Alıştırmalar:

3.1 Aşağıdaki soruların her birindeki boşlukları doldurun.

a) Yürütülecek eylemler ve eylemlerin yürütülmesi gereken sıra açısından bir sorunu çözmeye yönelik prosedüre …….. denir.

b) İfadelerin bilgisayar tarafından yürütülme sırasını belirtmeye ..... denir.

c) Tüm programlar üç tür kontrol ifadesi cinsinden yazılabilir bunlar: ……., …….. ve..... .

d) ……..Seçim ifadesi, bir koşul doğru olduğunda bir eylemi, bu koşul yanlış olduğunda başka bir eylemi yürütmek için kullanılır.

e) Parantez ({ ve }) içinde gruplandırılmış çeşitli ifadelere …….. denir.

f) …… Yineleme ifadesi, bir ifadenin veya ifadeler grubunun, bazı koşullar doğru kalırken tekrar tekrar yürütülmesi gerektiğini belirtir.

g) Belirli sayıda yinelemeye …….. yineleme denir.

h) Bir dizi ifadenin kaç kez tekrarlanacağının önceden bilinmediği durumlarda yinelemeyi sonlandırmak için …… değeri kullanılabilir.

3.1

a) Algoritma.

b) Program kontrolü.

c) Sıralama, seçme, yineleme (Sequence, selection, iteration).

d) if... else.

e) Bileşik ifade veya blok.

f) while

g) Sayıcı kontrollü.

h) Gözcü değer.

3.2 Her biri x tamsayı değişkenine 1 ekleyen dört farklı C ifadesi yazın.

```
x = x + 1;
x += 1;
++x;
x++;
```

3.3 Aşağıdakilerden her birini gerçekleştirmek için tek bir C ifadesi yazın:

a) *= operatörünü kullanarak değişkeni 2 ile çarpın.

b) = ve * operatörlerini kullanarak değişkeni 2 ile çarpın.

c) count değişkeninin değerinin 10'dan büyük olup olmadığını test edin. Eğer öyleyse, "Sayı 10'dan büyük" yazdırın.

d) Bölümün bölene bölünmesinden sonra kalanı hesaplayın ve sonucu bölüme atayın. Bu ifadeyi iki farklı şekilde yazın.

e) 123.4567 değerini iki haneli hassasiyetle yazdırın. Ne yazdırılır?

f) 3.14159 kayan nokta değerini ondalık noktanın sağındaki üç basamakla yazdırın. Ne yazdırılır?

a) x *= 2;

b) x = x * 2;

c)
```
if (count > 10) {
puts("Count 10 dan buyuk.");
}
```

d)
```
bolum %= bolen;
bolum = bolum % bolen;
```

e)
```
printf("%.2f", 123.4567);
```
123.46 yazdırılır

f)
```
printf("%.3f\n", 3.14159);
```
3.142 yazdırılır.

3.4 Aşağıdaki görevlerin her birini gerçekleştirmek için bir C ifadesi yazın.

a) x değişkenini int türünde olacak şekilde tanımlayın ve 1'e ayarlayın.

b) sum değişkenini int türünde olacak şekilde tanımlayın ve 0'a ayarlayın.

c) x değişkenini sum değişkenine ekleyin ve sonucu sum değişkenine atayın.

d) "Toplam:" ifadesini ve ardından toplam değişkeninin değerini yazdırın.

a) `int x = 1;`
b) `int sum = 0;`
c) `sum += x;` yada `sum = sum + x;`
d) `printf("Toplam: %d\n", sum);`

3.5 Alıştırma 3.4'teki ifadeleri, 1'den 10'a kadar olan tam sayıların toplamını hesaplayan bir programda birleştirin. Hesaplama ve artırma ifadeleri arasında döngü yapmak için while ifadesini kullanın. Döngü, x >10 olduğunda sona ermelidir.

```
1
2 #include <stdio.h>
3
4 int main(void) {
5    int x = 1;
6    int sum = 0;
7
8    while (x <= 10) {
9        sum += x;
10       ++x;
11   }
12
13    printf("Toplam: %d\n", sum);
14 }
```

3.6 Aşağıdaki görevlerin her birini gerçekleştirmek için tekli C ifadeleri yazın:

a) scanf ile x tamsayı değişkenini girin. %d dönüştürme özelliğini kullanın.

b) scanf ile y tamsayı değişkenini girin. %d dönüştürme özelliğini kullanın.

c) Tamsayı değişkeni i'yi 1'e ayarlayın.

d) Tamsayı değişkeninin üssünü 1'e ayarlayın.

e) Tamsayı değişkeninin üssünü x ile çarpın ve sonucu üsse atayın.

