HeaderFiles

November 30, 2021

1 Header Files

- header files and their purpose
- standard library header files
- header file content rules
- using header files
- implementation files

1.1 Header files and their purposes

- we've used library header files such as `iostream`, `string`, etc.
- the purpose of this chapter is to learn how to create your own header files and why
- as program grows bigger, code need to be divided into many files
- using function prototype (forward declaration) and defining functions after main is not scalable
- breaking solution code into many files has many advantages:
  - makes program easier to manage, read, update, debug
  - makes it easier to work in a team where each member works on a separate file
  - avoid conflicts while using version control such as git
- Generally, header files allow us to put declarations in one or more files and then include them wherever we need them
  - this can save a lot of typing in multi-file programs
  - helps us create our own library of important functions

1.2 Creating header files

- header best practices:
  - must contain header include guards (to avoid being included multiple times in the final binary)
  - typically contains struct and class definitions only
  - contains function prototypes
  - avoid function definitions
  - may include other header files
  - do NOT declare global variables
  - each header file should be as independent as possible with specific purpose
  - file has `.h` extension

1.3 Implementing header files

- typically a header file is implmented in a separate corresponding .cpp file
- functions and classes are implmented or defined in implementation files
- implementation file are regular .cpp files that must include the header being implemented
- must also include any library header files required to implment the functions
- syntax:

```
#include "headerFileName.h"
```

- note the double quotes `"headerName.h"` instead of `<headerName.h>` used for built-in headers

1.4 Using header files

- include user-defined header files similar to library header file
- include only the header files that are required
- syntax:

```
#include "headerFileName.h"
```

- use the functions and user-defined types, etc. defined in the included header file

1.5 Compiling multiple cpp files

- header files are not compiled; ONLY the CPP files
- compiling multiple file is similar to compiling single file using g++ compiler
  - simply list all the .cpp files separated by a space

```
g++ <switches: -std=c++17, etc.> -o programName inputFile1.cpp inputFile2.cpp ...
```

- or use the Makefile as shown in the following demo programs

1.5.1 Demo 1 - see folder `demos/header_files/header/`

- a simple demo program with one header file

1.5.2 Demo 2 - see folder demos/header_files/triangle/

- contains several header and cpp files
- a single-file `triangle.cpp` program provided in File IO chapter - demos/file_io/triangle/ is separated into several `header` and `cpp` files
- test functions are separated into `test.h` and `test.cpp` files
- utility functions are separated into `utility.h` and `utility.cpp` files
- main entry/driver function is in `main.cpp` file
- triangle definition and related functions are separated into `triangle.h` and `triangle.cpp` files

[ ]: