ALGORİTMA VE PROGRAMLAMA II

HAFTA#3

YZM 1106

Celal Bayar Üniversitesi Hasan Ferdi Turgutlu
Teknoloji Fakültesi

struct Değişkenlerini Kopyalama

- **struct** tipindeki bir değişkenin değeri aynı tipteki bir başka **struct** değişkenine atanabilir.
- Üyeleri tek tek atmaya **gerek yoktur**.
- Atama aşağıdaki şekilde yapılır:

```
degisken1 = degisken2;
```

Örnek: struct Kopyalama


```c
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>

struct Ogrenci
{
  int No;
  char Ad[50];
  char Soyad[50];
  int Cinsiyet;
  int FakulteBolum;
  float GenelOrtalama;
};

int main()
{
  struct Ogrenci ogrenci_bilgisi1, ogrenci_bilgisi2;

  ogrenci_bilgisi1.No = 1;
  strcpy(ogrenci_bilgisi1.Ad, "Ada");
  strcpy(ogrenci_bilgisi1.Soyad, "KILINC");
  ogrenci_bilgisi1.FakulteBolum = 11;
  ogrenci_bilgisi1.GenelOrtalama = 4.5;

  ogrenci_bilgisi2 = ogrenci_bilgisi1;

  printf("1.OgrenciNo: %d \n",ogrenci_bilgisi1.No);
  printf("2.OgrenciNo: %d \n",ogrenci_bilgisi2.No);
  return 0;
}
```

ogrenci_bilgisi2.No = ogrenci_bilgisi1.No;
ogrenci_bilgisi2.Ad = ogrenci_bilgisi1.Ad;
ogrenci_bilgisi2.Soyad = ogrenci_bilgisi1.Soyad;
…

struct Değişkenlerini Karşılaştırma

- struct tipindeki bir değişken aynı tipteki bir başka struct değişkeni ile direk karşılaştırılamaz. Ancak **sahip olduğu üyeler ile** karşılaştırma yapılabilir.
- Aşağıdaki karşılaştırma ***yanlıştır***:

```
if (degisken1 == degisken2) ….
```

Örnek: struct Değişkenlerini Karşılaştırma


```
struct Ogrenci
{
    int No;
    char Ad[50];
    char Soyad[50];
    int Cinsiyet;
    int FakulteBolum;
    float GenelOrtalama;
} ogr1, ogr2;

int main()
{
    ogr1.No = 139280;
    ogr2.No = 139281;

    if (ogr1.No == ogr2.No)
        printf("ogr 1 ve ogr 2 esit...");
    else
        printf("ogr 1 ve ogr 2 esit degil...");

    return 0;
}
```

struct Değişkenleri ve Fonksiyonlar

1. struct tipindeki bir değişken herhangi bir fonksiyona **parametre** olarak aktarılabilir.
2. Herhangi bir fonksiyon **geri dönüş değeri** olarak struct tipinde bir veri türü geriye dönebilir.

Örnek: struct ve Fonksiyonlar


```c
#include <stdio.h>
#include <stdlib.h>

struct Ogrenci
{
    int No;
    char Ad[50];
    char Soyad[50];
    int Cinsiyet;
    int FakulteBolum;
    float GenelOrtalama;
};

void OgrenciBilgisiYazdir(struct Ogrenci ogr)
{
    printf(" No: %d \n Ad: %s \n Soyad: %s \n Cinsiyet: %d \n Fakulte-Bolum: %d \n Genel Ortalama: %0.2f",
           ogr.No,
           ogr.Ad,
           ogr.Soyad,
           ogr.Cinsiyet,
           ogr.FakulteBolum,
           ogr.GenelOrtalama
          );
}

int main()
{
    struct Ogrenci ogrenci_test;
    OgrenciBilgisiYazdir(ogrenci_test);

    return 0;
}
```

struct Değişkenleri ve Diziler

1. **struct** tipi içerisinde;
   - **Dizi** türünde **üyeler** tanımlamak mümkündür.
2. Dizileri;
   - **struct** tipinde tanımlamak mümkündür.

Nesneye dayalı programlama

yaklaşımına önemli bir ADIM….

struct Değişkenleri ve Diziler

- **OgrenciNot** isimli bir struct oluşturalım.
  - Öğrenci **Numarasını**,
  - 2 tane **Vize Notunu**,
  - 2 tane **Quiz Notunu** ve
  - 1 tane **Final Notunu** içersin.

Örnek: struct Değişkenleri ve Diziler


```c
#include <stdio.h>
#include <stdlib.h>

struct OgrenciNot
{
    int No;
    int VizeNotlar[2];
    int QuizNotlar[2];
    int Final;
};

int main()
{
    struct OgrenciNot ogr;

    ogr.No = 129211;
    ogr.VizeNotlar[1] = 78;
    ogr.VizeNotlar[2] = 65;
    ogr.QuizNotlar[1] = 82;
    ogr.QuizNotlar[2] = 66;
    ogr.Final = 61;

    return 0;
}
```

struct Değişkenleri ve Diziler

- **OgrenciNot** isimli bir structtan
  - Öğrenci **Numarasını**,
  - 2 tane **Vize Notunu**,
  - 2 tane **Quiz Notunu** ve
  - 1 tane **Final Notunu** içersin.

