Section 2: Lecture 6

Introduction to friend

- C++ provides a way to enable a class or function to access the *private* parts of another class.
- This is done by using the friend keyword in the class declaration.

Friends of the Creature

```
class Creature {
friend void rejuvenate(Creature & c);
friend class Fred;
private:
   int yearOfBirth;
public:
   Creature(int year) {
            yearOfBirth = year;
            }
   int getYearOfBirth() {
            return yearOfBirth;
            }
};
```

Friends of the Creature

```
class Creature {
friend void rejuvenate(Creature & c);
friend class Fred;
private:
   int yearOfBirth;
public:
   Creature(int year) {
          yearOfBirth = 
          }
   int getYearOfBirth() {
          return yearOfBirth;
          }
};
```

The function rejuvenate can now access the private attribute yearOfBirth:

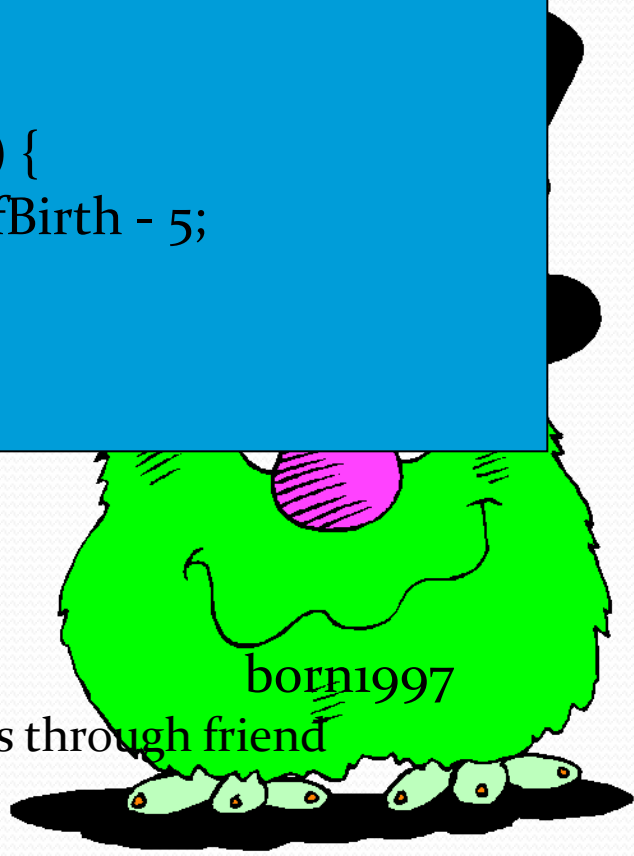
```
void rejuvenate(Creature & c) {
    c.yearOfBirth = c.yearOfBirth + 5;
    }
```

Friends of the Creature

```
class Creature {
friend void rejuvenate(Creature & c);
friend class Fred;
private:
   int yearOfBirth;
public:
   Creature(int year) {
          yearOfBirth = 
          }
   int getYearOfBirth()
          return yearOf
          }
};
```

The class Fred can now access the private attribute yearOfBirth:

```
class Fred {
    void mature(Creature &c ) {
       c.yearOfBirth = c.yearOfBirth - 5;
       }
// ...
    }
```



Private member of a class can be access by another class through friend

Friend Function

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
friend Complex sum (Complex A, Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
```

```
Complex sum (Complex A, Complex B)
{
Complex temp;
temp.real=A.real + B.real;
temp.imag= A.imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= sum (X,Y);
Z.putdata( );
}
```

Friend Function

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
friend Complex sum (Complex A, Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
Complex sum (Complex A, Complex B)
{
Complex temp;
temp.real=A.real + B.real;
temp.imag= A.imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= sum (X,Y);
Z.putdata( );
}
```

12 + 14 i

- Data member or member functions may be public, private or protected.
- Public means data members or member functions defining inside the class can be used at outside the class.(in different class and in main function)
- Private means data members and member functions can't be used outside the class.
- Protected means data member and member functions can be used in the same class and its derived class (at one level) (not in main function).

Passing Object

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
void sum (Complex A, Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
void complex : : sum ( Complex A, Complex B)
{
real = A.real + B.real;
imag= A.imag + B.imag;
}

void main( )
{
Complex X,Y,Z;
X.getdata( );
Y.getdata( );
Z.sum(X,Y);
Z.putdata( );
 }
```

12 + 14 i

Returning Object

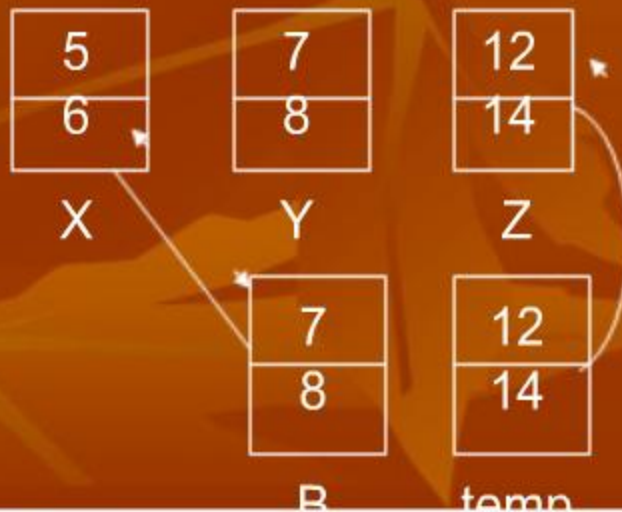
```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
Complex sum (Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
Complex Complex : : sum (Complex B)
{
Complex temp;
temp.real=real + B.real;
temp.imag= imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= X.sum (Y);
Z.putdata( );
}
```

Returning Object

```
#include<iostream.h>
class Complex
{
float real, imag;
public:
void getdata( );
void putdata( );
Complex sum (Complex B);
};
void Complex : : getdata( )
{
cout<<"enter real part:";
cin>>real;
cout<<"enter imaginary part:";
cin>>imag;
}
void Complex : : putdata( )
{
if (imag>=0)
cout<<real<<"+"<<imag<<"i";
else
cout<<real<<imag<<"i";
}
```

```
Complex Complex : : sum (Complex B)
{
Complex temp;
temp.real=real + B.real;
temp.imag= imag + B.imag;
return temp;
}
void main ( )
{
Complex X, Y, Z;
X.Getdata( );
Y. getdata( );
Z= X.sum (Y);
Z.putdata( );
}
```



12 + 14 i

A member function of one class can be friend of another class

```
Class x
{
...
...
Int fun1();                    //member function of x
.....
};

Class y
{
.....
.....
Friend int x::fun1();                    //fun1() of x is friend of y
...
}
```

All the member function of one class as the friend function of another class

```
Class z
{
……
Friend class x;        //all the member function of x
                       are friends to z
};
```

A function friendly to two classes

```
Class B;                //forward declaration
Class A
{  int a;
    public:

   void setvalue(int i){a=i;}
   friend void max(A,B);
};
Class B
{  int b;
    public:

   void setvalue(int i){b=i;}
   friend void max(A,B);
};
```

Continue….

```
Void max(A m,B n)
{
  If(m.a>=n.b)
  cout<<m.a;
  else
   cout<<n.b;
}
 int main()
{
 A aa;
 B bb;
 aa.setvalue(10);
 bb.setvalue(20);
 max(aa,bb);
 return 0;
}
```