BÖLÜM 11

SOYUT VERİ TİPLERİ

ANAHTAR KAVRAMLAR

Enum (Numalandırma), Fonksiyon İşaretçileri, Struct (Yapı)

ÖĞRENME HEDEFLERİ

1. Enum (numaralandırma) tanımlamayı ve kullanmayı bilir.
2. Fonksiyon işaretçilerini kullanarak fonksiyonlara veri tipi oluşturur.
3. Oluşturulmuş veri tipini başka bir fonksiyonda parametre olarak kullanır.
4. Struct (yapı) tanımlamayı ve kullanmayı bilir.
5. Struct içerisinde fonksiyon oluşturur.
6. Struct değişkenini fonksiyonda parametre olarak kullanır.

ONBİRİNCİ BÖLÜM

GİRİŞ

Bu bölümde C++ programlama dilinde soyut veri tanımlamada kullanılan,

enum (numaralandırma),

Fonksiyon işaretçileri,

struct (yapı)

kavramları anlatılmıştır. Numaralandırma aynı kümedeki elamanlara sabit tamsayı değerler atanması şeklinde yapılmaktadır. Küme elemalarının isimleri doğrudan kendilerine atanan tamsayı değerlerine karşılık olarak kullanılmaktadır. Fonksiyon işaretçileri ile veri türü ve parametre sayısı aynı olan fonksiyonları değer olarak alan fonksiyon veri tipi tanımlanır. Yapılar C++'da sınıflara benzer özelliklere sahip olup, bir kavrama ait değişkenler ve fonksiyonlarla oluşturulabilir.

1. SOYUT VERİ TİPLERİ

Soyut veri tipleri geliştirici tarafından çözülmeye çalışılan probleme özgü oluşturulan veri tipleridir. Problem çözümünün ilk aşamasındaki kavramsallaştırmanın programlama dillerindeki karşılıkları olan soyut veri tipleri, genel olarak kütüphane dosyalarının oluşturulması aşamasında geliştirilirler. Programlama dillerinin aktarım gücünün ölçütlerinden biri olan soyut veri tanımlama C++ dilinde,

- enum (numaralandırma),
- fonksiyon işaretçileri,
- struct (yapı),
- class (sınıf)

yapıları ile geliştirilmektedir.

Bu bölümde enum, fonksiyon işaretçileri ve struct ile veri tipleri açıklanacaktır. Nesne yönelimli dillerin temel programlama yapıları olan sınıflar bir sonraki bölümün konusudur.

2. ENUM (NUMARALANDIRMA)

C++ programlamada, enum (numaralandırma) tamsayı sabitlerini temsil eden adlandırılmış değerlerden oluşan bir veri türüdür. Değişkenlerin alabileceği farklı değerlerin sınırlı sayıda olması halinde programın okunabilirliğini artırmak için tercih edilir. Değişkenlere tek tek sayısal değerler vermek bazı durumlarda hataya sebep olabilmektedir. Bu durumu ortadan kaldırmak için enum kullanılabilir.

Numaralandırmanın C++'daki sözdizimi;

```
enum enum_adı {sabit1, sabit2, …};
```

enum_adı'nın elemanları sabit1, sabit2, … durumlarını içeren küme ismini ifade etmektedir. Örneğin günleri tutacak bir enum yapısında veri tipi (enum) ismi Gunler olacaktır.

Örnek:

```
#include <iostream>
using namespace std;
enum Gunler {Pazartesi, Sali, Carsamba, Persembe, Cuma, Cumartesi, Pazar};
int main () {
   cout<<"Pazartesi:"<<Pazartesi<<endl;
   cout<<"Sali:"<<Sali<<endl;
   cout<<"Carsamba:"<<Carsamba<<endl;
   cout<<"Persembe:"<<Persembe<<endl;
   cout<<"Cuma:"<<Cuma<<endl;
   cout<<"Cumartesi:"<<Cumartesi<<endl;
   cout<<"Pazar:"<<Pazar<<endl;
   return 0;
}
```

Çıktı:

```
Pazartesi:0
Sali:1
```

```
Carsamba:2
Persembe:3
Cuma:4
Cumartesi:5
Pazar:6
```

Program çıktısı enum içerisinde verilen isimlerin (const int Pazartesi = 0, Salı =1, …) gibi değerleri enum'daki bildirim sırasına göre aldıklarını göstermektedir. Bu anlamda enum içerisindeki değerler Gunler kümesini oluşturan tam sayı sabitlerdir.

