

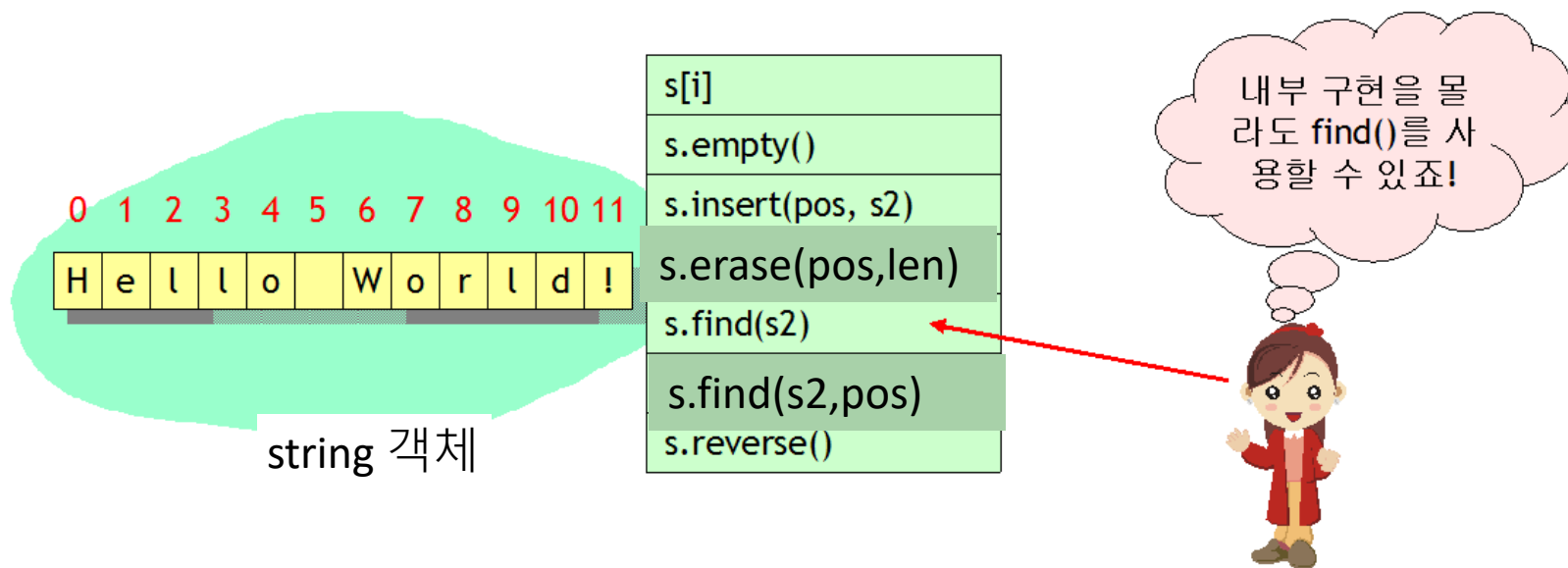
c with classes : C++ a string class example

국민대학교 임은진

클래스 사용의 예 : string 클래스

- C++에서는 문자열을 나타내는 클래스 string을 제공한다.

#include <string>



string class in <string>

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main(){
6      char s1[10], s2[10] = "xxx";
7
8      // s1 = "12";
9      strcpy(s1, "12");
10     cout << "length of " << s1 << " is "
11     // s2 = s1 + "ab";
12     strcpy(s2, s1);
13     strcat(s2, "ab");
14     cout << "length of " << s2 << " is "
15 }
```

length of 12 is 2
length of 12ab is 4

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(){
6      string s1, s2 = "xxx";
7
8      s1 = "12";
9      cout << "length of " << s1 << " is " << s1.length() << endl;
10
11     s2 = s1 + "ab";
12     cout << "length of " << s2 << " is " << s2.size() << endl;
13 }
```

Reference

+

 C library:

+

 Containers:

+

 Input/Output:

+

 Multi-threading:

-

 Other:

<algorithm>

<bitset>

<chrono>

<codecvt>

<complex>

<exception>

<functional>

<initializer_list>

<iterator>

<limits>

<locale>

<memory>

<new>

<numeric>

<random>

<ratio>

<regex>

<stdexcept>

<string>

<system_error>

<tuple>

<typeindex>

<typeinfo>

<type_traits>

<utility>

<valarray>

<string>

class

std::string

<string>

```
typedef basic_string<char> string;
```

String class

Strings are objects that represent sequences of characters.

