Lab-4

Data Structures

Khalid Mengal

Contents

- Passing Arguments to main function
- Header files
- makefile

Passing Parameters to **main**

- It is possible to give the **main** function a parameter list.
- **main** function header:

```
int main(int argc, char *argv[]);
```

where argc is an integer variable and argv is an array of pointers to characters.

3

Passing Parameters to **main**

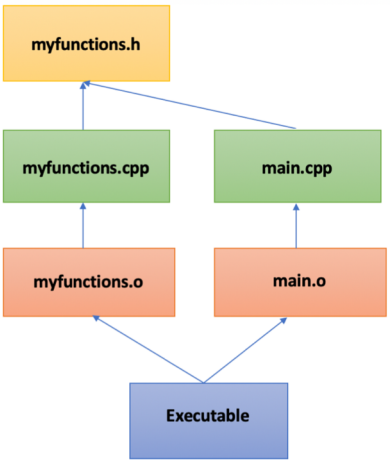
```
#include <iostream>
using namespace std;
/* Program to read and echo data from command line */
int main(int argc, char *argv[ ])
{
   for (int i = 1; i < argc; i++)
         cout<<argv[i]);
   return EXIT_SUCCESS;
}
```

4

Header Files in C++

- Contain declaration for Functions, structures, union, classes, constants etc.
- Can be imported/included in any other program
- Source file which include header file can access all declarations/definitions present in the header file
- Two types:
  - System Header Files: Comes with Compiler (e.g. iostream, cstdlib etc.)
  - User Header Files: Written by the programmer
- Usage:
  - #include<string> //System header files (from compilers include directory)
  - #include "filename" //User Header file (from current directory)

Header and cpp files



makefile

- makefile tells make utility how to compile and link a program
- A Simple makefile contains Rules:

```
target … : prerequisites …
        recipe
        …
        …
```

- Target: Usually the name of the file (generated by Program)
- Prerequisites: File(s) used as input to create the Target
- Recipe: Action(s) which are carried out when perquisites change

A Simple makefile

Makefile

```
1 output: main.o myfunctions.o
2     g++ main.o myfunctions.o -o output
3 main.o: main.cpp
4     g++ -c main.cpp
5 myfunctions.o:  myfunctions.cpp myfunctions.h
6     g++ -c myfunctions.cpp
7 clean:
8     rm *.o output
9
```

`g++ -c` Only run preprocess, compile, and assemble steps