Function and operator overloading

Contents

- Need of function overloading
- Need of operator overloading
- Overload +, -, unary -, increment operators.

Function overloading

- Same function but different signature.
- Signature means
    1. May be no of arguments.
    2. No of parameters.
    3. Sequence of parameters.
    4. return type.

Name mangling for overloaded functions

e.g.

int sum(int a,int b)	sum@1
float sum(float a,float b)	sum@2
int sum(int a,float b)	sum@3

Operator overloading

- Additional meaning is given to operators.
- Enhances power of extensibility.
- operator keyword is used to implement operator overloading.
- Following operators we are overloading.
- Plus (+)
- Subtraction(-)
- Unary –
- Pre increment and post increment

Class cComplex

- For operator overloading we are considering following class.

```
class cComplex
{
                int real,imag;
        public:
                ……………………
}
```

Operator overloading syntax

Syntax:

returntype operator # (parameterlist);

1. Here operator is keyword
2. # is placeholder

e.g.

cComplex operator +(cComplex c1);

Overload + operator

```
int main( )           //client code
{
        cComplex c1(1,1);
        cComplex c2(2,2);
        cComplex c3 = c1 + c2;
}
```

Declaration:

```
cComplex operator + (cComplex& c2);
```

Definition:

```
cComplex cComplex::operator + (cComplex& c2)
{
        cComplex temp;
        temp.real = real + c2.real;
        temp.imag = imag + c2.imag;
        return temp;
}
```

Need of assignment operator overloading

- cString s1(“Hello”);
- cString s2;
- s2=s1;

- It is ok. But it will perform shallow copy means member wise copy.
- So it will create problem of memory leakage and dangling pointer problem.
- So to overcome these problems we need to overload = operator.

Overloading assignment operator

```
class cString
{
        cString& operator = ( cString &);
}
cString & cString::operator = (cString& s1)
{
        if( this == &s)
        {
                return *this;
        }
        else
        {
                length=s1.length;
                delete[] ptr;
                ptr=new char[length + 1];
                strcpy(ptr,s1.ptr);
                return *this;
        }
}
```

Difference between Copy constructor and assignment operator overloading

```
cString s1(“Hello”);
cString s2=s1;      //call for copy constructor
cString s2(s1);     //call for copy constructor

cString s1(“hello”);
cString s2;
s2=s1;              //call for assignment operator overloading
```

Lab Assignments

- Perform addition for different types of data by using function overloading.
- Overload + , - ,unary minus and increment operator for cComplex class.
- Overload = operator for cString class.