BÖLÜM 13

STRİNG SINIFI VE KARAKTER DİZİSİ İŞLEMLERİ

ANAHTAR KAVRAMLAR

String Kütüphanesi, String Kapasite Metotları, String Erişim Metotları, String Düzenleyici Metotları

ÖĞRENME HEDEFLERİ

1. Bir string değişkeni tanımlayabilir.
2. Bir string değişkenine değer ataması yapabilir.
3. String sınıfına ait kapasite metotlarını bilir.
4. String sınıfına ait erişim metotlarını bilir.
5. String sınıfına ait düzenleyici metotlarını bilir.
6. Struct değişkenini fonksiyonda parametre olarak kullanır.

ONÜÇÜNCÜ BÖLÜM

GİRİŞ

String veri türü “Merhaba” gibi iki veya daha fazla karakterden oluşan metin içeriğine sahip nesneleri saklamak için kullanılır. C++ programlama dilinde tanımlı diğer temel veri türleri gibi kullanmadan önce tanımlanmalı ve değer ataması yapılmalıdır. String veri türüne ait bir değişken tanımlayabilmek ve metotlarına erişebilmek için <string> kütüphanesinin programa eklenmesi gerekmektedir. String değişkeninin değeri çift tırnak içine alınması gerekmektedir.

Metin içerikli değişkenler aynı zamanda karakter dizisi (C-tarzı string) olarak da oluşturulabilir. Ancak zamanla bu kullanımdan kaynaklanan bellek yönetimi sorunları nedeniyle daha güvenilir ve verimli string sınıfının kullanılması tavsiye edilmektedir.

String kütüphanesinde tanımlı çok çeşitli metotlar vardır. Bu metotlar string nesneleri üzerinde işlemler yapabilmeyi sağlar. Bu metotlar genel olarak üç ana başlık altında değerlendirilebilir. Kapasite metotları string nesnesinin uzunluğu, hafızada ne kadar alan kullandığı gibi bilgilere ulaşmayı sağlar. Erişim metotları string değişkenin belirli bir karakterine veya bir grup karakterine erişme amacıyla tanımlanmıştır. Düzenleyici metotları, iki farklı string nesnesini birleştirmeye veya üzerinde ekleme veya çıkarma gibi karakter işlemleri gerçekleştirmeye izin verir. Bu üç gruba girmeyen daha genel kullanıma sahip metotlar diğer metotlar başlığı altında incelenmiştir.

Metotların kullanımı aldıkları parametre sayısı ve veri türüne göre farklılıklar göstermektedir. Metotların kullanımı en genel ve sıklıkla kullanılan şekliyle anlatılmıştır. Diğer kullanım yöntemlerine bölüm sonunda verilen kaynaklardan ulaşmak mümkündür.

1. STRİNG DEĞİŞKENİ TANIMLAMA VE DEĞER ATAMA

Bir string nesnesi oluşturmak için kullanılabilecek farklı tür ve sayıda girdiye sahip birkaç yapıcı yöntemi vardır. En genel haliyle, bir string nesnesi üç farklı yöntem ile oluşturulabilir. İlk yöntem string sınıfının tek parametre alan yapıcı metodu kullanılarak gerçekleştirilir.

string str("Merhaba");

İkinci yöntem benzer şekilde yine string sınıfının iki parametre alan yapıcı metodu kullanarak gerçekleştirilir. Aşağıda verilen örnek kod bloğu üç adet 'w' karakterinden oluşan "www" değerine sahip string nesnesini oluşturur.

string str(3, 'w ');

Üçüncü yöntem atama işleci kullanılarak gerçekleştirilir.

string str="Merhaba";

Yukarıda anlatılan string nesnesi oluşturma ve değer atama yöntemleri ve diğer yöntemler aşağıda verilen örnek kod bloğunda gösterilmiştir.

Örnek:

```
string str1 ("Merhaba Ankara");
string str2 (str1);
string str3 (str1, 8, 6);
string str4 ("Merhaba Ankara", 8);
string str5 (3, 'w');
string str6 (3, 119); //'w' karakterinin ASCII karşılığı olan değeri 119
string str7 (str1.begin(), str1.begin()+7);
cout<<"str1= "<<str1<<endl;
cout<<"str2= "<<str2<<endl;
cout<<"str3= "<<str3<<endl;
cout<<"str4= "<<str4<<endl;
cout<<"str5= "<<str5<<endl;
cout<<"str6= "<<str6<<endl;
cout<<"str7= "<<str7<<endl;
```

Çıktı:

```
str1= Merhaba Ankara
str2= Merhaba Ankara
str3= Ankara
str4= Merhaba
str5= www
str6= www
str7= Merhaba
```

