5. DÖNGÜLER

Giriş

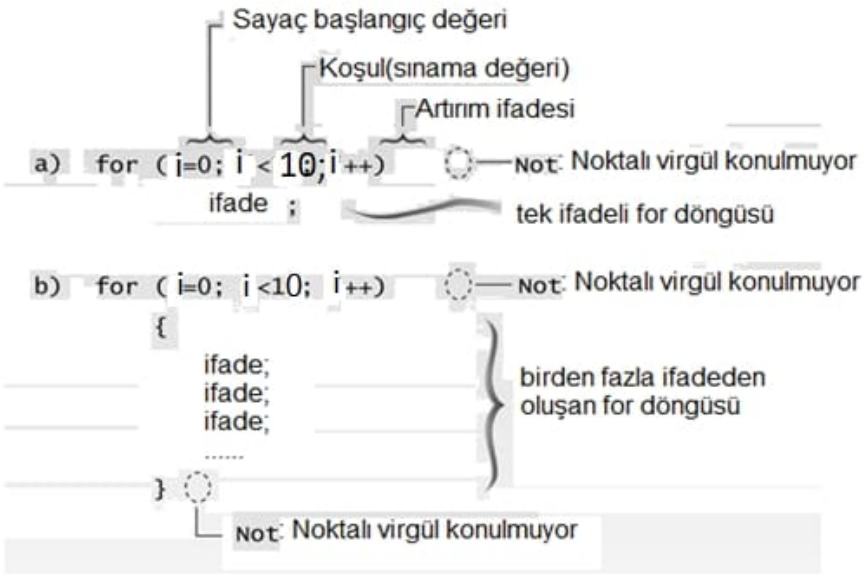
Döngüler, karar yapıları olduğu gibi program içerisinde yapılan hesaplamalara bağlı olarak akışı kontrol eder ve programın bir bölümünün belli sayıda tekrar etmesini sağlar. *Üç ve dördüncü bölümden hatırlanacağı gibi karar yapılarında kodun belli bölümünün çalıştırılıp belli bölümünün de çalıştırılmaması, program içerisinde belli deyimlerin doğru(True) ve yanlış(False) olmasına bağlıydı.* Döngülerde de programın belli bölümünün kaç defa çalışacağı karar yapılarında olduğu gibi program içerisinde belli deyimlerin doğru(True) ve yanlış(False) olmasına bağlıdır. Tekrarlar, her iterasyon sonunda kontrol edilen koşul doğru olduğu sürece devam eder. Koşulun yanlış olduğu noktada döngü biter ve akış döngünün sonundaki ifadeye geçer.

C++' ta for döngüsü, while döngüsü ve do….while döngüsü olmak üzere üç döngü çeşidi vardır.

5.1. for Döngüsü

for döngüsü hemen bütün programlama dillerinde kullanılan bir döngü çeşididir. for döngüsü anlaşılması en kolay döngü çeşitlerinden biridir. for döngünün, bütün döngü kontrol elemanları bir yerde toplanmıştır. Diğer döngü yapılarında kontrol elemanları döngünün dışında, döngünün başında, döngü gövdesinde, döngü sonunda yer alabilmektedir. Bu durum döngülerin anlaşılmasını, kodun okunabilirliğini for döngüsüne göre zorlaştırmaktadır. Buna rağmen ele alınan problemin özelliğine göre for döngüsü dışındaki döngüler de programlarda yağın olarak kullanılmaktadır. Bu bölümde, ilerleyen kısımlarda hangi tür problemlerin hangi döngülerle çözülmesi gerektiği konusunda açıklamalarda bulunulacaktır.

İpucu: for döngüsü genellikle döngüye girmeden önce döngü bloğundaki kodun kaç kere çalıştırılacağı bilindiği durumlarda kullanılır.



Şekil 5.1. *for döngüsü blok yapısı*

Şekil 5.1'de iki ayrı for döngüsü gösterilmiştir(a ve b). Her iki döngüde de for ifadesi döngüyü kontrol eder. for döngüleri, for anahtar kelimesi ile başlar. for parantezinde ilki *sayaç başlangıç değeri*, ikincisi *koşul* ve üçüncüsü *artırım* olmak üzere üç ifade yer alır. for döngülerinde sayaç başlangıç değeri, koşul ve artırım deyimleri genellikle aynı değişken ile tanımlanır. Değişkenin adı *Döngü Değişkenidir*. Şekil 5.1'de for

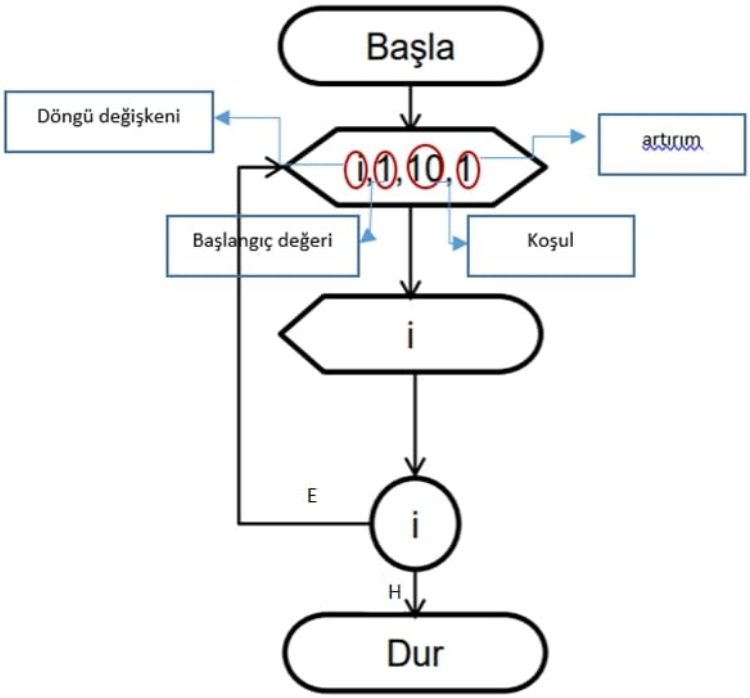
döngüsünde bu değişken i olarak belirlenmiştir $for(i = 0; i < 10; i++)$. for döngülerinde döngü değişkeni döngü gövdesindeki ifadeler çalıştırılmadan tanımlanmalıdır.

