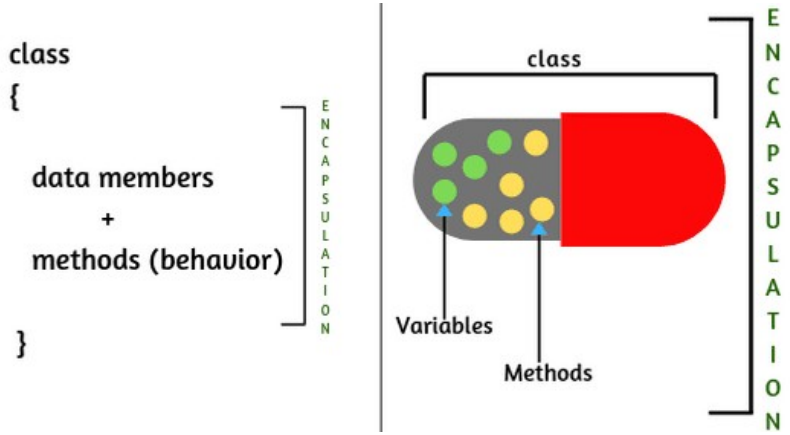
# Object Oriented Programming
# Data Members (Constant, Static)

CS(217) Object Oriented Programming

Abeeda Akram

# Class Data Members (TYPES)

1. Simple
2. Pointers Dynamic Memory
3. Constant
4. Static and Static Member functions



# Data Members(Constant)

- Does not change once initialized throughout life time of an object
- **Examples:**
  1. Student - Roll Number
  2. Employee – ID
  3. Point (fix axis) – X or Y (moves in line horizontal or vertical)
- **Initialization**
  - **With one fix value for all objects** (initialize in class definition)
  - **Different values for different objects** (initialize in constructors)

    **Use Member initializer syntax**

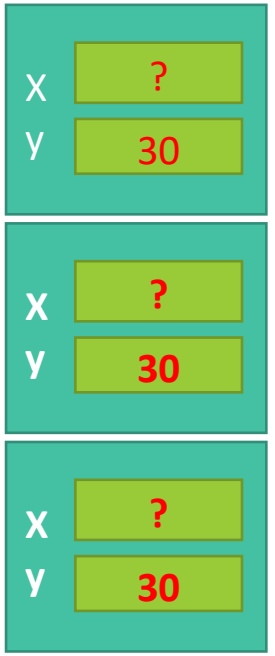
```
: Data member Name (value or variable or Expression)
```

# Data Members(Constant) Con..

- **With one fix value for all objects** (initialize in class definition)

```
class Point {
        int x;
        const int y = 30;
};
```

```
void main(){

    Point p;

    Point p2;

    Point p3;
}
```

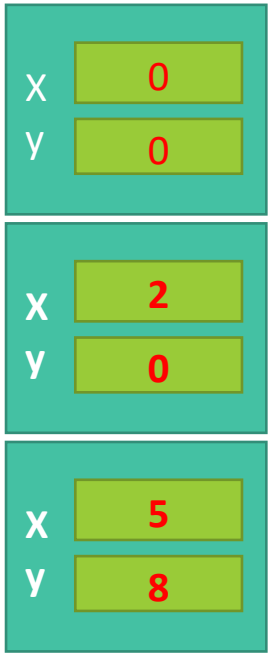


# Data Members(Constant) Con..

- **With one fix value for all objects** (initialize in class definition)

```
class Point {
        int x;
        const int y = 30;
    public:
    //Default constructor
    Point();
    //Parameterized constructors
    Point(int x);
    Point(int x, int y);
};
Point::Point(){ x = 0; y = 0;}
Point::Point(int x){ this->x = x;}
Point::Point(int x, int y){
    this->x = x;
    this->y = y;
}
```

```
void main(){

    Point p;

    Point p2(2);

    Point p3(5,8);

}
//Compiler Error: cannot change the
constant member using assignment once
initialized
```

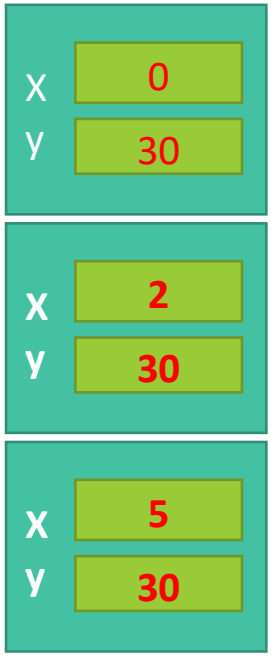


# Data Members(Constant) Con..

- **With one fix value for all objects** (initialize in class definition)

```
class Point {
        int x;
        const int y = 30;
    public:
    //Default constructor
    Point();
    //Parameterized constructors
    Point(int x);
    Point(int x, int y);
};
Point::Point(){ x = 0;}
Point::Point(int x){ this->x = x;}
Point::Point(int x, int y){
    this->x = x;
}
```

```
void main(){

    Point p;

    Point p2(2);

    Point p3(5,8);

}
//Same constant members data for all
objects
```



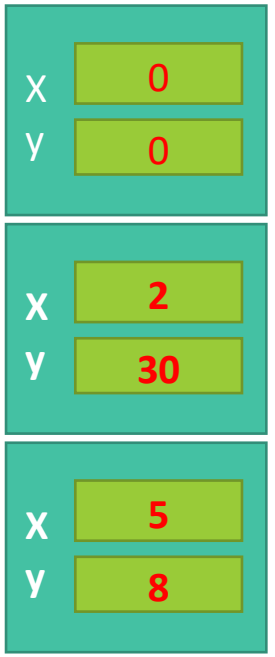
# Data Members(Constant) Con..

