ERCİYES ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ PROGRAMMING LABORATORY NOTLARI

3. HAFTA NOTU (C Programlama)

Bu not dersin 3. haftasında yapacağınız uygulamaya hazırlanmanızı sağlayacak olup temel kavramları anlatan bilgileri içermektedir.

DERSİN ÖĞRETİM ÜYESİ

Prof. Dr. Bahriye AKAY

DENEYİ YAPTIRAN ÖĞRETİM ELEMANI

Arş. Gör. Abdulbaki KARTAL

2026
KAYSERİ

1-) Deneyde Kullanılacak Kütüphaneler Ve Fonksiyonlar Hakkında Kısa Bilgi

Bu deney kapsamında gerçekleştirilecek uygulama, metin (karakter dizisi) tabanlı veri işleme üzerine kuruludur. Konsol üzerinden kullanıcıdan veri alınması, karakterlerin tek tek incelenmesi ve belirli kurallara göre dönüştürülmesi gibi işlemler yapılacaktır. Bu tür uygulamalarda, C programlama dilinin standart kütüphaneleri önemli bir rol oynamaktadır.

Kütüphaneler, belirli işlevleri hazır olarak sunan ve programın daha düzenli, güvenli ve okunabilir şekilde yazılmasını sağlayan yapılardır. Aşağıda, deney boyunca kullanılacak temel kütüphaneler ve içerdiği önemli fonksiyonlar açıklanmıştır.

1.1 stdio.h

stdio.h (Standard Input Output), C dilinin en temel ve en yaygın kullanılan kütüphanelerinden biridir. Bu kütüphane, program ile kullanıcı arasındaki etkileşimi sağlayan giriş ve çıkış işlemlerini gerçekleştirmek için kullanılır.

Konsol tabanlı uygulamalarda:

- Kullanıcıdan veri alma
- Ekrana formatlı veri yazdırma
- Metin tabanlı giriş-çıkış işlemleri

stdio.h aracılığıyla gerçekleştirilir.

Bu kütüphanede yer alan temel fonksiyonlar:

- printf() → Formatlı çıktı üretmek için kullanılır.
- scanf() → Kullanıcıdan veri almak için kullanılır.
- fgets() → Karakter dizisi okumak için tercih edilir.

Format belirteçleri sayesinde farklı veri türleri kontrollü şekilde ekrana yazdırılabilir:

- %d → Tamsayı
- %c → Karakter
- %s → Karakter dizisi

Bu yapı, programın kullanıcı ile kontrollü ve düzenli bir şekilde iletişim kurmasını sağlar.

1.2 string.h

Metin işleme gerektiren uygulamalarda karakter dizileri üzerinde çeşitli işlemler yapılması gerekir. C dilinde karakter dizileri üzerinde güvenli ve sistematik işlemler gerçekleştirebilmek için string.h kütüphanesi kullanılır.

Bu kütüphane:

- Metin uzunluğunu hesaplama
- Bir metni başka bir değişkene kopyalama
- Belirli karakterleri arama

gibi işlemleri kolaylaştırır.

Temel fonksiyonlar:

- strlen() → Karakter dizisinin uzunluğunu hesaplar.
- strcpy() → Bir karakter dizisini başka bir diziye kopyalar.
- strcspn() → Belirli karakterleri aramak için kullanılabilir.

Bu kütüphane sayesinde karakter dizileri üzerinde manuel işlem yapma gereksinimi azalır ve hata riski düşer.

1.3 ctype.h

Metin işleme uygulamalarında her karakter üzerinde kontrol yapılması gerekebilir. Bir karakterin harf mi, sayı mı veya büyük/küçük harf mi olduğunu belirlemek için ctype.h kütüphanesi kullanılır.

Bu kütüphane, karakter sınıflandırma ve dönüşüm fonksiyonlarını içerir.

Örneğin:

- isalpha() → Karakterin harf olup olmadığını kontrol eder.
- isupper() → Büyük harf kontrolü yapar.
- islower() → Küçük harf kontrolü yapar.
- tolower() → Karakteri küçük harfe dönüştürür.
- toupper() → Karakteri büyük harfe dönüştürür.

