C++ Programlama Dili Öğeleri

Yapılar (Structs)

BG-211 İleri Programlama

Yapılar (Structs)

- Yapılar, farklı tipteki (integer, float, double vb.) elemanlara sahip veri yapıları olarak tanımlanabilir.
- Bir yapı içerisinde sadece karakter, integer, float ya da double gibi temel tipteki verileri değil, aynı zamanda dizileri, işaretçileri ve hatta başka yapıları eleman olarak kullanabiliriz.
- Bir yapı içerisinde kullanılan her bir elemana “ÜYE” (Member) adı verilir.
- Genel olarak kullanımı;

```
struct tag {
    member 1;
    member 2;
    ...
    member m;
};
```

Yapılar (Structs)

- Tipik bir yapı tanımlaması şu şekilde olur;

```
struct account {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
};
```

- Tanımlanan yapıya “account” adı verilmiştir,
- Dört üyesi vardır: bir integer, bir karakter, bir karakter dizisi ve bir float değişken.
- Bu tanımlamayı yaptıktan sonra yapıya ait değişkenler tanımlayabiliriz;
  - struct account old_customer, new_customer;

Yapılar (Structs)

- Bir yapıya ait değişkenleri yapıyı tanımlarken oluşturmakta mümkündür,
  - “account” yapısı için kullanacak olursak;

```
struct account {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
} old_customer, new_customer;
```

- Bu yapıya ait old_customer ve new_customer’dan başka değişken yaratılmayacaksa da opsiyonel olan account tagı kullanılmayabilir;

```
struct {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
} old_customer, new_customer;
```

Yapılar (Structs)

- Bir yapı başka bir yapıya üye olabilir. Böyle bir durumda üye olan yapı, üyesi olduğu yapıdan önce tanımlanmış olmalıdır.

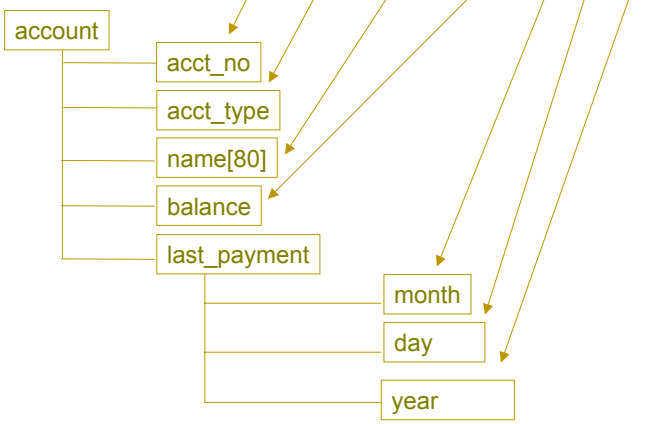
```
struct date {
    int month;
    int day;
    int year;
} ;
```

```
struct account {
    int acct_no;
    char acct_name;
    char name[80];
    float balance;
    struct date last_payment;
};
```


Yapılar (Structs)

- Yapı üyelerine ilk değer atanması da dizilerle yapıldığı şekilde yapılır.

```
struct account customer = {12345, 'Y', "Tolga", 1000.0, 5, 24, 90}
```



Yapılar (Structs)

- Yapılardan oluşan diziler yaratmak da mümkündür;

```
struct date {
    char name[80];
    int month;
    int day;
    int year;
};
```

```
struct date birthday[ ] = {   {"Amy", 12, 30, 73},
                              {"Gail", 5, 13, 66},
                              {"Nurettin", 7, 1, 12} };
```


Yapılar (Structs)

- Programlarda yapı üyelerine ulaşmak için;
  - variable.member

    notasyonu kullanılır.

```
struct account {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
} customer;
```

- Eğer "customer" değişkeninin "name" değerine ulaşmak istiyor isek;
  - customer.name

  yazılarak ulaşılabilir.

typedef

- typedef komutu kullanıcıların varolan veri tiplerine denk yeni veri tipleri tanımlamalarını sağlar.

  typedef int age;

  age male, female;

- Yapıları kullanırken sürekli olarak “struct tag” ifadesini kullanmak yerine, yapımıza typedef ile yeni bir ad vererek bu uzun ifadeyi kullanmak zorluğunu ortadan kaldırabiliriz;

```
typedef struct {
    int acct_no;
    ….
} record; ;
```

böylece;

struct account old_customer; yerine

record old_customer; yazabiliriz.

Yapılar ve İşaretçiler

- Bir yapıya ait işaretçi değişkenler tanımlayabiliriz;

  type *ptvar; (struct account *pc)

  ptvar = &variable; (pc = &customer)

- İşaretçi kullandığımızda, işaretçi üzerinden üyelere erişmek de mümkündür,

  ptvar -> member (pc->acct_no)

- Aşağıdaki 3 ifade de birbirine denk olacaktır;

  customer.acct_no, pc->acct_no, (*pc).acct_no

- Bir yapının dizi tipindeki üyelerine ulaşmak söz konusu olduğunda da benzer yöntemlerle birlikte birçok farklı ifadeyi kullanabiliriz;

  customer.name[2], pc->name[2], (*pc).name[2]

  *(customer.name+2), pc->(name+2)

  *((*pc).name+2) gibi

Yapıların Fonksiyona Aktarımı

- Yapının üyeleri teker teker bir fonksiyona gönderilip, döndürülebileceği gibi, yapı bir bütün olarak da bir fonksiyona gönderilip, fonksiyondan geri bir bütün olarak döndürülebilir.
- Bir yapının üyeleri teker teker bir fonksiyona gönderildiğinde üyelerin yazımı ve davranışları, standart veri tipleriyle tamamıyla aynı olacak, herhangi özel bir durum söz konusu olmayacaktır.
- Fonksiyona yapıyı gönderirken aynı dizilerde olduğu gibi yapının işaretçisini yollarız,
- İşaretçi kullandığımız durumda, yine dizilerde olduğu gibi, fonksiyon içinde yapı üzerinde yapılacak değişikliklerin fonksiyon dışında da etkili olacaktır.

```
1.  #include <iostream.h>
2.  #include <iomanip.h>
3.
4.  struct Time {
5.    int hour;     // 0-23 (24-hour clock format)
6.    int minute;   // 0-59
7.    int second;   // 0-59
8.  }; // end struct Time
9.
10. void printUniversal( const Time & );  // prototype
11. void printStandard( const Time & );   // prototype
12.
13. int main()
14. {
15.   Time dinnerTime;
16.
17.   dinnerTime.hour = 18;
18.   dinnerTime.minute = 30;
19.   dinnerTime.second = 30;
20.
21.   cout << "Dinner will be held at ";
22.   printUniversal( dinnerTime );
23.
24.   cout << " universal time,\nwhich is ";
25.   printStandard( dinnerTime );
26.
27.   cout << " standard time.\n";
28.   return 0;
29. }
```

```
1.  // print time in universal-time format
2.  void printUniversal( const Time &t )
3.  {
4.    cout << setw( 2 ) << t.hour << ":"
5.         << setw( 2 ) << t.minute << ":"
6.         << setw( 2 ) << t.second;
7.
8.  }
9.
10. // print time in standard-time format
11. void printStandard( const Time &t )
12. {
13.    cout << ( ( t.hour == 0 || t.hour == 12 ) ? 12 : t.hour % 12 )
14.         << ":" << setw( 2 ) << t.minute << ":"
15.         << setw( 2 ) << t.second
16.         << ( t.hour < 12 ? " AM" : " PM" );
17.
18. }
```

```
Dinner will be held at 18:30:30 universal time,
which is 6:30:30 PM standard time.
```