PROGRAM KONTROL KOMUTLARI

A karşılaştırma kontrol komutları

1. if koşulu
2. if else koşulu
3. if - else if koşulu
4. (? :) üçlü koşul
5. goto deyimi
6. switch deyimi

B tekrarlama kontrol komutları

7. do-while döngüsü
8. while döngüsü
9. for döngüsü

KARŞILAŞTIRMA KOMUTLARINDA OPERATÖR KULLANIMI

Burada kullanılan operatörler ilişki operatörleri ve sayısal değerleri veya karakterleri karşılaştırmak için kullanılırlar.

C dilinde karşılaştırma operatörleri karakter katarları(strings) için kullanılmazlar.

Bunların karşılaştırılması için standart kütüphanede strcmp( ) ve buna benzer birçok fonksiyon vardır.

Karşılaştırma (ilişkilendirme) operatörlerinin if içerisinde anlamı:

>	büyük mü?
>=	büyük veya eşit mi?
<	küçük mü?
<=	küçük veya eşit mi?
==	eşit mi?
!=	farklı mı?

Genelde karşılaştırma ve döngü deyimlerinde koşulun sınanması için kullanılırlar.

Karşılaştırma sonucu doğru(true) ya da yanlış (false) çıkar.

Koşul doğruysa olumlu varsayılarak, koşul sonunda istenenler yerine getirilir. Yanlış ise olumsuz varsayılır ve istenenler atlanır.

BOOLEAN SINAMA ÖRNEK

```
#include<iostream>
main()
{
bool a; //int a;
a=2<3;
std::cout<<a;
}
```

if (Eğer) Karşılaştırması

İf koşulu

if (şart) ifade;

if (x>y) pirintf ("x büyüktür");

if (şart)

{ ifade 1;

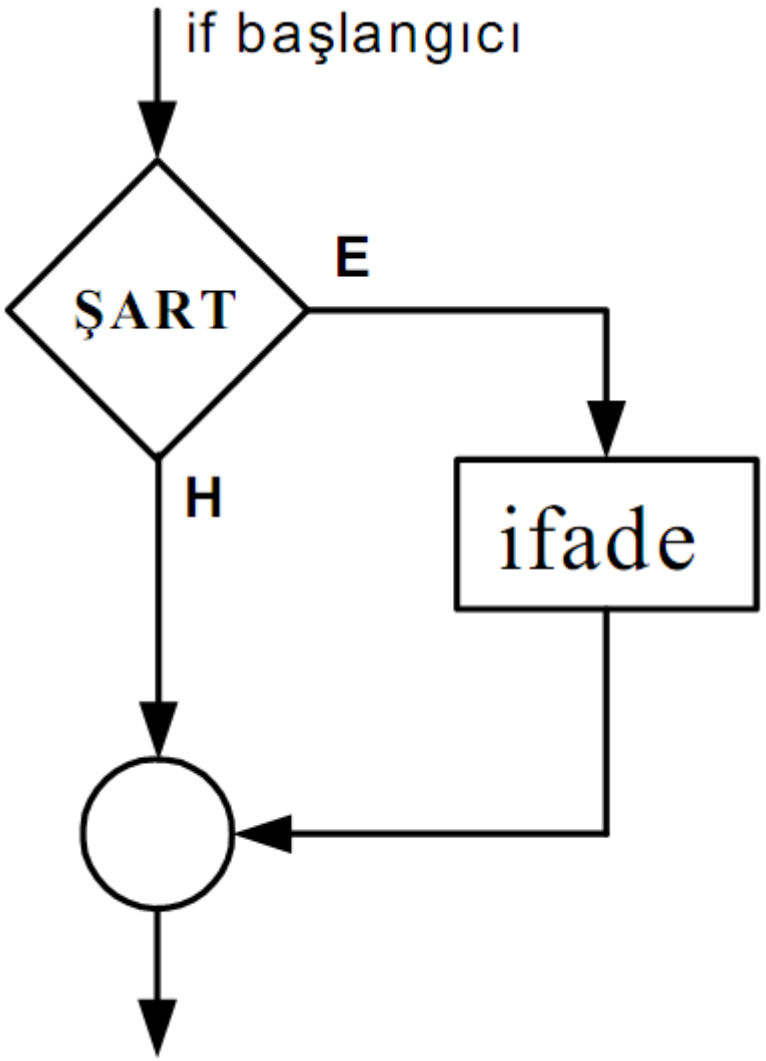
ifade 2; ...

