksal operatörler(AND,OR,NOT,XOR,..)
d. Bit operaörleri(belli bir sayının kaçıncı bitinin kaç olduğu hakkında bilgi verir)
e. Gösterici operatörleri
f. Özel amaçlı operatörler

2-Operand sayılarına göre

a. İki operand alanlar (binary)
b. tek operand alanlar (unary)
c. Üç opernd alanlar (ternary)

3-Operatörün konumuna göre yapılan sınıflandırma

a. Ara ek operatörler (infix)
b. Ön ek operatörle (prefix)
c. Son ek operatörleri (postfix)

C'nin bütün iki operand alan operatörleri infix'tir. Bir operatörün teknik olarak tanımlanması için bütün bu gruplardaki yerinin belirtilmesi gerekir. Örneğin 4 binary infix aritmetik operatördür.

ARİTMETİK OPEARATÖRLER

+, -, *, / binary, infix

% OPEARÖRÜ

Binary infix bir operatördür. Bölüm işlemindeki kalanı hesaplar.

++ VE -- OPERATÖRLERİ

++ arttırma -- eksiltme operatörüdür.

İkisi de unary operatörlerdir. Postfix ve prefix olarak kullanılabilir. Postfix ve prefiz kullanımda fark vardır. ++a => a = a + 1;

Bu operatörler başka hiçbir operatör olmadan tek başlarına kullanılmışsa aralarında fark olmaz.

a) İkili Operatörler

+ Toplama operatörü
- Fark operatörü
* Çarpma operatörü
/ Bölme operatörü
-- Bir azaltma
++ Bir arttırma
% Mod operatörü: bölme sonucundan kalanı verir. (k=(20%6)=2 olur)

örnek

```
#include<stdio.h>
#include<conio.h>
void main()
{
int x,y,z;
x=14;y=3;
z=x+y;printf("toplama=%d\n",z);
z=x-y;printf("çıkarma=%d\n",z);
z=x/y;printf("bölme=%d\n",z);
z=x*y;printf("çarpma=%d\n",z);
z=x%y;printf("mod=%d",z);
getche();
}
```

SORU :

En fazla dört basamak olabilen sayının basamak değerlerini yazdır.

% ve / işlemlerinin kullanımı.

ÇÖZÜM

```
#include <stdio.h>
main()
{
  int i,y;
      y=1985;
      i= y / 1000;
      printf("%d",i);
      y= y-i*1000;
      i= y / 100;
      printf(" %d",i);
      y = y-i*100;
      i= y / 10;
      printf(" %d",i);
      y = y-i*10;
      printf(" %d\n",y);
```

```
 i = 1928;   // Yöntem 2
      printf("%d ",i / 1000);
      printf("%d ",(i / 100) % 10);
      printf("%d ",(i / 10) % 10);
      printf("%d\n",i  % 10);
}
```

C++ da özel kullanımlar.

x+=5; → x=x+5;

x*=5; → x=x*5;

x-=y+5; → x=x-y+5;

x%=5; → x=x%5;

b)Tekli (Unary) Operatörler

C'de 4 adet tekli operatör kullanılır: Tam ve gerçel sayılarda kullanılır.

-	Herhangi bir sayının –1 ile çarpımıdır.
+	Bu operatör etkisizdir.
--	Değişkenin değerini 1 eksiltir.
++	Değişkenin değerini 1 arttırır.

