

Computer Programming

Dr. Deepak B Phatak

Dr. Supratik Chakraborty

Department of Computer Science and Engineering
IIT Bombay

Session: More on Structures

Quick Recap of Relevant Topics

- Brief introduction to object-oriented programming
- Structures as collections of variables of possibly different data types
- Accessing members of structures
- Programming using simple structures

Overview of This Lecture

- Common conventions when speaking of structures
- More features of structures
 - Structures as members of other structures
 - Initializing members of structures in structures
- Disallowed structure definitions
- Visibility of structure definitions in C++ programs

Acknowledgment

- Some examples in this lecture are from
An Introduction to Programming Through C++
by Abhiram G. Ranade
McGraw Hill Education 2014
- All such examples indicated in slides with the citation
AGRBook

Common Conventions

```
struct MyStructType {  
    int x;  
    char y;  
};  
  
MyStructType myVariable;
```

Common Conventions

```
struct MyStructType {  
    int x;  
    char y;  
};  
  
MyStructType myVariable;
```

“structure” refers to a specific structure type

The definition of structure MyStructType is given here.

Common Conventions

```
struct MyStructType {  
    int x;  
    char y;  
};  
  
MyStructType myVariable;
```

“structure” refers to a specific object of a structure type

The structure myVariable is used in the program.

Common Conventions

```
struct MyStructType {  
    int x;  
    char y;  
};  
  
MyStructType myVariable;
```

“structure” refers to a
an arbitrary object of
type MyStructType

**A structure of type
MyStructType needs 5
bytes of storage**

Common Conventions

```
struct MyStructType {  
    int x;  
    char y;  
};  
  
MyStructType myVariable;
```

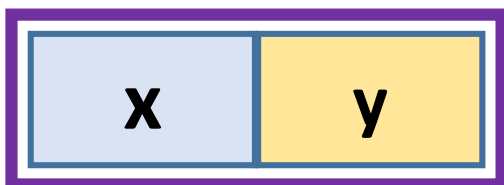
Member “x” of structure
“myVariable” is like an
object/variable of type int

myVariable.x = 12;

Points and Disks in 2D space [Ref. AGRBook]

- We want to represent points and disks in 2-dimensional space
- Every point has an x-coordinate and a y-coordinate
- Every disk has a center (point) and a radius

```
struct Point {  
    double x, y;  
};
```



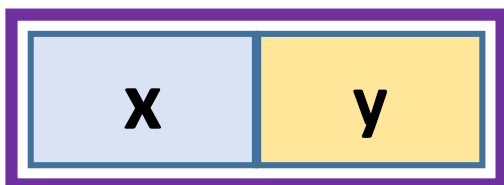
```
struct Disk {  
    Point center;  
    double radius;  
};
```



Points and Disks in 2D space [Ref. AGRBook]

- We want to represent points and disks in 2-dimensional space
- Every point has an x-coordinate and a y-coordinate
- Every disk has a center (point) and a radius

```
struct Point {  
    double x, y;  
};
```



```
struct Disk {  
    Point center;  
    double radius;  
};
```



Accessing Members of Structures in Structures

Point p1;

Disk d1;

p1



```
struct Point {  
    double x, y;  
};
```

d1



```
struct Disk {  
    Point center;  
    double radius;  
};
```

Accessing Members of Structures in Structures

Point p1;

Disk d1;

p1.x = 0.5; p1.y = 0.9;

p1



```
struct Point {  
    double x, y;  
};
```

d1



```
struct Disk {  
    Point center;  
    double radius;  
};
```

Accessing Members of Structures in Structures

Point p1;

Disk d1;

p1.x = 0.5; p1.y = 0.9;

d1.center = p1;

p1



```
struct Point {  
    double x, y;  
};
```

d1



```
struct Disk {  
    Point center;  
    double radius;  
};
```

Accessing Members of Structures in Structures

Point p1;

Disk d1;

p1.x = 0.5; p1.y = 0.9;

d1.center = p1;

d1.radius = 3.2;

p1



```
struct Point {  
    double x, y;  
};
```

d1



```
struct Disk {  
    Point center;  
    double radius;  
};
```

Accessing Members of Structures in Structures

Disk d1;

d1



```
struct Point {  
    double x, y;  
};
```

```
struct Disk {  
    Point center;  
    double radius;  
};
```