Bu fonksiyonlar, karakter dizisi üzerinde işlem yapılırken güvenli kontrol mekanizması sağlar ve kodun daha okunabilir olmasına katkıda bulunur.

2-) DENEYDE KULLANILACAK PROGRAMLAMA YAPILARI HAKKINDA KISA BİLGİ

2.1 Veri Türleri

C programlama dilinde her değişken, bellekte belirli bir alan kaplar ve bu alanın türü değişkenin veri tipine göre belirlenir. Veri türü, değişkenin hangi tür bilgiyi saklayabileceğini tanımlar.

Bu deney kapsamında kullanılan temel veri türleri şunlardır:

- int → Tamsayı değerleri temsil eder.
- char → Tek bir karakter saklar.
- char[] → Birden fazla karakteri ardışık şekilde saklayan karakter dizisini temsil eder.
- void → Bir fonksiyonun değer döndürmediğini belirtir.

Doğru veri türünün seçilmesi, belleğin kontrollü ve verimli kullanılmasını sağlar.

2.2 Sabit Tanımları (#define)

C dilinde sabit değerlerin tanımlanması için #define yönergesi kullanılır. Bu yapı, derleme aşamasında metinsel yer değiştirme yapar ve program içerisinde tekrar eden sabit değerlerin merkezi olarak tanımlanmasını sağlar.

Örneğin:

```
#define N 1024
```

Bu tanım sayesinde, dizi boyutları veya sabit sınırlar kod içerisinde tekrar tekrar yazılmadan kullanılabilir. Bu yaklaşım, kodun okunabilirliğini artırır ve bakımını kolaylaştırır.

2.3 Karakter Dizileri (String Yapısı)

C dilinde metinler, karakter dizileri (string) olarak temsil edilir. Karakter dizileri, aynı türden verilerin ardışık bellek alanlarında saklandığı yapılardır.

Örnek:

```
char metin[100];
```

Bu tanım, belirli bir uzunlukta karakter depolayabilen bir alan oluşturur. Karakter dizileri:

- Bellekte ardışık olarak tutulur.
- Sonlandırıcı karakter (\0) ile biter.
- Karakter karakter işlenebilir.

Bu özellikleri sayesinde metin tabanlı uygulamalarda temel veri yapısı olarak kullanılır.

2.4 Döngüler

Döngüler, belirli bir kod bloğunun tekrar tekrar çalıştırılmasını sağlar. Karakter dizileri üzerinde işlem yapılırken genellikle her karakterin tek tek incelenmesi gerekir. Bu işlem, döngüler yardımıyla gerçekleştirilir.

Örneğin:

```
for(int i = 0; i < uzunluk; i++)
```

Bu yapı sayesinde:

- Dizinin başından sonuna kadar ilerlenebilir.
- Her karakter üzerinde dönüşüm veya kontrol işlemi uygulanabilir.

Döngüler, sıralı veri yapıları üzerinde işlem yapmanın temel aracıdır.

2.5 Koşul Yapıları

Koşul yapıları, programın belirli şartlara göre farklı davranmasını sağlar. Metin işleme uygulamalarında, karakterin türüne göre farklı işlemler yapılabilir.

Örneğin:

```
if(isalpha(c))
```

Bu yapı ile:

- Karakterin harf olup olmadığı kontrol edilir.
- Belirli şartlara bağlı dönüşüm uygulanır.

Koşul yapıları, program akışının kontrol edilmesini ve mantıksal kararların uygulanmasını sağlar.

2.6 Fonksiyon Kullanımı

Fonksiyonlar, belirli bir görevi yerine getiren kod bloklarıdır. Programın modüler ve düzenli yazılmasını sağlar.

Örnek yapı:

```
void fonksiyonAdi(...)
{
   // işlemler
}
```

Fonksiyon kullanımı sayesinde:

- Kod tekrarının önüne geçilir.
- Program daha anlaşılır hâle gelir.
- Belirli işlemler ayrı bir yapı altında toplanır.