The standard `string` class provides support for such objects with an interface similar to that of a [standard container](#) of bytes, but adding features specifically designed to operate with strings of single-byte characters.

The `string` class is an instantiation of the [basic_string](#) class template that uses `char` (i.e., bytes) as its *character type*, with its default [char_traits](#) and [allocator](#) types (see [basic_string](#) for more info on the template).

Note that this class handles bytes independently of the encoding used: If used to handle sequences of multi-byte or variable-length characters (such as UTF-8), all members of this class (such as [length](#) or [size](#)), as well as its iterators, will still operate in terms of bytes (not actual encoded characters).

Member types

member type	definition
<code>value_type</code>	<code>char</code>
<code>traits_type</code>	char_traits < <code>char</code> >
<code>allocator_type</code>	allocator < <code>char</code> >
<code>reference</code>	<code>char&</code>
<code>const_reference</code>	<code>const char&</code>
<code>pointer</code>	<code>char*</code>
<code>const_pointer</code>	<code>const char*</code>
<code>iterator</code>	a random access iterator to <code>char</code> (convertible to <code>const_iterator</code>)
<code>const_iterator</code>	a random access iterator to <code>const char</code>
<code>reverse_iterator</code>	reverse_iterator < <code>iterator</code> >
<code>const_reverse_iterator</code>	reverse_iterator < <code>const_iterator</code> >
<code>difference_type</code>	ptrdiff_t
<code>size_type</code>	size_t

fx Member functions

(constructor)	Construct string object (public member function)
(destructor)	String destructor (public member function)
operator=	String assignment (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin <small>C++11</small>	Return const_iterator to beginning (public member function)
cend <small>C++11</small>	Return const_iterator to end (public member function)
crbegin <small>C++11</small>	Return const_reverse_iterator to reverse beginning (public member function)
crend <small>C++11</small>	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return length of string (public member function)
length	Return length of string (public member function)
max_size	Return maximum size of string (public member function)
resize	Resize string (public member function)
capacity	Return size of allocated storage (public member function)
reserve	Request a change in capacity (public member function)
clear	Clear string (public member function)
empty	Test if string is empty (public member function)
shrink_to_fit <small>C++11</small>	Shrink to fit (public member function)

Element access:

operator[]	Get character of string (public member function)
at	Get character in string (public member function)
back <small>C++11</small>	Access last character (public member function)
front <small>C++11</small>	Access first character (public member function)

Modifiers:

operator+=	Append to string (public member function)
append	Append to string (public member function)
push_back	Append character to string (public member function)
assign	Assign content to string (public member function)
insert	Insert into string (public member function)
erase	Erase characters from string (public member function)
replace	Replace portion of string (public member function)
swap	Swap string values (public member function)
pop_back <small>C++11</small>	Delete last character (public member function)

String operations:

c_str	Get C string equivalent (public member function)
data	Get string data (public member function)
get_allocator	Get allocator (public member function)
copy	Copy sequence of characters from string (public member function)
find	Find content in string (public member function)
rfind	Find last occurrence of content in string (public member function)
find_first_of	Find character in string (public member function)
find_last_of	Find character in string from the end (public member function)
find_first_not_of	Find absence of character in string (public member function)
find_last_not_of	Find non-matching character in string from the end (public member function)
substr	Generate substring (public member function)
compare	Compare strings (public member function)

fx Member constants

npos	Maximum value for size_t (public static member constant)
-------------	---

fx Non-member function overloads

operator+	Concatenate strings (function)
relational operators	Relational operators for string (function)
swap	Exchanges the values of two strings (function)
operator>>	Extract string from stream (function)
operator<<	Insert string into stream (function)
getline	Get line from stream into string (function)

String

여기서 사용

[이러한 함수들이

구현되어 있음)

string 클래스의 멤버 함수 중 일부

멤버 함수	설명
s[i]	i번째 원소
s.size()	문자열의 길이를 반환 (null 문자(=‘\0’) 제외)
s.clear()	문자열을 지우고 null 문자열(=“”)로 만들
s.empty()	s가 비어있으면 true 반환
s.insert(pos, s2)	s의 pos 위치에 문자열 s2를 삽입
s.erase(pos, len)	s의 pos 위치부터 len만큼의 문자들을 삭제 (424)
s.find(s2)	s에서 문자열 (혹은 문자)s2가 발견되는 첫번째 <u>인덱스를 반환</u>
s.find(s2, pos)	s의 pos 위치부터 문자열 (혹은 문자) s2가 발견되는 첫번째 <u>인덱스를 반환</u>