İpucu: 1. for döngülerinde for parantezinden sonra noktalı virgül konulmaz.

2. for gövdesinde tek bir ifade yer alıyor ise, bu ifadenin küme parantezi içerisine alınması gerekmez. Bu ifade küme parantezi içerisine alınırsa hata olmaz.

3. for gövdesinde birden fazla ifade yer alıyor ise bu ifadelerin küme parantezi içerisine alınması gerekir.

4. Küme parantezlerinin sonuna noktalı virgül konulmaz.



Şekil 5.2. *for döngüsü akış şeması*

5.1.1. for döngüsünün Çalışma Şekli

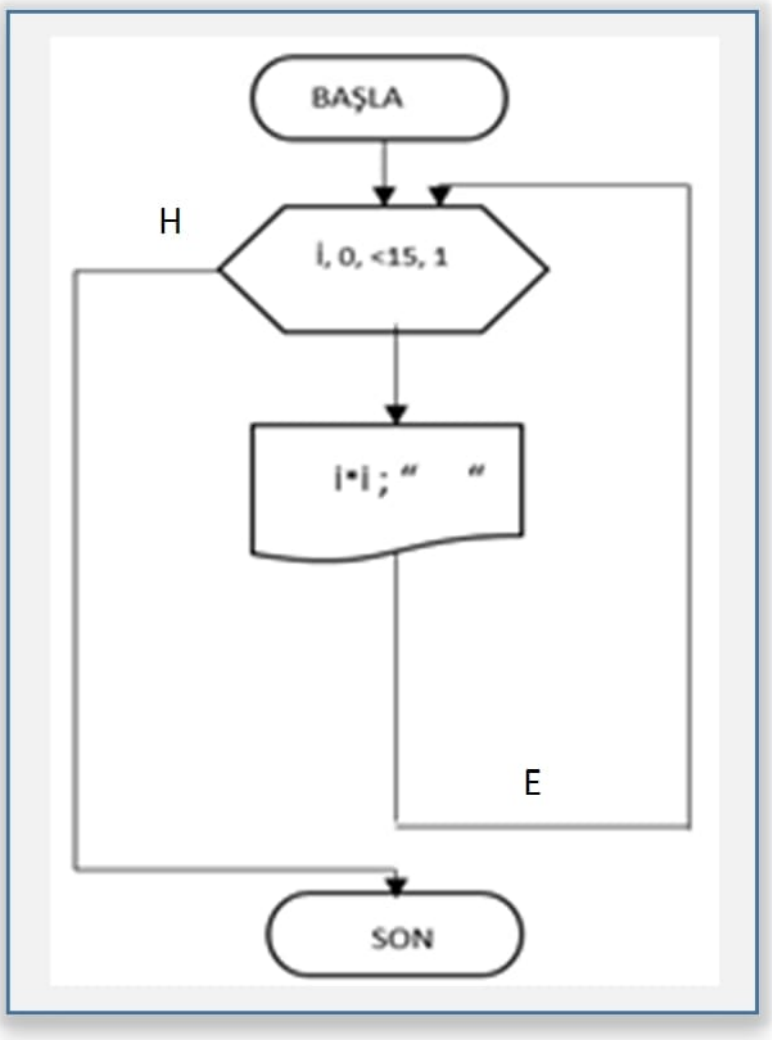
Programın çalışması, for anahtar kelimesine geldiğinde önce sayaç başlangıç değerine atama yapılır. Sonra koşul değeri ile başlangıç değeri test edilir. Eğer test doğru(True) çıkarsa döngü gövdesindeki ifadeler çalıştırılır. Sonra artırım değeri çalıştırılır ve test işlemi tekrarlanır. Sonuç doğru(True) çıkarsa döngü gövdesindeki ifadeler tekrar çalıştırılır. Bu işlem test sonucu yanlış(false) çıkana kadar tekrarlanır. Test sonucu yanlış çıktığında döngüden çıkılır ve programın akışı döngü bloğunu takip eden ilk ifadeye aktarılır.

İpucu: for döngüsüne girişte yapılan ilk testte(Başlangıç değeri koşul ile karşılaştırıldığında), test sonucu yanlış(false) çıkarsa döngü gövdesindeki ifadeler çalıştırılmadan programın akışı döngü bloğunu takip eden ilk ifadeye aktarılır.

5.1.2. Program 5.1

o ile 15 arasındaki sayıların karesini alan ve sonucu ekrana yazdıran algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız.

