CSE 165/ENGR 140 Intro to Object Orient Program

Lecture 2 – Programming in C++

Annoucements

- Reading:
  - Ch 1 and 2: focus on pages 23 - 46 and 83 – 118
- A little about Jurybox
  - juryboxapp.com
  - Web and mobile app
  - Tech stack (MERN)
    - React (JavaScript library for building User Interfaces)
    - Node.js (JavaScript runtime environment)
    - Express (Node.js web framework)
    - MongoDB (No SQL database)

Object oriented programming (OOP)

- Everything is an object
- A program is a bunch of objects telling each other what to do by sending messages
- Each object has its own memory made up of other objects
- Every object has a type
- All objects of a particular type can receive the same messages

Declaration vs. Definition (Ch. 2)

- Declaration
  - Gives a name (identifier) to a variable or function
  - Variable: `extern int a;` //extern means the variable will be defined later
  - Function: `int func1(int, int);`
- Definition
  - Allocates a memory location (storage) for a variable or function
  - Variable: `int a;`
  - Function: `int func1(int length, int width) {…};`
  - It is illegal to define a variable or function multiple times in a program

Declaration vs. Definition

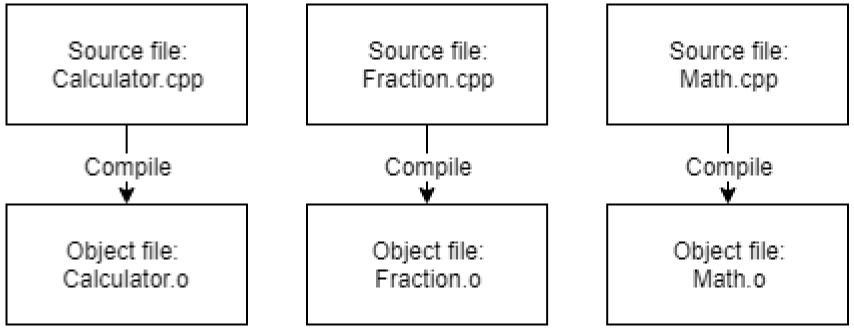
```
//: C02:Declare.cpp
// Declaration & definition examples

extern int i;               // Declaration without definition
extern float f(float);      // Function declaration

float b;                    // Declaration & definition
float f(float a) {          // Definition
    return a + 1.0;
}

int i;                      // Definition
int h(int x) {              // Declaration & definition
    return x + 1;
}

int main() {
    b = 1.0;
    i = 2;
    f(b);
    h(i);
}
```

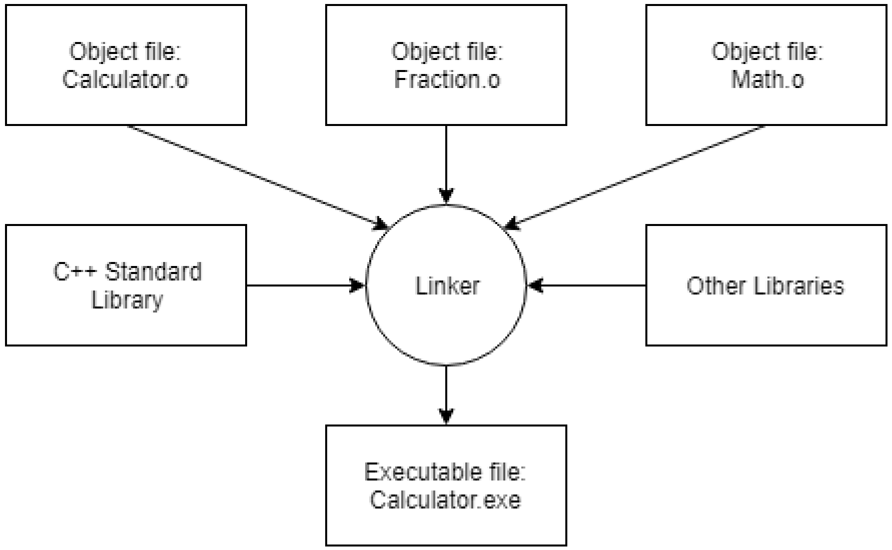
Writing C++ Code

- C++ source code can be written with a text editor, we don’t need a fancy IDE
- Example editors:
  - **gedit** is popular in Linux
  - **nano** is simple with less functionality

Writing C++ Code

- Compiler
  - checks your code for errors
  - converts the source code to object code: **xx.cpp -> xx.o**
- Linker
  - combines all the object code into an executable
  - **(xx.o, yy.o, zz.o) -> aaa.exe**





First Program

```cpp
//: C02:Hello.cpp
// Saying Hello with C++
#include <iostream>       // Stream declarations
using namespace std;

int main() {
  cout << "Hello, World! I am "
       << 8 << " Today!" << endl;
}
```

Output:

```
Hello, World! I am 8 Today!
```

Insertion and Extraction Operators

- Insertion Operator <<
  - cout << "This is output" << endl;
  - Inserts data into the output stream

- Extraction Operator >>
  - cin >> X;
  - Extracts data from the input stream

More About *iostream*

We can display integers in different bases:

```
//: C02:Stream2.cpp
#include <iostream>
using namespace std;

int main()
{
  // Specifying formats with manipulators:
  cout << “15 in decimal: "
       << dec << 15 << endl;
  cout << "in octal: " << oct << 15 << endl;
  cout << "in hex: " << hex << 15 << endl;
  cout << "a floating-point number: "
       << 3.14159 << endl;
 }
```

Output:

```
15 in decimal: 15
in octal: 17
in hex: f
a floating-point number: 3.14159
```

String Concatenation Example

```
//: C02:Concat.cpp
#include <iostream>
using namespace std;

int main()
{
  cout << "This is far too long to put on a "
    "single line but it can be broken up with "
    "no ill effects\nas long as there is no "
    "punctuation separating adjacent character "
    "arrays.\n";
}
```

Newline

Output:

This is far too long to put on a single line but it can be broken up with no ill effects
as long as there is no punctuation separating adjacent character arrays.