찾는 문자열(혹은 문자)가 없으면 string::npos (= -1) 을 반환

주의점 : 인자 개수 차이, pos를 default parameter를 줌


```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main(){
6
7     string s1;
8     string s2("I");
9     string s3 = s2;
10    string s4 = "you";
11
12    cout << "s1 : \"\" << s1 << "\" size \" << s1.size() << endl;
13    cout << "s2 : \"I\" << s2 << "\" size \" << s2.size() << endl;
14    cout << "s3 : \"I\" << s3 << "\" size \" << s3.size() << endl;
15    cout << "s4 : \"you\" << s4 << "\" size \" << s4.size() << endl;
16
17    s1 = s2 + " and " + s4;
18    cout << "s1 : \"I and you\" << s1 << "\" size \" << s1.size() << endl;
19    s1.clear();
20    cout << "s1 : \"\" << s1 << "\" size \" << s1.size() << endl;
21    cout << "is s1 empty? \" << s1.empty() << endl;
22
23    s4[0] = 'Y';
24    cout << "s4 : \"You\" << s4 << "\" size \" << s4.size() << endl;

```

string은 local 변수로 메모리 관리됨

문자 배열 (= string)

문자 배열 상수

escape sequence

문자 배열 상수

character reference type 이

```

s1 : "" size 0
s2 : "I" size 1
s3 : "I" size 1
s4 : "you" size 3
s1 : "I and you" size 9
s1 : "" size 0
is s1 empty? 1
s4 : "You" size 3

```

Reference

- [C library:](#)
- [Containers:](#)
- [Input/Output:](#)
- [Multi-threading:](#)
- [Other:](#)

<string>

- [class templates:](#)
- [classes:](#)
- [functions:](#)

string

string::string
string::~string

member functions:

- string::append
- string::assign
- string::at
- string::back
- string::begin
- string::capacity
- string::cbegin
- string::cend
- string::clear
- string::compare
- string::copy
- string::crbegin
- string::crend
- string::c_str
- string::data
- string::empty
- string::end
- string::erase
- string::find
- string::find_first_not_of
- string::find_first_of
- string::find_last_not_of
- string::find_last_of
- string::front
- string::get_allocator
- string::insert
- string::length
- string::max_size
- string::operator+=
- string::operator=
- string::operator[]

public member function

std::string::find

<string>

C++98

C++11

?

```
string (1) size_t find (const string& str, size_t pos = 0) const;
c-string (2) size_t find (const char* s, size_t pos = 0) const;
buffer (3) size_t find (const char* s, size_t pos, size_t n) const;
character (4) size_t find (char c, size_t pos = 0) const;
```

①
②
③
④

Find content in string

Searches the [string](#) for the first occurrence of the sequence specified by its arguments.

When *pos* is specified, the search only includes characters at or after position *pos*, ignoring any possible occurrences that include characters before *pos*.

Notice that unlike member [find_first_of](#), whenever more than one character is being searched for, it is not enough that just one of these characters match, but the entire sequence must match.

Parameters

str

Another [string](#) with the subject to search for.

pos

Position of the first character in the string to be considered in the search.

If this is greater than the [string length](#), the function never finds matches.

Note: The first character is denoted by a value of 0 (not 1): A value of 0 means that the entire string is searched.

s

Pointer to an array of characters.

If argument *n* is specified (3), the sequence to match are the first *n* characters in the array.

Otherwise (2), a null-terminated sequence is expected: the length of the sequence to match is determined by the first occurrence of a null character.

n

Length of sequence of characters to match.

c

Individual character to be searched for.

[size_t](#) is an unsigned integral type (the same as member type [string::size_type](#)).

Return Value

The position of the first character of the first match.

If no matches were found, the function returns [string::npos](#).