Ders Katkıları

Bu uygulama kapsamında kullanılan yapılar ve gerçekleştirilen işlemler, öğrencinin aşağıdaki teknik ve kavramsal kazanımları edinmesini sağlar:

1. Algoritmik Düşünme

- Metin üzerinde adım adım işlem yapma mantığını kavrama
- Karakter bazlı dönüşüm algoritması oluşturma
- İşlem sırasını planlama ve mantıksal akışı yapılandırma
- Bir problemi küçük ve yönetilebilir parçalara ayırabilme

2. Metin ve Karakter İşleme Yetkinliği

- Karakter dizilerinin bellekte nasıl tutulduğunu anlama
- Diziler üzerinde karakter karakter işlem yapabilme
- Büyük/küçük harf kontrolü gerçekleştirme
- Harf olmayan karakterlerin kontrolünü sağlama

3. Veri Yapıları ve Bellek Bilinci

- Karakter dizilerinin sonlandırıcı karakter ile bittiğini kavrama
- Sabit uzunluklu dizilerin sınırlarını dikkate alarak program yazma
- Bellekte ardışık veri yapılarının nasıl işlendiğini gözlemleme

4. Kontrol Mekanizmalarının Etkin Kullanımı

- Döngü yapıları ile tekrar eden işlemleri modelleme
- Koşul ifadeleri ile karar mekanizması oluşturma
- İç içe mantıksal kontrolleri yapılandırma

5. Modüler Programlama Bilinci

- Fonksiyonlar aracılığıyla kodu bölümlere ayırma
- Tekrar kullanılabilir yapı geliştirme
- Okunabilir ve sürdürülebilir kod yazma

6. Girdi-Çıktı Kontrolü

- Kullanıcıdan veri alma yöntemlerini öğrenme
- Formatlı çıktı üretme
- Programın kullanıcı ile etkileşimini yönetme

Örnek Uygulama;

3. hafta derste yapılacak uygulamadan bağımsız olarak değişken tanımlama, döngüler, koşul yapısı, fonksiyon tanımlama vb. becerilerin nasıl kullanıldığını hatırlatmak amaçlı basit bir not giriş uygulamasının kodları aşağıda verilmiştir.

```
#include <stdio.h>
#include <stdlib.h>
void notDurumu(float ortalama)
{if(ortalama >= 50)
    printf("Durum: GECTI\n");
  else
    printf("Durum: KALDI\n");}
int main(){
  int i, j, ogrenciSayisi;
  char isim[50];
  int vize, final;
  float ortalama;
  int notMatrisi[50][3];
  FILE *dosya;
  dosya = fopen("notlar.txt", "w");
  if(dosya == NULL)
  {  printf("Dosya olusturulamadi!\n");
    return 1;  }
  printf("Kac ogrenci gireceksiniz: ");
  scanf("%d", &ogrenciSayisi);
  for(i = 0; i < ogrenciSayisi; i++)
  { printf("\n%d. Ogrencinin ismi: ", i + 1);
    scanf("%s", isim);
    printf("%d. Ogrencinin vize notu: ", i + 1);
    scanf("%d", &vize);
    printf("%d. Ogrencinin final notu: ", i + 1);
    scanf("%d", &final);
    ortalama = vize * 0.4 + final * 0.6;
    notMatrisi[i][0] = vize;
    notMatrisi[i][1] = final;
    notMatrisi[i][2] = (int)ortalama;
    fprintf(dosya, "%s %d %d %.2f\n", isim, vize, final, ortalama);}
  fclose(dosya);
  printf("\nVeriler dosyaya yazildi.\n");
  printf("\nMatris Taramasi Sonucu:\n");
  printf("Vize  Final  Ortalama\n");
  for(i = 0; i < ogrenciSayisi; i++)
  {for(j = 0; j < 3; j++){
    printf("%5d ", notMatrisi[i][j]);}
    printf("\n");}
  dosya = fopen("notlar.txt", "r");
  if(dosya == NULL)
  {printf("Dosya okunamadi!\n");
    return 1;}
  printf("\nDosyadan Okunan Veriler:\n");
  while(fscanf(dosya, "%s %d %d %f", isim, &vize, &final, &ortalama) != EOF)
  { printf("Isim: %s  Vize: %d  Final: %d  Ortalama: %.2f\n", isim, vize, final, ortalama);
    notDurumu(ortalama);
    printf("\n"); }
  fclose(dosya);
  return 0;}
```