Object Oriented Programming with C++

10. Type Conversion

By: Prof. Pandav Patel

Second Semester, 2020-21 Computer Engineering Department Dharmsinh Desai University

```
int main() {
#include<iostream>
                                                               Number n = 10;
#include<string>
                                                               cout << n << endl;;
using std::cout;
                                                               n = 20;
using std::endl;
                                                               cout << n << endl;
using std::ostream;
                                                               return 0;
class Number {
  int num;
public:
  Number(int num) {
     cout << "constructor called\n";</pre>
     this->num = num;
  friend ostream &operator<<(ostream &strm, Number &n);
};
ostream & operator << (ostream & strm, Number & n) {
  strm << "num is: " << n.num;
  return strm;
```

constructor called num is: 10 constructor called num is: 20

```
#include<iostream>
#include<string>
using std::cout;
using std::endl;
using std::ostream;
class Number {
  int num;
public:
  Number(int num) {
     cout << "constructor called\n";</pre>
     this->num = num;
  friend ostream &operator<<(ostream &strm, Number &n);
};
ostream & operator << (ostream & strm, Number & n) {
  strm << "num is: " << n.num;
  return strm;
```

```
int main() {
    Number n = 10;
    cout << n << endl;;
    int i;
    // error: cannot convert 'Number' to 'int' in assignment
    i = n;
    cout << i << endl;
    return 0;
}</pre>
```

Conversation Function

- Enables conversion from a class type to another type.
- Conversion function is declared like member function with no parameters,
 - no explicit return type
- operator conversion-type-id() {}

• When such member function is declared in class X, it performs conversion from X to conversion-type-id

```
int main() {
#include<iostream>
#include<string>
                                                                Number n = 10;
                                                                cout << n << endl;;
using std::cout;
                                                                int i:
using std::endl;
                                                                i = n;
using std::ostream;
                                                                cout << i << endl;
                                                                return 0;
class Number {
  int num;
public:
  Number(int num) {
     cout << "constructor called\n";</pre>
     this->num = num;
  operator int() {
     cout << "conversion function called\n";</pre>
     return num;
                                                                            constructor called
  friend ostream &operator<<(ostream &strm, Number &n);
                                                                            num is: 10
                                                                            conversion function called
ostream & operator << (ostream & strm, Number & n) {
                                                                            10
  strm << "num is: " << n.num;
  return strm;
```

class Number {		class Fnumber {
int num;		float fnum;
public:		public:
Number(int num) {		Fnumber(float fnum) {
cout << "constructor called\n"		cout << "Fnum constructor called\n";
this->num = num;		this->fnum = fnum;
}		}
operator int() {		
cout << "conversion function of	called\n";	Fnumber(Number n) {
return num;		cout << "Fnum constructor 2 called\n";
}		this->fnum = n.get_num();
int get_num() {		}
return num;		
}		operator Number() {
friend ostream &operator<<(ostream &strm, Number &n);		cout << "conversion function 2 called\n";
} ;		return Number(int(fnum));
ostream &operator<<(ostream &sti	m, Number &n) {	}
strm << "num is: " << n.num;		
return strm;	int main() {	friend ostream &operator<<(ostream &strm, Fnumber &fn);
}	Number $n = 10$;	} ;
constructor called	cout << n << endl;;	ostream &operator<<(ostream &strm, Fnumber &fn) {
num is: 10	Fnumber $fn = 7.7f$;	strm << "Fnum is: " << fn.fnum;
Fnum constructor called	cout << fn << endl;	return strm;
Fnum is: 7.7	n = fn;	}
conversion function 2 called	cout << n << endl;	
num is: 7	fn = n;	
	cout << fn << endl;	Working Perfect Let's try to answer following Questions:
Fnum constructor 2 called	return 0;	☐ Why fn = n working without conversion function?
Fnum is: 7	}	□ Can we create Conversion function for it ?