Çözüm:



Şekil 5.3 *Program 5.1'in akış şeması*

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

int main()
{
    int i;

    for (i = 0; i < 15; i++)
        cout << i * i<<" ";
    system("PAUSE");
    return 0;
}
```

Program 5.1'in C++ kodu

Programın çıktısı aşağıdaki gibidir:

```
0 1 4 9 16 25 36 49 64 81 100 121 144 169 196
```

Programın Çalışması: for parantezinde, döngü değişkeninin başlangıç değeri sıfır olarak atanmıştır. Bu atamadan sonra koşul kontrol edilir. Başlangıç değeri(Sıfır) koşul değerinden küçük(On Dört) olduğu için

döngü gövdesindeki ***cout << i * i<<" ";*** çalıştırılır. Ekrana sıfır yazılır. Sonra artırım değeri bir artar(Bir Olur). Koşul test edilir. Koşul sağlandığı için döngü gövdesindeki ifade bir kere daha çalıştırılır. Ekrana bir yazılır. Bu işlem artım değeri on beş olana kadar tekrarlanır. Artım değeri 15'e ulaştığında önceki iterasyonlarda olduğu gibi bu değer koşul ile test edilir ve karşılaştırma sonucu yanlış(False) olduğu için programın çalışması döngü bloğunun sonundaki ifadeye aktarılır.

İpucu: Bir C++ programı içerisinde, çalışma sırası for parantezine geldiğinde;

1. Döngü değişkeni parantez içerisinde tanımlanmış ise önce bu değişkene ilk değer atanır(Başlangıç değeri) ve sonra başlangıç değeri koşul değeri ile karşılaştırılır. Karşılaştırma sonucu doğru(true) olursa for gövdesindeki ifadeler çalıştırılır.
2. for parantezinde başlangıç değeri atama işlemi bir defa gerçekleşir.
3. Döngünün ikinci defa çalıştırılıp çalıştırılmayacağına, for parantezi içerisindeki artırım değeri ifadesi çalıştırılarak ve artırım değerinin yeni değeri ile koşul değeri test edilerek karar verilir. Döngü gövdesindeki ifadelerin tekrar çalıştırılıp çalıştırılmayacağına her defasında bu kontrol yapılarak karar verilir.
4. For döngüsünden çıkış, artım değeri ile koşul değerinin karşılaştırılması sonucu elde edilecek yanlış(false) değerine ulaşıldığında gerçekleşir.

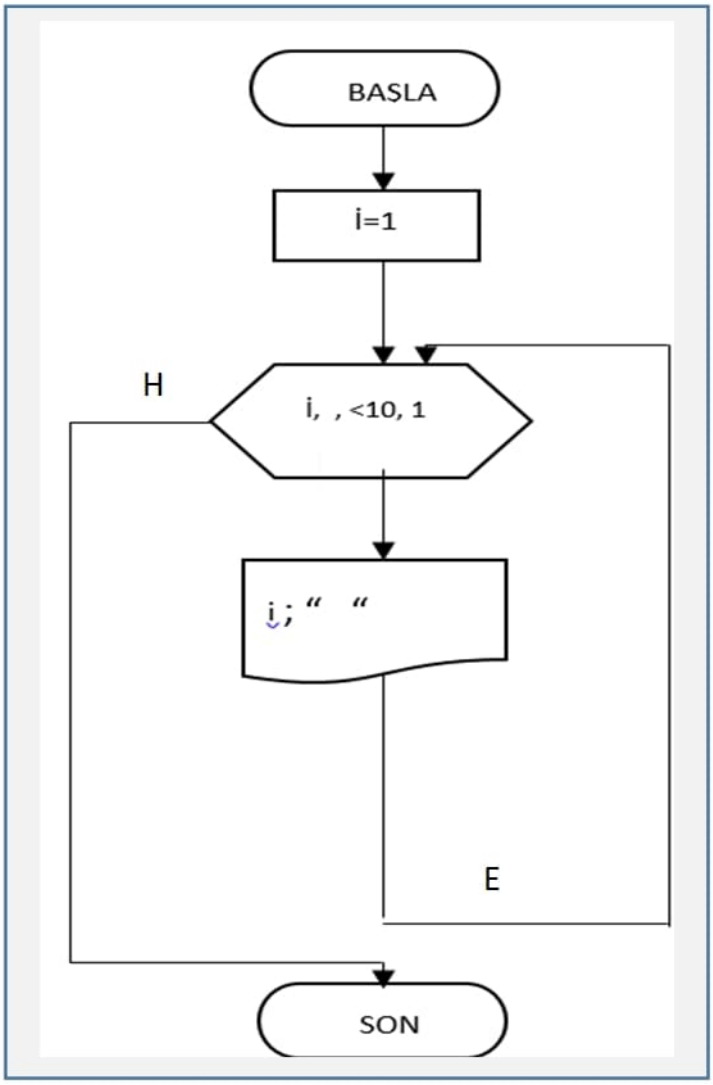
5.1.3. Program 5.2

Birden On'a kadar olan sayıları ekrana yazdıran algoritmanın akış şemasını ve C++ programını yazınız;

Not: döngünün başlangıç değerini for parantezinden önce, for parantezi dışında

tanımlayınız.

Çözüm:



Şekil 5.3 *Program 5.2'in akış şeması*

```
#include <iostream>
using namespace std;
int main()
{
    int i=1;
    for(;i<=10;i++){
        cout << i<<" ";
            }
    cout<<endl;
    system("PAUSE");
    return 0;
}
```

Program 5.2'in C++ kodu

Programın çıktısı aşağıdaki gibidir:

```
1 2 3 4 5 6 7 8 9 10
```

Problemin çözümü için çizilen akış şeması ve yazılan C++ koduna yakından bakılırsa döngünün başlangıç değeri for parantezinden önce atanmıştır. Problemi ilginç yapan da bu yaklaşımdır. for döngülerinde döngü değişkeninin tanımı, başlangıç değerinin atanması, döngüden çıkış koşulunun belirlenmesi ve artırım for parantezinin içerisinde yapılabileceği gibi(genellikle bu şekilde kullanılır) bütün bu tanımlamaların tamamı for parantezi dışında da yapılabilir. Program 5.2 için yazılan aşağıdaki C++ programını yazınız ve çalıştırınız.

```
#include <iostream>
using namespace std;
int main() {
    int i=1;
    for(;;){
        cout<<i<<" ";
        i++;
        if(i>10)
        break;

    }
    return 0;
}
```

Program çalıştırıldıktan sonra elde edilen ekran çıktısını bir önceki(program 5.2) programın ekran çıktısı ile karşılaştırınız.