Gunler içerisindeki isimlerden herhangi birine değer atanırsa bir sonraki ismin değeri, atanan değerden bir büyük sayı olacaktır. Örneğin,

```
enum Gun {Pazartesi, Sali=3, Carsamba, Persembe, Cuma, Cumartesi, Pazar};
```

enum yapısı verilmiş olması durumunda Pazartesi değeri değişmezken Sali= -3, Carsamba=-2 Perşembe

=-1 şeklinde devam edecektir.

Çıktı:

```
Pazartesi:0
Sali:-3
Carsamba:-2
Persembe:-1
Cuma:0
Cumartesi:1
Pazar:2
```

Enum içerisindeki isimlerin program tarafından sabit tamsayı değerlerine karşılık geldiğini gördüğümüze göre, bu değerlerden oluşturulan Günler veri tipi nasıl kullanılacak?

C++'da bir değişken bildirimi;

```
Veri_tipi değişken_ismi;
```

Değer ataması;

değişken_ismi = veri kümesi içerisindeki değerlerden biri;

şeklinde olmaktadır. Bu hatırlatmaya dayanarak,

```
Gun gun; // değişken bildirimi
gun = Persembe;
```

veya

```
Gun gun = Persembe;
```

şeklinde yapılabilir.

Programlamada enum'ların yaygın kullanıldığı durumlardan biri küme elemanlarının temsil ettiği değişkenlerde yapılması gereken işlerin farklılık göstermesidir. Renk kümesinin elemanlarının değerlerine göre farklı kod gruplarının seçimi aşağıda verilen örnek kod üzerinde gerçekleştirilmiştir.

Örnek:

```
#include <iostream>
using namespace std;
enum Renk{beyaz=3, siyah=5, turuncu};
int main() {
    Renk r = turuncu;
    switch(r)
    {
          case turuncu: cout << “Turuncu\n”; break;
          case beyaz: cout << “Beyaz\n”; break;
          case siyah : cout << “Siyah\n”; break;
          default: cout << “Renk yok\n”; break;
    }
    return 0;
}
```

Çıktı:

```
Turuncu
```

DİKKAT EDELİM

Yapı (struct) ile oluşturulan veri tipindeki değişkene değer ataması Veri tipi değişken = {değer1, değer2 ,...} biçiminde yapılabilir.

3. YAPI İÇERİSİNDE FONKSİYON KULLANMA

C++ programlama dilinde yapı içerisinde fonksiyon geliştirilerek tanımlanan kavrama ait yapılabilecek işler tek bir programlama biriminde tamamlanabilir. Yukarıdaki Dikdörtgen örneğini fonksiyon içeren yapıya dönüştürürsek;

Örnek:

```
#include<iostream>
using namespace std;
struct Dikdortgen{
    int yukseklik;
    int genislik;
    void alan(){
          int al = yukseklik * genislik;
          cout<<” Alan :”<<al<<endl;
    }
    void cevre(){
          int cev = 2* (yukseklik + genislik);
          cout<<” Cevre :”<<cev<<endl;
```

```
    }
};
int main(){
    Dikdortgen d = {3,4};//değişken bildirim ve değer atama
    d.cevre(); // fonksiyon çağırma
    d.alan();
    return 0;
}
```

Çıktı:

```
Cevre :14
Alan :12
```

DİKKAT EDELİM

Yapı (struct) içerisinde fonksiyon kullanarak veri tipindeki değişkenin hesaplanmak istenilen değerleri yapıyla bütünleştirilebilir.

