National Chiao Tung University
Department of Electrical and Computer Engineering
Computer Intelligence on Automation(C.I.A.) Lab

Laboratory Manual 05
Match 26, 2012
Prof. Hung-Pin (Charles) Wen

UEE1303(1070) S12: Object-Oriented Programming
Constructors and Destructors

What you will learn from Lab 5

In this laboratory, you will learn how to use constructor and copy constructor to create an object and use destructor to delete it.

TASK5-1 CONSTRUCTOR

✓ Please try to compile and execute the program lab5-1-1, and observe the results.

```cpp
// lab5-1-1.cpp
#include <iostream>

class Point2D
{
private:
    int x;
    int y;
    double value;

public:
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
};

void Point2D::assignPoint2D(int n1, int n2, double v)
{
    x = n1;
    y = n2;
    value = v;
}

void Point2D::displayPoint2D()
{
    std::cout << "(" << x << "," << y << ") = ";
    std::cout << value << std::endl;
}

int main()
{
    Point2D ptArray[10];
    for (int i=0;i<10;i++)
        ptArray[i].displayPoint2D();
    return 0;
}
```

✓ We add *constructor* to class Point2D and make program lab5-1-2 work as expect.

```
// lab5-1-2.cpp
…
class Point2D
{
private:
    int x;
    int y;
    double value;

public:
    Point2D();          // default constructor
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
};

Point2D::Point2D()
{
    x = 0;
    y = 0;
    value = 0.0;

}
…
```

➢ You can also use a parenthesized expression list to build your default constructor. In the above example, you can replace the declaration and definition of default constructor as Point2D():x(0), y(0), value(0.0) {}.

✓ Please modify your class Point2D to make the lab5-1-3 work.

```
// lab5-1-3.cpp
…
int main()
{
    Point2D pt1;
    Point2D pt2(1,2);
    Point2D pt3(3,2,1.9);

    pt1.displayPoint2D();
    pt2.displayPoint2D();
    pt3.displayPoint2D();
    pt3.assignPoint2D(2,1,0.0);
    pt3.displayPoint2D();

    return 0;
}
```

➢ The line `Point2D pt3(3,2,1.9)` and `pt3.assignPoint2D(2,1,0.0)` can both assign
the value to `pt3`'s private member. Can you explain their difference?

TASK5-2 DESTRUCTOR

✓ In class `Point2D`, we do not specific the destructor `~Point2D()` since the compiler will
generate one. However, if you use `new` or `delete` memory in the object, you need constructor
to allocate memory and destructor to release it.

✓ Please modify the class `Point2D` as `PointND`, which is used to record the N-dimensional
coordinate using an integer array.

```cpp
// lab5-2.cpp
#include <iostream>
#include <assert.h>

const int num = 10;

class PointND
{
private:
    int *coord;
    double value;

public:
    PointND();
    ~PointND();
    void assignValue(double v);
    void assignCoord(int *vec, int len);
    void displayPointND();
};

PointND::PointND()
{
    value = 0.0;
    coord = new int [num];
    for (int i=0;i<num;i++) coord[i] = 0;
}

PointND::~PointND()
{
    delete []coord;
}

void PointND::assignValue(double v)
{
    value = v;
}
```

National Chiao Tung University
Laboratory Manual 05
Department of Electrical and Computer Engineering
Match 26, 2012
Computer Intelligence on Automation(C.I.A.) Lab
Prof. Hung-Pin (Charles) Wen

```cpp
void PointND::assignCoord(int *vec, int len)
{
    assert(len <= num);          // make sure len <= num
    for (int i=0;i<len;i++)
        coord[i] = vec[i];
}

void PointND::displayPointND()
{
    std::cout << "(";
    for (int i=0;i<num;i++)
    {
        std::cout << coord[i];
        if (i!=num-1)
            std::cout << ", ";
    }
    std::cout << ") = " << value << std::endl;
}

int main()
{
    PointND pt1;
    pt1.displayPointND();

    PointND pt2;
    pt2.assignValue(1.0);
    pt2.displayPointND();

    int *vec = new int [num];
    for (int i=0;i<num;i++) vec[i] = i;

    PointND pt3;
    pt3.assignValue(4.3);
    pt3.assignCoord(vec,num);
    pt3.displayPointND();

    delete []vec;
    return 0;
}
```

