**Döngü Deyimleri**

Bir programın yinelemeli olarak çalıştırılmasını sağlayan deyimlere döngü deyimleri(loop statements) denilmektedir. C’de döngü deyimleri ikiye ayrılır:

1. While döngüleri
2. For döngüleri

**While Döngüleri**

While döngüleri de kendi içinde ikiye ayrılır:

1. Kontrolün başta yapıldığı while döngüleri
2. Kontrolün sonda yapıldığı while döngüleri(do-while döngüleri)

**Kontrolün Başta Yapıldığı while Döngüleri**

Bir koşul sağlandığı sürece yinelemeye yol açan döngülere while döngüleri denilmektedir. Genel biçimi şöyledir:

while(<int türden ifade>)

<deyim>;

while döngü deyiminin parantezi içerisinde ifade zero veya non-zero değerler olabilir aynı if deyiminde olduğu gibi burda da 0 false ve diğer tüm değerler ise true olarak kabul edilir. Bu durum Java ve C# gibi dillerde farklıdır. Parantez içerisinde ifade non-zero olduğu sürece while döngüsü içerisinde bulunan deyim veya deyimler çalıştırılır. Eğer while parantezi içerisindeki değer sıfırsa bu durumda döngü sonladırılır. while deyiminin kendisi de dışarıdan bakıldığında tek bir deyimdir. Örneğin:

#include <stdio.h>

int main(void)

{

int i = 1;

while(i < 10){

printf(“i = %d\n”, i);

i++;

}

printf(“Programin sonu\n”);

}

Burada:

while (i < 10) {

printf(“i = %d\n”, i);

i++;

}

deyimi tek bir deyimdir.

Bazen while döngüleri yanlışlıkla boş deyim ile kapatılabilmektedir:

#include <stdio.h>

int main(void)

{

int i = 1;

while (i < 10); //dikkat boş deyim

{

printf(“i = %d\n”,i);

i++;

}

printf(“Programin sonu\n”);

}

Bu durumda bir hata mesajı oluşmaz. Fakat şüphesiz program yanlış çalışır. Bazı durumlarda döngü deyimlerinde programcının algoritması gereği bilerek boş deyim koyması gerekebilir. Bu durumda programcı okunabilirlik açısından noktalı virgülü bir alt satıra koymalıdır. Örneğin:

while(i < 10)

;

Program içerisinde belirli bir sayı kadar dönen döngüler çok sık olarak kullanılmaktadır. while döngü deyimi ile belirli bir sayıda dönen örnek döngü kalıpları aşağıdaki biçimlerde oluşturulabilir:

1. n bir tamsayı türünden değişken olmak üzere n-kez dönen döngü while döngü deyimi ile şu şekilde oluşturulabilir:

while(n-- > 0)

<deyim>;

Burada --(azaltma) operatörünün sonek kullanımının ürettiği değer karşılaştırma işlemine sokulmaktadır. Örneğin:

#include <stdio.h>

int main(void)

{

int n;

printf(“Pozitif bir sayi giriniz?\n”);

scanf(“%d”, &n);

while(n-- > 0)

{

printf(“Merhaba\n”);

}

printf(“n = %d\n”,n);

printf(“Programin sonu\n”);

}

Bu kalıpta dikkat edilmesi gereken n değişkeninin içerisindeki değer döngünün her adımında azaltıldığından döngüden çıkıldıktan sonra n içerisindeki değer -1 olmaktadır. Dolayısıyla eğer n değişkeni içerisindeki değer döngüden sonra gerekiyorsa döngüye girmeden önce bir değişkene atanmalı ve o değişken ile döngüye girilmelidir. Örneğin:

#include <stdio.h>

int main(void)

{

int n;

printf(“Pozitif bir sayi giriniz?\n”);

scanf(“%d”,&n);

int temp = n;

while(temp-- > 0)

{

printf(“Merhaba\n”);

}

printf(“n = %d\n”,n);

printf(“temp = %d\n”,temp);