class Number {		clas
int num;		fl
public:		pub
Number(int num) {		-
cout << "constructor called\n	n.	
this- $>$ num = num;		,
}		}
operator int() {		_
cout << "conversion function	called\n";	F
return num;		
}		}
int get_num() {		
return num;		0
}		O
friend ostream &operator<<(ost	ream &strm, Number &n);	
};	tura. Nicosala au Osal (}
ostream & operator << (ostream & st	trm, Number &n) {	~
strm << "num is: " << n.num;		
return strm;	int main() {	}
constructor called	Number n = 10;	fr
num is: 10	cout << n << endl;;	} ;
Fnum constructor called	Fnumber $fn = 7.7f$;	ostr
Fnum is: 7.7 conversion function 2 called	cout << fn << endl;	S
constructor called	n = fn;	re
num is: 7	cout << n << endl;	}
Fnum constructor 2 called	fn = n;	
Destructor Called	cout << fn << endl;	For
Fnum is: 7	return 0;	- Co
Destructor Called	}	- Co

```
ss Fnumber {
loat fnum;
lic:
-number(float fnum) {
 cout << "Fnum constructor called\n";</pre>
 this->fnum = fnum;
Fnumber(Number n) {
 cout << "Fnum constructor 2 called\n";
 this->fnum = n.get num();
perator Number() {
 cout << "conversion function 2 called\n";
 return Number(int(fnum));
-Fnumber(){
 cout<<"Destructor Called"<<endl;
riend ostream &operator<<(ostream &strm, Fnumber &fn);
ream &operator<<(ostream &strm, Fnumber &fn) {
strm << "num is: " << fn.fnum;
eturn strm;
```

For fn = n statement compiler perform followings steps:

- Compiler creates temporary object by calling Fnum constructor 2
- Copies temporary to fn and destroys temporary

```
class A{
                              int main(){
                                                                                 В
public:
                                B o1;
  int a1;
                                A o2;
                                                                                 Constructor2 Called
  A(){
                                cout<<o1.get b1()<<endl;</pre>
                                                                                 Destructor Called
    a1=10;
                                o1=o2;
                                                                                 10
                                cout<<o1.get b1()<<endl;
                                                                                 Destructor Called
class B{
  float b1:
public:
                                            Just another Example
  B(){
    b1=5:
    cout<<"B"<<endl;
                                            Here also compiler creates temporary object of
                                            type B from o2 by calling constructor2
 B(A a){
                                            (conversion constructor) while we write o1=o2
   cout<<"Constructor2 Called"<<endl;
  b1=(int)a.a1;
                                            Then it copies temporary to o1 and destroys the
 float get_b1(){
                                            temporary object
   return b1;
 ~B(){
   cout<<"Destructor Called"<<endl;</pre>
```

```
class Fnumber;
                                                                   class Fnumber {
                                      int main() {
                                                                      float fnum;
                                        Number n = 10;
class Number {
                                                                   public:
                                         cout << n << endl;;
                                                                      Fnumber(float fnum) {
  int num;
                                         Fnumber fn = 7.7f;
                                                                        cout << "Fnum constructor called\n";</pre>
public:
                                        cout << fn << endl;
  Number(int num) {
                                                                        this->fnum = fnum;
                                        n = fn;
    cout << "constructor called\n":
                                        cout << n << endl;
    this->num = num;
                                        fn = n:
                                                                      Fnumber(Number n) {
                                        cout << fn << endl:
                                                                        cout << "Fnum constructor 2 called\n";
                                         return 0;
                                                                        this->fnum = n.get num();
operator Fnumber(){
    cout << "Conversion function from Number is called\n";</pre>
                                                                      operator Number() {
                                                                        cout << "conversion function 2 called\n";</pre>
    return Fnumber(float(num));
                                                                        return Number(int(fnum));
  operator int() {
    cout << "conversion function called\n";
                                                                      friend ostream &operator<<(ostream &strm, Fnumber &fn);
    return num;
  int get num() {
                                                                   ostream & operator << (ostream & strm, Fnumber & fn) {
                                                                      strm << "num is: " << fn.fnum:
    return num;
                                                                      return strm;
  friend ostream & operator << (ostream & strm, Number & n);
ostream & operator << (ostream & strm, Number & n) {
                                                                 error: return type 'class Fnumber' is incomplete c++
  strm << "num is: " << n.num;
  return strm;
```

```
class Fnumber;
                                                                    class Fnumber {
                                       int main() {
                                                                      float fnum;
                                         Number n = 10;
class Number {
                                                                    public:
                                         cout << n << endl;;
                                                                      Fnumber(float fnum) {
  int num;
                                         Fnumber fn = 7.7f;
                                                                         cout << "Fnum constructor called\n";</pre>
public:
                                         cout << fn << endl;
  Number(int num) {
                                                                         this->fnum = fnum;
                                         n = fn;
    cout << "constructor called\n":
                                         cout << n << endl;
    this->num = num;
                                         fn = n:
                                                                       Fnumber(Number n) {
                                         cout << fn << endl:
                                                                         cout << "Fnum constructor 2 called\n";</pre>
                                         return 0;
                                                                         this->fnum = n.get num();
Operator Fnumber();
operator int() {
     cout << "conversion function called\n";</pre>
                                                                       operator Number() {
                                                                         cout << "conversion function 2 called\n";
    return num;
                                                                         return Number(int(fnum));
  int get num() {
     return num;
                                                                      friend ostream & operator << (ostream & strm, Fnumber & fn);
  friend ostream & operator << (ostream & strm, Number & n);
                                                                    ostream & operator << (ostream & strm, Fnumber & fn) {
                                                                       strm << "num is: " << fn.fnum:
ostream & operator << (ostream & strm, Number & n) {
  strm << "num is: " << n.num;
                                                                       return strm;
  return strm;
                                                                    Number:: operator Fnumber(){
                                                                         cout << "Conversion function from Number is called\n";</pre>
                                                                         return Fnumber(float(num));
```

```
class Fnumber;
                                      int main() {
                                         Number n = 10:
class Number {
                                         cout << n << endl;;
  int num;
                                         Fnumber fn = 7.7f;
public:
                                         cout << fn << endl;
  Number(int num) {
                                         n = fn;
    cout << "constructor called\n":
                                         cout << n << endl;
    this->num = num;
                                         fn = n:
                                         cout << fn << endl:
                                         return 0;
Operator Fnumber();
operator int() {
     cout << "conversion function called\n";
    return num;
  int get num() {
     return num;
  friend ostream & operator << (ostream & strm, Number & n);
ostream & operator << (ostream & strm, Number & n) {
  strm << "num is: " << n.num;
  return strm;
```