```
1 2 3 4 5 6 7 8 9 10
```

Karşılaştırma sonucunda Yazılan bu yeni programın(elemanların tamamı for parantezi dışında tanımlanmıştır), çıktısı ile program 5.2'nin çıktısının aynı olduğunu göreceksiniz.

5.1.4. break Deyimi

break ifadesi döngü bloğu çalışırken belli koşullar altında döngüden tamamen çıkılmasını sağlar. Yukarıda en son yazılan C++ programında for parantezi ***for(; ;)*** şeklinde yazılmıştır. Aslında bu sonsuz bir döngü oluşturur. Yani program, sürekli döngü bloğu içerisinde çalışmaya devam eder ve sonsuz sayıda rakam üretir. Fakat bizim, yazdığımız bu programdan beklediğimiz çıktı, bir ile on arasındaki sayıların ekrana yazılmasıdır. Bunu sağlayabilmek için birden on 'a kadar sayılar ekrana yazıldıktan sonra, sonsuz olan bu döngüden çıkmak için **break** ifadesi kullanılmıştır.

İpucu: ***break*** deyimi, içerisinde yer aldığı döngüyü sonlandırmak için kullanılır. Bir döngü içerisinde çalışma sırası break deyimine geldiğinde program akışı döngünün tamamlanıp tamamladığına bakılmadan döngü sonundaki il ifadeye geçer.

5.1.5. continue Deyimi

continue deyimi döngü içerisinde program akışının başa dönmesini sağlar. Döngü içerisinde çalışma sırası ***continue*** deyimine geldiğinde, ***continue*** deyiminden sonraki satırlar çalıştırılmaz ve programın akışı döngü içerisindeki ilk ifadeye aktarılır.

5.1.6. Program 5.3

Bir ile yüz arasındaki sayılardan ilk 5 ve son beşini tek bir for döngüsü kullanarak ekrana yazdıran C++ programını yazınız.

Çözüm:

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

    for(int i=1;i<=100;i++){

        if(i>5 &&  i<95)
        continue;
        cout<<i<<" ";
    }
    return 0;
}
```

Programın çıktısı aşağıdaki gibidir:

```
1 2 3 4 5 95 96 97 98 99 100
```

Programın ürettiği çıktıdan da anlaşılacağı gibi ***if (i>5 && i<95)*** koşulu sağlandığında (parantezin içerisi bire eşit olduğunda) ***continue*** deyiminin altındaki ifade çalışmayacak, beşten büyük ve doksan beşten küçük sayılar ekrana yazılmayacaktır. Çünkü bu koşul doğru(true) olduğunda ***continue*** deyimi çalışacak ve program akışını döngünün başına geri döndürecektir.

İpucu: ***continue*** deyimi döngü içerisinden çıkmadan belli koşullar altında döngü gövdesindeki ifadelerin bir kısmının veya tamamının çalıştırılmamasını sağlar.

5.1.7. Program Blokları Arasında Tanımlanan Değişkenlerin Erişilebilirliği

Döngüler içerisinde birkaç ifadenin küme parantezleri ile sınırlandığı program bölümlerine ***program bloğu*** denir. Bu bloklar içerisinde tanımlanan değişkenlere, yalnızca tanımlı oldukları bloklar içerisinden erişilebilir. Blok içerisinde tanımlı olan değişkenlere blok dışından erişilemez. Bu yaklaşım aynı isimli değişkenlerin farklı bloklarda tanımlanıp kullanılabilmesini sağlar.

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

    for (i = 0; i < 15; i++)
    {
        int kare = i * i;
        cout << kare;
        cout << " ";
    }
    cout << endl;
    cout << kare << endl; // Kare değişkeni burada kullanılamaz
    system("PAUSE");
    return 0;
}
```

Program bloğu içerisinde tanımlanan değişken blok dışında kullanılamaz

5.1.8. for Parantezi İçerisinde Birden Fazla Başlangıç ve Artırım Deyimi Kullanmak

for parantezi içerisinde kullanılan deyimlerden başlangıç ve artırım deyimleri için birden fazla değişkene değer atanabilir. Koşul için bu kural geçerli değildir. Koşul değişkenine tek bir değer atanabilir. Parantez içerisindeki deyimlere birden fazla ilk değer ataması yapılacaksa atamalar virgül ile birbirinden ayrılmalıdır.

Aşağıda verilen örnekte i, j değişkenlerine farklı iki başlangıç değeri verilmiş ve bu atamalar virgül ile birbirinden ayrılmıştır. Diğer taraftan artırım deyimi için de i, j değişkenlerine farklı değerler atanmış ve bu atamalar da virgül ile birbirinden ayrılmıştır. Örnekten de görülebileceği gibi koşul olarak tek değişken ve tek bir atama yapılmıştır.

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

int main()
{
    int i, j;

    for (i = 14, j = 0; i >= 0; i--, j++)
    {
        cout << i * j << " ";
    }
    cout << endl;
    system("PAUSE");
    return 0;
}
```

Birden fazla ilk değer atama örneği

5.1.9 Farklı for Döngüsü Örnekleri

a-)

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    for (i = 0; i <10; i++); // Burada Noktalı virgül kullanılmamalıdır.
    {
        cout << i<<endl;
        cout << " ";
    }
    cout << endl;
    system("PAUSE");
    return 0;
}
```

b-)

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

int main()
{
    // i değişkeni döngüde tanımlanıyor
    for (int i = 0; i < 15; i++)
    {
        int kare = i * i;
        cout << kare;
        cout << " ";
    }
    cout << endl;

    cout << i << endl; // i değişkeni döngüde tanımlandı.Döngü dışında kullanılamaz

    system("PAUSE");
    return 0;
}
```

c-)