}

1. n bir tamsayı türünden ve i de n ile aynı türden bir değişken olmak üzere n-kez dönen döngü deyimi ile şu şekilde oluşturulabilir:

i = 0;

while(i < n)

{

//..

i++;

}

Burada döngü içerisinde n değişkeninin değeri herhangi bir şekilde değiştirilmediğinden döngüden sonra da kullanılabilmektedir. Örneğin:

#include <stdio.h>

int main(void)

{

int n;

printf(“Pozitif bir sayi giriniz?\n”);

scanf(“%d”,&n);

int i = 0;

while(i < n)

{

printf(“Merhaba\n”);

++i;

}

printf(“n = %d\n”,n);

}

1. n bir tamsayı ve i de n ile aynı türden bir değişken olmak üzere n-kez dönen while döngü deyimi ile şu şekilde oluşturulabilir:

i = 0;

while(i++ < n)

<deyim>;

Aslında bu kalıp 2.maddede anlatılan kalıbın başka bir yazılışıdır diyebiliriz. Bu kalıpta ++(arttırma) operatörünün son ek kullanımının ürettiği değer karşılaştırma işlemine sokulmaktadır. Örneğin:

#include <stdio.h>

int main(void)

{

int n;

printf(“Pozitif bir sayi giriniz?\n”);

scanf(“%d”,&n);

int i = 0;

while(i++ < n)

{

printf(“Merhaba\n”);

}

printf(“n = %d\n”,n);

}

while döngü deyimiyle sonsuz döngü şu şekilde oluşturulabilir:

while(1)

<deyim>

Parantez içerisindeki ifade 1 değerinde olduğundan koşul her zaman sağlanacaktır.

Dikkat aşağıdaki program sonsuz döngü değildir:

#include <stdio.h>

int main(void)

{

int n = 10;

int i = 0;

while(i-- < n)

;

printf(“n = %d\n”,n);

}

Burada i değeri int türünün negatif taraftan sınırları dışına çıktığında i değişkenin içerisinde int türü için pozitif en büyük değer olacağından döngü sonlanacaktır.

**Kontrolün Sonda Yapıldığı while Döngü Deyimi(do – while Döngüleri)**

Bu döngülerin çalışma biçimi kontrolün başta yapıldığı while döngüleriyle aynıdır. Yani while parantezi içerisindeki ifade doğru olduğu sürece döngü yinelenmektedir. Bu döngülerde tek fark kontrol noktasının sonda olmasıdır. Bu da bu döngünün bir defa dönmesini garanti altına almaktadır. Genel biçimi şu şekildedir:

do{

<deyim>

}while(<int türden ifade>);

Burada koşul gerçekleşmese de akış do-while döngü deyimine geldiğinde döngü içerisindeki deyim bir kez çalıştırılır. Örneğin:

#include <stdio.h>

int main(void)

{

int i;

int n;

i = 10;

n = 8;

do{

printf(“Dongu ici i = %d\n”,i);

++i;

}while(i < n);

printf(“Dongu disi i = %d\n”,i);

}

Burada i < n koşulu yanlış olmasına karşın döngü içerisindeki deyim bir kez çalıştırıldığından i değeri 11 olmaktadır.

do-while döngülerinde while parantezinin sonundaki noktalı virgül boş deyim belirtmez. Sentaksın bir parçasıdır.

do-while döngülerine seyrek gereksinim duyulmaktadır. Bazı durumlarda okunabilirliği ve algılanabilirliği arttırmaktadır.

Örnek Soru: 1’den klavyeden girilen n değerine kadar tamsayıların toplamını bulan C programını yazınız ve test ediniz.

#include <stdio.h>

int main(void)

{

int i = 0;

int n;

printf(“Bir sayi giriniz?\n”);

scanf(“%d”,&n);

int sum = 0;

while(i <= n){

sum += i;//sum += i++;

i++;

}

printf(“Sum : %d\n”,sum);

}