

# Constructors and Accessor Functions

Lecture 10

# Outline

- What is a constructor
- Comments on constructors
- Overloading constructors
- Classes & ADTs

# What is a constructor?

- ◆ It is a member function which initializes a class.
- ◆ A constructor has:
  - (i) the same name as the class itself
  - (ii) no return type

```
class rectangle {  
    private:  
        float height;  
        float width;  
        int xpos;  
        int ypos;  
  
    public:  
        rectangle(float, float);    // constructor  
        void draw();                // member  
function  
        void posn(int, int);        // member  
function  
        void move(int, int);        //member  
function  
};  
  
rectangle::rectangle(float h, float w)  
{  
    height = h;  
    width = w;  
    xpos = 0;  
    ypos = 0;  
}
```

# Comments on constructors

- ◆ A constructor is called automatically whenever a new instance of a class is created.
- ◆ You must supply the arguments to the constructor when a new instance is created.
- ◆ If you do not specify a constructor, the compiler generates a default constructor for you (expects no parameters and has an empty body).

## Comments on constructors (cont.)

```
void main()
{
    rectangle rc(3.0, 2.0);

    rc.posn(100, 100);
    rc.draw();
    rc.move(50, 50);
    rc.draw();
}
```

- ◆ *Warning:* attempting to initialize a data member of a class explicitly in the class definition is a syntax error.

# Overloading constructors

- ◆ You can have more than one constructor in a class, as long as each has a different list of arguments.

```
class rectangle {  
    private:  
        float height;  
        float width;  
        int xpos;  
        int ypos;  
    public:  
        rectangle(float, float); // constructor  
        rectangle();             // another constructor  
        void draw();             // draw member function  
        void posn(int, int);     // position member function  
        void move(int, int);     // move member function  
};
```

# Overloading constructors (cont.)

```
rectangle::rectangle()  
{  
    height = 10;  
    width = 10;  
    xpos = 0;  
    ypos = 0;  
}
```

```
void main()  
{  
    rectangle rc1(3.0, 2.0);  
    rectangle rc2();  
  
    rc1.draw();  
    rc2.draw();  
}
```




# Classes & ADTs

## ◆ Accessor Functions

- Functions that give you access to the values of the private member variables.

```
class School
{
public:
    ...
private:
    int    NumOfStudents;
    int    NumOf Classes;
    double Area;
}
```

```
int get_Students();
//Return the number of students in a school
int get_Classes();
//Return the number of classes in a school
double get_Area();
//Return the area of a school
```



# Classes & ADTs

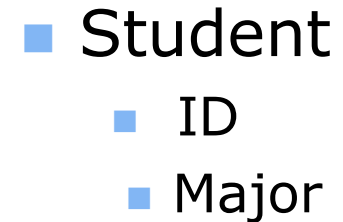
## ◆ Private members → need for Accessor Functions

```
int get_id();  
//returns the student id
```

```
char get_major();  
//returns the student major
```

```
void set_id(int new_id);  
//assigns a value to student id
```

```
void set_major(char new_major);  
//assigns a value to student major
```



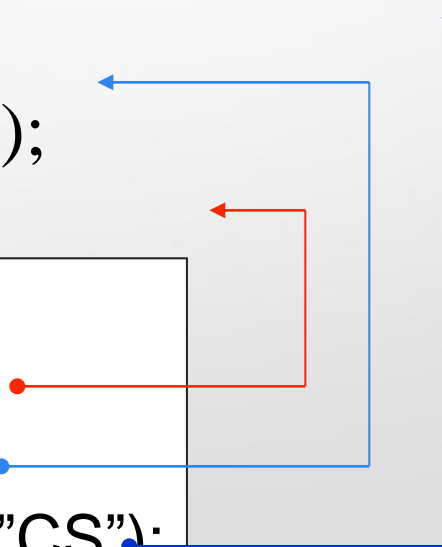
```
Student  
{  
public:  
    int id;  
    char major;  
};
```

# Classes & ADTs

## ◆ Can we overload member functions?

- void set(int the\_id, char the\_major[2]);
- void set(int the\_id);
- void set(double score);

```
Student new_student;  
new_student.set (16.0);  
new_student.set (555);  
New_student.set ((999,"CS");
```



*Search for matching data types and/or number of parameters*

# Classes & ADTs

```
Class DayOfYear
{
public:
    void    output();
private:
    int  month;
    int  day;
};
```

## Restriction:

Once you make a member variable private, the only way to access it or change its value is by using one of the member functions.

*private member variables*

```
int main()
{
    DayOfYear Today;
    cin >> Today.month;
    cout << Today.month;
    If (Today.month ==1)
        cout << "January";
}
```

**ILLEGAL!**

# Classes & ADTs

Class Sample

```
{  
public:  
    int  variable;  
    void  output();  
    void  input();  
private:  
    int  month;  
    int  day;  
    void  doStuff();  
};
```

*public members*

*private members*

**Public members** can be used in the main body of your program or in the definition of any function, even a non- member function.

# Summary

- A constructor method has the same name as the class.
- Constructors are used to create objects of the class.
- A no parameter default constructor is there if no user defined constructor exists.
- Private member methods are used to allow access to private members from outside the class.
- Overloaded methods have the same name but different list of arguments.
- Classes can have overloaded constructors or methods.



**Thank You**