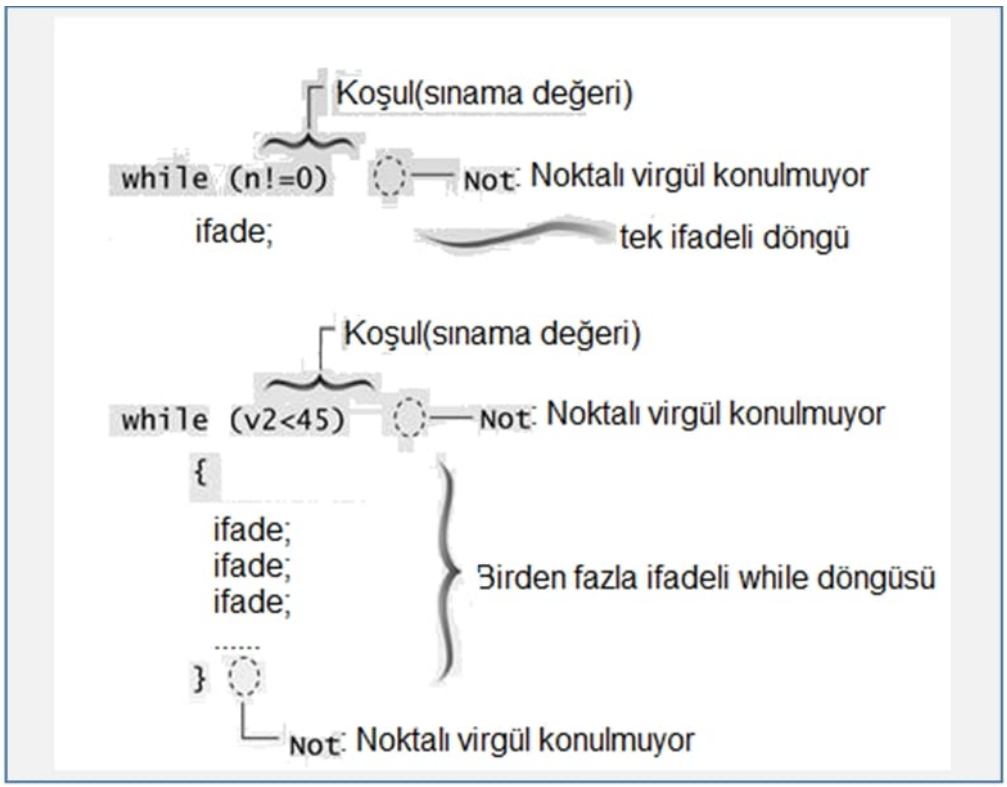
```
#include <iostream>
using namespace std;

int main()
{
    // for döngüsü 14'ten 0'a kadar geri geri sayar
    for (int i = 14; i >= 0; i--)
    {
        cout << i * i << " ";
    }

    cout << endl;

    system("PAUSE");
    return 0;
}
```

d-)

```
#include <iostream>
#include <locale.h>
using namespace std;
int main() {
    setlocale(LC_ALL,"Turkish");
    for(; ;){ //Sonsuz Döngü oluşturur.
        cout<<"İstanbul Üniversitesi"<<endl;
    }
    return 0;
}
```

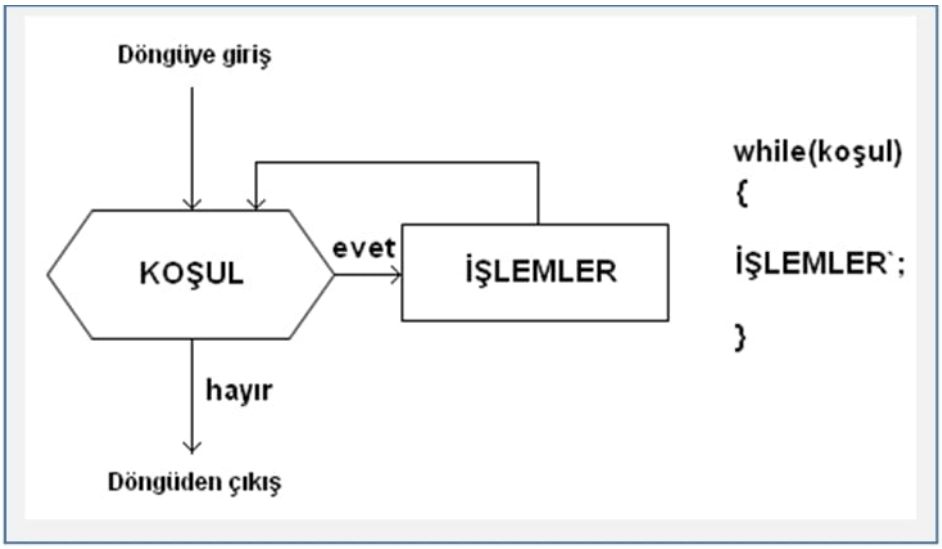
5.2. while Döngüsü

While döngüsü, daha çok döngüye girmeden önce döngü gövdesindeki ifadelerin kaç defa çalıştırılacağını bilmediğimiz durumda kullanılan bir döngüdür. Aşağıda while döngüsünün söz dizimi verilmiştir ***şekil 5.4.***



Şekil 5.4 *while döngüsünün söz dizimi*

While döngüsünde, while deyiminden sonra, parantez içerisinde koşul ifadesi yer alır. Fakat bu döngüde, ilk değer atama ve artırım deyimleri yer almaz. Bu açıdan bakıldığında while döngüsü for döngüsünün basitleştirilmiş halidir. while döngüsünde de koşul sınama değeri doğru(true) olduğu sürece, for döngüsünde olduğu gibi, döngü gövdesindeki ifadeler tekrar, tekrar çalıştırılır. Şekil 5.5'de while döngüsünün blok akış şeması verilmiştir.



