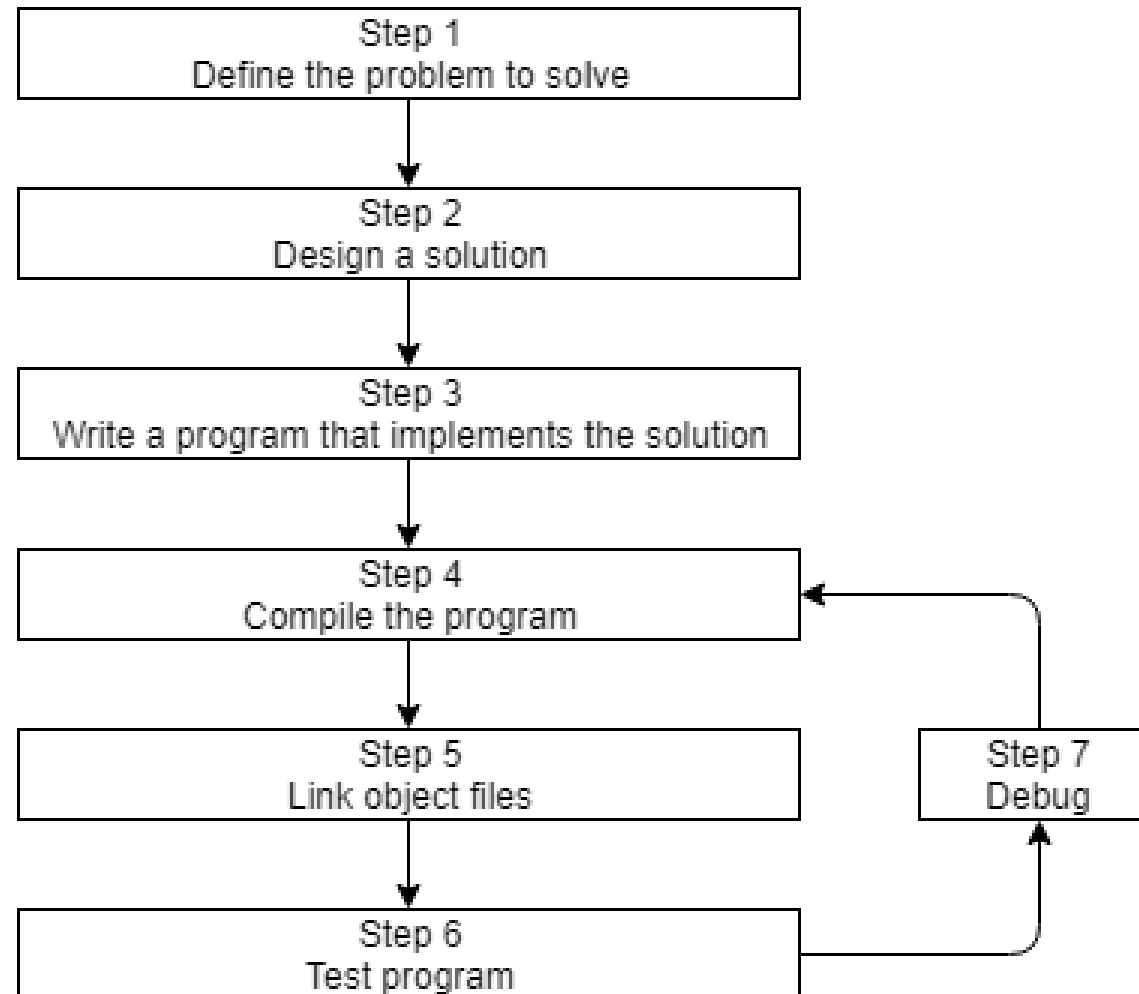


Programming Languages and Tools: Programming with C++ CS:3210:0003

Lecture/Lab #14

Developing C++ Programs



Compile and Link

- Compilation:

1. Check syntax
2. Convert C++ code -> machine language instructions
 .cpp File -> .o Object File

- Linking:

1. Resolve dependencies between files
2. Resolve library (packaged precompiled) files
 - C++ standard libraries (iostream, etc.)
 - Other libraries

- **Building:** source code -> executable (binary)