... }

if (x>y)

{ printf ("x büyüktür");

printf ("y küçüktür"); }

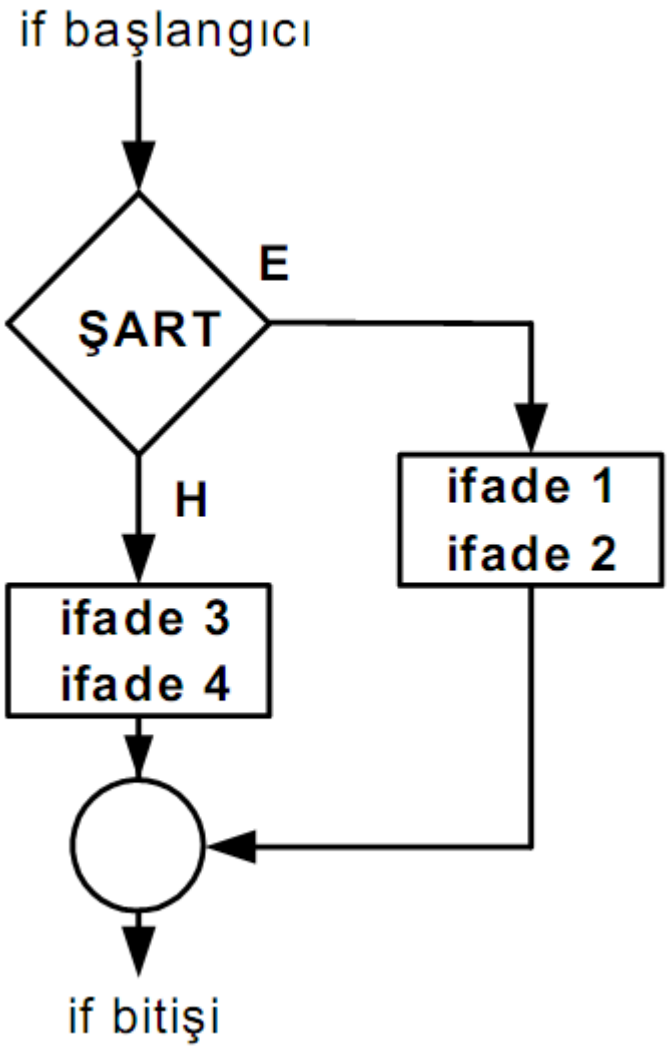


if-else koşulu

```
if (şart ) ifade 1;
else ifade 2;
```

```
if (x>y)
printf (“x büyüktür”)
else
printf (“y büyüktür veya eşittir”)
```

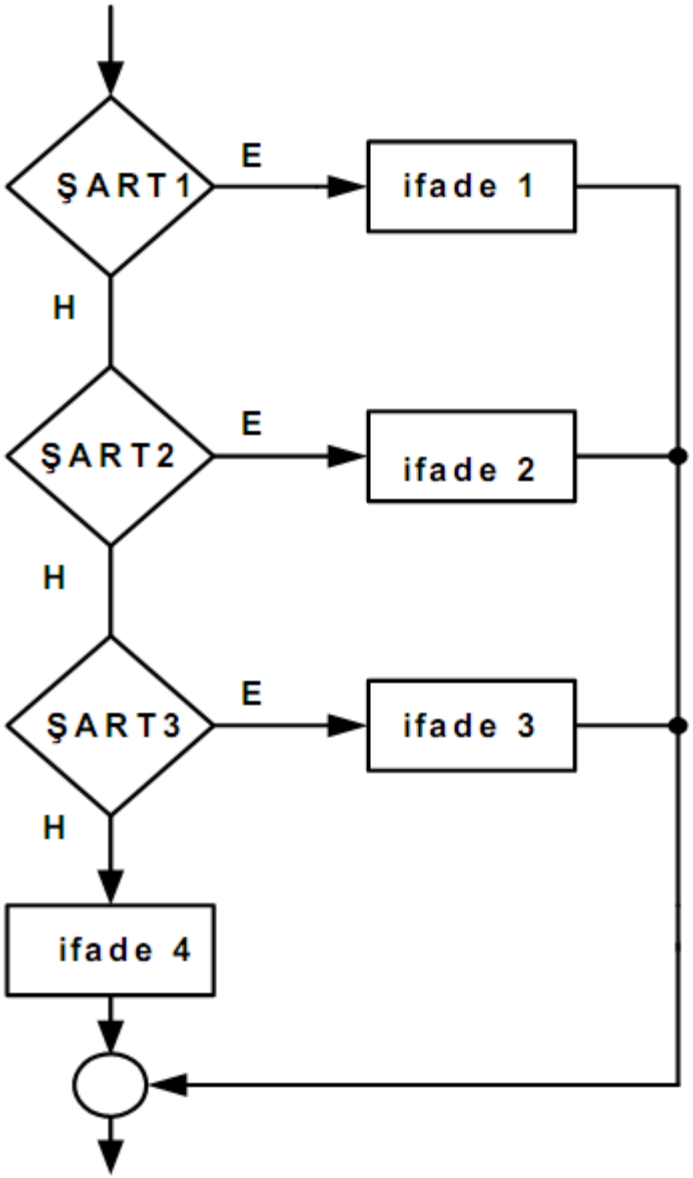
```
if (şart)
{
ifade 1; ifade2;
}
else
{
ifade 4; ifade 5;
}
```



İf else if koşulu

```
if (şart 1)
ifade 1;
else if (şart 2)
ifade 2;
else if (şart 3)
ifade 3;
.....
else
ifade 4;
```

```
if (x>y) printf ("x>y");
else if (x=y)
printf ("x=y");
else
printf ("x<y");
```



Örnek: Girilen bir tamsayının char tip sınırlarında olup olmadığını denetleyen program:

char→1 Byte → -128 den +127 e kadar

```c
#include<stdio.h>
#include<cstdlib>
#include "locale.h"
main()
{
system("cls");
int x;
char y;
setlocale(LC_ALL,"turkish");
printf("bir tamsayi giriniz:");
scanf("%d",&x);
printf("\n");
y=x;
if(x>=-128 && x<=127)
printf("girdiğiniz sayı char tipine uygundur.\nsayı=%d dir\n",x);
else
printf("girdiğiniz sayı char tipine uygun değildir=%d\n",y);
system("pause");
}
```

Örnek: Hava sıcaklığını size sorup sıcaklık değerine göre normal, sıcak yada soğuk bir gün diye cevap veren program.

20-30 derece→ Normal bir gün
>30 derece → Sıcak bir gün
20> derece → Soğuk bir gün

```c
#include<stdio.h>
#include<cstdlib>
#include "locale.h"
main()
{
setlocale(LC_ALL,"turkish");
int temp,k;
printf("Sıcaklık değerini giriniz\n");
scanf("%d",&temp);
system("cls");
if(temp<30 && temp>20)
printf("normal bir gün (Hava sıcaklığı=%d derece)\n",temp);
else if(temp>30)

printf("sıcak bir gün  (Hava sıcaklığı=%d derece)\n",temp);
else
printf("soğuk bir gün  (Hava sıcaklığı=%d derece)\n",temp);
system("pause");
}
```