Şekil 5.5*. while döngüsü blok akış şeması*

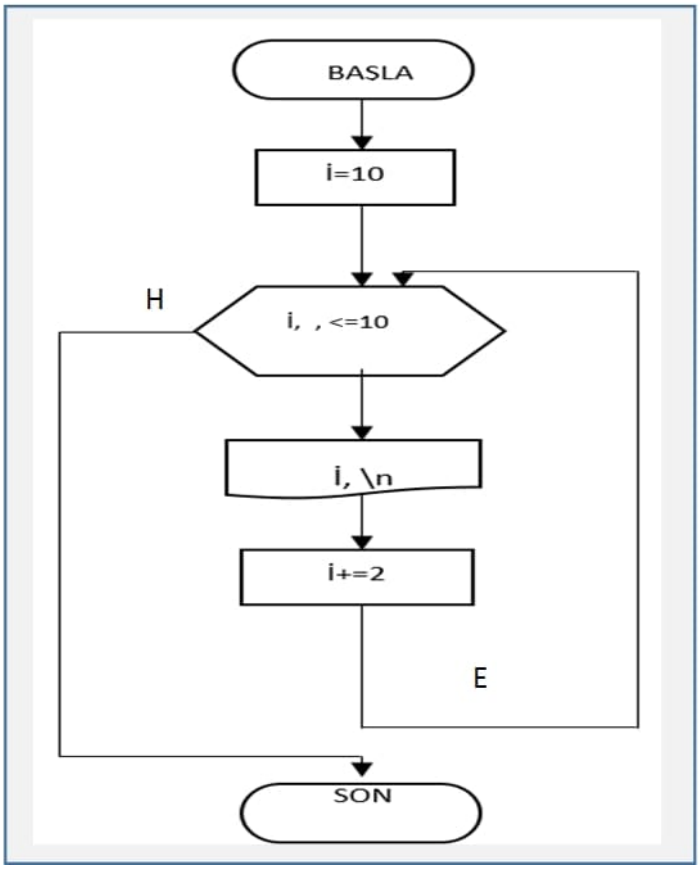
While döngüsünde ilk değer atama ifadesi olmasa da döngü değişkenine döngüye girmeden önce mutlaka bir ilk değer atanmalıdır. Aksi halde döngü içerisine, while parantezi içerisindeki koşul sağlanamayacağı için girilmez ve programın akışı doğrudan döngü bloğu sonuna aktarılır. While döngü bloğu içerisinde mutlaka döngü değişkeninin değeri değiştirilmelidir ki, döngüden çıkış değerine ulaşılabilsin.

While döngülerinde de for döngüsünde olduğu gibi döngü gövdesinde tekbir ifade yer alırsa bu ifadenin küme parantezleri içerisine alınması gerekmez. Fakat unutulmamalıdır bu tek ifade küme parantezleri içerisine alınsa da hata olmaz. Eğer while döngü gövdesinde birden fazla ifade yer alıyor ise bu ifadeler mutlaka küme parantezleri içerisine alınmalıdır.

5.2.1. Program 5.3

Sıfır ile 10 arasındaki çift sayıları ekrana yazdıran algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız.

Çözüm:



Şekil 5.6. program 5.3'ün akış şeması

```
#include <iostream>
using namespace std;
int main()
{
    int i = 0;
    while (i <= 10)
    {
        cout << i <<endl;

        i+=2;
    }
    cout << endl;
    system("PAUSE");
    return 0;
}
```

Program 5.3'ün C++ kodu

Program 5.3'ün ekran çıktısı:

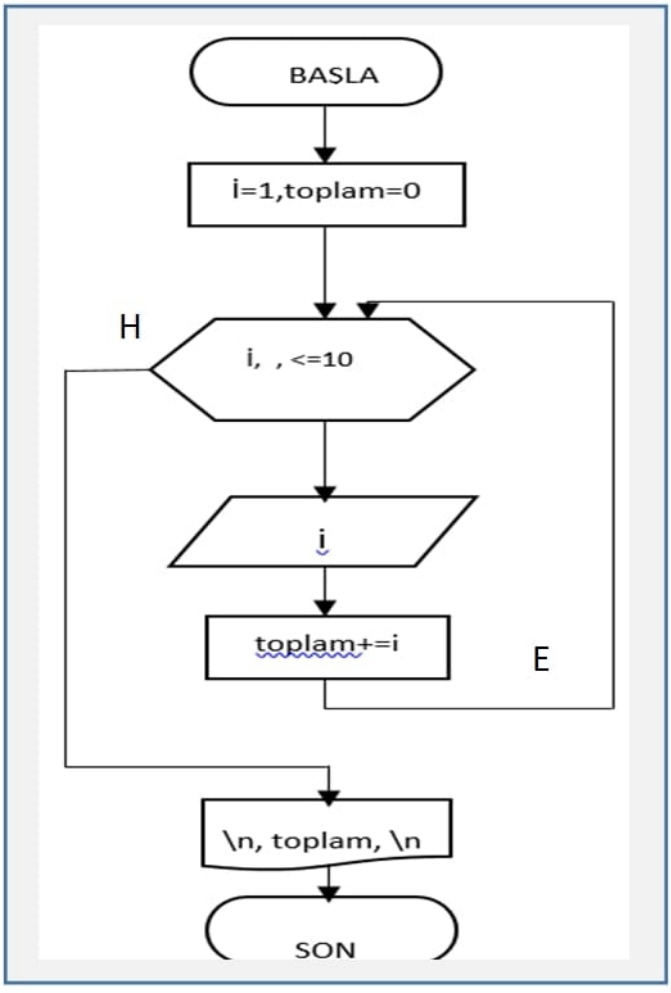
```
0
2
4
6
8
10
```

Şekil 5.6'dan veya program 5.3'ün C++ kodundan görüleceği gibi bu problemde, **for** döngülerinde olduğu gibi, döngü gövdesinin kaç defa çalıştırılacağı önceden bellidir. Burada bu örneğe yer verilmesinin nedeni ***while*** döngüleri ile bu türden problemle de çözülebilir. Programda döngüye girilmeden önce, döngü değişkenine ilk değer ataması yapılmıştır. Daha sonra while parantezindeki koşul test edilmiş ve koşul sağlandığı için döngü bloğundaki ifadeler çalıştırılmıştır. Bu işlem döngü değişkeni 12 olana kadar devam etmiş ve döngü değişkeni 12 olduğunda programın akışı döngü bloğunun dışındaki ilk ifadeye aktarılmıştır.

