Bilgisayar Programlama

Temel Kavramlar

04/03/2021

Ders İçeriği

- Temel Kavramlar
- Donanım ve Yazılım. Bilgisayar Organizasyonu. Makine dili. Assembly dili.
- Yüksek seviyeli diller. C program geliştirme ortamı.
- Algoritma tasarımına giriş.
- Akış diyagramları.
- Basit bir C programının yapısı.
- Karar yapıları: if, if-else, switch, koşullu operator.
- Döngüler: for, do-while, while. break ve continue deyimleri.
- Modüler C programlama. Fonksiyon tanımları. Fonksiyon prototipleri. Başlık dosyaları.
- Fonksiyonların değer ve referansla çağrılması. Rastgele sayı üretimi. Kapsama kuralları.
- Dizilere giriş. Dizi tanımlamaları. Dizilerin fonksiyonlara geçilmesi.
- Dizilerin sıralanması. Diziler içerisinde arama yapılması.
- Göstericilerin tanımlanması ve başlatılması. Gösterici operatörleri.
- const deyimi. sizeof operatörü. Gösterici aritmetiği. Gösterici dizileri.
- Dosya Giriş/Çıkış

Programlama Nedir?

- Program : Günlük hayattaki bir sorunu bilgisayarla çözmek, rutin işlemleri kolaylaştırmak için bilgisayarların isteğe uygun olarak özel bir takım işlemleri gerçekleştirmesi için programlanması gerekmektedir. İşte yazılan bu yazılımlar ile ortaya çıkan ürün bir programdır.
- Bilgisayar oyunu, muhasebe işlem programları ve ticari şirketlerde kullanılan paket programlar, bir şirkette kullanılan stok uygulaması, eğitim kurumlarının kullandığı öğrenci otomasyonları birer programdır.

Programlama Dili ve Çeşitleri

- Bilgisayarda çözülecek bir sorun için çözümün bilgisayara adım adım yazılmasını sağlayan biçimsel kuralları olan ve bu kurallara sıkı sıkıya bağımlılığı gerektiren bir tanımlar kümesidir.
- Yani, programcı ile bilgisayar arasında bir tercüman durumundadır.
- Düşük Seviyeli Diller
  - Makine Dilleri
  - Assembly
- Orta Seviyeli Diller
- Yüksek Seviyeli Diller
- Çok Yüksek Seviyeli Diller
- Yapay Zekaya Yönelik Diller

Yazılan kodları, makine dilinde değilse, makine diline çevirip koşmaya hazır hale getirmek, o dilin derleyicisinin veya yorumlayıcısının görevidir.

Düşük Seviyeli Diller

- İlk programlar makine dili ile hazırlanıyordu.
- Makine dili bir programlama dilidir ancak makine dili ile program yazmak çok zahmetli bir iştir. Çünkü makine dilinde sıfırlar (0) ve birler (1) vardır.
- Yani işlemlerde DOĞRU (1) yada YANLIŞ (0) durumlarına göre hareket edilmektedir.
- Ayrıca, makine dili programları anlaşılması zor olan ve tamamıyla donanıma hitap eden programlardır.

```
1011101100010001 0000000110111001 0000110100000000 1011010000001110
1000101000000111 0100001111001101 0001000011100010 1111100111001101
0010000001001000 0110010101101100 0110110001101111 0010110000100000
0101011101101111 0111001001101100 0110010000100001
```

Hello world

Düşük Seviyeli Diller

- Makine diline yakın Assembly programlama dilinde mikroişlemcilerin anlayacağı assembly kodları kullanılır.
- Bu ham şeklinde bulunan komutlar (MOV, ADD, PUSH gibi) mikroişlemcinin belli bir işlevi yerine getirmesini sağlamaktadır.
- Bu komutlara mnemonic adı verilir.

```
format mz

org 100h

mov ah,09

mov dx,yazi

int 21h

mov ah,00

int 16h

int 20h

yazi db "merhaba dunya$"
```

merhaba dunya

Orta Seviyeli Diller

- Orta seviye ve üstündeki dillerde yazılan programlar donanımdan bağımsız çalışırlar.
- Yani yazılan programlar farklı makinelerde de kullanılabilirler.
- Oldukça esnek olan bu diller hem üst hem alt seviye programlama yapabilirler.
- Yüksek seviye dillerine göre kullanımı daha zordur fakat programcıya daha özgür bir program geliştirme imkanı sunarlar.
- Örneğin: C ,C++, C# , Java ,ADA...

```c
#include <stdio.h>

int Yaz()
{

  printf("merhaba, dünya");

  return 0;

}
```

