ENM107 Bilgisayar Programlama - Ara Sınav Hazırlık Notları

Bu belge, ENM107 Bilgisayar Programlama dersinin ara sınavına kadar olan konuları özetlemektedir. Notlar, sağlanan ders materyallerine (PDF belgeleri) dayanarak hazırlanmıştır.

Bölüm 1: Bilgisayara Giriş ve Temel Kavramlar

(Kaynak: 959410735_enm107_bilgisayar_programlama-i.pdf)

Sınavdan önce bir bilgisayarın temel bileşenlerini bilmek önemlidir.

- **Donanım (Hardware):** Bilgisayarın fiziksel, dokunulabilir parçalarıdır (Örn: Anakart, işlemci, RAM, ekran kartı).
- **Yazılım (Software):** Bilgisayarın fiziksel olmayan, donanımın çalışmasını sağlayan komutlar bütünüdür.
  - **İşletim Sistemi (Operating System):** Donanım ve kullanıcı arasındaki iletişimi sağlayan temel yazılımdır (Örn: Windows 10, Linux, MacOS).
  - **Uygulama Yazılımları:** Belirli bir işi yapmak için kullanılan programlardır (Örn: Dev-C++, Microsoft Word).
- **Temel Windows İşlemleri:**
  - **Dosya/Klasör:** Bilgilerin saklandığı birimler.
  - **İşlemler:** Klasör oluşturma, ad değiştirme (F2 tuşu), kopyalama (Ctrl+C), kesme (Ctrl+X), yapıştırma (Ctrl+V), silme (Delete) ve özellikleri (Sağ tık -> Özellikler) yönetebilme.

Bölüm 2: Algoritma ve Akış Diyagramları

(Kaynak: algoritmalar.pdf)

Programlamanın temeli, bir problemi çözme adımlarını tasarlamaktır.

Algoritma

Herhangi bir sorunun veya problemin çözümü için izlenecek yolun adımlar halinde (sözlü veya yazılı) ifade edilmesidir.

- **Temel Özellikleri:**
  - `Başla` komutu ile başlar ve `Dur` (veya Bitti) komutu ile biter.
  - Adımlar net, sıralı ve anlaşılır olmalıdır.
  - Her kararın net bir sonucu olmalıdır.
- **Algoritma Örneği (İki Sayının Toplamı):**

```
Adım 1: Başla
Adım 2: Birinci sayıyı oku (sayi1)
Adım 3: İkinci sayıyı oku (sayi2)
Adım 4: İki sayıyı topla (toplam = sayi1 + sayi2)
Adım 5: Sonucu ekrana yaz (Yaz topla)
Adım 6: Dur
```

Akış Diyagramları (Flow Charts)

Algoritmaların, genel kabul görmüş semboller (şekiller) kullanılarak görsel olarak ifade edilmesidir.

- **Temel Semboller:**
  - **Elips (Başla / Bitti):** Algoritmanın başlangıç ve bitiş noktasını gösterir.
  - **Paralelkenar (Veri Giriş/Çıkış):** Dışarıdan veri okuma (`cin` veya `Oku`) veya ekrana veri yazma (`cout` veya `Yaz`) işlemleridir.
  - **Dikdörtgen (İşlem):** Matematiksel veya mantıksal hesaplamaların yapıldığı yerdir (Örn: `toplam = sayi1 + sayi2`).
  - **Baklava (Karar):** `if` gibi koşullu ifadelerin (Evet/Hayır veya Doğru/Yanlış) kontrol edildiği yerdir (Örn: `sayi > 0`).
  - **Altıgen (Döngü):** `for` veya `while` gibi tekrarlı işlemlerin başlangıcını ve koşulunu gösterir.

Bölüm 3: Programlama Ortamı

(Kaynak: c uygulamasının indirilmesi.pdf)

Ders uygulamalarında **Dev-C++** Tümleşik Geliştirme Ortamı (IDE) kullanılmaktadır. Bu ortam, kodlarımızı yazmak, derlemek (makine diline çevirmek) ve çalıştırmak için gerekli araçları sağlar.