Accessing Members of Structures in Structures

Disk d1;

d1.center.x = 0.5;

```
struct Point {  
    double x, y;  
};
```

d1



```
struct Disk {  
    Point center;  
    double radius;  
};
```

d1: Object of type Disk

Accessing Members of Structures in Structures

Disk d1;

d1.center.x = 0.5;

```
struct Point {  
    double x, y;  
};
```

```
struct Disk {  
    Point center;  
    double radius;  
};
```

d1



d1.center: Member “center” of d1 can be used as an object of type Point

Accessing Members of Structures in Structures

Disk d1;

d1.center.x = 0.5;

d1



```
struct Point {  
    double x, y;  
};
```

```
struct Disk {  
    Point center;  
    double radius;  
};
```

d1.center.x: Member “x” of d1.center can be used as an object of type double

Accessing Members of Structures in Structures

Disk d1;

d1.center.x = 0.5;

d1.center.y = 0.9;

d1.radius = 3.2;

d1



```
struct Point {  
    double x, y;  
};
```

```
struct Disk {  
    Point center;  
    double radius;  
};
```

Initializing Members of Structures in Structures

Disk d1 = {{0.5, 0.9}, 3.2};

const Disk d2 = {{1.0, 2.0}, 3.0};

d1



d2

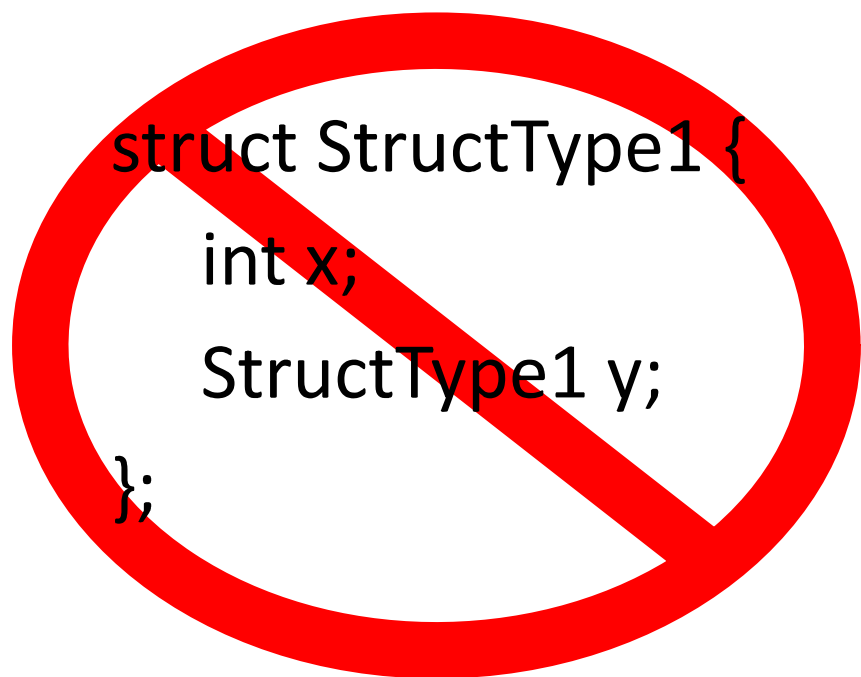


```
struct Point {  
    double x, y;  
};
```

```
struct Disk {  
    Point center;  
    double radius;  
};
```

Disallowed Structure Construction

- Structure type “StructType1” cannot have a member of the same structure type “StructType1”



Storage required for an object of type StructType1 would be infinite!

Visibility of Structure Types

- Where should we define structure types?
- If a structure type is used only in one function, it can be defined in the body of the function

```
void doSomethingWithDisks()  
{ struct Point { double x, y; };  
  struct Disk {Point center; double radius};  
  // Code that does something with points and disks  
  return;  
}
```

Visibility of Structure Types

- If a structure data type is used in multiple functions, it must be defined **outside and before** the functions in program file

```
struct Point {double x, y};
```

```
struct Disk {Point center; double radius};
```

```
void doOneThingWithPointsAndDisks() { ... return; }
```

```
void doAnotherThingWithPointsAndDisks() { ... return; }
```

```
void updatePointsAndDisks(Point &p, Disk &d) { ... return; }
```


Summary

- Common conventions when speaking of structures
- Additional features of structures
 - Structures as members in other structures
 - Accessing and initializing members of structures in structure
- Illegal to have a structure with a member of the same structure type
- Visibility of structure types in a C++ program