4. YAPI ÜRETİCİ FONKSİYONU

Yapı içerisinde yapı ismiyle aynı isimli kod grubuna üretici fonksiyon denir. Üretici fonksiyonların geri dönüş değeri yoktur. Yapı değişkeni oluşturulurken ilk değerlerin atamasının yapılmasında kullanılır.

```
struct Dikdortgen{
    int yukseklik;
    int genislik;
    // üretici fonksiyon tanımlama
    Dikdortgen (int y, int g){
      yukseklik = y;
      genislik = g;
    }
};
```

Üretici fonksiyona sahip bir yapı tipindeki değişkene değer atama;

```
Dikdortgen d = Dikdortgen(4, 5);
Dikdortgen d1 = {4, 5};
```

veya

```
Dikdortgen d2;
d2.yukseklik = 4;
d2.genislik = 5;
```

şeklinde yapılabilir.

DİKKAT EDELİM

Yapıya ait üretici fonksiyon yapının ismiyle aynı isimdeki özel bir fonksiyon olup geri dönüşü yoktur.

```
struct Uretici{
       int numara;
       string ad;
       Uretici(int x, string a){
          numara =x;
          ad = a;}
       }
};
```

BÖLÜM ÖZETİ

Bu bölümde, geliştiricilerin programlarında yeni veri tipleri tanımlamak amacıyla kullandıkları **enum, fonksiyon işaretçileri** ve **struct kavramları** ele alınmıştır.

Enum (enumeration), bir küme içerisindeki değişkenlerin sabit tamsayılarla ifade edilmesini sağlayan bir veri tipi tanımlama mekanizmasıdır. Bu yapı, değişkenlerin daha anlamlı ve okunabilir bir şekilde temsil edilmesine olanak tanır.

Fonksiyon işaretçileri, aynı dönüş tipine ve parametre yapısına sahip fonksiyonları temsil etmek için kullanılan veri tipleridir. Bu yapılar, fonksiyonların birer veri nesnesi gibi işlenmesini sağlar ve esnek programlama imkanları sunar.

Yapılar (struct), C++ dilindeki en karmaşık veri yapılarından biridir ve sınıflarla (class) birlikte gelişmiş veri modellemelerine olanak tanır. Bu bölümde, bir yapı (struct) tanımlama, bu yapıdan türetilen değişkenlerin bildirilmesi ve değer atanması üzerinde durulmuştur. Ayrıca, yapılarda fonksiyon tanımlama, üretici fonksiyon (constructor) oluşturma ve kullanma süreçleri detaylı bir şekilde açıklanmıştır.

Bu kavramlar, C++ programlamasında özelleştirilmiş veri tipleri oluşturmak ve bu tipler üzerinde etkin işlemler gerçekleştirmek için önemli bir temel oluşturmaktadır.

GÖZDEN GEÇİRELİM

1.

```
#include <iostream>
using namespace std;
enum Renk{sari, mavi=5, kirmizi, yesil, siyah};
int main() {
      Renk r = siyah;
      cout<<"Renk : "<< r <<endl;
return 0;
}
```

Yukarıda verilen kodun çıktısı nedir?

a. 8

b. 0

c. 3

d. 4

e. 5

2.

```
#include <iostream>
using namespace std;
enum Renk{sari=-1.2, mavi, kirmizi, yesil, siyah};
int main() {
      Renk r = kırmızı;
      cout<<"Renk :"<<r <<endl;
return 0;
}
```

Yukarıda verilen kodun çıktısı nedir?

a. Program derleme hatası verir.

b. -2

c. -1

d. 0

e. 1

```
#include <iostream>
using namespace std;
enum Gunler {pazar, pazartesi, sali, carsamba, persembe, cuma, cumartesi};
int main () {
   Gunler g = sali;
   switch(g){
      case pazartesi:{
            cout<<"Pazartesi ";
            break;
       }
      case sali:{
            cout<<"Sali";
      }
      case carsamba:{
            cout<<"Carsamba ";
      }
      case persembe:{
            cout<<"Persembe ";
      }
      case cuma: {
            cout<<"Cuma ";
            break;
      }
   }
return 0;
}
```

3. Yukarıda verilen programın çıktısı nedir?

a. Sali Carsamba Persembe Cuma

b. persembe

c. sali

d. carsamba

e. Cuma

```
float cikar(float x, float y){
    return x - y;
}
float carp(float x, float y){
    return x * y;
}
```

