Standard C++ Strings

The Standard C++ String Type

- Standard C++ defines its string type in the <string> header.

- Objects of type string can be declared and initialized in several ways:

```
string str1;
string str2 = “”;
string str3 = "New York";
```

- If the string is not initialized, it represents the empty string.

Most Common functions

Function	Command	Details	Return value
Length	lenght()	Length of the string	unsigned int
Append	append(*string*)	appending additional *string* at the end of its current string	string
Substring	substr (*position*, *length*)	copy *length* of a string from *position* into another string	string
Find	find (*string*, *position*);	Search a *string*, start at *position*	unsigned int (start position)

Most Common functions

Function	Command	Details	Return value
Erase	erase (*position*, *length*);	Delete part of a string from *position*, reduce its *length*	string
Replace	replace (*position*, length, string);	Replace part of a *string* from *position*, spans *length* characters	string

Examples

- A length function can be used to determine how many characters are stored in a string:

```
    string s = "ABCDEFG";
    cout << s.length() << endl;    // print 7 for s = "ABCDEFG"
```

- Substring:

```
 string s4 = s6.substr(5, 3);   // s4 is "FGH"
```

- The find() returns the index of the first   occurrence of a givens substring:

```
   string s7 = "Mississippi River Basin";
 cout << s7.find("si“,0) << endl; // prints 3
 cout << s7.find("so“,5) << endl; // prints 23, the length of
```

  - Note: If start search at position 0, you can write

```
     cout << s7.find("si“) << endl; // prints 3
```

Examples (cont'd.)

- Append: adding *string* at the end of its current string

```
string s6 = s + "HIJK";          // s6 is "ABCDEFGHIJK"
```

- The `erase()` and `replace()` function work like this:

```
s6.erase(4, 2); /                // change s6 to " ABCDGHIJK"
s6.replace(5, 2, "xyz");         // change s6 to " ABCDGxyIJK"
```

Basic Operators

Operator	Details	Return value
==, <, >	Compare strings	Boolean
+	Add additional *string* at the end of its current string (same as *append* functions)	string

Examples

C++ strings can be compared using the relational operators:

```
   if (s2 < s5) cout << "...";
 while (s4 == s3) ...;
```

- Also concatenate and append:

```
   string s6 = s + "HIJK";
 s2 += s5;
```

- Substring:

```
 s4 = s6.substr(5, 3);
```

Files

- File processing in C++ is very similar to ordinary interactive input and output.
- Because the same kind of stream objects are used.
- An input file is managed by ifstream in the same way as istream manages cin.
- An output file is managed by ofstream.
- The only different is that ifstream objects and ofstream objects have to be declared explicitly and initialized with the external name of the file which they manage.
- Also have to #include <fstream>.

Output File Stream

```
#include <iostream.h>
int main()
{
  string str = “my text”;
  cout << str;
  return 0;
}
```

Output to screen

Output to file named “myfile.dat”

```
#include <fstream.h>
int main()
{
  string str = “my text”;
 ofstream fout(“myfile.dat");
 fout << str;
  return 0;
}
```

Input File Stream

```
#include <iostream.h>
int main()
{
  int value = 0;
  cin >> value;
  return 0;
}
```

Read from keyboard

Read from file named “number.dat” (only one value)

```
#include <fstream.h>
int main()
{
 ifstream fin(“number.dat");
  int value = 0;
  fin >> value;
  return 0;
}
```

Question: How to read until the end of file?

Extraction Operator to Control a Loop

- The expression

```
cin >> x
```

has a value that can be interpreted as boolean, depending upon whether the input is successful.

```
int main() {
  int n;
        while (cin >> n)
                cout << "n = " << n << endl;
}
```

Input File Stream (Con’t)

To read until the end of file

```
#include <fstream.h>
int main()
{
 ifstream fin(“number.dat");
  int value = 0;
  while (fin >> value)
 {
        …
 }
  return 0;
}
```

Read from file named “number.dat” until the end of file

Read/Write

```
#include <fstream.h>
int main()
{
   int value = 0;
   ifstream fin(“number.dat");
   ofstream fout_odd(“odd.dat");
   ofstream fout_even(“even.dat");
   while (fin >> value)
   {
     if(value%2) {
        fout_odd << value;
     }else {
        fout_even << value;
     }
   }
   return 0;
}
```

Number.dat

```
12 11 20 15 35 10 69
71 23 80
```

odd.dat

```
11 15 35  69 71 23
```

even.dat

```
12 20 10 80
```