- **Different values for different objects** (initialize in constructors)

**Use Member initializer syntax** : Data member Name (value or variable or Expression)

```
class Point {
        int x;
        const int y = 30;
    public:
    //Default constructor
    Point();
    //Parameterized constructors
    Point(int x);
    Point(int x, int y);
};
Point::Point():y(0){ x = 0;}
Point::Point(int x){ this->x = x;}
Point::Point(int x, int y) :y(y)
{
    this->x = x;
}
```

```
void main(){

    Point p;

    Point p2(2);

    Point p3(5,8);

}
```




```
//Different constant members data for
all objects
```

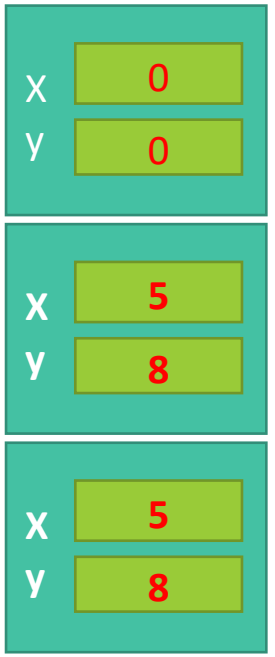
# Data Members(Constant) Con..

- **Different values for different objects** (initialize in constructors)

**Use Member initializer syntax** : Data member Name (value or variable or Expression)

```
class Point {
        int x;
        const int y;
    public:
    Point(int x = 0, int y = 0);
    Point(const Point & p);
};
//Default and Parameterized
Point::Point(int x = 0, int y =
0) :y(y) { this->x = x;}
//Copy Constructor
Point::Point(const Point & p) :y(p.y)
    if(this ! = &p)
        this->x = p.x;
}
```

```
void main(){

    Point p;

    Point p2(5,8);

    Point p3 = p2;

}

//Different constant members data for
all objects
```



# Static Variables

- Special variables which Exist between function calls
- Use **Static** (reserve word in C++) for creation
  1. Life time
     - **Starts** when declaration statement executed
     - **Ends** when Main ends (Same as global variables)
  2. Scope **Local** or **global** as normal variables
  3. Used to track function level tasks for changing function behavior accordingly

```
void fun()
{
    static int a = 10;
     a++;
     cout << a;
}
```

```
void main()
{
     fun();
     fun();
     fun();
}
```

1. **How many time function called?**
2. **Functions can communicate in future time with same functions.**
3. **Can Change behavior: Do one operation in first ten calls and then other for rest of calls.**

# Data Members (Static)

Have only one copy at class level

1. **Their memory is allocated with class definition**
2. **Can be accessed without creating objects**
3. **Shared** between all objects
4. All objects can read and write into that memory

**Use:** Objects can **Communicate** with one another using static members

- Accessible
  - by name in all member functions as normal data members
  - through all objects as normal members using member access operator (**.**)
- **If declared public**
  - Can be initialized and updated from outside the class
  - Also accessible at class level with scope resolution operator (**::**)

# Data Members (Static) Con..

```
class Point {
        const int y;
    public:
        static int x;
    Point(int x = 0, int y = 0);
    void printPoint() const;
};

// accessible as normal data members
Point::Point(int x = 0, int y = 0):y(y)
{this->x = x;}

void Point::printPoint() const {
    cout<< " X: " << this->x;
    cout<< " Y: " << this->y;
}
```


```
void main(){
    //Access at class level without object
    Point::x = 10
    cout << Point::x;

    Point p;
    p.printPoint();
    //only non-static members become
    object property

    Point p2(5,8);
    p2.printPoint();

    p2.x = 33;
    //accessible as normal data members at
    object level
}
```


# Data Members (Static) Con..

```
class Point {
        const int y;
    public:
        static int x;
    Point(int x = 0, int y = 0);
    void printPoint() const;
};
//Can be initialized outside the class
int Point::x = 50;

Point::Point(int x = 0, int y = 0):y(y)
{this->x = x;}
void Point::printPoint() const {
    cout<< " X: " << this->x;
    cout<< " Y: " << this->y;
}
```


```
void main(){


    Point p;
    p.printPoint();


    Point p2(5,8);
    p.printPoint();

}
```


# Static functions

Used to initialize and update **private static** members at class level

- Can be invoked using
  1. Class name and scope resolution operator (**::**) outside class (**If declared public)**
  2. At object level like other member functions

Cannot access any thing at object level

1. **Cannot call non-static functions**
2. **Cannot read or write into non-static data members**
3. **Do not have this pointer**
4. **Cannot be declared as const function.**

Non Static functions can call static functions

# Data Members (Static) and Static functions

```
class Point {
        const int y;
        static int x;
    public:
    //static function
        static void incrementX();
        static void printX();
};
//Initialized outside the class
int Point::x = 50;
//Do not use static while definig
void Point::incrementX() {  x++; }

void Point:: printX()
{ cout << Point::x; }
```


```
void main(){

    Point:: incrementX();
    Point:: printX();

    Point p;
    p.printX();
    p.incrementX();


    Point p2(5,8);
    p.printPoint();

}
```


# Data Members (Static) and Static functions

```
class Point {
        const int y;
        static int x;
    public:
    //static function
        static void incrementX();
        static void printX();
};
//Initialized outside the class
int Point::x = 50;
//Do not use static while definig
void Point::incrementX() {  x++; }

void Point:: printX()
{ cout << x; }
```


```
void main(){

    Point:: incrementX();
    Point:: printX();

    Point p;
    p.printX();
    p.incrementX();


    Point p2(5,8);
    p.printPoint();

}
```