Bölüm 4: C++ Programlamanın Temelleri

(Kaynak: uygulamalar.pdf)

- **Temel Program Yapısı:**

```
// Gerekli kütüphaneleri dahil et
#include <iostream> // Giriş/Çıkış işlemleri için

// Standart isim alanını kullan
using namespace std;

// Ana fonksiyon (Programın başladığı yer)
int main() {

    // Kodlar buraya yazılır
    cout << "Merhaba Dünya" << endl; // Ekrana yazı yazma

    // Programın başarılı bittiğini belirt
    return 0;
}
```

- `#include <iostream>` : `cin` (giriş) ve `cout` (çıkış) komutlarını kullanabilmek için eklenmesi zorunlu kütüphanedir.
- `cout` **(Çıkış):** Ekrana veri yazdırmak için kullanılır. `<<` operatörü ile kullanılır.
  - `endl` veya `\n` : Bir alt satıra geçmeyi sağlar.
  - `\t` : Bir "tab" boşluğu bırakır.
- `cin` **(Giriş):** Kullanıcıdan veri almak için kullanılır. `>>` operatörü ile kullanılır.
- **Yorum Satırları:**
  - `//` : Tek satırlık yorum.

    - `/* ... */` : Çok satırlık yorum.
- `setlocale(LC_ALL, "Turkish");` : Programda Türkçe karakterlerin (ç, ğ, ı, ö, ş, ü) konsolda doğru görüntülenmesi için `main` fonksiyonunun başında kullanılır.

Bölüm 5: Veri Tipleri ve Değişkenler

(Kaynak: veri tipleri.pdf, uygulamalar.pdf)

- **Değişken (Variable):** Program çalışırken verileri geçici olarak hafızada tutmak için kullanılan isimlendirilmiş alanlardır.
- **Temel Veri Tipleri:**
    - `int` : Tamsayılar (Örn: 5, -10, 0). (4 byte)
    - `double` / `float` : Ondalıklı sayılar (Örn: 3.1416, 5.0). (8 byte / 4 byte)
    - `char` : Tek karakterler (Örn: 'A', 'z', '1'). (1 byte)
    - `bool` : Mantıksal değerler (`true` veya `false`).
- **Değişken Tanımlama:** `veri_tipi degisken_adi;` (Örn: `int tsayi;`)
- **Değer Atama:** `tsayi = 4;`
- **İlk Değer Atama (Inline):** `int tsayi2 = 6;`

Bölüm 6: Operatörler

(Kaynak: uygulamalar.pdf)

- **Aritmetik Operatörler:**
    - `+` (Toplama), `-` (Çıkarma), `*` (Çarpma), `/` (Bölme)
- **Modulus (Kalan) Operatörü:** `%`
    - Bir sayının bir sayıya bölümünden kalanı verir.
    - Örn: `10 % 3` işleminin sonucu `1` 'dir.
    - Bir sayının çift mi tek mi olduğunu anlamak için kullanılır: `sayi % 2 == 0` (Sayı çiftse sonuç 0'dır).
- **Atama Operatörleri:**
    - `=` : Sağdaki değeri soldaki değişkene atar.
    - `+=` , `-=` , `*=` , `/=` : İşlemi yapıp değişkene atar (Örn: `toplam += i;` ifadesi `toplam = toplam + i;` ile aynıdır).
- **Artırma/Azaltma Operatörleri:**
    - `++` : Değişkenin değerini bir artırır (Örn: `i++`).
    - `--` : Değişkenin değerini bir azaltır (Örn: `n--`).

Bölüm 7: Karar Yapıları (Kontrol İfadeleri)

(Kaynak: uygulamalar.pdf, 599433928_enm107_bilgisayar_programlama_uygulamalari_2022.pdf)

Programın akışını belirli koşullara göre yönlendiren yapılardır.

