Object Oriented Programming with C++

10. Type Conversion

By: Prof. Pandav Patel

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

```
#include<iostream>
                                                             int main() {
                                                                Number n = 10;
#include<string>
                                                                cout << n << endl;;
                                                                n = 20;
using std::cout;
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;
                                                                           constructor called
  return strm;
                                                                           num is: 10
                                                                           constructor called
                                                                           num is: 20
```

```
#include<iostream>
                                                        int main() {
                                                          Number n = 10;
#include<string>
                                                          cout << n << endl;;
using std::cout;
                                                          int i;
using std::endl;
                                                          // error: cannot convert 'Number' to 'int' in assignment
using std::ostream;
                                                          i = n;
                                                          cout << i << endl;
class Number {
                                                          return 0;
  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;
```

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

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

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

```
cout << "Fnum constructor called\n";</pre>
     cout << "Fnum constructor 2 called\n";
     cout << "conversion function 2 called\n";
  friend ostream & operator << (ostream & strm, Fnumber & fn);
ostream & operator << (ostream & strm, Fnumber & fn) {
     cout << "Conversion function from Number is called\n":
```

Interesting reads

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



