C++ String Fonksiyonları Kılavuzu

Kapsamlı Referans Dökümanı

20 Eylül 2025

İçindekiler

1 Giriş

C++ dilinde string işlemleri için iki ana yaklaşım bulunmaktadır:

- `std::string` sınıfı (C++ tarzı)
- C-tarzı string fonksiyonları (`<cstring>` kütüphanesi)

Bu kılavuz, her iki yaklaşımın da detaylı kullanımını kapsamaktadır.

2 std::string Sınıfı

2.1 Temel Yapıcılar (Constructors)

```cpp
#include <iostream>
#include <string>

int main() {
    // Bo  string
    std::string str1;

    // Literal ile ba latma
    std::string str2("Merhaba D nya");
    std::string str3 = "C++ String";

    // Karakter tekrar
    std::string str4(10, 'A');  // "AAAAAAAAAA"

    // Kopyalama
    std::string str5(str2);
    std::string str6 = str3;

    // Alt string ile ba latma
    std::string str7(str2, 0, 7);  // "Merhaba"

    return 0;
}
```

Listing 1: String Yapıcı Örnekleri

2.2 Temel String Fonksiyonları

2.2.1 Uzunluk ve Boyut

```cpp
std::string str = "Programlama";

// Uzunluk alma
size_t len1 = str.length();     // 11
size_t len2 = str.size();       // 11 (length() ile ayn )

// Kapasite bilgisi
size_t cap = str.capacity();    // Ayr lan bellek boyutu
size_t max_size = str.max_size(); // Maksimum boyut

// Bo  kontrol
```

```cpp
bool isEmpty = str.empty();      // false
```

Listing 2: Uzunluk Fonksiyonları

2.2.2 Karakter Erişimi

```cpp
std::string str = "C++ Guide";

//  ndeks  operat r  (bounds checking yok)
char ch1 = str[0];               // 'C'
char ch2 = str[str.length()-1]; // 'e'

// at() fonksiyonu (bounds checking var)
char ch3 = str.at(0);            // 'C'
// char ch4 = str.at(100);       // std::out_of_range exception

//  lk  ve son karakter
char first = str.front();        // 'C'
char last = str.back();          // 'e'

// C-tarz  string'e d  n  t  rme
const char* cstr = str.c_str(); // Null-terminated
const char* data_ptr = str.data(); // C++11'den itibaren null-terminated
```

Listing 3: Karakter Erişim Yöntemleri

2.3 String Değişiklikleri

2.3.1 Ekleme İşlemleri

```cpp
std::string str = "Merhaba";

// Sona ekleme
str += " D nya";                    // "Merhaba D nya"
str.append("!");                   // "Merhaba D nya!"
str.push_back('?');                // "Merhaba D nya!?"

// Belirli pozisyona ekleme
str.insert(8, "G zel ");            // "Merhaba G zel  D nya!?"
str.insert(0, "Selam ");           // "Selam Merhaba G zel  D nya!?"

// Karakter tekrar  ile ekleme
str.append(3, '!');                // "Selam Merhaba G zel  D nya
   !?!!!"
```

Listing 4: String Ekleme Fonksiyonları

2.3.2 Silme İşlemleri

```cpp
std::string str = "Merhaba D nya Programlama";

// Son karakter silme
str.pop_back();                    // "Merhaba D nya Programlam"

// Belirli pozisyondan silme
```

```
str.erase(8, 6);                    // "Merhaba Programlam" (8. pozisyondan 6 karakter)
str.erase(15);                      // "Merhaba Program" (15. pozisyondan sonras n  sil)

// Tamam n  temizleme
str.clear();                        // ""
bool isEmpty = str.empty();         // true
```

Listing 5: String Silme Fonksiyonları

2.3.3 Değiştirme İşlemleri

```
std::string str = "C++ Programlama Dili";

// Belirli b l m  de i tirme
str.replace(0, 3, "Java");          // "Java Programlama Dili"
str.replace(5, 11, "Geli tirme");   // "Java Geli tirme Dili"

// Karakter ile de i tirme
str.replace(0, 4, 10, 'X');         // "XXXXXXXXXX Geli tirme Dili"
```

