Object Oriented Programming

Classes: constructor overloading and `this` keyword

Mr. Usman Wajid

usman.wajid@nu.edu.pk

February 22, 2023

National University
of Computer & Emerging Sciences

this keyword

this keyword

It will overwrite the access from local copy of a variable to the global variable

```
Student(int rollNo, string name, char section)
    {
        this->rollNo = rollNo;
        this->name = name;
        this->section = section;
}
```

this keyword Example

```
class Student {

        private:
        int rollNo=0;
        string name="N/A";
        char section='-';
        public:
        Student(int rollNo, string name, char section){
                this->rollNo = rollNo;
                this->name = name;
                this->section = section;
        }
        void display();
};

void Student::display(){
        cout<<"roll no: "<<rollNo<<endl;
        cout<<"name: "<<name<<endl;
        cout<<"section: "<<section<<endl;
}

int main() {

        Student ali={1,"ali imran", 'A'};
        ali.display();

        return 0;

}
```

Default Values of Constructor Parameter

- If the values are not passed to the constructor parameters while declaring an object, then the default parameters will be initialized
- Example,

```
Student(int rollNo=0, string name="N/A", char
    section='-'){
        this->rollNo = rollNo;
        this->name = name;
        this->section = section;
}
```

Default Values of Constructor Parameter Example

```
class Student {
        private:
        int rollNo=0;
        string name="N/A";
        char section='-';
        public:
        Student(int rollNo=0, string name="N/A",
              char section='-'){
                 this->rollNo = rollNo;
                 this->name = name;
                 this->section = section;
        }
        void setRollNo(int x){  rollNo = x;
        }
        void setName(string y){ name = y;
        }
        void setSection(char z){section = z;
        }
        int getRollNo(){return  rollNo;
        }
        string getName(){return  name;
        }
        char getSection(){return  section;
        }
        void display(){
        cout<<"roll no: "<<getRollNo()<<endl;
        cout<<"name: "<<getName()<<endl;
        cout<<"section: "<<getSection()<<endl;
        }
};
```

```
main() {

    Student ali;
    ali.setName("ali imran");
    ali.display();
    return 0;
```

Constructor Overloading

Constructor Overloading

Similar to function overloading, constructor can also be overload, i.e., a constructor with same name but with different parameters such as type and number of arguments

```
Student(){
        this->rollNo = 0;
        this->name = "N/A";
        this->section = '-';
}
Student(int rollNo){
        this->rollNo = rollNo;
        this->name = "N/A";
        this->section = '-';
}

Student(string name){
        this->rollNo = 0;
        this->name = name;
        this->section = '-';
}

Student(char section){
        this->rollNo = 0;
        this->name = "N/A";
        this->section = section;
}
```

```
int main() {

        Student ali;
        ali.display();

        Student zain={02};
        zain.display();

        Student waqas={"Waqas"};
        waqas.display();

        Student aysha={'D'};
        aysha.display();
}
```

Home Practice

- Make a class Triangle with the following attributes,
- Overload constructors
  1. **default constructor**, that automatically assign values to all the three sides automatically
  2. **constructor that accepts single int value:** that assign value to the single side only and the rest of two sides are assigned values automatically
  3. **constructor that accepts two int values:** That assign values to the two sides of the triangle that may not be equal. The third side is assigned values as the average (int) value
  4. **constructor that accept three int values:**
- Use set and get functions for the triangle side_a, side_b and side_c
- use a print function for printing the triangle using the get functions of each side