

프로그래밍 언어 – string

최백준 choi@startlink.io

string

string

STL

3

```
string s1;
```

```
char c[] = "c string";
```

```
string s2(c);
```

```
string s3 = c;
```

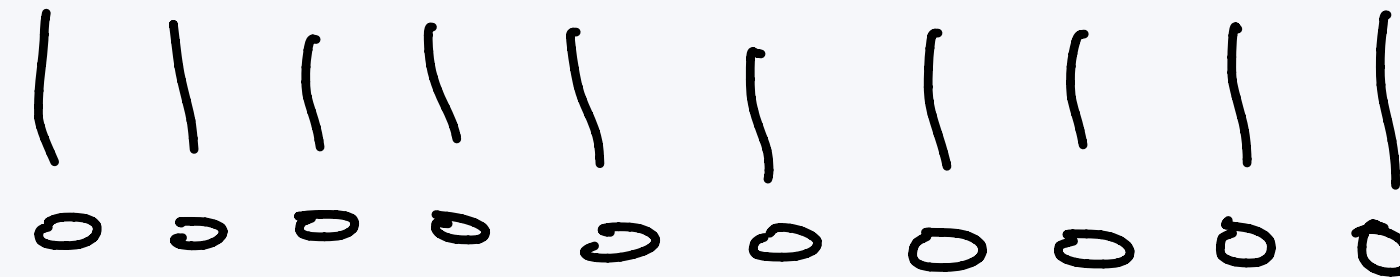
```
c[1] = '\\0';
```

```
string s4(c);
```

```
string s5 = c;
```

```
string s6(10, '!');
```

```
string s7 = "abcdefg";
```



string

STL

c string

c string

c

c

!!!!!!!!!!

abcdefg

string

STL

```
string s1, s2;  
cin >> s1 >> s2;  
cout << s1 << ' ' << s2;
```

단어의 개수

<https://www.acmicpc.net/problem/1152>

- <https://gist.github.com/Baekjoon/32d83ef9853706fb2386>

문자열 분석

<https://www.acmicpc.net/problem/10820>

- <https://gist.github.com/Baekjoon/228b31a4f2fb84e49f0a>

정수의 개수

<https://www.acmicpc.net/problem/10821>

- <https://gist.github.com/Baekjoon/dcd8751c1ff39204483a>
- getline의 세 번째 인자는 구분자

string

STL

```
string s1, s2;  
cin >> s1 >> s2;  
cout << s1 << ' ' << s2;
```

string

STL

10

```
string s = "abc";  
printf("%s\n",s.c_str());
```

```
s += "def";  
printf("%s\n",s.c_str());
```

```
s = "";  
printf("%s\n",s.c_str());
```

```
s = "abcdefghijklmnopqrstuvwxyz";  
printf("%s\n",s.c_str());
```

string

STL

11

```
string s = "book";  
cout << s << ": " << s.size() << '\n';  
cout << s << ": " << s.length() << '\n';
```

```
s = "";
```

0

```
cout << s << ": " << s.size() << '\n';  
cout << s << ": " << s.size()-1 << '\n';
```

-1

string

STL

book: 4

book: 4

: 0

: 18446744073709551615

단어 길이 재기

<https://www.acmicpc.net/problem/2743>

- <https://gist.github.com/Baekjoon/ca6054e85bad2a517002>

string

STL

14

```
string s = "book";  
cout << s << ": " << s.empty() << '\n';
```

```
s = "";  
cout << s << ": " << s.empty() << '\n';
```