Reading Input

```
//: C02:Numconv.cpp
#include <iostream>
using namespace std;

int main()
{
  int number;
  cout << "Enter a decimal number: ";
  cin >> number; //Read input from user
  cout << "value in octal = 0"
       << oct << number << endl;
  cout << "value in hex = 0x"
       << hex << number << endl;
}
```

Output:

```
Enter a decimal number: 128
value in octal = 0200
value in hex = 0x80
```

String class

- Allows you to manipulate the content of a character array
- Needs to be included at the beginning of a program

```
#include <string>
#include <iostream>
using namespace std;

int main()
{
  string proclamation, day; // Empty strings
  string greeting = "Hello, World."; // Initialized
  string iam("I am"); // Also initialized
  day = "Today"; // Assigning to a string
  proclamation = greeting + " " + iam; // Combining strings
  proclamation += " 8 "; // Appending to a string
  cout << proclamation + day + "!" << endl;
}
```

Output:

```
Hello, World! I am 8 Today!
```

Wake up!

https://youtu.be/nMJdsQL_Bco

File Input/Output

- To read from or write to a file, we need to include:
  - #include <fstream>
- Before **reading** from a file, we need to define and open a file to be read:
  - ifstream myfile(<*file_name*>); //**ifstream:** Stream class to read from *file_name*
- Before **writing** to a file, we need to define and open a file to be written:
  - ofstream myfile(<*file_name*>); //**ofstream:** Stream class to write on *file_name*
- Close the file after finishing the operations with it:
  - myfile.close();

File Input (Read)

- Once a file is opened, we can read the content by lines:
  - getline (myfile, *line*); // read current line of file and put it in *line*
  - It discards the newline character at the end
  - After reading a line, getline will start at the next line when it is called again
  - You don’t need to increment the line number in your code

File Output (Write)

- Once a file is opened, we can write the content onto the file as if writing to console:
  - myfile << "Writing to file is similar\n";
  - Instead of ***cout***, we use the instance variable that contains the file object, in this case ***myfile***

File IO Example

```
// Read/write file
#include <string>
#include <fstream>
using namespace std;

int main()
{
  ifstream input("file.txt"); // Open for reading
  ofstream output("file_out.txt"); // Open for writing
  string myString;
  while(getline(input, myString)) // Discards newline char
  {
    output << myString << "\n"; // ... must add newline back
  }
}
```

File IO Example

```
// Read an entire file into a single string
#include <string>
#include <iostream>
#include <fstream>
using namespace std;

int main() {
  ifstream input("FillString.cpp");
  string myString, line;
  while(getline(input, line))
    myString += line + "\n";
  cout << myString;
}
```

File IO Example

```
// Example using is_open
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  ifstream infile;
  infile.open ("test.txt");
  if (infile.is_open()) // Check if the file is open
  {
    while (!infile.eof()) // Check if it reaches the end of file
        cout << (char) infile.get(); // Read character by character
    infile.close();
  }
  else
  {
    cout << "Error opening file";
  }
  return 0;
}
```

Vector (Array list)

- It works similarly as arrays
- Elements in a vector of size N are accessed by their indices [0…N-1]
- We can change the size of a vector dynamically
  - We don’t need to worry about the size as the number of data grows
- Vector is a **template class**
  - It can work with any data type
  - vector<data_type> myVector
    - vector<int> scores

STL Vector

- There is a vector class in the Standard Template Library in C++
- Member functions of STL Vector class (given a vector V):
  - **resize(n)**: Resize V, so that it has space for n elements
  - **size()**: Return the number of elements in V
  - **front()**: Return a reference to the first element of V
  - **back()**: Return a reference to the last element of V
  - **push_back(e)**: Append a copy of the element e to the end of V, thus increasing its size by one
  - **pop_back()**: Remove the last element of V, thus reducing its size by one
  - **insert(i,e)**: Insert a copy of the element e to the i^{th} position of V
  - **erase(i)**: Remove the element at the i^{th} position of V

Vector Example

```
//: C02:Fillvector.cpp
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

int main()
{
  vector<string> v;
  ifstream in("Fillvector.cpp");
  string line;
  while (getline(in, line)) //getline returns true if read successfully
    v.push_back(line); // Add the line to the end of v
  // Add line numbers:
  for(int i = 0; i < v.size(); i++)
    cout << i + 1 << ": " << v[i] << endl;
}
```

Vector Example

```
//: C02:GetWords.cpp
// Break a file into whitespace-separated words
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

int main()
{
  vector<string> words;
  ifstream in("GetWords.cpp");
  string word;
  while(in >> word) // Extraction operator reads until white space
    words.push_back(word);
  for(int i = 0; i < words.size(); i++)
    cout << words[i] << endl;
}
```

Vector Example

```
//: C02:Intvector.cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
  vector<int> v;
  for(int i = 0; i < 10; i++)
    v.push_back(i);
  for(int i = 0; i < v.size(); i++)
    cout << v[i] << ", ";
  cout << endl;

  for(int i = 0; i < v.size(); i++)
    v[i] = v[i] * 10; // Assignment
  for(int i = 0; i < v.size(); i++)
    cout << v[i] << ", ";
  cout << endl;
}
```

What's the output?

Output

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
0, 10, 20, 30, 40, 50, 60, 70, 80, 90,
```