Yüksek Seviyeli Diller

- Yüksek seviyeli dilde, Assembly dili veya makine dilindeki birçok satır tek bir komutla gösterilir.
- Böylece program daha kısa bir sürede yazılır.
- Bu dillerin komutları İngilizce bir kelimenin tamamından alındığından öğrenmek kolaydır.
- Örneğin: Pascal, Basic, Fortran...

```
program merhaba(output);
begin
  WriteLn('merhaba, dünya');
end.
```

Çok Yüksek Seviyeli Diller

- Bu programlama dilleri programlama hâkimiyetini azaltırlar, bunun yanında en hızlı ve en etkili programlama dilleri bu kategoridedir.
- Çok yüksek seviye programlama dillerini kullanmak ve öğrenmek kolaydır fakat bu tip diller ile yapabilecekleriniz kısıtlıdır.
- Örneğin: VisualBasic, VB.NET, Acces , Foxpro

```
Private Sub Button1_Clicked() Handles Button1.Click

    MsgBox("Merhaba, Dünya!")

End Sub
```

Yapay Zekaya Yönelik Diller

- Beşinci nesil programlama dilleridir.
- Yapay zeka programlama dillerinin çalışma mantığı diğer dillere göre farklıdır.
- Diğer dillerde bir problemin çözülmesi için gerekenler adım adım yazılır ve programa nasıl yapacağı öğretilirdi. Kısacası bir algoritma oluşturulur ve program buna göre yazılırdı.
- Yapay zekada ise koşulları ve kısıtlamaları programa verdiğinizde, çözümü programın kendisi bulur.
- Açıkça kodlamanın yerine bildirimsel yöntemle çalışır.
- Örneğin: Prolog, OPS5 ve Mercury

Program Geliştirme Süreci

İyi bir programın nitelikleri

- Estetik olarak görselliği ön plana çıkmalıdır.
- Kullanıcı açısından kullanımı kolay olmalıdır.
- İşlem ve hesaplamaları doğru yapmalıdır.
- Hızlı çalışmalıdır.
- Kolayca değiştirilebilmeli ve güncellenebilmelidir.
- Fazla kod yazılmadan etkin bir kodlamaya sahip olmalıdır.
- Yaygın kullanılan işletim sistemlerinde çalışabilmelidir.
- Büyük programlar için çoklu kullanıcı desteği olmalıdır.
- Ticari yazılan programlar ise iyi belgelenerek, lisanslı satılmalıdır.

Program Tasarımı

- Bir yazılım geliştirirken takip edilmesi gereken adımlar şunlardır:

1. Gereksinimlerin belirlenmesi: Problemin tanımı verilir.
2. Analiz: Problemin çözümü için gerekli tüm girdi ve çıktılar analiz edilmelidir.
3. Dizayn: Problemin çözümünde kullanılacak uygun algoritmanın adım adım tanımlanması yapılmalıdır.
4. Akış Diyagramı: Algoritmaya göre uygun akış diyagramı çizilmelidir.
5. Kod Yazımı: Algoritmanın herhangi bir programlama dilinde yazılarak kaynak dosyanın hazırlanması gerekir.

Program Tasarımı

6. Test: Bu basamakta ise yazılan programın bölümleri ve tamamı çalışır halde test edilir.
7. Doğrulama: Programın örnek girdilerle doğru çıktı ürettiği gözlenmelidir.
8. Bakım: Yazılan programda bulunan hatalar ayıklanır veya gerekli güncellemeler yapılır.
9. Belgeleme: Yazılan program için belgeleme yapılarak, toplu çoğaltmalara karşı engelleme konulur.