4. **Fonksiyonlarını değer olarak alacak işaretci fonksiyonu aşağıdakilerden hangisinde doğru tanımlanmıştır?**
   a. typedef float (*Islem)(float x ,float y);
   b. typedef float Islem(float x ,float y);
   c. typedef float (*Islem)(int x ,int y);
   d. typedef (*Islem)(float x ,float y);
   e. typedef vod (*Islem)(float x ,float y);

5. **void hesapla(float x, float y, Islem op,string islem){**
   cout<<islem<<" "<<op(x,y)<<endl;
}

Fonksiyonunun çağrılması aşağıdakilerden hangiside doğru yapılmıştır.
   a. Islem op = topla; hesapla(1.0, 2.0, op, "topla");
   b. Islem op = topla; cout<<hesapla(1.0, 2.0, op, "topla");
   c. op Islem = topla; hesapla(1.0, 2.0, op, "topla");
   d. op topla = Islem; hesapla(1.0, 2.0, op, "topla");
   e. Islem topla = op; hesapla(1.0, 2.0, op, "topla");

6. **Kişilerin ad, soyad, yas bilgilerini tutacak yapı aşağıdakilreden hangisinde doğru tanımlanmıştır.**
   a.
```
struct Kisi{
    string ad;
    string soyad;
    int yas;
    };
```
   b.
```
Kisi struct {
    string ad;
    string soyad;
    int yas;
    };
```
   c.
```
Kisi {
    string ad;
    string soyad;
    int yas;
    } struct;
```
   d.
```
Kisi{
    int ad;
    int soyad;
    string yas;
    };
```
   e.
```
struct Kisi(
    string ad;
    string soyad;
    int yas;
    );
```

7. **struct Daire{**
   float r;
};
Yukarıda verilen Daire koduna aşağıdakilerden hangisi eklenirse, Daire d = Daire(1.5); ifadesi çalışır.
   a. Daire (float x) {r = x;}
   b. void Daire(float x){r =x;}
   c. void Daire(float x){x =r;}
   d. Daire (float x) {x = r;}
   e. Yeni kod eklemek gerekmez.

8. **İki boyutlu düzlemde bir nokta x ve y eksen değerleri ile verilmektedir. Bir noktanın değerlerinin girileceği veri tipi aşağıdakilerden hangisinde tanımlanmıştır.**

a.
```
struct Nokta{
        float x;
        float y;
};
```

b.
```
enum Nokta{
        float x;
        float y;
};
```

c.
```
Nokta(float x, float y);
```

d.
```
Nokta enum {
        float x;
        float y;
};
```

e.
```
enum Nokta{
        float x;
        float y;
};
```

9. **C++ dilinde yapı (struct) aşağıdakilerden hangisi ile sonlandırılır?**

a. ;

b. :

c. }

d. .

e. :

10.
```
#include <iostream>
using namespace std;
int main() {
    student{
      int num;
      string name;
};
student stu;
stu.num = 123;
stu.name = "Ali";
cout << stu.num << endl;
cout << stu.name << endl;
return 0;
}
```

Yukarıda verilen kodun çıktısı nedir?

a. Derleme hatası verir.

b. 123 Ali

c. 123

d. Ali

e. Ali 123

Yanıt Anahtarı: 1-A, 2-A, 3-A, 4-A, 5-A, 6-A, 7-A, 8-A, 9-A, 10-A

Kaynakça

Deitel, P., Deitel, H. (2017). C++ How to Program. 10. Baskı. Pearson. Essex, İngiltere.

Dikici, M. (2017). C++ Programlama Dili: Yapısal Programlama, Nesnelerle Programlama ve Jenerik Programlama. Seçkin Yayıncılık. İstanbul, Türkiye.

Çobanoğlu, B. (2021). C/C++ Programlama Eğitim Kitabı. Kodlab. İstanbul, Türkiye.

Demir, S., Eryiğit, R., Ar, Y., Tuğrul, B., Gök, O., Kul, S., . . . Yağanoğlu, M. (2019). Programlama Temelleri. Atatürk Üniversitesi Açıköğretim Fakültesi.

Kaleli, C., Bilge, A., Ergin, S., & Şora Günal, E. (2016). Bilgisayar ve Programlamaya Giriş. Eskişehir: Anadolu Üniversitesi.