Bern University
of Applied Sciences

394661-FS2020-0 - C++ Programming I

EXERCISE-08

TABLE OF CONTENTS

1 Introduction

This exercise of 394661-FS2020-0 will focus on the basic concepts of operator overloading. In C++, operators are implemented as functions. By using function overloading on the operator functions, you can define your own versions of the operators that work with different data types including classes that you've written! The effort in implementing relevant operators will be rewarded by the ease of consumption of the class.

You will learn the following topics when completing this exercise:

- Operator overloading
- Conversion operators / stream insertion
- Copy assignment operator
- Move assignment operator

2 Excercises

Create CMake-Projects with C++ 11 compiler support and Debug/Release build options for the exercise. Add additional files manually to the project to gain full control over the included project files. Separate the implementation from the declaration in a header and source file, respectively.

2.1 Extended Class `Vector`

Run the provided test program with your class `Vector` from exercise-05. It's the same test program as for exercise-05, but with extended test cases. Fix the compile errors by implementing the missing operators in class `Vector`:

- Implement the output stream operator « to print the first 5 elements of a vector. You can either implement
  1. a conversion operator "`operator const char *()`" as shown in the lecture slides (requires an additional string member!) or
  2. overload the « operator. See "stream extraction and insertion" (`http://en.cppreference.com/w/cpp/language/operators`)
- Implement a copy assignment operator which checks for self-assignment
- Overload the operator + and * to enable element wise addition and multiplication of vector with the same size, respectively
- Implement a move assignment operator to optimise the performance of your class

Test your classes with the following test program (`ex08.cpp`):

```
#include <iostream>

#include "vector.h" // Your implementation of vector

using namespace std;

int main(int argc, char *argv[])
{
    cout << " ***************************** " << endl;
    cout << " ******** VECTOR TEST-2 ******** " << endl;
    cout << " *****************************\n" << endl;

    // 1) Initialisation
    Vector v1; // empty vector
    Vector v2(100); // vector with 100 elements initialised to 0!
    Vector v3(100, 42) ; // vector with 100 elements initialised to 42!

    cout << "v1 has size " << v1.size() << endl;
    cout << "v2 has size " << v2.size() << endl;
    cout << "v3 has size " << v3.size() << "\n" << endl;

    // 2) Element access
    cout << "v1 contains value: " << v1.at(0) << endl; // -> warning ! --> without
        correction an old value in the memory is returned
    cout << "v2 contains value: " << v2.at(0) << endl;
    cout << "v3 contains value: " << v3.at(0) << endl;
    cout << "v3 contains value: " << v3.at(142) << endl; // -> warning ! --> without
        correction an old value in the memory is returned

    // 3) Add Element
    v1.push_back(1);
    v1.push_back(2);
    v1.push_back(3);
    v1.push_back(4);
    v1.push_back(5);

    cout << "\n" << "v1 has size " << v1.size() << " and contains: " << endl;

    for (int i = 0; i < v1.size(); ++i)
    {
        cout << i << ": " << v1.at(i) << endl;
    }

```

```
42      // 4) Remove Element
43      v1.pop_back();
44      v1.pop_back();
45
46      cout << "\n" << "v1 has size " << v1.size() << " and contains: " << endl;
47
48      for (int i = 0; i < v1.size(); ++i)
49      {
50          cout << i << ": " << v1.at(i) << endl;
51      }
52
53      // 4) Clear Elements
54      v1.clear();
55      cout << "\n" << "v1 has size " << v1.size() << endl;
56
57      // 5) Check Copy Constructor
58      Vector vCopy(v3);
59      cout << "vCopy has size " << vCopy.size() << "\n" << endl;
60      cout << "vCopy contains value: " << vCopy.at(0) << "\n" << endl;
61
62      // 6) Check Move Constructor
63      Vector vMove = move(v3);
64      cout << "vMove has size " << vMove.size() << "\n" << endl;
65      cout << "vMove contains value : " << vMove.at(0) << "\n" << endl;
66
67      cout << "v3 has size " << v3.size () << "\n" << endl;
68
69      // 7) Check Copy Assignment Operator <<Operator
70      vMove = vMove; // Copy assign - Check Self-Assignment
71      cout << "Self Assigned Copy has size " << vMove.size()  << endl;
72      cout << "Self Assigned Copy contains value " << vMove.at(0) << "\n" << endl;
73
74      v2 = vMove; // Copy assign
75      cout << "Assigned Copy has size " << v2.size() << endl;
76      cout << "Assigned Copy contains value " << v2 << "\n" << endl;
77
78      // 8) Check Move Assignment Operator and <<Operator
79      v2 = v2 + vMove; // Move assign!
80      cout << "Vector Addition has size " << v2.size() << endl;
81      cout << "Vector Addition contains value\n" << v2 << "\n" << endl; // --> 84
82
83      v3 = v2 * v2; // Move assign!
84      cout << "Vector Multiplication has size " << v3.size() << endl;
85      cout << "Vector Multiplication contains values\n" << v3 << "\n" << endl; // --> 7056
86
87
88      cout << " **************************** " << endl;
89      cout << " **** VECTOR TEST-2 PASSED ***** " <<  endl;
90
91      return 0;
92  }
```

3 Submission

Submit your source code (as a zip-file) to Ilias EXERCISE-08 **before the deadline** specified in Ilias.