Strings and Stream I/O
C Strings

- In addition to the string class, C++ also supports old-style C strings.
- In C, strings are stored as null-terminated character arrays.

```cpp
char * str1 = "What is your name? ";
char str2[] = "How old are you? ";
char arr1[] = {'H','o','w',' ','o','l','d',' ','a','r','e',' ','y','o','u','?',' '};
```

```
What is your name? \
```

```
How old are you? \
```

```
How old are you? \
```
C Library String Functions

```c
#include <string.h>
#include <stdlib.h>

void F() {
    char message[100];

    // copy string into message
    strcpy(message, "The following error has occurred: ");

    // concatenate string to message
    strcat(message, "Out of memory ");

    // compute length of the string
    int msgLen = strlen(message);

    // convert a string to an integer
    int num = atoi("-272");
}
```
int strlen(const char *str) {
    int len = 0;
    const char * p = str;
    while (*p != '\0') {
        ++len;
        ++p;
    }
    return len;
}

void strcpy(char * dest, const char * src) {
    char * d = dest;
    const char * s = src;
    while (*s != '\0') {
        *d++ = *s++;
    }
    *d = '\0';
}

void strcat(char * str1, const char * str2) {
    strcpy(str1 + strlen(str1), str2);
}
#include <string.h>
#include <iostream>
using namespace std;

void F() {
    const char * A = "Hello";
    const char * B = ", ";
    const char * C = "how are you today?"
    char * message;

    int msgLen = strlen(A) + strlen(B) + strlen(C);
    message = new char[msgLen + 1];

    strcpy(message, A);
    strcat(message, B);
    strcat(message, C);

    cout << message << endl;
    delete [] message;
}
Using C Strings with C++ Strings

- Sometimes you need to convert a C string to a C++ string, and vice versa

  - C string to C++ string
    - Initialize a C++ string with a C string
      ```cpp
      string name("fred");
      ```
    - Assign a C string to a C++ string
      ```cpp
      name = "fred";
      ```

  - C++ string to C string
    - All C runtime library functions require C strings
    - The string class has a method named `c_str` that returns a pointer to a C string
      ```cpp
      int x = atoi(numStr.c_str());
      ```
C Library Character Classification Functions

- `#include <ctype.h>`
- `isalpha`
- `isdigit`
- `isalnum`
- `isspace`
- `islower`
- `isupper`
- `isprint`
- `iscntrl`
- many others
Command-Line Arguments

- Command-line arguments are passed as parameters to the main function
  ```c
  int main(int argc, char *argv[])
  ```
- You can omit these parameters if you don't need to process command-line arguments
- The first parameter, `argc`, is a count of how many command-line arguments there are
- The second parameter, `argv`, is an array of C strings that are the values of the command-line arguments
  - `argv[0]` contains the name of the program executable
  - `argv[1]` .. `argv[argc-1]` contain the actual command-line arguments
Command-Line Arguments

printargs.cpp

```cpp
#include <iostream>
using namespace std;

int main(int argc, char *argv[]) {
    for (int i=0; i < argc; ++i) {
        cout << argv[i] << endl;
    }
}
```

$ ./printargs Computer Science 240
./printargs
Computer
Science
240
$
Stream I/O

- A stream is an ordered sequence of bytes
- Streams are a useful abstraction for doing I/O
- Input comes from the keyboard, files, network connections, and memory
- Output goes to the screen, files, network connections, and memory
- All of these I/O sources and destinations can be viewed as byte streams
C++ Stream I/O Library

- C++ has an extensive set of classes for doing stream-based I/O
- In class we will learn enough about stream I/O to do the programming projects
- The textbook provides much more information than will be covered in class
  - If you want to put C++ on your resume, Read The Book!
Stream Class Hierarchy

- ios
  - istream
    - ifstream
    - istrstream
  - iostream
  - ofstream
  - ostringstream
    - fstream
    - strstream
The C++ stream classes support both formatted and unformatted I/O

**Formatted I/O**
- Overloaded stream insertion `<<` and extraction `>>` operators

**Unformatted I/O**
- Read/write one character at a time
- Read/write one line at a time
Reading Lines From a Text File

```cpp
#include <string>
#include <fstream>
#include <iostream>
using namespace std;

bool ReadLine(istream & stream, string & line) {...}

int main(int argc, char * argv[]) {
  if (argc == 2) {
    ifstream file(argv[1]);
    if (file) {
      string line;
      while (ReadLine(file, line)) {
        cout << line << endl;
      }
      file.close();
      return 0;
    }
    else {
      cout << "Could not open file " << argv[1] << endl;
    }
  } else {
    cout << "Invalid arguments" << endl;
  }
  return -1;
}
```
bool ReadLine(istream & stream, string & line) {
    const int BUF_SIZE = 1000;
    char buf[BUF_SIZE];
    stream.getline(buf, BUF_SIZE);
    if (stream) {
        line = buf;
        return true;
    } else {
        return false;
    }
}
ReadLine (version 2)

bool ReadLine(istream & is, string & line) {
    if (!is) {
        return false;
    }

    line = "";
    while (true) {
        char c;
        is.get(c);
        if (is) {
            if (c == '\n') {
                return true;
            } else {
                line += c;
            }
        } else {
            return true;
        }
    }
}
#include <string>
#include <fstream>
#include <iostream>
using namespace std;

void WriteHTMLPage(ostream & stream) {
    ...
}

int main(int argc, char * argv[]) {
    if (argc == 2) {
        ofstream file(argv[1]);
        if (file) {
            WriteHTMLPage(file);
            file.close();
            return 0;
        }
    } else {
        cout << "Could not open file " << argv[1] << endl;
    }
}
else {
    cout << "Invalid arguments" << endl;
}
return -1;
void WriteHTMLPage(ostream & stream) {
    stream << "<HTML>" << endl;
    stream << "<HEAD>" << endl;
    stream << "<TITLE>CS 240</TITLE>" << endl;
    stream << "</HEAD>" << endl;
    stream << "<BODY>" << endl;
    stream << "<H1>Welcome to CS 240!</H1>" << endl;
    stream << "</BODY>" << endl;
    stream << "</HTML>" << endl;
}
#include <string>
#include <strstream>
#include <iostream>
using namespace std;

void WriteHTMLPage(ostream & stream) {
    stream << "<HTML>" << endl;
    stream << "   <HEAD>" << endl;
    stream << "      <TITLE>CS 240</TITLE>" << endl;
    stream << "   </HEAD>" << endl;
    stream << "   <BODY>" << endl;
    stream << "      <H1>Welcome to CS 240!</H1>" << endl;
    stream << "   </BODY>" << endl;
    stream << "</HTML>" << endl;
}

int main() {
    ostrstream stream;
    WriteHTMLPage(stream);
    stream << ends;  // writes '\0' to the stream
    cout << stream.str();
    return 0;
}
```cpp
#include <string>
#include <sstream>
#include <iostream>
using namespace std;

int main(int argc, char * argv[]) {
    if (argc == 2) {
        istringstream stream(argv[1]);
        int count = 0;
        while (true) {
            string word;
            stream >> word;
            if (stream) {
                ++count;
            } else {
                break;
            }
        }
        cout << count << " words" << endl;
        return 0;
    } else {
        cout << "Invalid arguments" << endl;
        return -1;
    }
}
```