Algoritmalar ve Akış Diyagramları

- Bilgisayara verilecek iki sayıyı toplayıp sonucu ekrana yazacak bir program için algoritma geliştirmek isteyelim.
- Sorun hakkında anlaşılamayan tüm belirsiz noktalar açıklığa kavuşturulmalıdır.
- Örneğin sayılar bilgisayara nereden verilecek, Klavye, Dosya vb.
- Bu ve buna benzer soru ve tereddütleriniz varsa sorunun sahibine bunları sormalı ve sistem analizi yapmalısınız.
- Sonra bulacağımız çözümü algoritma haline dönüştürebiliriz

1. BAŞLA
2. A sayısını oku
3. B sayısını oku
4. TOPLAM=A + B işlemini yap
5. TOPLAM değerini ekrana yaz
6. SON

Algoritmalar ve akış Diyagramları

- Bir başka örnek; Klavyeden girilecek iki sayıdan büyük olanından küçük olanını çıkarıp sonucu ekrana yazacak program için bir algoritma geliştirelim.

1. BAŞLA
2. A sayısını oku
3. B sayısını oku
4. Eğer A büyüktür B ise SONUC=A-B değilse SONUC=B-A
5. SONUC değerini ekrana yaz
6. SON

- Algoritmalar doğal dille yazılabileceği için fazlaca biçimsel değildir.
- Algoritmalar belli bir kurallar bütününü ifade ettiği için bir algoritmada aşağıdaki ifadelerin mutlaka doğrulanması gereklidir;
- ¬ Netlik ¬ Etkinlik ¬ Sonluluk ¬ Giriş/Çıkış Bilgileri

Algoritmalar

- Netlik: Algoritmada bulunan anlatım satırları kesin olmalıdır. Kesin olmayan anlatımlar algoritmada bulunmamalıdır. Başka bir deyişle her işlem (komut) açık olmalı ve farklı anlamlar içermemelidir.

  z←x + y

  sayı←sayı + 1

- Etkinlik: Algoritmada, her komut, bir kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır.
  - Algoritmada tekrar anlatımlar olmamalıdır.
  - Bir algoritma bünyesinde ne kadar az tekrar varsa algoritmanın etkinliği o kadar artar.
  - Kaçınılmaz tekrarlarda ise bir algoritmayı etkin hale getirebilmek için; tekrar anlatımların alt algoritma yapılması gerekmektedir.

Algoritmalar

- Sonluluk: Her türlü olasılık için algoritma sonlu adımda bitmelidir.
  - Her algoritmanın bir bitiş ya da geriye dönüş noktası olmalıdır.
  - Ana algoritmada bitiş noktası END, alt algoritmalarda ise geriye dönüş noktası RETURN komutları ile sağlanır.
  - İşletim sistemleri gibi bazı programlar istisnai olarak sonsuza dek çalışırlar.

Algoritmalar

- Giriş/Çıkış Bilgisi: Bir algoritmada mutlaka Giriş ve Çıkış bilgisi olmalıdır.
  - Giriş bilgisi, algoritmaya dışarıdan bilgi aktarımını, Çıkış bilgisi ise, algoritma içinde oluşan sonuçların algoritma dışına çıkartılabilmesi işlemidir.
  - Genelde Giriş ve Çıkışı işlemleri için Read ve Write (veya Print) kullanılır.
  - Bir bilginin okunabilmesi için değişken kullanılır.
  - Read Değişken Değişken ile belirtilene dışardan değer oku.
  - Write Değişken Değişken ile belirtilendeki değeri dışarıya yaz.

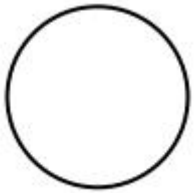
Akış Diyagramları

- Akış Diyagramı; bir algoritmanın belirli bir anlamı olan şekillerle ifade edilmesidir.
- Önceki konuda eğer dikkat edildiyse algoritmaların, doğal dille yazıldığı için herkes tarafından anlaşılamayabilir ya da istenmese de başka anlamlar çıkarılabilir oluşudur.
- Ancak akış diyagramlarında her bir şekil standart belli bir anlam taşıdığı için farklı yorumlanıp anlaşılması olası değildir.
- Bir algoritmanın ifade edilebilmesi için sıklıkla kullanılan şekiller ve anlamları şunlardır:

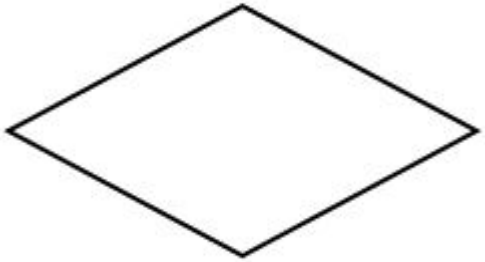
Akış Diyagramları

Bir algoritmanın başladığı veya bittiği konumu gösterir.	
Bir algoritmada aritmetik işlem yapılmasını sağlayan şekildir. Bu dörtgen kutu içerisine yapılmak istenen işlem yazılır.	
Algoritmada bir bilginin ekrana yazılacağı konumu gösteren şekildir. Ekrana yazılacak ifade ya da değişken bu şekil içerisine yazılır.	
Bir algoritmada başka bir yerde tanımlanmış blokun yerleştiği konumu gösteren şekildir. Kutu içerisine blokun adı yazılabilir.	

