



string CLASS FUNCTIONS

An Introduction

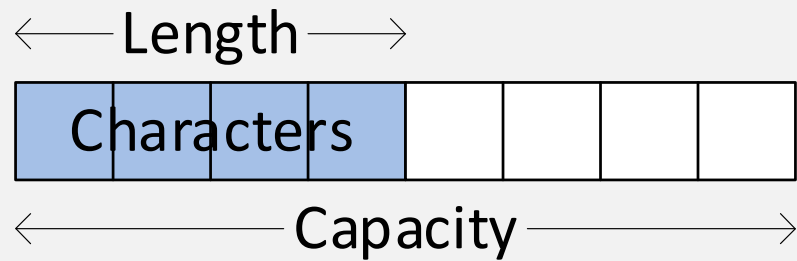


C++ `string` CLASS REVIEW

- `#include <string>`
- Class data and functions called members
- Header prototypes non-member functions
- Hides data members
- Binds function calls to `string` objects
- `size_t` is a portable type suitable for saving a size
- Input with `getline(cin, s);`
- output with `cout << s << endl;`



BASIC OPERATIONS: length AND capacity



```
size_t length();  
size_t size();  
size_t capacity();
```

```
string s;
```

```
for (size_t i = 0; i < s.length(); i++)
```

```
for (size_t i = 0; i < s.size(); i++)
```

```
cout << s.length() << " " <<  
      s.capacity() << endl;
```

```
11 15
```



A PREREQUISITE: OVERLOADED OPERATORS

- Overloaded operators: operator 😊
- Function name is “operator” followed by the operator
 - `char& operator[] (size_t pos);`
 - `string operator+(const string& lhs, const string& rhs);`
- Overloaded operator functions allow a novel calling syntax:
 - `s1 😊 s2`



CHARACTER ACCESS

- `char& operator[](size_t pos);`
 - Does NOT validate the index
- `char& at(size_t pos);`
 - Validates the index
- `char& front();`
- `char& back();`
- An r-value expression is a value
- An l-value expression is an address

```
for (size_t i = 0; i < s.length(); i++)  
    cout << s[i] << endl;
```

```
for (size_t i = 0; i < s.size(); i++)  
    cout << s.at(i) << endl;
```

```
s[0] = 'X';
```

```
s.at(0) = 'Z';
```

```
c = s.front();      s.front = 'A';  
c = s.back();       s.back() = 'B';
```



ASSIGNMENT

```
string& operator=(const string& rhs);  
string& operator=(const char* s);  
string& operator=(char c);
```

```
string s1("hello");  
string s2;  
s2 = s1;  
s2 = "hello";  
s2 = 'X';
```



CONCATENATION

```
string operator+(const string& lhs, const string& rhs);  
string operator+(const string& lhs, const char* s);  
string operator+(const string& lhs, char c);
```

```
string s1("hello");  
string s2(" world");  
string s3;  
s3 = s1 + s2;  
s3 = s1 + " world";  
s3 = s1 + s2 + '!';
```

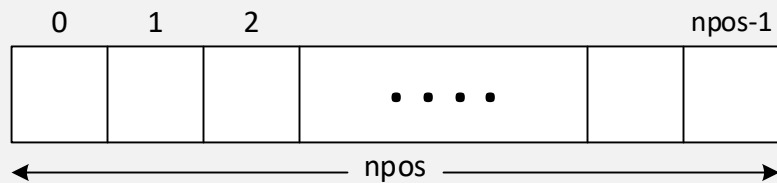


CONCATENATION WITH ASSIGNMENT

```
string& operator+=(const string& rhs);  
string& operator+=(const char* s);  
string& operator+=(char c);
```

```
string s1("hello");  
string s2(" world");  
s1 += s2;  
s1 += " world";  
s1 += ' ';  
s1 += "world";
```


THE `string` CLASS CONSTANT `npos`



- `string s(...);`
- `int i = s.length();`
- $0 \leq i \leq s.length()-1$
- `string::npos`



THE `find` AND `rfind` FUNCTIONS

- `size_t find(const string& str, size_t pos = npos);`
- `size_t find(const char* str, size_t pos = npos);`
- `size_t find(const char c, size_t pos = npos);`
- Returns `npos` if the string or character is not found

```
string s = "Hello, World!";  
size_t index = s.find("World");  
if (index != string::npos)  
    ...
```

```
string s = "Hello, World!";  
size_t index = s.rfind('l');  
if (index != string::npos)  
    cout << s.rfind('l', 5) << endl;
```

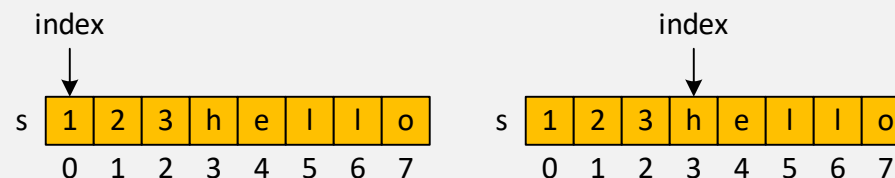


CONVERTING BETWEEN NUMBERS AND STRINGS

- `string to_string(int val);`
- `string to_string(double val);`
- `string s1 = to_string(123);`
- `string s2 = to_string(3.14);`
- `int stoi(const string& str, size_t* index = nullptr, int base = 10);`
- `double stod(const string& str, size_t* index = nullptr);`
- `int i = stoi(s1);`
- `int i = stoi("0XAF27", 16);`
- `double d = stod(s2);`

THE `index` ARGUMENT

```
string s = "123hello";  
int index = 0;  
int counter = stoi(s, &index);
```



- Passed by pointer, making it an INOUT argument
- The function sets it to the location of the first non-convertible character
- The function throws an exception if the string begins with a non-number character: “x123hello”