Girilen tarihteki günün adını veren programı yazınız.

```
#include <studio.h>
main ( )
{
   int gun, ay, yıl ;
  long gt ;
  printf(“Tarihi gir”) ; scanf ( “%d %d %d “,&gun)
/* oncekı yıllardakı gun sayısını hesapla */
  gt=( yıl*1)*365 + yıl/4;
/* bu yildaki aylardaki gunleri ekle */
if (ay==2)  gt = gt + 31 ;
else if (ay ==3) gt = gt + 31 + 28 ;
else if (ay ==4) gt = gt + 31 + 28 +31;
else if (ay ==5) gt = gt + 31 + 28 +31+ 30 ;
else if (ay ==6) gt = gt + 31 + 28 +31+ 30 +31;
else if (ay ==7) gt = gt + 31 + 28 +31+ 30 +31+ 30 ;
else if (ay ==8) gt = gt + 31 + 28 +31+ 30 +31+ 30 + 31 ;
else if (ay ==9) gt = gt + 31 + 28 +31+ 30 +31+ 30 + 31+30 ;
else if (ay ==10)
gt = gt + 31 + 28 +31+ 30 +31+ 30 + 31+30 + 31;
else if (ay ==11)
gt = gt  + 31 + 28 +31+ 30 +31+ 30 + 31+30 + 31+ 30 ;
else if (ay ==12)
gt = gt + 31 + 28 +31+ 30 +31+ 30 + 31+30 + 31+ 30 +31;
/*Bu ayı ekle */
gt = gt+ gun;
if(yıl%4==0 && ay>2) gt =gt+1;
gt=gt %7,
if(gt==1)
printf(“Pazar”);
else if(gt==2)
printf(“Pazartesi”);
else if(gt==3)
printf(“Salı”);
else if(gt==4)
printf(“Carsamba”);
else if(gt==5)
printf(“Persembe”);
else if(gt==6)
printf(“Cuma”);
else if(gt==7)
printf(“Cumartesi”);
}
```

```c
#include <stdio.h>
main ( ) {
long int gun, ay, yil ;  long gt ;
  printf("Gün Ay Yıl olarak Tarihi giriniz\n") ;
  scanf ( "%ld%ld%ld",&gun,&ay,&yil) ;
  gt=( yil)*365 + yil/4;
if (ay==2)  gt = gt + 31 ;
else if (ay ==3) gt = gt + 59 ;
else if (ay ==4) gt = gt + 90;
else if (ay ==5) gt = gt + 120 ;
else if (ay ==6) gt = gt + 151;
else if (ay ==7) gt = gt + 181 ;
else if (ay ==8) gt = gt + 212 ;
else if (ay ==9) gt = gt + 242 ;
else if (ay ==10) gt = gt + 273;
else if (ay ==11) gt = gt  + 303 ;
else if (ay ==12) gt = gt + 334;

gt = gt+ gun;/*Girlen ayın günlerini ekleme */
if(yil%4==0 && ay>2) gt=gt+1;
gt=gt%7;
if(gt==2)       printf("Pazar");
if(gt==3)       printf("Pazartesi");
if(gt==4)       printf("Salı");
if(gt==5)       printf("Carsamba");
if(gt==6)       printf("Persembe");
if(gt==0)       printf("Cuma");
if(gt==1)       printf("Cumartesi");
}
```

FONKSİYONLAR

[geri dönüş değeri] <fonksiyon ismi>([paramatre])
{
    ANABLOK
}

Her fonksiyon ard arda tanımlanır. İç içe fonksiyon tanımlanamaz...

Yanlış fonksiyon kullanımına örnek:

```
main()
{
fonk()
{   }
}
```

Olması gereken:

```
main()
{
}
fonk()
{
}
```

Hiçbir fonksiyon 1'den fazla tanımlanamaz. En azından aynı isme sahip olmaz.

```
#include <stdio.h>
int fonk1();
int fonk2();
int fonk3();
main()
{       printf("Ben main'im\n");
        fonk1();
        fonk2();        }

int fonk1()
{       printf("Ben 1.fonksiyonum\n");        }

int fonk2()
{       printf("Ben 2. fonksiyonum\n"); fonk3();     }

int fonk3()
{       printf("Ben 3. fonksiyonum\n");       }
```

FONKSİYONLARIN GERİ DÖNÜŞ DEĞERLERİ(RETURN VALUE)

```
int x;

x = fonk();
```

Bir fonksiyonun çalışması bittikten sonra onu çağıran fonksiyona gönderdii değere geri dönüş değeri denir.

Fonksiyonların geri dönüş değerleri aritmetik işlemlere sokulabilir.

```
x = fonk() + a;
```
gibi..

GERİ DÖNÜŞ DEĞERİ OLUŞTURULMASI VE return ANAHTAR SÖZCÜĞÜ

```
#include <stdio.h>
int fonk()
{
        printf("Ben fonk'um\n");
        return 100;          }

main()
{
        int a;
        a = fonk();
        printf("fonk'un geri dönüş değeri=%d\n",a);      }
```

return anahtar sözcüğünün iki işlevi vardır:
1-Fonksiyonun çalışmasını bitirir. Bu durumda akış, onu çalıştıran fonksiyonda devam edecektir.
2-Geri dönüş değeri oluşturur.