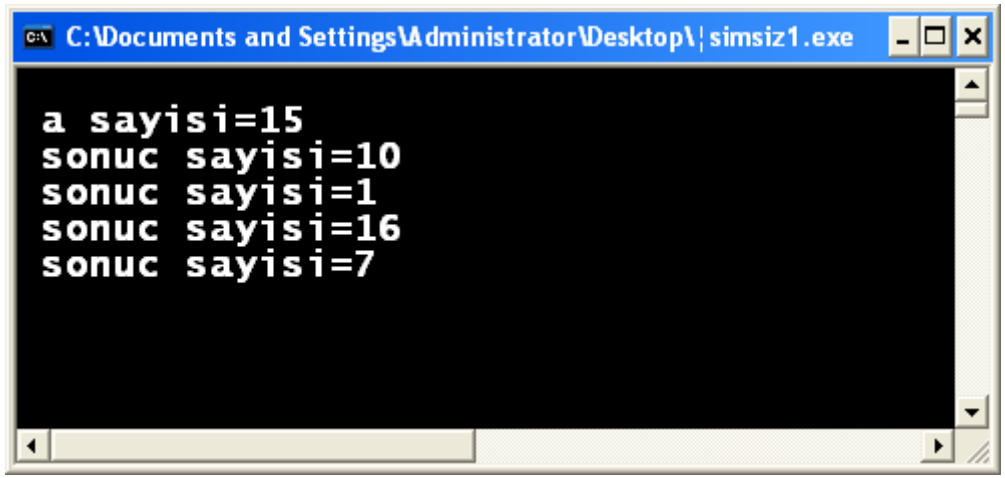
-- ve ++ operatörleri değişken sağ ve sol tarafına göre farklı işlevleri vardır.

x++	→	x=x+1;	//(eşitle ve sonra arttır)
++x	→	x+1=x;	//(arttır ve eşitle)
x--	→	x=x-1;	//(eşitle ve sonra azalt)
--x	→	x-1=x;	//(azalt ve sonra eşitle)

örnek

```
#include <stdio.h>
#include <conio.h>
main()
{
int a,b,sonuc;
a=10; // scanf("%i", &a);
a+=5;
printf("\n a sayısı=%d”, a);
a=10;
sonuc=a++;
printf("\n sonuc sayısı=%d”, sonuc);
b=5;
sonuc=a%b;
printf("\n sonuc sayısı=%d”, sonuc);
a=10;
sonuc=++a+b;
printf("\n sonuc sayısı=%d”, sonuc);
sonuc=a-(--b);
printf("\n sonuc sayısı=%d”, sonuc);
getch ();
}
```

ÇÖZÜM

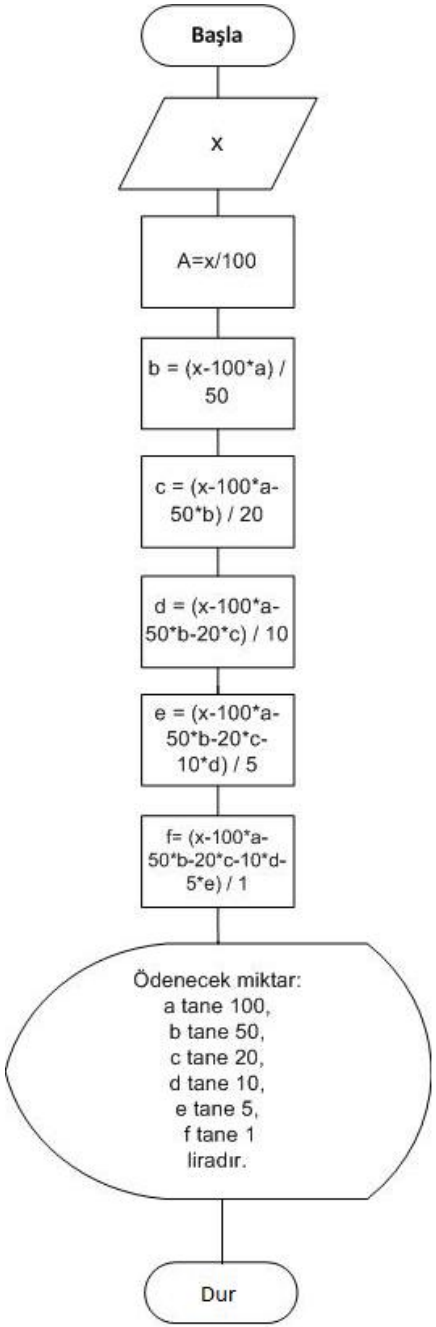

C:\Documents and Settings\Administrator\Desktop\simsiz1.exe
a sayisi=15
sonuc sayisi=10
sonuc sayisi=1
sonuc sayisi=16
sonuc sayisi=7