2. KAPASİTE METOTLARI

Üzerinde çalışılan string nesnesinin uzunluğu, hafızada ne kadar alana gereksinim duyduğu gibi bilgilere ulaşmak için kapasite metotları altında size, length, max_size, ve capacity metotları tanımlanıştır. Ayrıca string nesnesinin boş olup olmadığını öğrenmek için empty metodu ve gerektiğinde değişkenin sakladığı bütün karakterlerini silmek için tanımlanmış clear metotları mevcuttur.

size ve length Metotları

Hem size hem de length metotları bir string değişkenindeki boşluklar ve noktalama işaretleri dahil olmak üzere toplam karakter sayısını döndürür.

Örnek:

```
string str = “Bu metinde 33 karakter mevcuttur. “;
cout<<str.length()<<endl;
cout<<str.size()<<endl;
```

Çıktı:

```
33
33
```

max_size Metodu

Bir string değişkenin saklayabileceği en büyük uzunluk değerini döndürür. Bu sistem tarafından sağlanabilecek uzunluğa karşılık gelmektedir. Bu değerin aşılması durumunda length_error hatası alınır. Metot çalıştırılan bilgisayarın özelliklerine ve kullanılan işletim sistemine bağlı olarak farklı değer üretebilmektedir.

Örnek:

```
string str = “Merhaba Ankara!”;
cout<<str.size()<<endl;
cout<<str.max_size()<<endl;
```

Çıktı:

```
15
4611686018427387903
```

capacity Metodu

Capacity metodu string değişkeni için ne kadarlık bir alanın hafızada ayrıldığı bilgisini döndürür. Dönen değerin string uzunluğu ile aynı olması gerekmemektedir. Değişkene sonradan eklenebilecek karakter işlemlerini optimize etmek için değişken uzunluğuna eşit veya daha uzun olabilir.

Örnek:

```
string str = “Merhaba Ankara!”;
cout<<str.capacity()<<endl;
cout<<str.length()<<endl;
str += “!”;
cout<<str.capacity()<<endl;
cout<<str.length()<<endl;
```

Çıktı:

```
15
15
30
16
```

Yukarıdaki örnekte oluşturulan string nesnesinin hem capacity hem de uzunluğu 15 iken değişkene bir karakter eklenmesi durumunda capacity 30'a uzunluğu ise 16'a çıkmıştır.

empty Metodu

Bir string değişkenin içeriğinin boş olup olmadığını kontrol eder. Eğer çağrıldığı değişkenin uzunluğu sıfır ise 1 değeri döndürür değilse 0 değeri döndürür.

Örnek:

```
string str1 = "";
string str2 = "Merhaba";
if(str1.empty())
cout<<"Bos string";
if(str2.empty())
cout<<"Dolu string";
```

Çıktı:

Bos string

clear Metodu

Bir string değişkenin içeriğini boşaltır. Böylece uzunluğu sıfır olan bir string halini alır.

Örnek:

```
string str = "Merhaba";
str.clear();
if(str.empty())
cout<<"Boş string";
```

Çıktı:

Boş string

Erişim Metotları

Bazı durumlarda string değişkenini oluşturan karakterlere erişmek ve üzerinde birtakım değişiklikler yapmak gerekmektedir. String nesnesinin son karakterine erişmek için back metodu tanımlanmıştır. Benzer şekilde ilk karakterine front metodu ile erişilir. Herhangi bir karaktere erişmek için at ve [indeks] yapıları tercih edilebilir.

back Metodu

Bir string değişkeninin en son karakterine erişmek için kullanılır.

Örnek:

```
string str = "0123456789";
cout<<str.back()<<endl;
```

Çıktı:

9

front Metodu

Bir string değişkeninin ilk karakterine erişmek için kullanılır.

Örnek:

```
string str = "0123456789";
cout<<str.front()<<endl;
```

Çıktı:

0

at Metodu

Köşeli parantez veya "at" metodu kullanarak iki farklı yöntem ile bir string değişkenin bütün karakterlerine erişilebilir. Değişkenin ilk karakterinin indeks değeri sıfırdır. String değişkenin sınır değerleri dışındaki konumlara erişmemeye dikkat edilmesi gereklidir.

Örnek:

```
string str = "0123456789";
cout<<str.at(0)<<endl;
cout<<str.at(5)<<endl;
cout<<str.at(9)<<endl;
```

Çıktı:

0

5

9

Benzer sonucu aşağıda verilen kod bloğu da gerçekleştirmektedir.