5.2.2. Program 5.4

Bir while döngüsü içerisinde klavyeden sıfır girinceye kadar girilen sayıları toplayıp sonucu ekrana yazdıran algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız?

Çözüm:



Şekil 5.7 *Program 5.4'ün akış şeması*

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

int main()
{
    int i = 1,toplam=0;

    while(i!=0)
    {
        cout << "0'dan farkli bir sayi girin: ";
        cin>>i;
    /*  toplam=toplam+i;  toplama işlemi için
        bu ifadede kullanılabilir. */
        toplam+=i;
    }

    cout <<endl<<toplam<<endl;

    system("PAUSE");
    return 0;
}
```

Program 5.4'ün C++ kodu

Program 5.4'ün Program 5.3'ten farkı, Program 5.4'te döngüden çıkmak için kontrolün tamamen kullanıcının seçeneğine sunulmasıdır. Program çalışırken döngü içerisinden her iterasyonda bir sayı girilmesi, girilen sayının toplama eklenmesi öngörülmüştür. Programın(Program 5.4), daha önce bu bölümde yazılan uygulamalardan farkı, önceden döngü gövdesindeki tekrar sayısının belirlenmemiş olması ve döngüden çıkışın tamamen kullanıcıya bırakılmış olmasıdır. Kullanıcı klavyeden sıfır girdiğinde döngüden çıkılacak ve programın çalışması döngü sonundaki ilk ifadeye(toplama sonucunun ekrana yazdırılması) aktarılacaktır. Bu yaklaşım program kontrolünün uygulamadan kullanıcıya geçmesini birazda olsa sağlamıştır.

5.3. Sonsuz Döngüler

Sonsuz döngü, döngü çalışmaya başladıktan sonra, döngü bloğunun dışarıdan müdahale edilmediği sürece çalışmaya devam etmesi anlamına gelmektedir. Aşağıda iki farklı sonsuz döngü örneğinin C++ kodları verilmiştir.

```
#include <iostream>
using namespace std;
int main() {
    int i=0;
    while(1)
     cout<<i++;
    return 0;
}
```

Döngünün sonsuz olması while parantezindeki (1) den kaynaklanmaktadır. Bu koşul programın çalışması sırasında sağlanmaz.

```
9745984599460046014602460346044605460646
2746284629463046314632463346344635463646
5746584659466046614662466346644665466646
8746884689469046914692469346944695469646
1747184719472047214722472347244725472647
4747484749475047514752475347544755475647
7747784779478047814782478347844785478647
0748084809481048114812481348144815481648
3748384839484048414842484348444845484648
```

Programın ürettiği çıktıdan bir kesit