Compile and Link

- `g++ file.cpp -o file.bin`
 - compiles `file.cpp`,
 - links included libraries,
 - creates binary `file.bin`
- `g++ -c file.cpp`
 - compiles `file.cpp` to object code `file.o`
- `g++ -o file.bin file.o`
 - links included libraries to binary `file.bin`

Separate Compilation

- To compile multiple files, use g++ with multiple arguments:
`g++ file1.cpp ... filen.cpp -o binname.bin`
- Compiler doesn't remember definitions from `filei.cpp` while compiling `filej.cpp`
- Forward declarations work across files compiled together

Activity

- Create a folder `workspace/calc`
- Copy `prob8.cpp` from `class/homeworks/hw1/solutions/` to `workspace/calc/`
- Separate `prob8.cpp` into 3 files and compile to `calc.bin`:
 1. `menu.cpp` will contain the `Menu` function
 2. `choice.cpp` will contain the `Choice` function
 3. `main.cpp` will contain `main()`

Activity

- Create a folder `workspace/bin`
- Copy `prob3.cpp` from `class/homeworks/hw2/solutions/` to `workspace/bin/`
- Separate `prob3.cpp` into 3 files and compile to `bin.bin`:
 1. `binio.cpp` will contain the `inputint()` and `printBin()` and any helper functions
 2. `conv.cpp` will contain the `dec2Bin()` and `bin2Dec()` and any helper functions
 3. `menu.cpp` will contain `menu()` and the functions to perform the menu ops
 4. `main.cpp` will contain `main()`

Header Files

- Store and propagate declarations to code in `.cpp` files.
 - Ex: `iostream` stores declaration of `std::cout`
- Extension: `.h`
- Header files:
 1. Header guard
 2. Declarations
- To access header files (don't compile them):
`#include "headerName.h"`
- Definitions go in corresponding `.cpp` file

Header Files

- Include header files in corresponding implementation files
 - Because compilers can usually check whether the types match
- Preprocessor replaces `#include<header>` by contents of header
- Don't define functions inside header files
 - If you do, linker will have to deal with multiple definitions
- Tell the preprocessor where to look
 1. `#include <headerName>` looks in include directories
 2. `#include "headerName"` looks locally first, then in include directories

The One Definition Rule (ODR)

- Within the same scope, an entity (function/variable/type) can have only one definition
- This applies across multiple files if they are compiled together
- If violated, compiler error
- Harder to fix with included files
- Don't `#include .cpp` files
 - Instead compile them

Activity

- Create a folder `workspace/bin2` (by copying `workspace/bin`)
- Move all declarations to header files, `#include` the right header files in each file, and compile to `bin.bin`:
 1. `binio.cpp` will contain definitions of `inputint()` and `printBin()` and any helper functions, and `binio.h` the declarations
 2. `conv.cpp` will contain definitions of `dec2Bin()` and `bin2Dec()` and any helper functions and `conv.h` the declarations
 3. `menu.cpp` will contain `menu()` and functions to perform the menu ops and `menu.h` the declarations
 4. `main.cpp` will contain `main()`

Classes in Header Files

- Classes can/should be split into header (.h) and implementation (.cpp) files
 - Define class `ClassName` in `ClassName.h`
 - Declare functions inside class
 - Define functions in .cpp file using syntax:
`returnType ClassName::functionName(args) { ... }`

Activity

- Create a folder `workspace/tennis` and copy `03-18/05tennis6.cpp`, `Points.txt` and `Points2.txt` into it
- Separate it into:
 1. `Score.h` will contain definitions of `GamePoints`, `Score()`, `printScore()` and declarations of `printGamePoints()`, `incrp1()`, `incrp2()`
 2. `Score.cpp` will contain definitions of `printGamePoints()`, `incrp1()`, `incrp2()`
 3. `main.cpp` will contain `main()`

GNU make

- GNU make
 - determines which pieces of a large program need to be recompiled
 - issues the commands to recompile them
- From terminal, `make --version`
- Project directory contains Makefile
 - Build project by running `make` from terminal
- Makefile contains rules with syntax:
targets: prerequisites
 recipe
 recipe
 ...

GNU make

- Multiple targets are specified, call a specific one by running `make targetName`
- By default, it runs the topmost target
- A target is run only if one of it's prerequisites are newer than it
- A **target** is two things:
 1. An identifier for a case in the `Makefile`
 2. The name of the file created from `makeing` the rule
- For `.PHONY` targets, 2. doesn't apply