Kullanım biçimi => return [ifade]

return İFADESİ NASIL OLUŞTURULUR?

return'ün yanındaki ifade önce derleyici yarafından programcının erişemediği geçici bir bölgeye alınır,oradan kullanılır.Örneğin a = fonk(); işleminde şunlar yapılır:
temp = fonk();

a = temp; Bu işlem bize aksettirilmez.

Fonksiyonun geri dönüş türü aslında geçici bölgenin türünü gösterir.return ifadesinin oluşturulması da geçici bölgeye yapılan gizli bir atamadır.

```
long fonk()
{
        printf("Ben fonk'um\");
        return 123000L;
}

main()
{
        long a;
        a = fonk();
        printf("%ld\n",a);
}
```

return anahtar sözcüğü kullanılmışsa fonksiyon ana bloğu bittiğinde sonlanır. Fonksiyonda return ile belirli bir değer verilmemişse rastgele bir geri dönüş değeri verilecektir.

Bir fonksiyonun geri dönüş değerine sahip olması kullanılmasını gerektirmez.

Fonksiyon başında void yazılırsa foksiyonun geri dönüş değerinin olmadığı analtılır.

```
void fonk()
{
 }
```

Böyle fonksiyonlarda return anahtar sözcüğü fonksiyonu sonlandırmak için kullanılır. void bir fonksiyonun geri dönüş değeri olmadığı için geri dönüş değeri kullanılmaya çalışılmamalıdır.

```
void fonk()
{       printf("selam\n");
        return;     }

main()
{       int a;
        a = fonk();/*bu kullanım tamamen yanlış*/
        printf("%d\n",a);/*bu kullanım tamamen yanlış*/
}
```

void FONKSİYONA NEDEN İHTİYAÇ VARDIR?

1-void anahtar sözcüğü okunabilirliği arttırır.
2-Geri dönüş değerine sahip olmayan fonksiyonları void olarak tanımlarsak yeni derleyicilerin vereceği uyarılardan kurtuluruz.

Parametre yerine void koyarsak fonksiyonun parametre almadığını gösteririz.

main fonksiyonunun da bir geri dönüş değeri vardır. main'e geri dönüş değeri çıkış kodu olarak(exit code) gönderilir. Bu değer işletim sisteminden daha sonra istenebilir. Ancak böyle bir bilgiye seyrek olarak ihtiyaç duyulur.

NESNELERİN FAALİYET ALANLARI VE ÖMÜRLERİ

Faaliyet alanları:

Bir nesnenin derleyici tarafından tanınabildiği program aralığını gösterir. 3 tür faaliyet alanı vardı.

1-Blok faaliyet alanı:Yalnızca bir blokta tanınır.
2-Fonksiyon faaliyet alanı:Bir fonksiyonun her yerinde tanınır.
3-Dosya faaliyet alanı:Programın her tarafında tanınır.

NESNELERİN FAALİYET ALANLARINA GÖRE SINIFLANDIRILMASI

1-Yerel değişkenler:

Blokların başlarında tanımlanmış değişkenler yerel değişkenlerdir. Bunlar tanımlandıkları blokta kullanılırlar.(blok faaliyet alanı)

```
void main(void)
{
int a;
        {
        int b;
        }
}
```

Burda a main fonksiyonunun tamamında b ise sadeec içteki blok'ta geçerlidir.Farklı faaliyet alanlarına sahip değişkenler aynı ada sahip olabilir.