```
#include <iostream>
using namespace std;
int main() {
    int i=0;
    while(true)
     cout<<i++;
    return 0;
}
```

Yanda verilen bu program da sonsuz döngü oluşturacaktır çünkü while parantezinin içi hiçbir zaman ***false*** olmayacaktır.

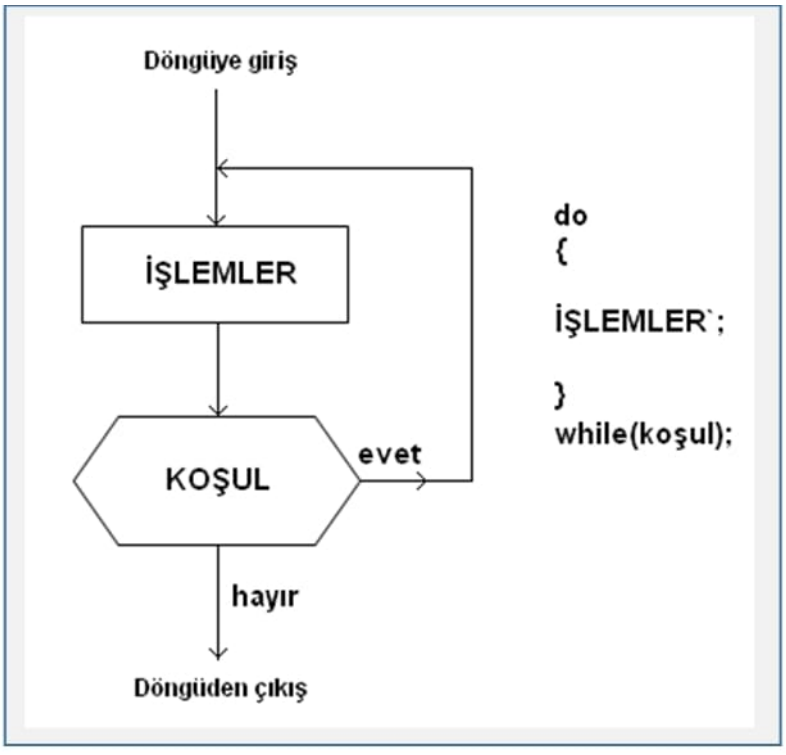
```
9745984599460046014602460346044605460646
2746284629463046314632463346344635463646
5746584659466046614662466346644665466646
8746884689469046914692469346944695469646
1747184719472047214722472347244725472647
4747484749475047514752475347544755475647
7747784779478047814782478347844785478647
0748084809481048114812481348144815481648
3748384839484048414842484348444845484648
```

Programın ürettiği çıktıdan bir kesit(önceki programla aynı çıktı üretilmiştir)

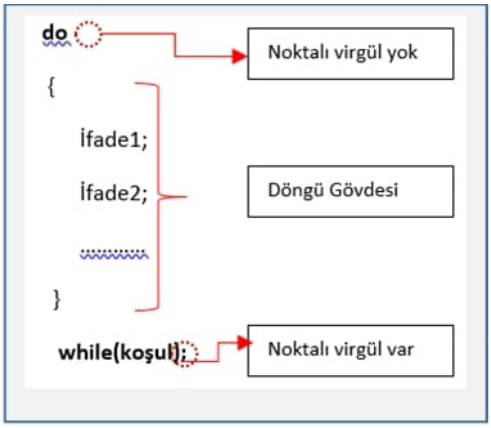
5.4. do….while Döngüsü

"***do … while***" döngüsü diğer döngüler gibi aynı işlemleri birçok kez tekrarlamak için kullanılır. Farklı olarak, bu döngüde koşul sınaması yapılmadan döngüye girilir ve döngü gövdesindeki ifadeler en az bir kere işletilir. ***do….while*** döngülerinde koşul kontrolü döngü sonundaki ***while*** parantezindeki deyim ile yapılır. Bu parantezin sonunda noktalı virgül vardır. Bu döngü yapısında da koşul sağlandığı sürece çevrim tekrarlanır.

Koşul tek bir karşılaştırmadan oluşabileceği gibi birden çok koşulun mantıksal operatörler ile birleştirilmesi ile de oluşturulabilir.



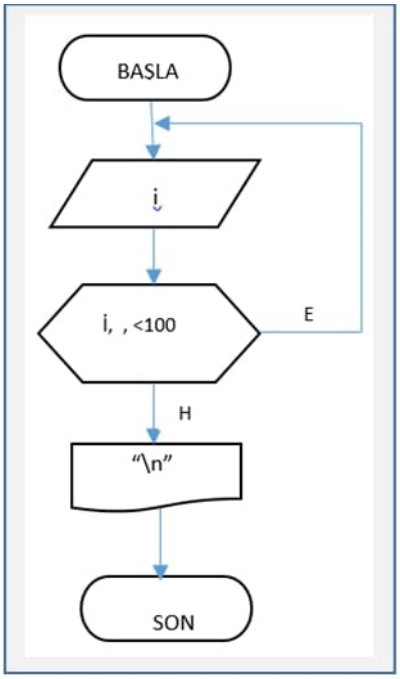
Şekil 5.8 *do….while döngüsünün akış şeması*



Şekil 5.9. *do….while döngüsünün söz dizimi*

5.4.1. Problem 5.5

Bir do….while döngüsü içerisinde, kullanıcının 100'den büyük veya 100'e eşit bir sayı girişine kadar klavyeden sayılar girmesini sağlayan algoritmanın akış şemasını çiziniz ve C++ kodunu yazınız?



Şekil 5.10 *Problem 5.5'in akış şeması*

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

int main()
{
    int i;

    do
    {
        cout << "100'den kucuk bir sayi girin: "
        cin >> i;
    } while (i < 100);

    cout << endl;

    system("PAUSE");
    return 0;
}
```

Problem 5.5'in C++ kodu

5.4.2. Problem 5.6

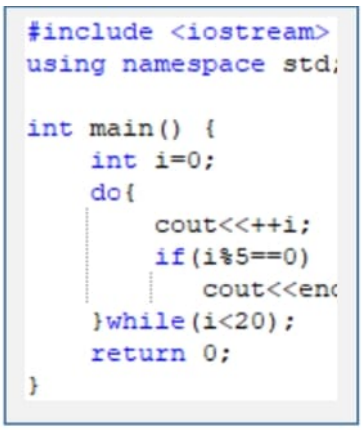
Bir do….while döngüsü içerisinde klavyeden girilen sayının karesini alan, sonucu ekrana yazdıran ve işleme devam edip etmeme tercihini kullanıcının seçimine bırakan algoritmanın C++ kodunu yazınız?

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    char c;

    do
    {
        cout << "Bir sayi girin: ";
        cin >> i;
        cout << "Girilen sayinin karesi: " << i * i << endl << endl;

            cout << "Baska bir sayi daha girmek istiyor musunuz? (e/h)? ";
            cin >> c;

    } while (c != 'h');

    cout << endl;
    system("PAUSE");
    return 0;
}
```

Problem 5.6'nın C++ Kodu

5.4.3. Problem 5.7

Birden yirmiye kadar olan sayıları her satırda 5 sayı olacak şekilde ekrana yazılmasını sağlayan algoritmanın C++ kodunu yazınız?



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

int main() {
    int i=0;
    do{
        cout<<++i;
        if(i%5==0)
            cout<<endl;
    }while(i<20);
    return 0;
}
```

Problem 5.7'nin C++ Kodu

Bölüm Özeti

Programların geliştirilmesi sırasında, genel olarak döngülerin kullanılma nedenlerini öğrendik. Yapısal programlamada, döngülerin daha kısa kod yazılması için önemli araçlar olduğunu gördük.

Daha sonra C++ 'ın desteklediği üç döngü yapısını inceledik: for döngüleri, while döngüleri ve do while döngüleri. Bu döngülerin birbirleri arasındaki farkları ve hangi döngünün hangi tür problemle karşılaşınca tercih edileceğini kavradık. Her üç döngünün blok akış şemalarının çizilmesi ve C++ kodlarının yazılması da ayrıca bu bölümde ele alındı.

Bundan başka, sonsuz döngülerin nasıl oluştuğu ve çalışma şekilleri de gözden geçirildi.

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

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

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

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

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