Örnek: Üç sayının ortalamasının bulunması

```
#include “stdio.h”
void main()
{
  float a,b,c,ort;
  printf(“a sayısını giriniz “);  scanf(“%f”, &a);
  printf(“b sayısını giriniz “);  scanf(“%f”, &b);
  printf(“c sayısını giriniz “);  scanf(“%f”, &c);
  ort = (a+b+c)/3.0;
  printf(“Ortalaması = &f\n”, ort);
}
```

Örnek: Para ödeme algoritması ve programının oluşturulması
(Hangi paradan kaç tane olabiliyor)

Adım1 *Başla*
Adım2 *Oku (x)*
*/*Toplam ödenecek miktar*/*
Adım3 *a = x / 100*
/ a: Ödenecek 100'lük kağıt para sayısı*/*
Adım4 *b = (x-100*a)*
/ 50 / b: Ödenecek 50'lik kağıt para sayısı*/*
Adım5 *c = (x-100*a-50*b) / 20*
/ c: Ödenecek 20'lik kağıt para sayısı*/*
Adım6 *d = (x-100*a-50*b-20*c) / 10*
/ d: Ödenecek 10'luk kağıt para sayısı*/*
Adım7 *e = (x-100*a-50*b-20*c-10*d) / 5*
/ e: Ödenecek 5'lik kağıt para sayısı*/*
Adım8 *f = (x-100*a-50*b-20*c-10*d-5*e) / 1*
/ f: Ödenecek 1'lik madeni para sayısı*/*
Adım 9 *Dur*



```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
	int x,a,b,c,d,e,f;
	printf("Para miktarini giriniz:");
	scanf("%d",&x);

	a=x/100;
	b=(x-a*100)/50;
	c=(x-a*100-b*50)/20;
	d=(x-a*100-b*50-c*20)/10;
	e=(x-a*100-b*50-c*20-d*10)/5;
	f=(x-a*100-b*50-c*20-d*10-e*5)/1;

printf("Odenecek miktar: \n %d tane 100\n %d tane 50\n %d tane 20\n
%d tane 10\n %d tane 5\n %d tane 1 liradir.\n",a,b,c,d,e,f);
	system("pause");
	return 0;
}
```

ŞÜPHELİ KODLAR

++ veya -- operatörlerinin bilinçsizce ve kötü kullanımları derleyiciler arasında yorum farklıklarına yol açarak taşınabilirliği bozar.Böyle kodlardan kaçınmak gerekir.

1-Üç tane + operatörü boşluk olmaksızın yanyana getirilmemelidir.(+++)
2-Bir değişken ++ veya -- ile kullanılmışsa bir daha aynı ifade içerisinde ++ veya -- operatörleriyle gözükmemelidir.(hata: b = ++a + ++a;)
3-

```
int multiply(int a, int b)
{return a * b;}

void main(void)
{
int a, b = 10;
a = multiply(b, ++b);/*hata:derleyicinin parametreleri ne sırayla aktardığı derleyiciye göre değişir.*/
printf("%d\n", a);
}
```

Bir fonksiyon çağırılırken parametrelerden birinde ++ veya -- kullanılmışsa diğer parametrelerde aynı değişken kullanılmamalı, çünkü parametre aktarım sayısı her sistemde aynı olmayabilir.

```
fonk(++a);/*Doğru:önce a arttırılır sonra yeni a değeriyle fonk çağırılır*/
fonk(a++);/*Doğru:önce fonk çagırılır sonra a arttırılır*/
```

OPERATÖRLER ARASI ÖNCELİK İLİŞKİSİ

Bir operatörün diğerine göre bir öncelik sırası vardır. Bu sıra operatörlerin öncelik tablosu denilen bir tabloyla belirtilir.

Tabloda üstteki satırda bulunanlar attakilerden daha önceliklidir. Aynı satırda bulunanlar eşit önceliklidir. Aynı öncelikli operatörlerle soldan sağa ya da sağdan sola işlem yapılır.