2-Global değişkenler

Tüm blokların dışında tanımlanan değişkenlere global değişkenler denir. Dosya faaliyet alanı kuralına uyar. Kaynak kodun her yerinde tanınır.

```
#include <stdio.h>
int a;

void fonk(void)
{               a = 20; }

main()
{
        a = 10;
        fonk();
        printf("%d\n", a);/*ekrana 20 değeri basılır*/
}
```

3-Parametre değişkenleri

Bir fonksiyon parametre alabilir. Bunun için parametre değişkenlerinin tanımlanması gerekir. İki ayrı yöntemsel parametre bildirimi yapılır.

a.Eski biçim:

```
void fonk(a,b)
int a;
long b;
{
 }
```

b.Yeni biçim

```
void fonk(int a,long b)
{
 }
```

Bu biçimde parametre değişkenler aralarına virgül konularak tanımlanır.

```
void fonk(int a,b)/*YANLIŞ TANIMLAMA*/
```

Bu değişkenler fonksiyon faaliyet alanı kuralına uyarlar. Parametre değişkenine sahip bir fonksiyon aynı sayıda değişkenle çağırılmalıdır.

PARAMETRE AKTARIM KURALI

Parametreli bir fonksiyon çağırıldığında derleyici önce parametrelerden parametre değerlerine karşılık atama yapar, daha sonra programın akışı fonksiyona geçirilir.

```
#include <stdio.h>

int add(int a,int b)
{
        return a + b;
}

main()
{
        int x = 10, y = 20, z;

        z = add(x, y);
        printf("%d\n", z);
}
```

Fonksiyonlar sabitlerle de çağırılabilrler.

fonk(10, 20); gibi..

?: KARŞILAŞTIRMA OPERATÖRÜ

C dilinde if-else karşılaştırma deyiminin yaptığı işi sınırlı olarak yapan bir operatördür. Genel yazım biçimi:

(koşul deyimi)? deyim1 : deyim2;

- İlk önce koşul deyimi sınanır;olumluysa deyim1, aksi durumda deyim2 değerlendirilir. Deyim1 ve deyim2'de atama işlemi yapılamaz,ancak koşul deyiminde atama işlemi de yapılabilir.

(şart)? (E1) : (E2);

şart› doğru (1) ise E1 işlem görür.
şart› yanlış (0) ise E2 işlem görür.

x=10 y=5 ise
z=(x>y)? 20:15; (z=20)
z= (x>y) ? x:y; (z=x=10)

İki sayının karşılaştırılmasında ?
İşareti ile karşılaştırma

```c
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>
#include<locale.h>
main()
{
int x,y,c;
setlocale(LC_ALL,"turkish");
printf("1.sayıyı giriniz: ");
scanf(" %d",&x);
printf("\n");
printf("2.sayıyı giriniz: ");
scanf("%d",&y);
printf("\n");
c=(x<y) ? x:y;
system("cls");
printf("küçük olan sayı=%d dir\n",c);
getche();
}
```

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 : printf("Bir");
   case 2 : printf("İki");
   default : printf("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.

```
#include <stdio.h>
main()
{
  char islem;
  int s1, s2, s3;
  printf("Önce işlemi sonra sayıları girin ");
  scanf("%c%d%d",&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 : printf ("Hatalı işlem");
  }
  printf("\nSonuç = %d",s3);
}
```

1-12 arasında girilen sayıya göre mevsimlerin ismini veren programı yapınız

```
scanf("%d", &ay);
        switch (ay) {
                case 3:
                case 4:
                case 5: printf("ilkbahar"); break;
                case 6:
                case 7:
                case 8: printf("yaz"); break;
                case 9:
                case 10:
                case 11: printf("sonbahar"); break;
                case 12:
                case 1:
                case 2: printf("kış"); break;
        }
```

YAPILMASI İSTENENLER

1: Sınav notunu harfe dönüştüren programı yazınız (switch-case).
(>=90 :AA, 85-89:BA, 80-84:BB, 75-79:CB, 70-74:CC, 60-69:D, <60 :Başarısız)

2: Klavyeden alınan iki sayının toplamının 80-100 aralığında olup olmadığını test eden programı yapınız

3: Girilen bir sayının birler basamağı rakamı 5'ten küçük ve tek sayı ise Kırmızı yazan 5'e eşit büyük ve çift ise Mavi yazan diğer durumlarda Mor yazan programı yazınız.

4: Kullanıcıdan iki sayi girmesini isteyen ve büyük olan sayının karesini alıp ekranda gösteren programı yapınız.

5: Yaşı girilen bir kişinin seçimlerde oy verme yeterliliği olup olmadığını test eden program.