Listing 6: String Değiştirme Fonksiyonları

2.4 String Arama Fonksiyonları

```
std::string text = "C++ ile programlama   reniyorum  . C++ g   l   bir dil.";

//  lk  bulma
size_t pos1 = text.find("C++");             // 0
size_t pos2 = text.find("programlama");     // 8
size_t pos3 = text.find("Python");          // std::string::npos

// Son bulma
size_t last_pos = text.rfind("C++");        // 31

// Belirli pozisyondan sonra arama
size_t pos4 = text.find("C++", 1);          // 31

// Karakter arama
size_t char_pos = text.find('i');           // 5

//  lk  karakter setinden birini bulma
size_t vowel_pos = text.find_first_of("aeiouAEIOU"); // 9 ('i' karakteri)

// Son karakter setinden birini bulma
size_t last_vowel = text.find_last_of("aeiouAEIOU");  // son sesli harf

// Karakter setinde olmayan ilk karakter
size_t not_alpha = text.find_first_not_of("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ ");
```

Listing 7: String Arama İşlemleri

2.5 Alt String İşlemleri

```
std::string sentence = "C++ programlama dili    reniyorum   ";

// Alt string alma
std::string lang = sentence.substr(0, 3);         // "C++"
std::string verb = sentence.substr(20, 11);       // "   reniyorum   "
std::string from_pos = sentence.substr(4);        // "programlama dili
      reniyorum   "

// String kar  la t rma
std::string str1 = "Apple";
std::string str2 = "Banana";
std::string str3 = "Apple";

int cmp1 = str1.compare(str2);    // Negatif de er (str1 < str2)
int cmp2 = str1.compare(str3);    // 0 (e it)
int cmp3 = str2.compare(str1);    // Pozitif de er (str2 > str1)

// B l msel kar  la t rma
int partial_cmp = str1.compare(0, 2, str3, 0, 2); //  lk  2 karakteri
   kar  la t r
```

Listing 8: Alt String (Substring) İşlemleri

3 String Dönüşüm Fonksiyonları

3.1 Sayı Dönüşümleri

```
#include <string>

// String'den say ya d n     m
std::string number_str = "12345";
std::string float_str = "123.45";
std::string hex_str = "0xFF";

int num = std::stoi(number_str);           // 12345
long long_num = std::stol(number_str);     // 12345L
float f_num = std::stof(float_str);        // 123.45f
double d_num = std::stod(float_str);       // 123.45

// Hex string'den d n     m
int hex_num = std::stoi(hex_str, nullptr, 16); // 255

// Say dan string'e d n     m
int value = 42;
double pi = 3.14159;

std::string int_to_str = std::to_string(value);    // "42"
std::string double_to_str = std::to_string(pi);    // "3.141590"
```

Listing 9: String-Sayı Dönüşümleri

4 C-Tarzı String Fonksiyonları

4.1 Temel Fonksiyonlar (cstring)

```cpp
#include <cstring>
#include <iostream>

int main() {
    char str1[50] = "Merhaba";
    char str2[50] = "D nya";
    char str3[100];

    // Uzunluk hesaplama
    size_t len = strlen(str1);              // 7

    // Kopyalama
    strcpy(str3, str1);                     // str3 = "Merhaba"
    strcat(str3, " ");                      // str3 = "Merhaba "
    strcat(str3, str2);                     // str3 = "Merhaba D nya"

    // G venli kopyalama (C11)
    strncpy(str3, str1, sizeof(str3) - 1);
    str3[sizeof(str3) - 1] = '\0';

    // Kar  la t rma
    int cmp = strcmp(str1, str2);           // Negatif (str1 < str2)
    int cmp_n = strncmp(str1, "Mer", 3);    // 0 (ilk 3 karakter e it)

    // Arama
    char* found = strstr(str3, "D nya");     // "D nya" pointer'
    char* char_found = strchr(str1, 'h');   // 'h' karakterinin pointer'

    char* last_found = strrchr(str1, 'a');  // Son 'a' karakterinin
   pointer'

    return 0;
}
```

Listing 10: C-Tarzı String Fonksiyonları

5 İleri Düzey String İşlemleri

5.1 String İteratörler

```cpp
#include <string>
#include <algorithm>
#include <iostream>

std::string text = "C++ Programming";

// Forward iterat r
for (auto it = text.begin(); it != text.end(); ++it) {
    std::cout << *it << " ";
}

// Reverse iterat r
```

```cpp
for (auto rit = text.rbegin(); rit != text.rend(); ++rit) {
    std::cout << *rit;
}

// Range-based for loop (C++11)
for (const char& c : text) {
    std::cout << c;
}

// Algoritma kullan m
std::transform(text.begin(), text.end(), text.begin(), ::toupper);
// text  imdi  "C++ PROGRAMMING"
```