Örnek:

```
string str = "0123456789";
cout<<str[0]<<endl;
cout<<str[5]<<endl;
cout<<str[9]<<endl;
```

Çıktı:

0

5

9

3. DÜZENLEYİCİ METOTLAR

Oluşturulan string nesnesine bazı durumlarda eklemeler ve çıkarmalar yapmak mümkündür. Hem append metodu hem de aşırı yüklenmiş olan "+" işleci kullanarak var olan string değişkenine ekleme yapmak mümkündür. Eğer string nesnesinin sonuna tek bir karakter eklenmek isteniyorsa push_back metodu kullanılabilir. En sondan tek bir karakter silme işlemini ise pop_back metodu gerçekleştirir. İki farklı string değişkeninin değerleri değiştirilmek isteniyorsa swap metodu kullanılmalıdır.

append Metodu

Bir string nesnesinin sonuna ilave karakterler ekleyerek string nesnesinin genişletilmesini sağlar. Aynı sonuç '+' işleci kullanılarak da elde edilebilir.

Örnek:

```
string str1 = "Merhaba ";
string str2 = "Ankara";
string str3 ="!";
str1.append(str2);
```

```
string str4 = str1 + str3;
cout<<str1<<endl;
cout<<str4<<endl;
```

Çıktı:

Merhaba Ankara
Merhaba Ankara!

push_back Metodu

Bir string nesnesinin sonuna ilave bir karakter ekler. Böylece string nesnenin uzunluğu bir artar.

Örnek:

```
string str1 = "Merhaba Ankara";
cout<<str1<<" "<<str1.length()<<" karakterden oluşmaktadır."<<endl;
str1.push_back('!');
cout<<str1<<" "<<str1.length()<<" karakterden oluşmaktadır."<<endl;
```

Çıktı:

Merhaba Ankara 14 karakterden oluşmaktadır.
Merhaba Ankara! 15 karakterden oluşmaktadır.

pop_back Metodu

Bir string nesnesinin sonundan bir karakter siler. Böylece string nesnenin uzunluğu bir azalır. pop_back metodu parametre almaz.

Örnek:

```
string str1 = "Merhaba Ankara!";
cout<<str1<<" "<<str1.length()<<" karakterden oluşmaktadır."<<endl;
str1.pop_back();
cout<<str1<<" "<<str1.length()<<" karakterden oluşmaktadır."<<endl;
```

Çıktı:

Merhaba Ankara! 15 karakterden oluşmaktadır.
Merhaba Ankara 14 karakterden oluşmaktadır.

DİKKAT EDELİM

pop_back gibi bazı metotları kullanabilmek için string değişkenine değer ataması yapılması gerekmektedir. Eğer string değişkeni boş ise hata mesajı ile karşılaşılır.

insert ve erase Metotları

Bir string nesnesinin istenilen yerine ilave karakterlerin yerleştirilmesi insert metodu ile gerçekleştirilir. Eğer string nesnesinden bazı karakterlerin çıkarılması isteniyorsa erase metodu tercih edilir.

Örnek:

```
string str1 = "Merhaba Ankara";
cout<<str1<<endl;
str1.erase(8,6);
```

```
cout<<str1<<endl;
str1.insert(8, “Antalya”);
cout<<str1<<endl;
```

Çıktı:

```
Merhaba Ankara
Merhaba
Merhaba Antalya
```

replace Metodu

Bir string nesnesinin istenilen kısmın çıkarılmasını ve yerine yeni karakterlerin eklenmesini sağlar.

Örnek:

```
string str1 = “Merhaba Ankara”;
cout<<str1<<endl;
str1.replace(8, 6, “Mersin”);
cout<<str1<<endl;
```

Çıktı:

```
Merhaba Ankara
Merhaba Mersin
```

swap Metodu

İki farklı string nesnesinin birbirleriyle değerlerinin değiştirilmesini sağlar.

Örnek:

```
string str1 = “Ankara”;
string str2 = “Antalya”;
cout<<”str1= “<<str1<<endl;
cout<<”str2= “<<str2<<endl;
str1.swap(str2);
cout<<”swap metodundan sonra”<<endl;
cout<<”str1= “<<str1<<endl;
cout<<”str2= “<<str2<<endl;
```

Çıktı:

```
str1= Ankara
str2= Antalya
swap metodundan sonra
str1= Antalya
str2= Ankara
```