- `if` : Sadece koşul doğruysa çalışır.

- `if-else` : Koşul doğruysa `if` bloğu, yanlışsa `else` bloğu çalışır.
- `if-else if-else` : Birden fazla koşulun sıralı olarak kontrol edilmesi gerektiğinde kullanılır.
- **Mantıksal Operatörler:** `if` içinde birden fazla koşulu birleştirmek için kullanılır:
  - `and` (veya `&&`): Her iki koşul da doğru olmalı.
  - `or` (veya `||`): Koşullardan en az biri doğru olmalı.
- **Uygulama Örneği (Girilen Sayı Pozitif/Negatif/Sıfır mı):**

```
#include <iostream>
#include <cstdlib> // setlocale için gerekli olmayabilir ama bazı IDE'lerde gerekebil

using namespace std;

int main(int argc, char** argv) {
    setlocale (LC_ALL, "Turkish");
    int x;
    cout << "Bir sayı giriniz: ";
    cin >> x;

    if (x > 0) {
        cout << "Girilen Sayı pozitiftir." << endl;
    }
    else if (x < 0) {
        cout << "Girilen Sayı negatiftir." << endl;
    }
    else {
        cout << "Girilen Sayı Sıfırdır." << endl;
    }

    return 0;
}
```

- `switch-case` : Bir değişkenin alabileceği birden fazla belirli değere göre farklı işlemler yapmak için kullanılır (Genellikle `if-else if` yapısına alternatiftir). `break` komutu bir `case` bittiğinde yapıdan çıkmak için kullanılır.

Bölüm 8: Döngüler

(Kaynak: uygulamalar.pdf, 464444589_enm107_bilgisayar_programlama_uygulamalari_20222.pdf)

Bir kod bloğunu belirli bir koşul sağlandığı sürece veya belirli bir sayıda tekrar etmek için kullanılır.

- `for` **Döngüsü:** Genellikle tekrar sayısı bilindiğinde kullanılır.
  - `for (başlangıç_değeri; koşul; artış_miktarı)`
- **Uygulama Örneği (** `for` **ile 1'den N'e Kadar Olan Sayıların Toplamı):**

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
    setlocale(LC_ALL,"Turkish"); // Türkçe karakter desteği

    int toplam = 0;
    int n;
```

```
    cout << "Bir sayı giriniz (N): " << endl;
    cin >> n;

    // i, 1'den başlar, n'e eşit olana kadar (n dahil) çalışır
    // ve her adımda i bir artar.
    for (int i = 1; i <= n; i++) {
        toplam += i; // toplam = toplam + i;
    }

    cout << "1'den " << n << "'e kadar sayıların toplamı: " << toplam << endl;

    return 0;
}
```

- `while` **Döngüsü:** Koşul `true` (doğru) olduğu sürece çalışmaya devam eder. Koşul, döngüye girmeden önce kontrol edilir.
- **Uygulama Örneği (`while` ile Faktöryel Hesabı):**

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
    setlocale(LC_ALL, "Turkish");

    int faktoryel = 1, n;

    cout << "Faktöryeli alınacak sayıyı girin: " << endl;
    cin >> n;

    if (n > 0) {
        int temp_n = n; // Orijinal n değerini korumak için geçici değişken
        while (temp_n > 1) {
            faktoryel *= temp_n; // faktoryel = faktoryel * temp_n;
            temp_n--;
        }
        cout << n << " sayısının faktöryeli: " << faktoryel << endl;
    }
    else {
        cout << "Girilen n değeri pozitif olmalıdır." << endl;
    }

    return 0;
}
```

- `do-while` **Döngüsü:** `while` döngüsüne benzer, ancak koşul döngü bloğu çalıştıktan *sonra* (sonda) kontrol edilir. Bu nedenle, koşul yanlış (false) olsa bile **en az bir kez** çalışır.
- `break` **:** İçinde bulunduğu döngüyü (for, while, do-while) veya `switch` yapısını anında sonlandırır.