Listing 11: String İteratörler

5.2 String Algoritmaları

```cpp
#include <string>
#include <algorithm>
#include <cctype>

std::string str = "  C++ Programming  ";

// Trim fonksiyonu (sol bo luklar  sil)
auto ltrim = [](std::string &s) {
    s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](unsigned char
    ch) {
        return !std::isspace(ch);
    }));
};

// Trim fonksiyonu (sa  bo luklar  sil)
auto rtrim = [](std::string &s) {
    s.erase(std::find_if(s.rbegin(), s.rend(), [](unsigned char ch) {
        return !std::isspace(ch);
    }).base(), s.end());
};

// Her iki yandan trim
auto trim = [&](std::string &s) {
    ltrim(s);
    rtrim(s);
};

trim(str); // str = "C++ Programming"

// T m harfleri b y k yapma
std::transform(str.begin(), str.end(), str.begin(), ::toupper);

// T m harfleri k     k yapma
std::transform(str.begin(), str.end(), str.begin(), ::tolower);
```

Listing 12: String Algoritmaları

6 String Performance ve Bellek Yönetimi

6.1 Kapasile Yönetimi

```
std::string str;

// Bellek  n  ay rma
str.reserve(1000);  // 1000 karakter i in bellek ay r

std::cout << "Kapasite: " << str.capacity() << std::endl;
std::cout << "Boyut: " << str.size() << std::endl;

// String b y tme
for (int i = 0; i < 500; ++i) {
    str += 'x';
}

// Fazla belle i serbest b rakma
str.shrink_to_fit();

// Boyutu de i tirme
str.resize(100);        // 100 karaktere k salt
str.resize(200, 'A');   // 200 karaktere uzat, yeni karakterler 'A'
```

Listing 13: String Kapasite Optimizasyonu

7 String Utilities ve Helper Functions

```
#include <string>
#include <vector>
#include <sstream>

// String split fonksiyonu
std::vector<std::string> split(const std::string& str, char delimiter) {
    std::vector<std::string> tokens;
    std::stringstream ss(str);
    std::string token;

    while (std::getline(ss, token, delimiter)) {
        tokens.push_back(token);
    }
    return tokens;
}

// String replace all fonksiyonu
std::string replace_all(std::string str, const std::string& from,
                        const std::string& to) {
    size_t start_pos = 0;
    while ((start_pos = str.find(from, start_pos)) != std::string::npos)
    {
        str.replace(start_pos, from.length(), to);
        start_pos += to.length();
    }
    return str;
}

```

```cpp
// Kullan m    rnei
int main() {
    std::string sentence = "C++,Java,Python,JavaScript";
    auto languages = split(sentence, ',');

    std::string text = "Hello World Hello";
    std::string new_text = replace_all(text, "Hello", "Hi");
    // new_text = "Hi World Hi"

    return 0;
}
```

Listing 14: Faydalı String Yardımcı Fonksiyonları

8 En İyi Uygulamalar

8.1 Performance İpuçları

1. **Reserve kullanın**: Eğer string'in yaklaşık boyutunu biliyorsanız, `reserve()` kullanarak bellek realokasyonunu önleyin.

2. **const referans kullanın**: Fonksiyon parametrelerinde `const std::string&` kullanın.

3. **Move semantics**: C++11 move semantics'ini kullanarak gereksiz kopyaları önleyin.

4. **String literals**: C++14 string literals (`""s`) kullanın.

```cpp
#include <string>
using namespace std::string_literals;

//  yi : const referans kullan m
void process_string(const std::string& str) {
    // String kopyalanmaz
}

//  yi : Move semantics
std::string create_string() {
    std::string result = "Hello World"s;
    return result;  // Move semantics otomatik
}

//  yi : Reserve kullan m
std::string build_large_string() {
    std::string result;
    result.reserve(10000);  // Bellek  n  ay rma

    for (int i = 0; i < 1000; ++i) {
        result += "Some text ";
    }
    return result;
}
```

Listing 15: Performance Optimizasyonları

9 Özet Tablosu

İşlem	**std::string**	**C-tarzı**
Uzunluk	`str.length(), str.size()`	`strlen()`
Kopyalama	`str1 = str2`	`strcpy()`
Ekleme	`str1 += str2`	`strcat()`
Karşılaştırma	`str1.compare(str2)`	`strcmp()`
Arama	`str.find()`	`strstr()`
Alt string	`str.substr()`	Manuel pointer aritmetiği

Tablo 1: String Fonksiyonları Karşılaştırması

10 Sonuç

C++ string işlemleri için `std::string` sınıfı genellikle tercih edilmelidir çünkü:

- Bellek yönetimi otomatik
- Buffer overflow koruması
- Daha zengin fonksiyon seti
- STL algoritmaları ile uyumlu
- Exception safety

C-tarzı string fonksiyonları yalnızca legacy kod ile çalışırken veya performans kritik uygulamalarda dikkatli kullanılmalıdır.