CSE 165/ENGR 140
Intro to Object Orient Program
Lecture 2 – Programming in C++
Announcement

- Lab #1 on 1/30
- UCMCROPS site is up
- Reading assignment
  - Ch. 3
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

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.
//: 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 is 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
The compiler on your system converts the source code to object code: 
\[(xxx.cpp) -> (xxx.o/xxx.obj)\]
The linker combines all the object code into an executable program. 
\[(xxx.o, yyy.o, zzz.o) -> (aaa)\]

Examples
- Compile and link: g++ hello.cpp -o hello
- Compile only: g++ -c hello.cpp
- Link only: g++ hello.o -o hello
//: 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!
More About `iostream`

```cpp
//: 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;
    cout << "non-printing char (escape): "
        << char(27) << endl;
}
```

*Output:*

15 in decimal: 15
in octal: 17
in hex: f
a floating-point number: 3.14159
non-printing char (escape): ←
String Concatenation Example

```cpp
//: 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";
}

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

```cpp
//: 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";
    cout << oct << number << endl;
    cout << "value in hex = 0x";
    cout << 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.

```c++
//: C02:HelloStrings.cpp
#include <string>
#include <iostream>
using namespace std;

int main() {
    string s1, s2; // Empty strings
    string s3 = "Hello, World."; // Initialized
    string s4("I am"); // Also initialized
    s2 = "Today"; // Assigning to a string
    s1 = s3 + " " + s4; // Combining strings
    s1 += " 8 "; // Appending to a string
    cout << s1 + s2 + "!" << endl;
}
```

**Output:**
Hello, World! I am 8 Today!
To read from or write on a file, we need to include a built-in system class:
- `#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();`
Once a file is opened, we can read the content by lines:

- `getline (myfile, line);` // read current line of file and save it into variable `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

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

int main() {
    ifstream in("Scopy.cpp"); // Open for reading
    ofstream out("Scopy2.cpp"); // Open for writing
    string s;
    while(getline(in, s)) // Discards newline char
        out << s << "\n"; // ... must add it back
}
File IO Example

//: C02:FillString.cpp
// Read an entire file into a single string
#include <string>
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream in("FillString.cpp");
    string s, line;
    while(getline(in, line))
        s += line + "\n";
    cout << s;
}
Example using `is_open`

```cpp
#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`
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\textsuperscript{th} position of V.
- `erase(i)`: Remove the element at the i\textsuperscript{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 << ":: " << v[i] << endl;
}
Vector Example

```cpp
#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) // Read every word in the file
        words.push_back(word);
    for(int i = 0; i < words.size(); i++)
        cout << words[i] << endl;
}
```
//: 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;
}