CSCI 2170 **Inheritance**

1. **Inheritance**
   - a relationship among classes whereby a class derives properties from a previously defined class.
   - Inheritance provides a means of deriving a new class from existing classes called base classes.
   - Inheritance describes the ability of a class to derive properties from a previously defined class.
   - **base class** – a class from which another class is derived.
   - **derived class** – a class that inherits the members of another class called the base class.
   - **Purpose of inheritance** – inheritance provides a method for coping with the complexity of a problem by allowing the use of existing code.

Access Control and Inheritance:

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.
We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class inherits all base class methods with the following exceptions:
Constructors, destructors and copy constructors of the base class.
Overloaded operators of the base class.
The friend functions of the base class.

2. **Three data access types in a class**
   - All classes can contain a public section, private section and protected section.
   - Public portions of a class are accessible to all clients of the class.
   - Private portions of a class are accessible only by members and friends of the class.
   - Protected portions of a class are accessible by members of the class, friends of the class and all derived classes.

```
class   MyType
{
public:
        // declarations of visible members, can be used everywhere, no restriction

protected:
         // declarations of protected members
        // only accessible by member functions of MyType, friends of MyType, and
        // member functions and friends of classes derived from MyType

private:
    // can only be used by the member functions of MyType and friends of MyType
};
```

3. A derived class has data members and member functions of the base class in addition to members it defines.
4. Derived classes can revise any inherited member functions.
5. A derived class inherits all members of its base class, except constructors and destructor, friend functions, overloaded operators
6. A derived class's constructor executes **after** the base class's constructor.
7. A derived class's destructor executes **before** the base class's destructor.
8. When a base class constructor requires parameters, the derived class constructor must explicitly call the base constructor and pass necessary parameters.

Example **on order of execution of constructor and destructor of base and derived classes:**

```
class C1
{
public:
        C1(int n);
        ~C1();
protected:
        int *pi, intNum;
};

C1::C1(int n) : intNum(n)
{
        cout << intNum << " integers are
        allocated" << endl;
        pi = new int [intNum];
}

C1::~C1()
{
        cout << intNum << " integers are
        released''<< endl;
        delete [] pi;
}
```

```
class C2 : public C1
{
public :
        C2(int n);
        ~C2();
private:
        char *pc;
        int charNum;
};

C2::C2(int n) : C1(n), charNum (n)
{
        cout << charNum
        << " characters are allocated."
        << endl;
        pc = new char [charNum];
}

C2::~C2()
{
        cout << charNum
         << " characters are released"
        << endl;
        delete [] pc;
}

int main ()
{
        C2   array(30);

        return 0;
}
```