string

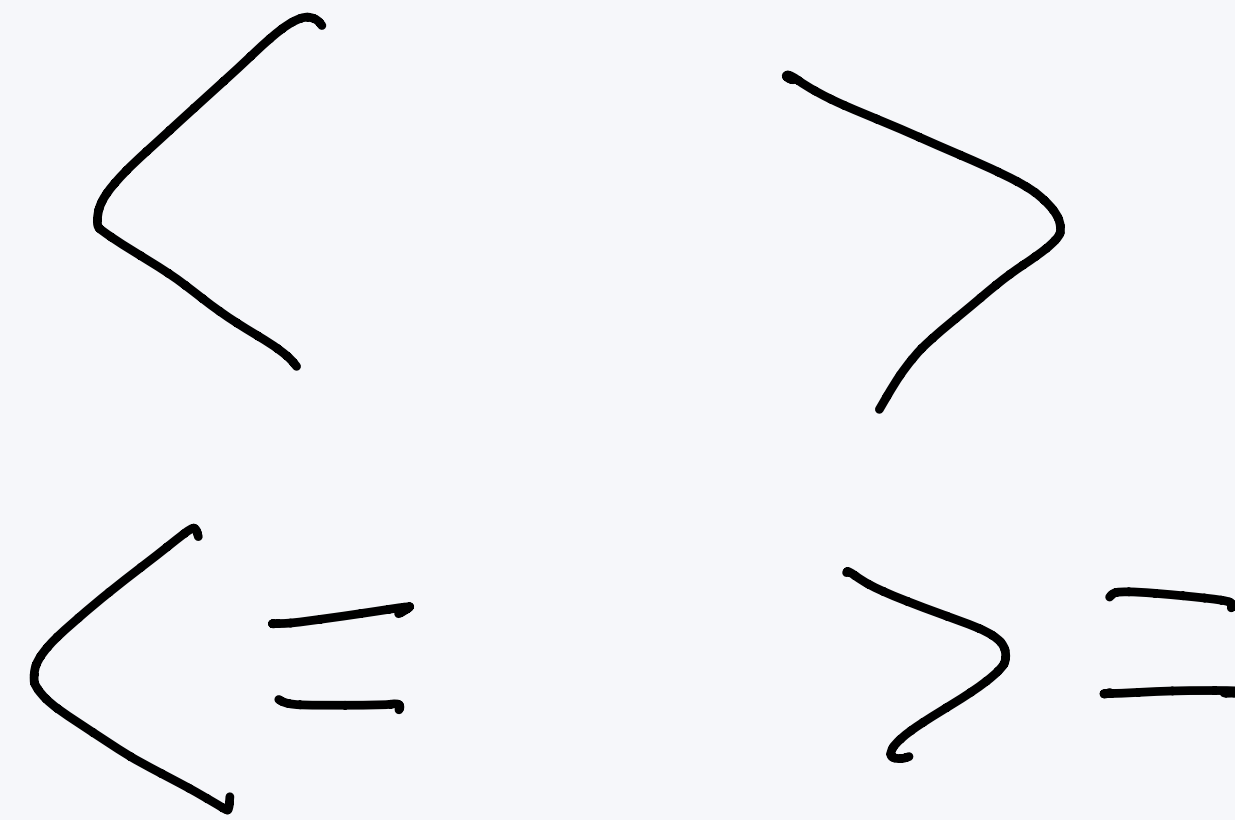
STL

```
string s1 = "string";
```

```
string s2 = "stirng";
```

```
if (s1 == s2) {  
    cout << "s1 == s2" << '\n';  
} else if (s1 != s2) {  
    cout << "s1 != s2" << '\n';  
}
```

```
if (s1 < s2) {  
    cout << "s1 < s2" << '\n';  
} else if (s1 > s2) {  
    cout << "s1 > s2" << '\n';  
}
```



string

STL

```
string a = "Hello";
```

```
string b = "World";
```

```
string hello_world = a + " " + b;
```

```
hello_world += "!";
```

```
cout << hello_world << '\n';
```

Hello World!

string

STL

17

```
string a = "Hello";
```

```
string b = "World";
```

```
string hello_world = a + " " + b;
```

```
hello_world.push_back('!');
```

```
cout << hello_world << "\n";
```

string

18

STL

```
string a = "He";  
a.append(2, 'l'); // "Hell"  
a.append("o").append(1, '_'); // "Hello_"
```

```
string b = "";  
const char *c = "World";  
b.append(c); // "World"
```

```
string hello_world = a; // "Hello "  
hello_world.append(b); // "Hello World"  
hello_world.push_back('!'); // "Hello World!"
```

```
cout << hello_world << '\n';
```

string

STL

```
string s = "e"; // "e"
```

```
s.insert(0, "H"); // "He"
```

```
s.insert(2, "o"); // "Heo"
```

```
s.insert(2, 2, 'l').append(" "); // "Hello "
```

```
string world = "Half the World Away";
```

```
s.insert(6, world, 9, 5).push_back('!'); // "Hello World!"
```