Diğer String Metotları

Bu bölümde yukarıda anlatılan ana gruplara girmeyen diğer string metotlarına yer verilmiştir. Hem find hem de rfind metotları bir string değişkeni içerisinde başka bir string nesnesinin olup olmadığını arar. Find metodu arama işlemine string değişkenin başından başlarken rfind metodu sondan başlar. Tek bir karakter bir string nesnesi içerinde aranacak ise find_first_of veya find_last_of metotları kullanılmalıdır. Bazı durumlarda bir string nesnesinin bir bölümü elde edilmek istenir. Bazen daha büyük bir string nesnesini bölerek daha küçük alt nesneler oluşturulması gerekebilir. Bu gibi durumlarda substr metodu tercih edilir. String nesnelerin alfabetik olarak sıralanması için karşılaştırılması en sık

ihtiyaç duyulan yöntemlerden biridir. Compare metodu iki string değişkenini karşılaştırarak üç farklı sonuç değeri döndürür. Eğer dönen değer sıfır ise bu değişkenlerin aynı karakterlerden oluştuğu anlaşılır.

find Metodu

Bir string değişkeni içerisinde parametre olarak gönderilen karakter dizisi var ise bulunduğu ilk konum indeksini döndürür. Bulunmaması durumunda string değişkenin uzunluğundan daha büyük bir değer döndürür. Parametre olarak ya bir karakter ya da başka bir string nesnesi alabilir.

Örnek:

```
string str1 = “Merhaba Ankara!”;
string str2 = “kara”;
cout<<str1.find(str2)<<endl;
cout<<str1.find(“Anka”)<<endl;
cout<<str1.find(“Kara”)<<endl;
```

Çıktı:

```
10
8
18446744073709551615
```

rfind Metodu

Bir string değişkeni içerisinde parametre olarak gönderilen karakter dizisi var ise bulunduğu en son konum indeksini döndürür. Bulunmaması durumunda string değişkenin uzunluğundan daha büyük bir değer döndürür.

Örnek:

```
string str1 = “Merhaba Ankara Ankara!”;
string str2 = “kara”;
cout<<str1.rfind(str2)<<endl;
cout<<str1.rfind(“Anka”)<<endl;
cout<<str1.rfind(“Kara”)<<endl;
```

Çıktı:

```
17
15
18446744073709551615
```

find_first_of Metodu

Bir string değişkeni içerisinde parametre olarak aldığı karakterin bulunduğu ilk indeks değerini döndürür.

Örnek:

```
string str1 = “Merhaba Ankara”;
cout<<str1.find_first_of(‘a’)<<endl;
```

Çıktı:

```
4
```

find_last_of Metodu

Bir string değişkeni içerisinde parametre olarak aldığı karakterin bulunduğu en son indeks değerini döndürür.

Örnek:

```
string str1 = “Merhaba Ankara”;
cout<<str1.find_last_of(‘a’)<<endl;
```

Çıktı:

```
13
```

substr Metodu

substr metodu çağrıldığı string değişkeninkinden parametre olarak aldığı başlangıç ve uzunluk değerini kullanarak bir alt parçasını oluşturur. Metodun aldığı ilk parametre başlangıç indeksini ve ikinci parametre uzunluğu ifade eder. Eğer uzunluk parametresi verilmez ise başlangıç indeksinden değişkenin sonuna kadar olan parça alınır.

Örnek:

```
string str1 = “Merhaba Ankara”;
string str2 = str1.substr(8, 6);
string str3 = str1.substr(8);
cout<<str2;
cout<<str3;
```

Çıktı:

```
Ankara
Ankara
```

DİKKAT EDELİM

Başlangıç parametresi için negatif bir değer göndermek çalışma zamanı hatasına sebep olacaktır. Eğer başlangıç parametresi için string nesnesinin uzunluğundan daha büyük bir değer girilmesi durumunda metot boş string döndürecektir.

compare Metodu

String nesnesi ile parametre değerini karşılaştırır. Eğer her ikisi aynı değere sahip ise sıfır, nesne parametre değerinden sözlükte daha önce ise negatif sonra ise pozitif değer döndürür. Burada dikkat edilmesi gereken husus, karşılaştırma işlemi ASCII tablosundaki karakterlerin diziliş sırasına göre yapılmaktadır. ‘A’ karakterinin değeri 65 iken ‘a’ karakterinin değeri 97’dir. Bu durumda “Armut” kelimesi “armut” kelimesinden önce gelmektedir. Benzer şekilde “Elma” kelimesi “armut” kelimesinden öncedir. Karşılaştırma işlemi ‘<’, ‘<=’, ‘<’, ve ‘>=’ işleçleri kullanılarak da gerçekleştirilebilir.

