Queens College, CUNY, Department of Computer Science

Object Oriented Programming in C++

CSCI 211 / 611

Summer 2018

Instructor: Dr. Sateesh Mane

September 6, 2018

Project with multiple files

- This lecture is about program compilation, for a project with multiple files.
- **There is nothing "C++" in this lecture.**
- This lecture is about **software organization**, for a (possibly) large project.

1 Project with multiple source files

- In most real-life applications, a project is too large for all the code to fit into one file.
- Hence the source code is placed in multiple files.
- In this section, we make some general observations.
- **If even a small part of the program code is edited, the entire project must be recompiled, if all the code is in one file.**
  1. This is time consuming, for a large project.
  2. If the code is in multiple source files, we only need to recompile those files which are affected by the change.
  3. The rest of the project need not be recompiled.
- **Multiple developers can work simultaneously on different source files.**
  1. If all the code is in one file, only one person can work on the file at a time.
  2. A large project typically has multiple software developers, working on different parts of the code.
  3. If the code is in multiple source files, the individual developers can perform their work simultaneously.
  4. This asumes, of course, that the code changes by the individual developers do not affect each other. *This is a serious management problem for large projects.*
- **A source file can be included in multiple different projects.**
  1. Actually, we see this all the time without even thinking about it.
  2. The I/O and math libraries are included in almost every C++ project.
  3. We do not write a separate math library (for example) for every C++ project.
  4. The same concept applies to a source file of our own.
  5. The same file may be useful in many different projects.
  6. There is no need to copy and paste it for each project.
  7. Furthermore, if the file is upgraded, all the projects benefit from the upgrade.
  8. *Convsersely, if a bug is introduced into the file, all the projects are affected by the bug. This is also a serious management problem for large projects, especially when writing heavily used software libraries which will be used by many people.*
- **There is nothing "C++" in this lecture.**
- This lecture is about *software organization*, for a (possibly) large project.

2 Forward declarations

- Functions in one file may call functions defined in a different file.
- Clearly, this is a problem.
- To solve this problem, C++ supports **forward declarations.**
- A forward declaration is just the function signature.
- We place the forward declarations in **header files.**
- A header file is usually denoted using a ".h" suffix.
- For a project with multiple files of source code, each source file includes suitable header files which contain all the relevant forward declarations to compile the source code.
- The syntax to include a header file "header_filename.h" is as follows.

```
#include "header_filename.h"
```

- The overall set of project files consists of a set of source ".cpp" files and header ".h" files.

3 Project: example with header and source files

- Consider functions to swap two inputs. The forward declarations are as follows.

```
void swap(int &u, int &v);
void swap(double &u, double &v);
void swap(string &u, string &v);
```

- We place the function declarations in a ".h" header file.
- We place the function bodies in a ".cpp" source file.
  1. The source file `swap.cpp` has a statement `#include "swap.h"` to include the header file.
  2. Every source file which calls the swap functions will write `#include "swap.h"` to access the forward declarations.
  3. The compiler knows the function bodies are available "somewhere" in the overall project.
- The project contains three files.
  1. Header file `swap.h`.
  2. Source file `swap.cpp`.
  3. Main program `main.cpp`.
- The main program includes the header file "`swap.h`" and calls the "`swap`" functions.
- The actual function bodies of the "`swap`" functions are in a different file.
- The compiler knows that the function bodies of the "`swap`" functions are available "somewhere" in the overall project.
- **See next page(s).**

3.1 Header file "swap.h"

```
#include <string>                  // required, to declare what "string" object means
using namespace std;

void swap(int &u, int &v);
void swap(double &u, double &v);
void swap(string &u, string &v);
```

3.2 Source file "swap.cpp"

```
#include "swap.h"                    // include header file

void swap(int &u, int &v)
{
  int t = u;
  u = v;
  v = t;
}

void swap(double &u, double &v)
{
  double t = u;
  u = v;
  v = t;
}

void swap(string &u, string &v)
{
  string t = u;
  u = v;
  v = t;
}
```

3.3 Main program "main.cpp"

```
#include <iostream>                    // system headers
#include <string>

#include "swap.h"                      // user-defined header file

using namespace std;

int main()
{
  int a = 3, b = 4;
  double x = 2.2, y = 3.3;
  string s("abcd"), t("alpha");

  cout << "original:" << endl;
  cout<< a << "   " << b << endl;
  cout<< x << "   " << y << endl;
  cout<< s << "   " << t << endl;

  swap(a, b);
  swap(x, y);
  swap(s, t);

  cout << "swap:" << endl;
  cout<< a << "   " << b << endl;
  cout<< x << "   " << y << endl;
  cout<< s << "   " << t << endl;

  return 0;
}
```