Object Oriented Programming

Lecture

this Pointer

this Pointer

```
class Rectangle
{
    int width, height;
public:
    void set_width(int a);
    void set_height(int b);
    int area();
};
```

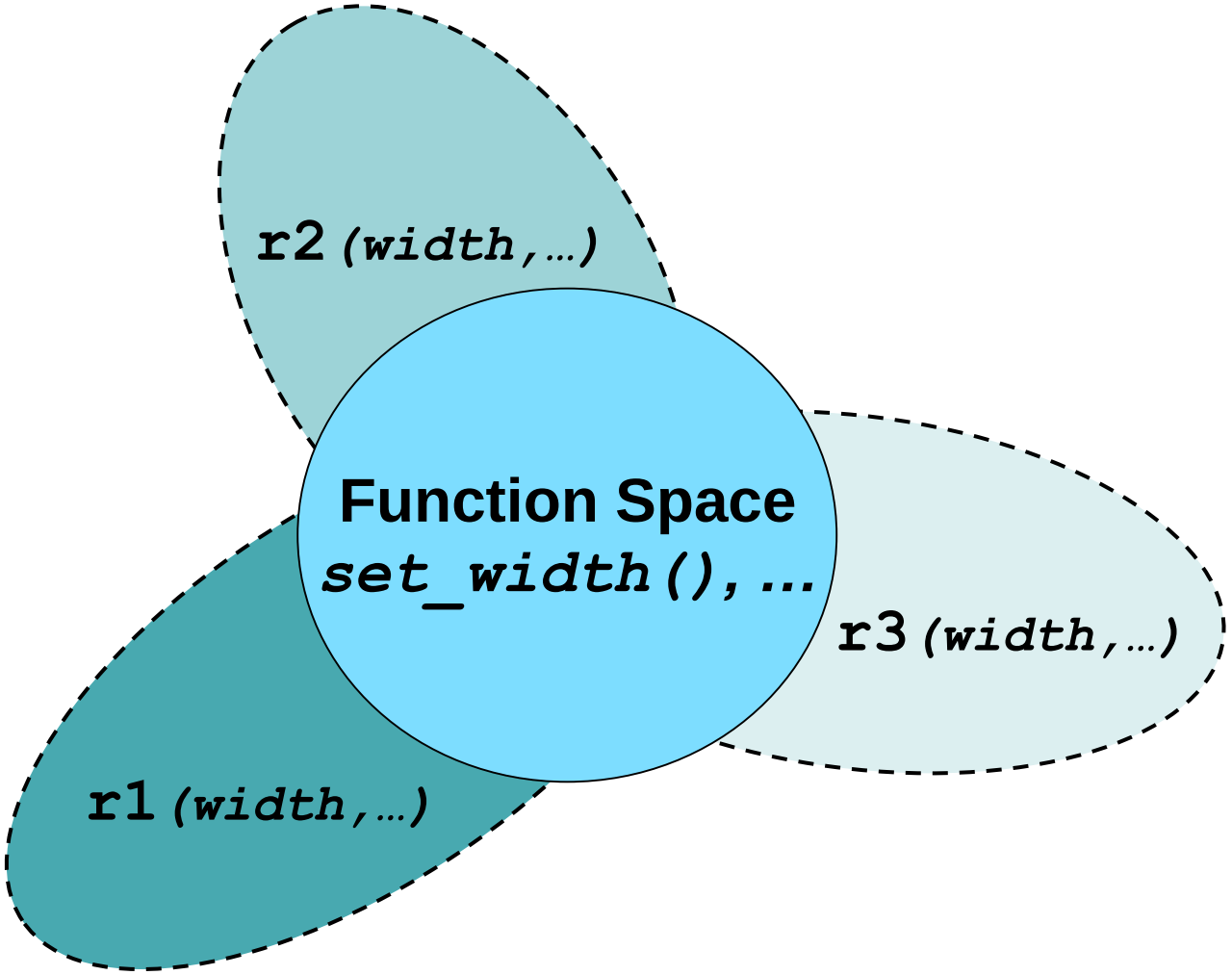
```
void Rectangle::set_width(int a)
{
    width = a;

}

void Rectangle::set_height(int b)
{
    height = b;

}

int Rectangle::area()
{
    return width * height;
}
```

`this` Pointer

- The compiler reserves space for the functions defined in the class
- Space for data is not allocated (*since no object is yet created*)