Akış Diyagramları

Klavyeden Bilgisayara bilgi girilecek konumu belirten şekildir. Girilecek bilginin hangi değişkene okunacağını kutu içerisine yazabilirsiniz.	
Giriş - Çıkış komutunun kullanılacağı yeri belirler. Kutu içerisine hangi değişken veya değişkenlere okuma mı? yoksa yazma mı? yapılacağını belirtmeniz gerekir.	
Bilginin Yazıcıya yazılacağı konumu gösteren şekildir.	
Bir algoritmanın birden fazla alana yayılması durumunda bağlantı noktalarını gösteren şekildir. Tek girişli veya tek çıkışlı olarak kullanılırlar.	

Akış Diyagramları

• Bir işlemin belli bir sayıda veya belli bir koşul doğru olduğu sürece tekrar edilmesini sağlayan döngü komutunu gösteren şekildir. • Bu döngüde altıgen içerisine ya koşul yada döngünün başlangıç, adım ve sonlanma değerlerini belirtebilirsiniz. • DÖNGÜ olarak belirlenen blokta da tekrar edilmek istenen komutlar yer almaktadır.	
• Bir algoritmada bir kararın verilmesini ve bu karara göre iki seçenekten birinin uygulanmasını sağlayan şekildir. • Burada eşkenar dörtgen içerisine kontrol edilecek mantıksal koşul yazılır. • Program akışı sırasında koşulun doğru olması durumunda "Evet“ yazılan kısma Yanlış olması durumunda "Hayır“ yazılan kısma sapılır. Tek girişli ve çift çıkışlı bir şekildir.	

Örnek 1: Akış Diyagramı

- Dışarıdan girilen iki sayıyı yer değiştirip çıktı olarak veren algoritmanın akış diyagramını yapalım.

Örnek 2: Akış Diyagramı

- Dışardan girilecek bir N sayısı için 1 den N'ye kadar olan sayıların toplamını alıp çıktı olarak veren algoritmayı akış diyagramı olarak ifade edelim.

Örnek 3: Akış Diyagramı

- Dışardan girilen iki sayıyı büyükten küçüğe doğru sıralayan programın akış diyagramını çizelim.

Örnek 4: Akış diyagramı

- Dışardan girilen bir sayının N’ninci kuvvetini alan bir akış diyagramını çizelim.

C Programlama Dili

- Verilen C programı derlendikten sonra, ekrana 'Merhaba Dünya!' yazısını basan yalın bir C programıdır.
- Satır başlarına yerleştirilen 1:, 2: 3: ... rakamlarının yazılmasına gerek yoktur.
- Bu rakamlar sadece daha sonra program ile ilgili açıklama yapılırken, ilgili satırda bulunan kodlar izah edilirken kullanılacaktır.
- Bu programın bilgisayarda ilk.c adı ile kaydedilmiştir.

```
01:  /* ilk.c: ilk C programi */
02:  #include <stdio.h>
03:
04:  main()
05:  {
06:     printf("Merhaba Dünya!\n");
07:  }
```

C Programlama Dili

- /* ... */
  - Programda, 1. satırda /* ... */ sembolleri görülmektedir.
  - Bu ifadeler arasında yazılan herhangi bir metin, işlem vb. satırlar, derleyici tarafından işlenmez (değerlendirilmez).
  - Yani /* */ ifadeleri açıklama operatörüdür.
- #include <stdio.h>
  - 2. satırdaki #include deyimi, programda eklenecek olan başlık dosyanını işaret eder. Bu örnekte verilen başlık dosyası (header file) stdio.h dir. #include <stdio.h> ifadesi stdio.h dosyasının derleme işlemine dahil edileceğini anlatır.

```
01: /* ilk.c: ilk C programi */
02: #include <stdio.h>
03:
04: main()
05: {
06:     printf("Merhaba Dünya!\n");
07: }
```