Handwritten diagram illustrating string construction:

Initial state: `s = "e"` (index 0: 'e')

Step 1: `s.insert(0, "H")` → `"He"` (index 0: 'H', index 1: 'e')

Step 2: `s.insert(2, "o")` → `"Heo"` (index 0: 'H', index 1: 'e', index 2: 'o')

Step 3: `s.insert(2, 2, 'l')` → `"Heo"` (index 0: 'H', index 1: 'e', index 2: 'o')

Step 4: `s.append(" ")` → `"Heo "` (index 0: 'H', index 1: 'e', index 2: 'o', index 3: ' ')

Handwritten diagram illustrating string construction:

Initial state: `world = "Half the World Away"` (index 0: 'H', index 1: 'a', index 2: 'l', index 3: 'f', index 4: ' ', index 5: 't', index 6: 'h', index 7: 'e', index 8: ' ', index 9: 'W', index 10: 'o', index 11: 'r', index 12: 'l', index 13: 'd', index 14: ' ', index 15: 'A', index 16: 'w', index 17: 'a', index 18: 'y')

Step 1: `s.insert(6, world, 9, 5)` → `"Hello World"` (index 0: 'H', index 1: 'e', index 2: 'l', index 3: 'l', index 4: 'o', index 5: ' ', index 6: 'W', index 7: 'o', index 8: 'r', index 9: 'l', index 10: 'd')

Step 2: `s.push_back('!')` → `"Hello World!"` (index 0: 'H', index 1: 'e', index 2: 'l', index 3: 'l', index 4: 'o', index 5: ' ', index 6: 'W', index 7: 'o', index 8: 'r', index 9: 'l', index 10: 'd', index 11: '!')

string

20

STL

```
string str = "10";  
int number = stoi(str);  
print(str, number);
```

```
number = stoi(str, 0, 2);  
print(str, number);
```

```
str = "ffff";  
number = stoi(str, 0, 16);  
print(str, number);
```

ffff

10000 - 1
4 3 2 1 0

$16^4 - 1 = 2^{16} - 1$

65535

string

STL

```
str = "21 Guns";  
number = stoi(str);  
print(str, number);
```

21

```
str = "3[.141592";  
number = stoi(str);  
print(str, number);
```

3

string

STL

22

```
/*  
str = "2147483648";  
number = stoi(str);  
print(str, number);
```

```
str = "hello";  
number = stoi(str);  
print(str, number);  
*/
```

더하기

<https://www.acmicpc.net/problem/10822>

- <https://gist.github.com/Baekjoon/c5a8983b405951c9e4a2>

더하기 2

<https://www.acmicpc.net/problem/10823>

- <https://gist.github.com/Baekjoon/308a504b631264363a20>
- string을 표준 입출력 처럼 사용하려면 stringstream 을 사용한다

Cin Cout

string

STL

- unsigned long: stoul
- unsigned long long: stoull
- float: stof
- double: stod
- long double: stold

long long
Stoll

string

STL

```
int n1 = 1;
```

```
int n2 = 2;
```

```
string s1 = to_string(n1);  
string s2 = to_string(n2);
```

"1"
"2"

```
cout << s1 + ' ' + s2 << '\n';  
           |      2
```

string

27

STL

```
long long l1 = 2147483647;  
long long l2 = 2147483647;
```

```
s1 = to_string(l1);
```

```
s2 = to_string(l2);
```

```
cout << s1 + ' ' + s2 << '\n';
```

2147483647 _ 2147483647

string

STL

28

```
double d = 3.141592;  
float f = 65358979.0;
```

```
s1 = to_string(d);  
s2 = to_string(f);
```

```
cout << s1 + ' ' + s2 << '\n';
```

네 수

<https://www.acmicpc.net/problem/10824>

- <https://gist.github.com/Baekjoon/88d1ecca80dd87b17dd2>