this Pointer

- Rectangle ***r1, r2, r3;***

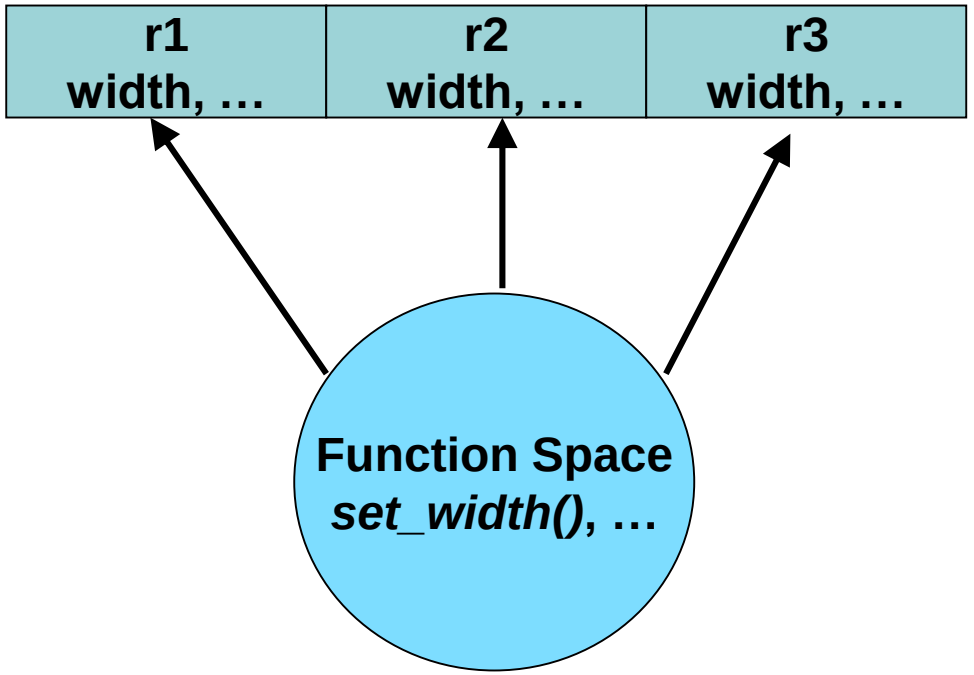


this Pointer

- Function space is **common** for every object
- Whenever a new object is created:
  - Memory is reserved for variables only
  - Previously defined functions are used over and over again

this Pointer

- **Memory layout for objects created:**



Question

How does the functions know on which object to act?

this Pointer

Answer:

- Address of each object is passed to the calling function
- This address is deferenced by the functions and hence they act on correct objects

r1 width, …	r2 width, …	r3 width, …	r4 width, …
address	*address*	*address*	*address*

The variable containing the “self-address” is called ***this*** pointer

Passing *this* Pointer

- Whenever a function is called the ***this*** pointer is passed as a parameter to that function
- Function with *n* parameters is actually called with *n+1* parameters

Example

```
void Rectangle::set_width(int a)
```

is internally represented as

```
void Rectangle::set_width(int a, Rectangle* const this)
```

Compiler Generated Code

```
Rectangle::set_width(int a)
{
 width = a;
}
```

is internally represented as

```
Rectangle::set_width(int a, Rectangle* const
 this)
{
 this->width = a;
}
```

this Pointer

There are situations where designer wants to use ***this*** pointer **explicitly**

Case 1: When local variable's name is same as member's name

```
class Rectangle
{
    int width, height;
public:
    void set_width(int width);
    void set_height(int height);
    int area();
};
```

```
void Rectangle::set_width(int width)
{
    width = width;

}

void Rectangle::set_height(int height)
{
    height = height;

}

int Rectangle::area()
{
    return width * height;
}
```

Case 1: When local variable's name is same as member's name

```
class Rectangle
{
    int width, height;
public:
    void set_width(int width);
    void set_height(int height);
    int area();
};
```

```
void Rectangle::set_width(int width)
{
    this->width = width;

}

void Rectangle::set_height(int height)
{
    this->height = height;

}

int Rectangle::area()
{
    return width * height;
}
```

Case 2: To return reference to the calling object

```
class Rectangle
{
    int width, height;
public:
    Rectangle& set_width(int width);
    Rectangle& set_height(int height);
    int area();
};
```

Rectanle& is the return type that the return *this i.e the object is of Rectangle data type that is it's class

```
Rectangle& Rectangle::set_width(int width)
{
    this->width = width;
    return *this;

}
```

Rectangle& is the return type that the return *this i.e the object is of Rectangle data type that is it's class

```
Rectangle& Rectangle::set_height(int height)
{
    this->height = height;
    return *this

}

int Rectangle::area()
{
    return width * height;
}
```

Case 2: To return reference to the calling object

```
class Rectangle
{
    int width, height;
public:
    Rectangle& set_width(int width);
    Rectangle& set_height(int height);
    int area();
};
```

```
int main()
{

Rectangle r1;
r1.set_width(10).set_height(10);
cout << r1.area();
return 0;
}
```

```
Rectangle& Rectangle::set_width(int width)
{
    this->width = width;
    return *this;

}

Rectangle& Rectangle::set_height(int height)
{
    this->height = height;
    return *this

}

int Rectangle::area()
{
    return width * height;
}
```

Case 2: To return reference to the calling object

```
class Rectangle
{
    int width, height;
public:
    Rectangle& set_
    Rectangle& set_
    int area();
};
```

```
Rectangle& Rectangle::set_width(int width)
{
    this->width = width;
    return *this;
```

```
int height)
```

When a reference to a local object is returned, the returned reference can be used to ***chain function*** **calls** on a single object.

```
int main()
{

Rectangle r1;
r1.set_width(10).set_height(10);
cout << r1.area();
return 0;
}
```

```
}

int Rectangle::area()
{
    return width * height;
}
```

Question

```
MyClass t1, t2;
MyClass t3 = t1;  // ----> (1)
t2 = t1;          // -----> (2)
```

Which of the following two statements call copy constructor and which one calls assignment operator?

Thanks a lot