[size_t](#) is an unsigned integral type (the same as member type [string::size_type](#)).

```

25 s1 = "abcdefghijklmnopqrstuvwxyz";
26 int i = s1.find('c'); => 2인 페이지의 4번째 함수 사용됨!
27 cout << "c is the " << i << "-th alphabet.\n";
28 i = s1.find('c', i+1);
29 cout << "the next c is the " << i << "-th alphabet.\n";
30 i = s1.find("efg"); 문자 배열 함수 = char pointer (string 이 아님)
31 cout << "efg starts at " << i << "-th alphabet.\n";
32 cout << s2 << " < " << s4 << " : " << (s2 < s4) << endl;
33 cout << s2 << " == " << s4 << " : " << (s2 == s4) << endl;
34
35 cout << "write a word to insert : ";
36 cin >> s3; => 타입이 string
37 s1.insert(0, s3);
38 cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
39 s1.erase(1, 3);
40 cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
41
42 return 0;
43 }

```

String에 대해 '<'
가 구현되어 있음
(사전순으로 앞에 있는지를 비교)

↓
문자열의 내용 같았는지 비교

같은 길이 앞부분
↑ return 되는 것

```

c is the 2-th alphabet.
the next c is the -1-th alphabet.
efg starts at 4-th alphabet.
I < You : 1
I == You : 0
write a word to insert : cprogram
s1 : "cprogramabcdefghijklmnopqrstuvwxyz" size 34
s1 : "cgramabcdefghijklmnopqrstuvwxyz" size 31

```

string::npos (= -1)

example : wheel of fortune

- 빈칸으로 구성된 문자열이 주어지고 사용자는 문자열에 들어갈 글자들을 하나씩 추측해서 맞추는 게임이다.




```
1  #include <iostream>
2  #include <string>
3  #include <cstdlib>
4  #include <ctime>
5  using namespace std;
6
7  #define DICT_LEN 3
8  #define MAX_TRIES 6
9
10 int main(){
11
12     string dictionary[] = {"space", "wheel", "program"};
13     srand(time(NULL));
14     string prob = dictionary[rand()%DICT_LEN];
15     int length = prob.length();
16     int tries = 0;
17     string answer(length, '-');
18
19     cout << "current state : " << answer << endl;
```

String 배열 문들 7 있음
↗

current state : -----

```

20 while (tries < MAX_TRIES && answer != prob){
21     char c;
22     cout << "guess a letter : ";
23     cin >> c;
24     if (answer.find(c) != string::npos){
25         cout << c << " is guessed previously.\n";
26         continue;
27     }
28     int pos = prob.find(c);
29     if (pos == string::npos){
30         cout << c << " is not in the target string.\n";
31         tries++;
32         continue;
33     }
34     do {
35         answer[pos] = c;
36         pos = prob.find(c, pos+1);
37     } while (pos != string::npos);
38     cout << "current state : " << answer << endl;
39     if (answer == prob){
40         cout << "You WON!!!\n";
41         break;
42     }
43 }
44 if (tries >= MAX_TRIES){
45     cout << "You lost : the answer was " << prob << endl;
46 }

```

prob : "program"
 answer : "-----"

```

current state : -----
guess a letter : e
e is not in the target string.
guess a letter : p
current state : p-----
guess a letter : r
current state : pr---r--
guess a letter : m
current state : pr---r-m
guess a letter : o
current state : pro-r-m
guess a letter : g
current state : progr-m
guess a letter : a
current state : program
You WON!!!

```

```

20 while (tries < MAX_TRIES && answer != prob){
21     char c;
22     cout << "guess a letter : ";
23     cin >> c;
24     if (answer.find(c) != string::npos){
25         cout << c << " is guessed previously.\n";
26         continue;
27     }
28     int pos = prob.find(c);
29     if (pos == string::npos){
30         cout << c << " is not in the target string.\n";
31         tries++;
32         continue;
33     }
34     do {
35         answer[pos] = c;
36         pos = prob.find(c, pos+1);
37     } while (pos != string::npos);
38     cout << "current state : " << answer << endl;
39     if (answer == prob){
40         cout << "You WON!!!\n";
41         break;
42     }
43 }
44 if (tries >= MAX_TRIES){
45     cout << "You lost : the answer was " << prob << endl;
46 }

```

prob : "space"
answer : "-----"

```

current state : -----
guess a letter : r
r is not in the target string.
guess a letter : h
h is not in the target string.
guess a letter : p
current state : -p---
guess a letter : p
p is guessed previously.
guess a letter : a
current state : -pa--
guess a letter : e
current state : -pa-e
guess a letter : i
i is not in the target string.
guess a letter : o
o is not in the target string.
guess a letter : u
u is not in the target string.
guess a letter : t
t is not in the target string.
You lost : the answer was space

```