Lecture 5

Classes and Objects

University of Anbar
College of Computer Science and Information Technology
Department of Computer Science
Object Oriented Programming
Second Class
Dr. Ruqayah R. Al-Dahhan

Outlines:

- Classes
- Classes that use variables of other classes
- Objects
- Static members of classes

Classes

Functions defined within a `class` or `struct` are `inline` by default.

`inline` functions should be small, and those that are defined within a `class` should be one or two lines at most.

We can define `class` member functions outside the class definition. They are then no longer `inline` by default. Only the function prototype needs to be included within the `class`.

Classes

```
class Point {
 public:
    void print(); // prototype inside class
 private:
    int x,y;
};


void Point::print() {  // embodiment elsewhere
   cout << "(" << x << "," << y << ")";
}
```

The scope resolution operator `::` is used to define functions outside the class declaration.

Classes

```
class Point {
 public:
    void print();                  // not inline
    void init(int u, int v){ // inline
       x = u;y = v;
    }
 private:
   int x,y;
};

void Point::print() {
   cout << "(" << x << "," << y << ")";
}
```

Classes

A **class** is a ***user-defined type*** that contains ***data*** as well as the set of ***functions*** that manipulate the data.

We often have a collection of "accessor" methods or functions - sometimes known as "get" and "set" methods or functions.

Note: Data members of a class cannot be initialized when they are declared inside the class.

These data members should be initialized using specific functions: "set" functions (like `init()` in the `Point` class).

Classes

Member functions can also be **overloaded**.

```
class Point {
 public:
   void init(int u, int v) {
      x = u; y = v;
   }
   void print();
   void print(int s);

 private:
   int x,y;
};
```

Classes

```
void Point::print() {
   cout << “(” << x << “,” << y << “)”;
}

void Point::print(int s) {
   cout << s;
   print();
}
```

Classes

```
int main(){
    Point w;
    w.init(4,7);
    w.print();
    cout << endl;
    w.print(1);
 }
```

```
Output:                 (4,7)
                        (4,7)
```

Class scope

Within the second form of the print function, there is a call to the other function print (it has different arguments).

```
void Point::print(int s) {
    cout << s;
    print();    //No scope operator is required here.

}
```

Class scope

If there is a global function `print`, **not contained within any class**, and we want to call it within a class member function, then we use the scope operator on its own - external scope.

```
void print() {
   cout << " The global print function";
}

void Point::print(int s) {
   cout << s;
   ::print();
}
```

Classes can contain other classes.

```
char c;

class Y {
 public:
   char c;
};

class X {
 public:
   char c;
   Y y;
};
```

```
int main () {
   X x;
   c = 'A';
   x.c = 'B';
   x.y.c = 'C';
}
```

Objects

C++ programming
•is **object-oriented**- the programming unit is the **class**.

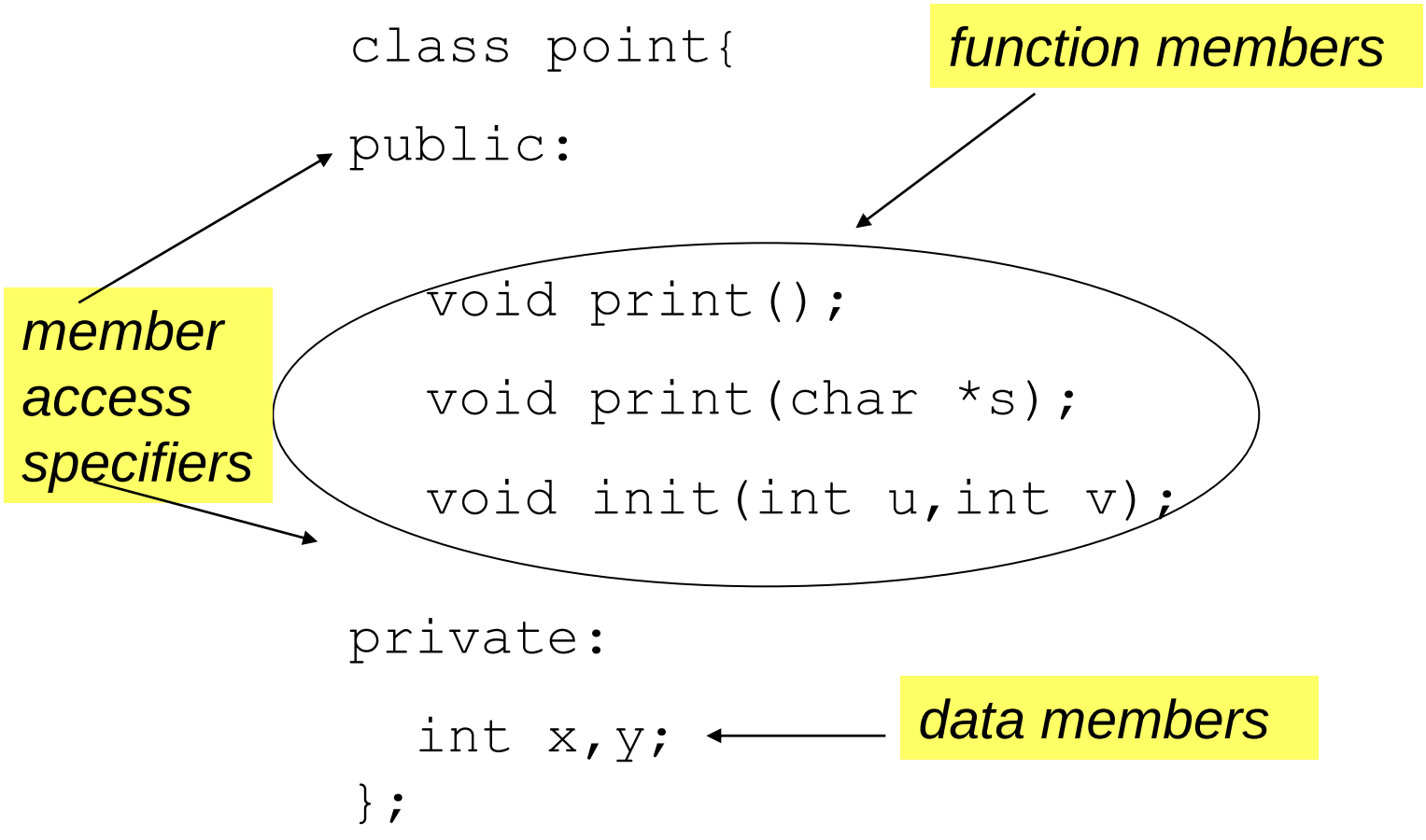
An instance of a type is called an **object**.

```
//j an integer object
int j;

//w is a point object
point w;
```

Objects

A class is a blueprint for all its objects.



Static members of classes

If a variable within a class is declared `static`, then there is only one instance of that variable in the program.

A `static` variable is common to **all** class variables.

(Unlike normal instance variables which are separate for each instantiation)

Static members of classes

```
class P {
 public:
   static char c;
};

char P::c = 'W';

int main () {
   P x,y;
   cout << x.c;
   x.c = 'A';
   cout << y.c;
}
```

Correct -
but missleading:
`x.c` and `y.c` are the same thing.

Static members of classes

It is better to refer to the static member as `P::c`

```
int main () {
    P x;
    P::c = 'A';
    cout << P::c;
}
```

Summary

A class in C++ is a form of struct whose default access specification is private.

Classes have **public** and **private** members that provide data hiding.

The scope resolution operator **::** allows member function of various classes to have the same names as used globals.

Static data members are shared by all variables of that class type.