C++: Functions, Program Compilation, Libraries

Modified from CS101 slides, which are by JPC and JWD © 2002 McGraw-Hill, Inc.
Translation System

- Set of programs used to develop software
- A key component of a translation system is a translator
- Some types of translators
  - Compiler
    - Converts from one language to another
  - Linker
    - Combines resources
- Examples
  - Microsoft Visual C++®, C++Builder®, g++, Code Warrior®
    - Performs compilation, linking, and other activities.
Software Development Activities

- Editing
- Compiling
- Linking with precompiled files
  - Object files
  - Library modules
- Loading and executing
- Viewing the behavior of the program
Software Development Cycle

1. Edit
2. Think
3. Source Program
4. Compile
5. Link
6. Load
7. Execute
8. Library routines
9. Other object files
IDEs

- Integrated Development Environments or IDEs
  - Supports the entire software development cycle
    - E.g., MS Visual C++, Borland, Code Warrior
  - Provides all the capabilities for developing software
    - Editor
    - Compiler
    - Linker
    - Loader
    - Debugger
    - Viewer
Function Prototypes

Before a function can appear in an invocation its interface must be specified

- *Prototype* or complete definition

```c
FunctionType FunctionName ( ParameterList )
```

A description of the form the parameters (if any) are to take

```c
int Max(int a, int b);
```
Function Prototypes

- Before a function can appear in an invocation its interface must be specified
  - Prototypes are normally kept in library header files

 identifier name of function

\[
\text{FunctionType FunctionName ( ParameterList )};
\]

\[
\text{int Max(int a, int b));
\]
Libraries

Library
- Collection of functions, classes, and objects grouped by commonality of purpose
- Include statement provides access to the names and descriptions of the library components
- Linker connects program to actual library definitions

Examples
- String: STL’s string class
- Graphics libraries
- GUI libraries (like Qt, Windows Forms)
Basic Translation Process

Source program

Process preprocessor directives to produce a translation unit

Check translation unit for legal syntax and compile it into an object file

Link object file with standard object files and other object files to produce an executable unit

Executable Unit
Some Standard Libraries

- **fstream**
  - File stream processing
- **assert**
  - C-based library for assertion processing
- **iomanip**
  - Formatted input/output (I/O) requests
- **ctype**
  - C-based library for character manipulations
- **math**
  - C-based library for trigonometric and logarithmic functions

**Note**
- C++ has many other libraries
Library Header Files

- Describes library components

- Typically contains
  - Function prototypes
    - Interface description
  - Class definitions

- Sometimes contains
  - Object definitions
    - Example: `cout` and `cin` in `iostream`
Library Header Files

- Typically do not contain function definitions
  - Definitions are in source files
  - Access to compiled versions of source files provided by a linker
```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    cout << "Enter Quadratic coefficients: ";
    double a, b, c;
    cin >> a >> b >> c;
    if ( (a != 0) && (b*b - 4*a*c > 0) ) {
        double radical = sqrt(b*b - 4*a*c);
        double root1 = (-b + radical) / (2*a);
        double root2 = (-b - radical) / (2*a);
        cout << "Roots: " << root1 " " << root2;
    } else {
        cout << "Does not have two real roots";
    }
    return 0;
}
```
#include <iostream>
#include <fstream>  // file stream library
using namespace std;

int main() {
    ifstream fin("mydata.txt");
    int ValuesProcessed = 0;
    float ValueSum = 0;
    float Value;
    while (fin >> Value) {
        ValueSum += Value;
        ++ValuesProcessed;
    }

    // to be continued next slide...
}

```cpp
#include <iostream>
#include <fstream> // file stream library
using namespace std;

int main() {
    ifstream fin("mydata.txt");
    int ValuesProcessed = 0;
    float ValueSum = 0;
    float Value;
    while (fin >> Value) {
        ValueSum += Value;
        ++ValuesProcessed;
    }

    if (ValuesProcessed > 0) {
        ofstream fout("average.txt");
        float Average = ValueSum / ValuesProcessed;
        fout << "Average: " << Average << endl;
        return 0;
    } else {
        cerr << "No list to average" << endl;
        return 1;
    }
}
```
# Program Structure

1. **Include Necessary Files**:
   - Libraries, headers, etc.

2. **Classes and Functions**:
   - Define classes, functions, and methods.

3. **Main Function**:
   - Entry point of the program.
   - Main logic of the program.

4. **Variables and Data**:
   - Define variables, arrays, etc.

5. **Control Structures**
   - Loops, conditionals, etc.

6. **Error Handling**
   - Handle errors and exceptions.

7. **Other Considerations**
   - Performance, readability, etc.

---

### Example 1: Reading and Writing to Text Files

```cpp
#include <fstream>
#include <iostream>

int main() {
    std::ifstream sin("in1.txt");  // extract from in1.txt
    std::ofstream sout("out1.txt");  // insert to out1.txt

    std::string s;

    while (sin >> s) {  // what's this do?
        sout << s << std::endl;
    }

    sout.close();  // close the output stream

    return 0;
}
```

### Example 2: File Operations

- **Open**: Use `ifstream` or `ofstream` to open a file.
- **Read/Write**: Use `>>` for reading and `<<` for writing.
- **Close**: Don't forget to close the streams.

---

### Tips

- **File Paths**: Use absolute or relative paths.
- **Error Checking**: Implement checks for file open and stream errors.
- **Performance**: Optimize reading and writing operations.

---

### Notes

- **Version Control**: Use version control systems like Git.
- **Documentation**: Keep your code well-documented.
- **Testing**: Write tests to ensure functionality.

---

### Related Resources

- [File Handling in C++](https://en.cppreference.com/w/cpp/io/io)
ifstream sin("in1.txt");  // extract from in1.txt
ofstream sout("out1.txt");  // insert to out1.txt

string s;
while (sin >> s) {  // what's this do?
    sout << s << endl;
}
sin.close();  // done with in1.txt
sout.close();  // done with out1.txt
ifstream sin("in1.txt");    // extract from in1.txt
ofstream sout("out1.txt");  // insert to out1.txt

string s;
while (sin >> s) {   // what's this do?
    sout << s << endl;
}

sin.close();       // done with in1.txt
sout.close();      // done with out1.txt

sin.open("in2.txt");    // now extract from in2.txt
sout.open("out.txt", // now append to out2.txt
         (ios_base::out | ios_base::app));
```cpp
ifstream sin("in1.txt");  // extract from in1.txt
ofstream sout("out1.txt"); // insert to out1.txt

string s;
while (sin >> s) {  // what's this do?
    sout << s << endl;
}

sin.close();  // done with in1.txt
sout.close();  // done with out1.txt

sin.open("in2.txt"); // now extract from in2.txt
sout.open("out.txt",  // now append to out2.txt
    ios_base::out | ios_base::app);
while (sin >> s) {  // what's this do?
    sout << s << endl;
}

sin.close();  // done with in2.txt
sout.close();  // done with out2.txt
```