Örnek:

```
string str1 = “abc”;
string str2 = “abc”;
string str3 = “bcd”;
cout<<str1.compare(str2)<<endl;
cout<<str1.compare(str3)<<endl;
cout<<str1.compare(“aaa”)<<endl;
```

Çıktı:

```
0
-65793
258
```

DİKKAT EDELİM

Eğer karşılaştırılan iki string değişkeni birbirinden farklıysa, programın her çalıştırılmasında farklı negatif ve pozitif değerler görmek mümkündür.

BÖLÜM ÖZETİ

Bu bölümde, C++ programlama dilinin standart string kütüphanesi detaylı bir şekilde ele alınmıştır. **String** veri türü, temel veri türlerinden biri olmayıp, metin tabanlı verileri saklamak ve işlemek amacıyla tanımlanmıştır. C++ dilinde string oluşturmanın iki temel yöntemi bulunmaktadır.

Birinci yöntem, **karakter dizileri** kullanılarak oluşturulan ve "C-tarzı string" olarak adlandırılan yapılardır. Bu tür string yapısı, önceki bölümlerde incelenmiştir. İkinci yöntem ise, C++ standart kütüphanesinde tanımlı **string sınıfı** kullanılarak string oluşturulmasıdır. Bu yaklaşım, daha güvenilir ve verimli bellek yönetimi sağladığından, modern C++ uygulamalarında yaygın olarak tercih edilmektedir.

Bölümde öncelikle bir string nesnesinin nasıl tanımlanacağı ve bu nesnelere değer atanmasının farklı yolları ele alınmıştır. String kütüphanesindeki metotlar, işlevselliklerine göre üç ana kategori altında incelenmiştir: **kapasite metotları**, **erişim metotları** ve **düzenleyici metotlar**. Bu kategorilere girmeyen işlevler ise "diğer metotlar" başlığı altında açıklanmıştır.

String nesnelerinin tanımlanması ve farklı değer atama yöntemlerinin yanı sıra, bir string nesnesinin uzunluk gibi özelliklerine erişim sağlama yöntemleri ele alınmıştır. Ardından, string nesnelerinin nasıl birleştirileceği, karşılaştırılacağı ve değerlerinin nasıl değiştirileceği üzerinde durulmuştur. Son olarak, bir string içerisinde belirli bir örüntünün nasıl aranacağına dair yöntemler detaylandırılmıştır.

Bu bölüm, C++ string sınıfının sunduğu özellikleri ve bu özelliklerin nasıl kullanılacağını anlamak için kapsamlı bir kaynak niteliğindedir.

GÖZDEN GEÇİRELİM

1. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string txt = "Merhaba Ankara!";

   cout << txt.size();

   a. 12

   b. 13

   c. 14

   d. 15

   e. 16

2. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string txt = "Merhaba Ankara!";

   txt.erase(7, 7);

   cout << txt;

   a. Merhaba

   b. Merhaba!

   c. Ankara

   d. Ankara!

   e.

3. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string str1 ="3";

   string str2="3";

   string str3 = str1 + str2;

   cout << str3;

   a. 3

   b. 6

   c. 33

   d. "3""3"

   e. "33"

4. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string str ("Merhaba Ankara!");

   str.pop_back();

   cout << str;

   a. Merhaba

   b. Merhaba Ankara

   c. Ankara

   d. Ankara!

   e. !

5. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string str ("Merhaba Ankara!");

   str.replace(10, 5, "talya");

   cout << str;

   a. Merhaba Antalya

   b. Merhaba Antalya!

   c. Merhaba talya!

   d. Merhaba Ankaratalya!

   e. Merhaba Ankaratalya

6. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string str1 ("Merhaba Ankara!");

   string str2 ("Antalya");

   str1.swap(str2);

   cout << str1;

   a. Merhaba Antalya!

   b. Merhaba Ankara!

   c. Merhaba Ankara Antalya!

   d. Antalya

   e. Antalya!

7. **Verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string txt = "Merhaba Ankara!";

   cout << txt.length();

   a. 2
   b. 3
   c. 14
   d. 15
   e. 16

8. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string txt = "Merhaba Ankara!";

   cout << txt.find_first_of('a');

   a. 3
   b. 4
   c. 5
   d. 6
   e. 7

9. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

   string txt = "Merhaba Ankara!";

   cout << txt.find_last_of('a');

   a. 10
   b. 11
   c. 12
   d. 13
   e. 14

10. **Aşağıda verilen kod parçasının ekran çıktısı aşağıdakilerden hangisidir?**

    string str1 ("Merhaba Ankara!");

    string str2 = str1.substr(8,7);

    cout << str2;

    a. Merhaba
    b. Ankara
    c. Merhaba!
    d. kara!
    e. Ankara!

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

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. Erzurum: Atatürk Üniversitesi Açıköğretim Fakültesi.

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