CSE240 – Introduction to Programming Languages (online)

Lecture 09:
Programming with C++ | Getting Started

Javier Gonzalez-Sanchez

javiergs@asu.edu
javiergs.engineering.asu.edu
Office Hours: By appointment

jgs

Outline

1. **Getting Started**
   - namespace,
   - classes,
   - scope(public, protected and private),
   - input and output,
   - scope resolution operator

2. **Memory management**
   - static,
   - constructor and destructor,
   - delete

3. **Inheritance and polymorphism**

Anatomy of a Program

- Data and the operations, that manipulate data, are **encapsulated** in classes.

```
class FooBar {

  int variable;

  void method() {

  }

};

int x;

int main(){

  FooBar anObject;

}
```

Anatomy of a Program

- Data and the operations, that manipulate data, are **encapsulated** in classes.
- You can defined **public**, **private**, and **protected** components. And, encapsulated data can only be accessed (from outside the class) through their **interface** (operations defined as **public**)

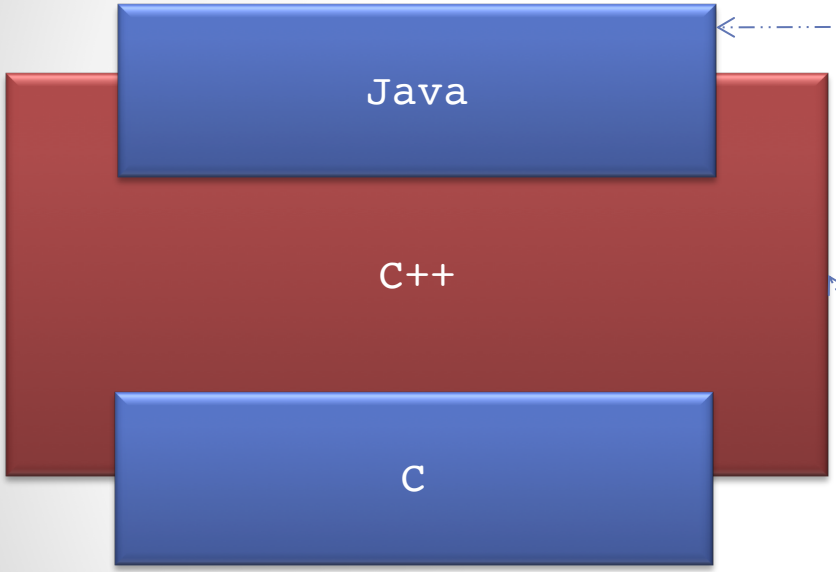
```
class FooBar {

private:
   char a;
   int variable;
protected:
   int anotherVariable;
public:
   void method1() { }
   void method2() { }
};

int x;
int main(){
   FooBar anObject;
}
```

Anatomy of a Program

- Data and the operations, that manipulate data, are **encapsulated** in classes.
- You can defined **public**, **private**, and **protected** components. And, encapsulated data can only be accessed (from outside the class) through their **interface** (operations defined as **public**)
- Classes are organized in **hierarchies: use and inherit.**

```
class Shape {

};

class Rectangle: public Shape {
  private:
    FooBar a;
};

class FooBar {

};

int main(){
  FooBar anObject;
}
```

Inside each C++ class, it is similar to C program



In **Java**,
attributes and **methods** are always in one file.

Java have all related information in one place.

In **C++**,
implementations of **member functions** can be **in** the **same file** than the class definition (for short functions) **or outside** of the class definition.

C++ argument: structurally clearer to separate implementation from definition

Example in One File: queue.cpp


```
1.  #include <iostream>
2.  using namespace std;
3.  class Queue {
4.    private:
5.      int queue_size;
6.    protected:
7.      int *buffer;
8.      int front;
9.      int rear;
10.   public:
11.     Queue(int v) {
12.       cout<<"constructor\n";
13.     }
14.     void enqueue(int v) {
15.       cout<<"enqueue\n";
16.     }
17.     int dequeue(void){
18.       cout<<"dequeue\n"; return 5;
19.     }
20. };
```

```
21. int main(){
22.   Queue q1(5);
23.   Queue *q2 = new Queue(5);
24.
25.   // Access Object
26.   q1.enqueue(2);
27.   q1.enqueue(8);
28.
29.   // Access Object Pointer
30.   q2->enqueue(25);
31.   int x = q2->dequeue();
32.   return 0;
33. }
```

Example in Two Files: time.h


```
1.  class Time {

2.    public:
3.      Time(); // constructor
4.      void setTime( int, int ); // set hour, minute
5.      void printMilitary();     // print military time format
6.      void printStandard();     // print standard time format
7.
8.    private:
9.      int hour;      // 0 - 23
10.     int minute;    // 0 - 59

11. };
```

Example in Two Files: time.cpp

jgs

```
1.  #include <iostream>
2.  #include "time.h"
3.  using namespace std;

4.  Time::Time() {  // constructor
5.    hour = minute = 0;
6.  }

7.  void Time::setTime( int h, int m) {  // Set a new mil Time
8.    hour = ( h >= 0 && h < 24 ) ? h : 0;
9.    minute = ( m >= 0 && m < 60 ) ? m : 0;
10. }

11. void Time::printMilitary() {  // Print time in military format
12.   cout << ( hour < 10 ? "0" : "" ) << hour << ":"
13.        << ( minute < 10 ? "0" : "" ) << minute; // add "0"
14. }
```

Example in Two Files: time.cpp

```
15. void Time::printStandard() {     // Print in standard format
16.   cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
17.        << ":" << ( minute < 10 ? "0" : "" ) << minute
18.        << ( hour < 12 ? " AM" : " PM" ) << endl; //endl is equal to "\n"
19. }
20.
21. int main() {
22.   Time t;  // new is not mandatory - instantiate object t of class Time
23.   cout << "The initial military time is ";
24.   t.printMilitary();
25.   cout << "\nThe initial standard time is ";
26.   t.printStandard();
27.   t.setTime(15, 27);
28.   cout << "\n\nMilitary time after setTime is ";
29.   t.printMilitary();
30.   cout << "\nStandard time after setTime is ";
31.   t.printStandard();
32.   return 0;
33. }
```

CSE240 – Introduction to Programming Languages (online)

Javier Gonzalez-Sanchez
javiergs@asu.edu
Fall 2017