f) i değişkenini 1 artırın.

g) while ifadesinin koşulunda y'den küçük veya ona eşit olup olmadığını görmek için i'yi test edin.

h) printf ile tamsayı değişkeninin üssünü çıkışa gönderin.

a) `scanf("%d", &x);`

b) `scanf("%d", &y);`

c) `int i = 1;`

d) `int power = 1;`

e) `power *= x;`

f) `++i;`

g) `while (i <= y)`

h) `printf("%d", power);`

3.7 Önceki alıştırmadaki ifadeleri kullanarak x'in y üssünü hesaplayan bir C programı yazın. Programda bir while yineleme kontrol ifadesi bulunmalıdır.

```
1
2 #include <stdio.h>
3
4 int main(void) {
5    printf("%s", "ilk tamsayiyi girin ");
6    int x = 0;
7    scanf("%d", &x);
8    printf("%s", "ikinci tamsayiyi girin: ");
9    int y = 0;
10   scanf("%d", &y);
11
12    int i = 1;
13    int power = 1;
14
15    while (i <= y) {
16       power *= x;
17       ++i;
18    }
19
20    printf("%d\n", power);
21 }
```

3.8 Aşağıdakilerin her birindeki hataları bulun ve düzeltin:

a)
```
while (c <= 5) {
product *= c;
++c;
```

b)
```
scanf("%.4f", &value);
```

c)
```
if (gender == 1) {
puts("Woman");
}
else; {
puts("Man");
}
```

a) Hata: While gövdesinin kapanış sağ parantezinin eksik olması.

Düzeltme: ++c; ifadesinden sonra kapanış sağ parantezini ekleyin.

b) Hata: scanf dönüştürme spesifikasyonunda kullanılan hassasiyet.

Düzeltme: Dönüştürme belirtiminden .4'ü kaldırın.

c) Hata: if…else ifadesinin else kısmından sonra gelen noktalı virgül

Mantık hatası: Noktalı virgül ile biten ifade her zaman yürütülür.

Düzeltme: else'den sonra noktalı virgülü kaldırın.

3.9 Desimal 100'den 1'e kadar olan tam sayıların toplamını hesaplaması gereken aşağıdaki while yineleme ifadesinde (z'nin 100 değerine sahip olduğunu varsayalım) yanlış olanı bulun?

```
while (z >= 0) {
sum += z;
}
```

while ifadesinde z değişkeni ile ilgili koşul herzaman doğru olur ve sonlandırma koşulu hiçbir zaman sağlanmaz. Bu nedenle sonsuz bir döngü oluşturulur. Sonsuz döngüyü önlemek için z'nin 0 olacak şekilde azaltılması gerekir.

4.1 Aşağıdaki ifadelerin her birindeki boşlukları doldurun.

a) Sayaç kontrollü yinelemede, ... bir talimat grubunun kaç kez tekrarlanması gerektiğini saymak için kullanılır.

b)..... ifadesi, bir yineleme ifadesinde yürütüldüğünde, döngünün bir sonraki yinelemesinin hemen gerçekleştirilmesine neden olur.

c) ..... ifadesi, bir yineleme ifadesinde veya bir anahtarda yürütüldüğünde, ifadeden hemen çıkılmasına neden olur.

d) ..... belirli bir değişkeni veya ifadeyi, alabileceği sabit değerlerinin her biri için test etmek amacıyla kullanılır.

a) control değişkeni veya bir sayaç.

b) continue.

c) break.

d) switch – case seçme ifadesi.

4.2 Aşağıdakilerin doğru mu yanlış mı olduğunu belirtin. Cevap yanlışsa nedenini açıklayın.

a) switch seçimi ifadesinde varsayılan durum (default) gereklidir.

b) switch ifadesinin varsayılan durumunda (default case) “break” ifadesi gereklidir.

c) (x > y && a < b) ifadesi, x > y doğruysa veya a < b doğruysa doğrudur.

d) || ifadesini içeren bir ifade operatörü, işlenenlerinden biri veya her ikisi de doğruysa doğrudur.

a) yanlış. Varsayılan durum (default) isteğe bağlıdır. Varsayılan eyleme gerek yoksa, default terimine de gerek yoktur.

b) Yanlış. Break ifadesi switch ifadesinden çıkmak için kullanılır. Break deyimi hiçbir durumda gerekli değildir.

c) Yanlış. && operatörünü kullanırken ifadenin tamamının doğru olması için her iki ilişkisel ifadenin de doğru olması gerekir.

d) Doğru.