OPERATÖRLERİN ÖNCELİKLERİ

OPERATÖRLER	AYNI BAĞINTIDA KULLANILDIĞINDA
( ) [ ] -> .	SOLDAN SAĞA
! ~ ++ -- + - * & (type) sizeof	SAĞDAN SOLA
* / %	SOLDAN SAĞA
+ -	SOLDAN SAĞA
<< >>	SOLDAN SAĞA
< <= > >=	SOLDAN SAĞA
== !=	SOLDAN SAĞA
&	SOLDAN SAĞA
^	SOLDAN SAĞA
|	SOLDAN SAĞA
&&	SOLDAN SAĞA
||	SOLDAN SAĞA
?:	SAĞDAN SOLA
= += -= *= /= %= &= ^= |= <<= >>=	SAĞDAN SOLA
,	SOLDAN SAĞA

İLİŞKİSEL OPERATÖRLER

< > <= >=

== !=

C'de 6 ilişkisel operatör vardır.Hepsi binary infix operatörlerdir. Aritmetik operatörlerden daha düşük önceliklilerdir.

İlişkisel operatörlerin ürettiği değer önerme doğruysa 1 yanlışsa 0'dır

```
main()
{
        int a;
        a =10 > 5;
        printf("a=%d\n", a);/*a=1*/
}
```

MANTIKSAL OPERATÖRLER(&&)

C'de 3 tane mantıksal operatör vardır.

AND && (VE)

OR || (VEYA)

NOT ! (DEĞİL)

AND OPERATÖRÜ

A	B	A&&B
0	0	0
1	0	0
0	1	0
1	1	1

Mantıksal operatörlerin hepsi önce operandlarını doğru ya da yanlış olarak yorumlar, eğer sonuç doğruysa 1, yanlışsa 0 sayısal değerini üretir. Yorumlamada kural:eğer operand 0 dışı bir değerse doğru olarak, 0 ise yanlış olarak yorumlanır.Uygulama da ilişkisel operatörlerle birlikte kullanılırlar.

&& operatörünün önce sol tarafı tam olarak bitirilir.D aha sonra sağ tarafı yapılır ve bitirilir.Eğer sol tarafın sayısal değeri 0 ise sağ tarafın yapılmasına gerek kalmaz.

Örneğin x > 10 && fonk() burada x 10'dan küçükse fonk hiç çağırılmayacaktır.

OR OPERATÖRÜ(||)

OR işlemi iki operand da yanlışsa yanlış,operandlardan en az birisi doğruysa doğru sonucunu üretir.

a	b	a||b
0	0	0
0	1	1
1	0	1
1	1	1

Bu operatör de 1 ya da 0 tamsayı değerini üretir.

```
void main(void)
{
        int x,y;
        scanf("%d",&y);
        x = y < 10 || y > 50;
        printf("%d\n",x);                    }
```

OR operatörünün önce sol tarafı yapılır.Eğer sol taraf değeri 0 dışı bir değerse sağ tarafın yapılmasına gerek kalmaz.

NOT OPERATÖR(!)

Bu operatör unary prefixtir.Zaten öncelik tablosunun ikinci düzeyi tamamen unary operatörlere ayrılmıştır.

a	!a
0	1
1	0

Yani bu operatör operand 0 ise 1 , 0 dışı herhangibir değerse 0 yapar.

```
void main(void)
{
        int x,a = 10;

        x = !!a;
        printf("%d\n", x);    /*a = 10, !a = 0, !(!a) = 1, ekrana 1 basılır*/
}
```

ATAMA OPERATÖRÜ(=)

Bu operatçr binary, infix bir operatördür.Atama operatörünün sol tarafındaki operandın nesne olması gerekir.

Buradan hareketle ++ ve -- operatörlerinin operandlarının da nesne olması gerekir.

Atama operatöründe elde edilen değer sağ taraftaki operandın sayısal değeridir.

```
void fonk(int n)
{
        printf("%d\n", n);
}
void main (void)
{
        int x = 10, y;

        fonk(y = x);
}
```

z = y = x = 10; ifadesi doğru olduğu gibi,
z = (y = 10) + 2; ifadesi de doğru ve geçerlidir.