C Programlama Dili

- main()
  - 4. satırdaki main() özel bir fonksiyondur.
  - Ana program bu dosyada saklanıyor anlamındadır.
  - Programın yürütülmesine bu fonksiyondan başlanır.
  - Dolayısıyla her C programında bir tane main() adlı fonksiyon olmalıdır.
- printf()
  - 6. satırdaki printf() standart kütüphane bulunan ekrana formatlı bilgi yazdırma fonksiyondur.
  - stdio.h dosyası bu fonksiyonu kullanmak için program başına ilave edilmiştir.

```
01:  /* ilk.c: ilk C programi */
02:  #include <stdio.h>
03:
04:  main()
05:  {
06:     printf("Merhaba Dünya!\n");
07:  }
```

Printf() Fonksiyonunun Kullanımı

Örnek kullanım şekli	Ekranda yazılacak ifade
printf("Element: Aluminyum");	Element: Aluminyum
printf("Atom numarası = %d",13);	Atom numarası = 13
printf("Yoğunluk = %f g/cm3",2.7);	Yoğunluk = 2.7 g/cm3
printf("Erime noktası = %f derece",660.32);	Erime noktası = 660.32 derece

C Kodlarının Temel Özellikleri

- Bir C programı aşağıda verilen özellikleri mutlaka taşımalıdır.
  - Yazılımda kullanılacak olan her fonksiyon için ilgili başlık dosyası programın başına ilave edilmelidir.
  - Her C programı main() fonksiyonunu içermelidir.
  - Program içinde kullanılacak olan değişkenler ve sabitler mutlaka tanımlanmalıdır.

C Kodlarının Temel Özellikleri

- Bir C programı aşağıda verilen özellikleri mutlaka taşımalıdır.
  - Satırın sonuna ; işareti konmalıdır.
  - Her bloğun ve fonksiyonun başlangıcı ve bitişi sırasıyla { ve } sembolleridir.
  - C dilinde yazılan kodlarda küçük-büyük harf ayrımı vardır (case sensitive).
  - Örneğin A ile a derleyici tarafından farklı değerlendirilir.
  - Açıklama operatörü /* */ sembolleridir.

Kod Yazımı için Bazı Tavsiyeler

- Program açıklamalarını ve döküman hazırlama işini program yazıldıkça yapın! Bu unutulmaması gereken çok önemli hususdur.
- Değişken, sabit ve fonksiyon adları anlamlı kelimelerden seçilip yeterince uzun olmalıdır. Eğer bu isimler bir kaç kelimeden oluşacak ise, kelimeler alt çizgi (_) ile ayrılmalıdır veya her kelime büyük harfle başlamalıdır. Örneğin:

```
int son_alinan_bit;
void KesmeSayisi();
float OrtalamaDeger = 12.7786;
```

- Sabitlerin bütün harflerini büyük harfle yazın. Örneğin

```
#define PI 3.14;
const int STATUS = 0x0379;
```

Kod Yazımı için Bazı Tavsiyeler

- Her alt yapıya girerken birkaç boşluk veya TAB tuşunu kullanın.

  Bu okunabilirliği arttıracaktır. Örneğin:

```
k = 0;
for(i=0; i<10; i++)
{
      for(j=0; j<i; j+=2)
      {
            do{
                  if( j>1 )  k = i+j;

                  x[k] = 1.0/k;
              }while(k!=0);
      }
}
```

Kod Yazımı için Bazı Tavsiyeler

- Aritmetik operatörler ve atama operatörlerinden önce ve sonra boşluk karakteri kullanın. Bu, yazılan matematiksel ifadelerin daha iyi anlaşılmasını sağlayacaktır. Örneğin:

```
h_max = pow(Vo,2) / (2*g);
Tf    = 2*Vo/g;
Vy    = Vo - g*t;
y     = Vo*t - (g*t*t)/2.0;
z     = ( a*cos(x) + b*sin(x) )*log( fabs(y) );
```

- Program bittikten sonra tekrar tekrar programınızı inceleyerek, programınızı daha iyi şekilde yazma yollarını arayın ve aynı fonksiyonları daha kısa algoritmalarla ve/veya daha modüler şekilde elde etmeye çalışın.