Bölüm 9: Fonksiyonlar

(Kaynak: uygulamalar.pdf)

- **Fonksiyon:** Belirli bir işi yapmak için tasarlanmış, tekrar kullanılabilen, isimlendirilmiş kod bloklarıdır. `main` de bir fonksiyondur.

- **Neden Kullanılır?** Kod tekrarını önlemek, programı modüler (parçalara ayrılmış) hale getirmek ve okunabilirliği artırmak.
- **Fonksiyon Yapısı:** `dönüş_tipi fonksiyon_adı(parametre1_tipi, parametre2_tipi, ...)`
- `dönüş_tipi` **:** Fonksiyonun çağrıldığı yere döndüreceği verinin tipidir.
  - `int` , `double` vb. olabilir.
  - Eğer fonksiyon bir değer döndürmeyecekse `void` kullanılır.
- `return` **:** Fonksiyondan bir değer döndürmek için kullanılır.
- **Uygulama Örneği (Girilen Sayının Karesini Alan Fonksiyon):**

```
#include <iostream>

using namespace std;

// Fonksiyon Tanımı:
// Bu fonksiyon 'int' tipinde bir parametre alır (a)
// ve 'int' tipinde bir sonuç döndürür.
int kare(int a) {
    int sonuc;
    sonuc = a * a;
    return sonuc; // Hesaplanan sonucu geri döndür
}

int main(int argc, char** argv) {
    setlocale(LC_ALL, "Turkish");

    int n;
    cout << "Karesi alınacak sayıyı giriniz: " << endl;
    cin >> n;

    // Fonksiyonun Çağrılması:
    // kare(n) komutu ile fonksiyona n değeri gönderilir
    // ve fonksiyondan dönen değer ekrana yazdırılır.
    cout << n << " sayısının karesi: " << kare(n) << endl;

    return 0;
}
```

Bölüm 10: Diziler (Arrays)

(Kaynak: uygulamalar.pdf)

- **Dizi:** Aynı veri tipinden birden fazla değeri hafızada sıralı bir şekilde tutan veri yapılarıdır.
- **Tanımlama:** `veri_tipi dizi_adı[eleman_sayısı];`
  - Örn: `int sayi[5];` // 5 adet tamsayı tutabilen bir dizi.
- **İndeks (Indis):** Dizi elemanlarına erişmek için kullanılan numaradır.
  - **ÇOK ÖNEMLİ:** Dizilerde indeks **0'dan (sıfır)** başlar.
  - `sayi[5];` dizisinin elemanları: `sayi[0]` , `sayi[1]` , `sayi[2]` , `sayi[3]` , `sayi[4]`
- **Değer Atama:** `sayi[0] = 4;`
- **Döngüler ve Diziler:** Diziler, eleman sayıları belirli olduğu için genellikle `for` döngüleri ile birlikte kullanılır.

- **Uygulama Örneği (Diziye Eleman Girme ve Yazdırma):**

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
    setlocale(LC_ALL, "Turkish");

    // 5 elemanlı bir tamsayı dizisi tanımla
    int sayi[5];

    cout << "Diziye 5 adet sayı giriniz:" << endl;

    // Döngü ile diziye elemanları alma
    // İndeks 0'dan 4'e kadar (5 dahil değil) gider.
    for (int i = 0; i < 5; i++) {
        cout << "Dizinin " << i + 1 << ". elemanını girin: ";
        cin >> sayi[i]; // Örn: i=0 için sayi[0]'a değer girilir
    }

    cout << "\nDizinin elemanları yazdırılıyor:" << endl;

    // Döngü ile dizi elemanlarını ekrana yazdırma
    for (int j = 0; j < 5; j++) {
        cout << "Dizinin " << j + 1 << ". elemanı: " << sayi[j] << endl;
    }

    return 0;
}
```