BİTLER ÜZERİNDE İŞLEM YAPAN OPERATÖRLER

Bitler üzerinde işlem yapmak, bir tamsayı yada karakter değişkenin (short,int,long ve char) bir bütün olarak sayısal değeri üzerinde değil de doğrudan bitlerini sınamak, değiştirmek ve öteleme yapmak anlamına gelmektedir. Örneğin, bilgisayarın iletişim kanalından alınan bir kısa tamsayının ikinci bitinin ne olduğu öğrenilmek isteniyorsa, bu operatörleri kullanmaya gerek vardır. Bu operatörler, kullanıcıya işlemcinin birleştirici dili düzeyinde(assembly) , bitler üzerinde çalışma olanağı verir.

Bit üzerinde işlem yapan altı operatör vardır. Bunlar:

& bit düzeyinde VE (bitwise AND)

l bit düzeyinde VEYA (bitwise OR)

^ bit düzeyinde YA DA (bitwise XOR)

~ bir'e tümleme (one's complement)

<< sola öteleme (left shift)

>> sağa öteleme (right shift)

VE OPERATÖRÜ(&): Genelde bir değişkenin bazı bitlerini sıfırlama için kullanılır.

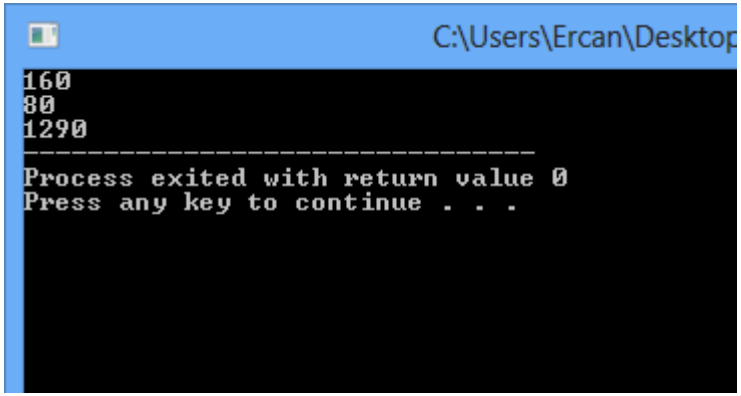
VEYA OPERATÖRÜ(|): Bir sayısal değişkenin bazı bitlerini birlemek için kullanılır.

YA DA OPERATÖRÜ (^): Karşılaştırılan bitlerin değeri birbirinden farklı ise 1, aynı ise 0 üretir. Bir değişkenin kendisi ile YA DA'lanması sayısal değerini sıfırlar. Çünkü bütün karşılıklı bitler aynı olacağından, bütün bitler sıfırlanacaktır.

Değer 1	Değer 2	Çıktı cevabı
0	0	0
0	1	1
1	0	1
1	1	0

```
a :    00000000000000000000000010000000
 ~a :  11111111111111111111111101111111

a :            00000000000000000000000010000000
 b :           00000000000000000000000011001011
--------------------------------------
a&b :          00000000000000000000000010000000
a|b :          00000000000000000000000011001011
a^b :          00000000000000000000000001001011
a<<1 :         00000000000000000000000100000000
b>>2 :         00000000000000000000000011001011
a>>>3:         00000000000000000000000000010000
```

```
#include <iostream>
using namespace std;
main()
{
unsigned int x, y, z;

x = 0xA0;
y = 0x50;
cout<<x<<endl;
cout<<y<<endl;
z = ( x << 3 ) + ( y >> 3 );
cout<<z;
}
```



$x >> y$ $\quad x / 2^y$

$x << y$ $\quad x * 2^y$

```cpp
#include <iostream>
using namespace std;
int main() {
   unsigned short a = 0xFFFF;
// pattern 1111 ...
   unsigned short b = 0xAAAA;
// pattern 1010 ...

   cout  << hex << ( a & b ) << endl;
// prints "aaaa", pattern 1010 ...
}
```

Bütün operatörler basit atama operatörü ile birlikte kullanılmaz; geçerli olan bitişik atama operatörleri aşağıdaki çizelgede verilmiştir:

Operatör	tanımı
+=	ekleyerek,atama
-=	çıkararak atama
*=	çarparak atama
/=	bölerek atama
%=	bölüp,kalanını atama
<<=	sola öteleyerek,atama
>>=	sağa öteleyerek,atama
&=	bit düzeyinde VE'leyerek atama
l=	bit düzeyinde VEYA'layarak atama
^=	bit düzeyinde YADA'layarak atama
=~	bit düzeyinde tümleme ve atama

X*= y+1; ifadesi ile x=x*(y+1); aynıdır…

& ve * İŞARETÇİ OPERATÖRLERİ

- İşaretçi (pointer), bir değişkenin bellekteki adresidir. İşaretçi değişken bu adresin saklanacağı özel bir değişkendir. Bu tip değişkenlere yalnızca adresler veya diğer işaretçi değişkenler atanabilir.

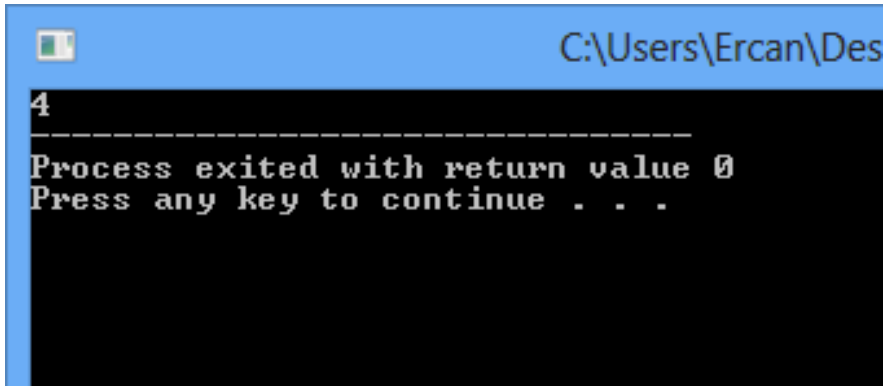
* KARAKTERİ : Bu işaret iki amaçla kullanılır.ilki, işaretçilerin bildiriminde, ikincisi bir işaretçi değişkenin işaret ettiği bellek gözüne erişmekte.

& KARAKTERİ: Daha önce bit üzerinde VE işlemi yapan operatör olarak kullanılmıştı.ancak işaretçi operatör olarak bir değişkenin önüne koyularak ta kullanılır. Bu değişkenin değeri ile değil de bellekte bulunduğu adresi ile ilgileniyor anlamına gelir.

SIZEOF OPERATÖRÜ: Bir değişkenin veya veri tipinin bellekte kaç sekizli yer kapladığını verir. Farklı tipte değişkenler kullanılan bir bağıntının sonucunun kaç sekizli yer kaplayacağını öğrenmek için de kullanılabilir.

Sizeof nesne ve sizeof(tip) olarak iki farklı şekilde kullanılabilir.

```cpp
# include<iostream>
# include<string.h>
using namespace std;

int main(){
int *a = new int[5];
    a[0] = 12;
    a[1] = 5;
    a[2] = 43;
    a[3] = -12;
    a[4] = 100;
 cout<<sizeof(a);
    return 0;
}
```



ÖDEV 3 (gönderdiğiniz mailde ödev 3 olduğunu belirtiniz)

1 Yarıçapı ve yüksekliği girilen bir silindirin hacmini hesaplayıp ekranda gösteren programı C++ ile yapınız.

2. Önce adınızı sonra soyadınızı soran ve arkasından adınız soyadınız birleşik olarak merhaba diyen programı yazınız.

3. Klavyeden ardı ardına girilen 5 sayıyı toplayan programı yazınız.

4. Klavyeden girilen iki pozitif tamsayıdan birincisinin ikincisi cinsinden kuvvetini alan programı hazır fonksiyon kullanmadan yazınız.

5. Türkiye kelimesinin harf harf ASCII koduna karşılık gelen sayılarını gösteren programı yazınız

6. Saatte ortalama 60 km yol giden bir aracın, klavyeden girilen mesafeyi kaç saatte gideceğini hesaplayan programı yazınız.

7. Girilen iki sayının bölenini ve kalanını veren programı yapınız.

(Not: Sadece kaynak dosyaları gönderilecektir exe dosyaları gönderilmeyecektir.)