error: conversion from 'Number' to 'Fnumber' is ambiguous

```
class Fnumber {
  float fnum;
public:
  Fnumber(float fnum) {
    cout << "Fnum constructor called\n":
    this->fnum = fnum;
  Fnumber(Number n) {
    cout << "Fnum constructor 2 called\n";</pre>
    this->fnum = n.get num();
  operator Number() {
     cout << "conversion function 2 called\n";</pre>
     return Number(int(fnum));
  friend ostream &operator<<(ostream &strm, Fnumber &fn);
ostream & operator << (ostream & strm, Fnumber & fn) {
  strm << "num is: " << fn.fnum:
  return strm;
Number:: operator Fnumber(){
    cout << "Conversion function from Number is called\n"
    return Fnumber(float(num));
```

```
class Fnumber;
                                                                    class Fnumber {
                                       int main() {
                                                                       float fnum;
                                         Number n = 10;
class Number {
                                                                    public:
                                         cout << n << endl;;
                                                                       Fnumber(float fnum) {
  int num;
                                         Fnumber fn = 7.7f;
                                                                          cout << "Fnum constructor called\n";</pre>
public:
                                         cout << fn << endl;
  Number(int num) {
                                                                         this->fnum = fnum;
                                         n = fn;
     cout << "constructor called\n":
                                         cout << n << endl;
     this->num = num;
                                         fn = n:
                                                                       /*
                                         cout << fn << endl:
                                                                       Fnumber(Number n) {
Operator Fnumber();
                                         return 0;
operator int() {
                                                                          cout << "Fnum constructor 2 called\n";
     cout << "conversion function called\n":
                                                                          this->fnum = n.get num();
                                                                       } */
     return num;
  int get num() {
                                                                       operator Number() {
                                                                         cout << "conversion function 2 called\n";
     return num;
                                                                          return Number(int(fnum));
  friend ostream & operator << (ostream & strm, Number & n);
ostream &operator<<(ostream &strm, Number &n) {
                                                                       friend ostream &operator<<(ostream &strm, Fnumber &fn);
  strm << "num is: " << n.num;
  return strm;
                                                                    ostream &operator<<(ostream &strm, Fnumber &fn) {
                             constructor called
                                                                       strm << "num is: " << fn.fnum;
                             num is: 10
                                                                       return strm;
                             Fnum constructor called
                             num is: 7.7
                             conversion function 2 called
                                                                    Number:: operator Fnumber(){
                             constructor called
                                                                          cout << "Conversion function from Number is called\n";
                             num is: 7
                                                                         return Fnumber(float(num));
                             conversion function from Number is called
                             Fnum constructor called
                             num is: 7
```

Interesting reads

- Conversion constructor vs. conversion operator: precedence
 - https://stackoverflow.com/questions/1384007/conversion-constructor-vs-conversion-operator-precedence