- **100 tane öğrenciyi tanımlayalım.**

Örnek: struct Değişkenleri ve Diziler


```
struct OgrenciNot
{
    int No;
    int VizeNotlar[2];
    int QuizNotlar[2];
    int Final;
};

int main()
{
    struct OgrenciNot ogr[100];

    ogr[0].No = 129211;
    ogr[0].VizeNotlar[1] = 78;
    ogr[0].VizeNotlar[2] = 65;
    ogr[0].QuizNotlar[1] = 82;
    ogr[0].QuizNotlar[2] = 66;
    ogr[0].Final = 61;

    return 0;
}
```

typedef Kullanımı

- typedef deyimi C dilinde **değişken tanımlama** yaparken kullanılan int, float, char gibi değişken isimlerini değiştirmeye yarar.
  - Bu sayede kodlar programcının anadiline daha fazla yaklaşmış olur.
- struct deyimi ile beraber kullanıldığında oluşturduğunuz yapıyı bir değişken türü olarak tanımlayıp o yapının çoğaltılmasını sağlar. Bu yapıdan değişken tanımlamak için tekrar struct deyiminin kullanılmasına gerek kalmaz.

Örnek: typedef Kullanımı


```c
#include <stdio.h>
#include <stdlib.h>

typedef struct
{
    int No;
    char Ad[50];
    char Soyad[50];
    int Cinsiyet;
} Ogrenci;

int main()
{
    Ogrenci ogr;
    ogr.No = 1000;
    strcpy(ogr.Ad, "Ada");
    strcpy(ogr.Soyad, "KILINC");
}
```

struct içerisinde struct Kullanımı

- C programlama dili **struct** içerisindeki bir üyenin yine bir **struct tipinde** olmasına izin verir. Bunu yapmaktaki temel amaç:
  - Kaynak kodun **tekrar kullanılabilirliğini arttırmak**
  - Kaynak kodun **okunabilirliğini arttırmak**
  - Kaynak kodu **sadeleştirmek**

Nesneye dayalı programlama

yaklaşımına önemli bir ADIM….

Örnek: struct içerisinde struct Kullanımı

- **OgrenciNot** isimli bir struct oluşturun
  - 2 Vize notu, 2 Quiz notu, 1 final notu olsun (Tüm alanlar int olabilir)
- **Iletisim** isimli bir struct oluşturun
  - Telefon ve Eposta üyeleri olsun (Karakter dizisi)
- **Ogrenci** isimli bir struct oluşturun
  - No, Ad, Soyad, Cinsiyet, **Iletisim** ve **OgrenciNot** üyeleri olsun

Örnek: struct içerisinde struct Kullanımı


```
typedef struct
{
    int VizeNotlar[2];
    int QuizNotlar[2];
    int Final;
} OgrenciNot;

typedef struct
{
    char telefon[50];
    char eposta[50];
} Iletisim;

typedef struct
{
    int No;
    char Ad[50];
    char Soyad[50];
    int Cinsiyet;
    Iletisim OgrIletisim;
    OgrenciNot Notlar;
} Ogrenci;
```

```
int main()
{
    Ogrenci ogr;
    ogr.No = 1000;
    strcpy(ogr.OgrIletisim.eposta, "drdenizkilinc@gmail.com");
    strcpy(ogr.OgrIletisim.telefon, "05306663654");

    ogr.Notlar.QuizNotlar[1] = 65;
    ogr.Notlar.QuizNotlar[2] = 54;
    ogr.Notlar.Final = 78;
}
```

Birlikler (Union)

- Birlikler de yapılar gibi sürekli belleğe yerleşen nesnelerdir.
- Birlikler yapılara göre daha az kullanılırlar. Bir programda veya fonksiyonda değişkenlerin aynı bellek alanını paylaşması için ortaklık bildirimi union deyimi ile yapılır. Bu yer, birliğin en büyük alanı kadardır.
- Bu da belleğin daha verimli kullanılmasına imkan verir.
- Bu tipte bildirim yapılırken struct yerine union yazılır.

Birlikler (Union) (devam…)

Örnek:

```
union test
{
    int x;
    char y[10];
} p
```

- Bu tanıma göre x tamsayısı için bellekte 4 baytlık yer ayrılmaktadır.
- Birlik içinde yer alan y değişkeni için 10 baytlık yer ayrılmıştır.
- Bu durumda, birlik için en fazla 10 baytlık bir yer ayrılmıştır. Ayrılan bu alan, birliğin her bir alanı tarafından ortak kullanılır.

KAYNAKLAR

- N. Ercil Çağıltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Milli Eğitim Bakanlığı "Programlamaya Giriş ve Algoritmalar Ders Notları", 2007
- Problem Solving and Program Design in C, Hanly, Koffman
- http://www.AlgoritmaveProgramlama.com

Algoritma ve Programlama

İYİ ÇALIŞMALAR…