

```

1 #include "regular.h"
2
3 int Regular::m_cp_times = 0, Regular::m_def_times = 0;
4
5 Regular::Regular()
6 {
7     cout << "    Default Constructor Called (" << ++m_def_times
8         << " times)" << endl;
9     m_example_member = new int;
10    *m_example_member = 0;
11 }
12
13 Regular::Regular(const int example_member): Regular()
14 {
15     cout << "    Parameterized Constructor Called" << endl;
16     *m_example_member = example_member;
17 }
18
19 Regular::Regular(const Regular& ex): Regular()
20 {
21     cout << "    Copy Constructor (" << ++m_cp_times
22         << " times)" << endl;
23     *m_example_member = *ex.m_example_member;
24 }
25
26 Regular::~Regular()
27 {
28     delete m_example_member;
29     m_example_member = nullptr;
30 }
31
32 Regular& Regular::operator=(const Regular& ex)
33 {
34     cout << "    Copy Assignment Operator" << endl;
35     *m_example_member = *ex.m_example_member;
36     return *this;
37 }
38
39 void Regular::get_summary()
40 {
41     cout << "    Total Copies: " << m_cp_times
42         << "    Total Default Calls: " << m_def_times << endl;
43 }

```

regular.cpp

```

1 #include "movesemantics.h"
2
3 int MoveSemantics::m_cp_times = 0, MoveSemantics::m_def_times = 0;
4
5 MoveSemantics::MoveSemantics()
6 {
7     cout << "    Default Constructor Called (" << ++m_def_times
8         << " times)" << endl;
9     m_example_member = new int;
10    *m_example_member = 0;
11 }
12
13 MoveSemantics::MoveSemantics(const int example_member):
14     MoveSemantics()
15 {
16     cout << "    Parameterized Constructor Called" << endl;
17     *m_example_member = example_member;
18 }
19
20 MoveSemantics::MoveSemantics(MoveSemantics&& ex)
21 {
22     cout << "    Move Constructor" << endl;
23     m_example_member = ex.m_example_member;
24     ex.m_example_member = nullptr;
25     // nullptr is the new C++11 constant for null pointers.
26 }
27
28 MoveSemantics::MoveSemantics(const MoveSemantics& ex):
29     MoveSemantics()
30 {
31     cout << "    Copy Constructor (" << ++m_cp_times
32         << " times)" << endl;
33     *m_example_member = *ex.m_example_member;
34 }
35
36 MoveSemantics::~MoveSemantics()
37 {
38     delete m_example_member;
39     m_example_member = nullptr;
40 }
41
42 MoveSemantics& MoveSemantics::operator=(MoveSemantics&& ex)
43 {
44     cout << "    Move Assignment Operator" << endl;
45     m_example_member = ex.m_example_member;
46     ex.m_example_member = nullptr;
47     return *this;
48 }
49
50 MoveSemantics& MoveSemantics::operator=(const MoveSemantics& ex)
51 {
52     cout << "    Copy Assignment Operator" << endl;
53     *m_example_member = *ex.m_example_member;
54     return *this;
55 }
56
57 void MoveSemantics::get_summary()
58 {
59     cout << "    Total Copies: " << m_cp_times
60         << "    Total Default Calls: " << m_def_times << endl;
61 }

```

movesemantics.cpp

```

1 #include <iostream>
2 #include "movesemantics.h"
3 #include "regular.h"
4 using namespace std;
5
6 template <class T>
7 T some_func(T in) { return in; }
8
9 template <class T>
10 void run_example()
11 {
12     cout << "    First Part" << endl;
13     T b;
14     T c(some_func<T>(b));
15
16     cout << endl << "    Second Part" << endl;
17     b = some_func<T>(c);
18
19     cout << endl << "    Third Part" << endl;
20     b = move(c);
21     T d(move(b));
22
23     cout << endl << "    Fourth Part" << endl;
24
25     int i = 5;
26     T e(i);
27     e = 4;
28
29     cout << "    TOTALS: " << endl;
30     T::get_summary();
31 }
32
33 int main()
34 {
35     cout << "No Move Semantics Example: " << endl << endl;
36     run_example<Regular>();
37     cout << endl << endl << "With Move Semantics: " << endl << endl;
38     run_example<MoveSemantics>();
39     return 0;
40 }

```

example.cpp

```

$ /usr/bin/g++ -g -Wall -W -pedantic-errors -std=c++11 *.cpp
$ ./example

```

No Move Semantics Example:

```

First Part
Default Constructor Called (1 times)
Default Constructor Called (2 times)
Copy Constructor (1 times)
Default Constructor Called (3 times)
Copy Constructor (2 times)

```

```

Second Part
Default Constructor Called (4 times)
Copy Constructor (3 times)
Default Constructor Called (5 times)
Copy Constructor (4 times)
Copy Assignment Operator

```

```

Third Part
Copy Assignment Operator
Default Constructor Called (6 times)
Copy Constructor (5 times)

```

```

Fourth Part
Default Constructor Called (7 times)
Parameterized Constructor Called
Default Constructor Called (8 times)
Parameterized Constructor Called
Copy Assignment Operator

```

```

TOTALS:
Total Copies: 5    Total Default Calls: 8

```

With Move Semantics:

```

First Part
Default Constructor Called (1 times)
Default Constructor Called (2 times)
Copy Constructor (1 times)
Move Constructor

```

```

Second Part
Default Constructor Called (3 times)
Copy Constructor (2 times)
Move Constructor
Move Assignment Operator

```

```

Third Part
Move Assignment Operator
Move Constructor

```

```

Fourth Part
Default Constructor Called (4 times)
Parameterized Constructor Called
Default Constructor Called (5 times)
Parameterized Constructor Called
Move Assignment Operator

```

```

TOTALS:
Total Copies: 2    Total Default Calls: 5

```

output.txt