4.3 Aşağıdaki görevlerin her birini gerçekleştirmek için bir ifade veya bir dizi ifade yazın:

a) for ifadesini kullanarak 1 ile 99 arasındaki tek tam sayıları toplayın. Tamsayı değişkenleri sum ve count'u kullanın.

b) 333.546372 değerini 15 karakterlik bir alan genişliğine 1, 2, 3, 4 ve 5 hassasiyetiyle yazdırın. Çıktıyı sola hizalayın. Yazdırılan beş değer nedir?

c) pow fonksiyonunu kullanarak 2,5'in 3 üssü değerini hesaplayın. Sonucu 10 konumlu alan genişliğinde 2 hassasiyetle yazdırın. Yazdırılan değer nedir?

d) Bir while döngüsü ve sayaç değişkeni x'i kullanarak 1'den 20'ye kadar olan tam sayıları yazdırın. Satır başına yalnızca beş tam sayı yazdırın. [İpucu: x % 5 hesaplamasını kullanın. Bu 0 olduğunda yeni satır karakteri yazdırın, aksi halde sekme karakteri yazdırın.]

e) For ifadesini kullanarak Alıştırma 4.3(d)'yi tekrarlayın.

a)

```
int sum = 0;
int count = 1;
for (count = 1; count <= 99; count += 2) {
sum += count;
}
```

b)

```
printf("%-15.1f\n", 333.546372); // prints 333.5
printf("%-15.2f\n", 333.546372); // prints 333.55
printf("%-15.3f\n", 333.546372); // prints 333.546
printf("%-15.4f\n", 333.546372); // prints 333.5464
printf("%-15.5f\n", 333.546372); // prints 333.54637
```

c) `printf("%10.2f\n", pow(2.5, 3)); // prints 15.63`

d)

```
int x = 1;
while (x <= 20) {
   printf("%d", x);
      if (x % 5 == 0) {
         puts("");
      }
      else {
         printf("%s", "\t");
      }
++x;
}
```

Veya

```
int x = 1;
while (x <= 20) {
   if (x % 5 == 0) {
      printf("%d\n", x++);
   }
   else {
       printf("%d\t", x++);
   }
}
```

Veya

```
int x = 0;
while (++x <= 20) {
   if (x % 5 == 0) {
      printf("%d\n", x);
   }
   else {
      printf("%d\t", x);
   }
}
```

e)

```
int x = 1;
for (x = 1; x <= 20; ++x) {
   printf("%d", x);
   if (x % 5 == 0) {
      puts("");
   }
   else {
      printf("%s", "\t");
   }
}
```

Veya

```
int x = 1;
for (x = 1; x <= 20; ++x) {
   if (x % 5 == 0) {
      printf("%d\n", x);
   }
   else {
      printf("%d\t", x);
   }
}
```

4.4 Aşağıdaki kod bölümlerinin her birindeki hatayı bulun ve nasıl düzeltileceğini açıklayın:

a)
```
x = 1;
while (x <= 10);
   ++x;
}
```

b)
```
for (double y = .1; y != 1.0; y += .1) {
   printf("%f\n", y);
}
```

c)
```
switch (n) {
   case 1:
      puts("The number is 1");
   case 2:
      puts("The number is 2");
      break;
   default:
      puts("The number is not 1 or 2");
      break;
}
```

d) Aşağıdaki kod 1'den 10'a kadar olan değerleri yazdırmalıdır.

```
n = 1;
while (n < 10) {
   printf("%d ", n++);
}
```

a) Hata: while başlığından sonra gelen noktalı virgül sonsuz döngüye neden olur.

Düzeltme: Noktalı virgülü { ile değiştirin veya ; ve } operatörlerinin her ikisini de kaldırın.

b) Hata: For iterasyon ifadesini kontrol etmek için kayan noktalı sayı kullanılmaz. Ayrıca sayı tanımlayıcı bazı C versiyonlarında for ifadesinin dışına alınmalı.

Düzeltme: Bir tamsayı kullanın ve istediğiniz değerleri elde etmek için uygun hesaplamayı yapın. Mesela duruma göre sonucu 10'a yada 100'e bölün. Örneğin;

```
int y = 1;

for (y = 1; y != 10; ++y) {
   printf("%f\n", (float) y / 10);
}
```

c) Hata: İlk durum için (case 1) ifadelerde break ifadesi eksik.

Düzeltme: İlk durum için (case 1) ifadelerin sonuna bir break ifadesi ekleyin. Case 2: ifadesinin, case 1: ifadesi her çalıştırıldığında yürütülmesini istiyorsanız, bu durumda bu bir hata değildir. Case 1 ifadesinde break eklemeye gerek kalmaz.

d) Hata: While yineleme-devam koşulunda kullanılan uygun olmayan ilişkisel operatör.

Düzeltme: < yerine <= kullanın.