TASK5-3 COPY CONSTRUCTOR

✓   Copy constructor is a constructor used to create a new object as a copy of an existing object.

```cpp
// lab5-3.cpp
#include <iostream>
```

```
#include <assert.h>

/* 1. class PointND defined in lab5-2
   2. add the definition of copy constructor to the class*/

PointND::PointND(const PointND &pt)
{
   value = pt.value;
   coord = new int [num];
   for (int i=0;i<num;i++) coord[i] = pt.coord[i];
}

int main()
{
   int *vec = new int [num];
   for (int i=0;i<num;i++) vec[i] = i;

   PointND pt1;
   pt1.assignValue(4.3);
   pt1.assignCoord(vec,num);
   pt1.displayPointND();

   PointND pt2(pt1);
   pt2.displayPointND();

   delete []vec;
   return 0;
}
```

TASK 5-4 EXERCISE

1. *CONSTRUCTORS AND DESTRUCTOR

✓ Please add the *constructor* and *copy constructor* to the class Complex you defined in program ex4-1.

✓ The main structure of the program becomes,

```
// Complex.h
#ifndef COMPLEX_H
#define COMPLEX_H


/* Write class definition for Complex including constructors*/


#endif
```

```
// Complex.cpp
#include <iostream>
using std::cout;


#include "Complex.h"


// Member-function definitions for class Complex.
```

```
// ex5-1.cpp
#include <iostream>
using std::cout;
using std::endl;


#include "Complex.h"


int main()
{
    Complex a(1.0, 7.0), b(9.0, 2.0), c; // create three Complex objects
    a.printComplex(); // output object a
    cout << " + ";
    b.printComplex(); // output object b
    cout << " = ";
    c = a.add(b); // invoke add function and assign to object c
    c.printComplex(); // output object c
    cout << endl;

    Complex d(c);     // use copy constructor to create object d
    d.printComplex(); // output object d


    return 0;
}
```

✓   Please add the ***constructor***, ***copy constructor*** and ***destructor*** to the class Matrix you defined in
    program ex4-2. Note that you must use dynamic memory allocation to create the object.

✓   The main structure of the program becomes,

```
// Matrix.h
```

```
#ifndef MATRIX_H
#define MATRIX_H


/* Write class definition for Matrix and add constructors and destructor*/


#endif
```

```
// Matrix.cpp
#include <iostream>
using std::cout;


#include "Matrix.h"


// Member-function definitions for class Matrix.
```

```
// ex5-2.cpp
#include <iostream>
using std::cout;
using std::endl;
using std::cin;
#include "Matrix.h"


int main()
{
    int n;
    cout << "Enter n for n x n matrix: " << endl;
    cin >> n;


    Matrix A(n), B(n);     // create two Matrix objects
    A.assignElements(); // assign elements in Matrix A randomly
    cout << "A = ";
    A.printMatrix(); // output object A
    cout << endl;


    B.assignElements(); // assign elements in Matrix B randomly
    cout << "B = ";
    B.printMatrix(); // output object B
```

```cpp
   cout << endl;


   Matrix tA(A);          // use copy constructor to build tA
   cout << "tA= ";
   tA.transposeMatrix(); // transpose Matrix tA
   cout << endl;


   Matrix tB(B);          // use copy constructor to build tB
   cout << "tB= ";
   tB.transposeMatrix(); // transpose Matrix tB
   cout << endl;


   Matrix C;
   C.multiplyMatrix(A,B); // C = A * B
   cout << "A*B = ";
   C.printMatrix(); // output object C
   cout << endl;


   return 0;
}
```

2. VECTOR – INTEGER SET

✓　Please define a class Vec to allow the main function work successfully.

```cpp
int main()
{
   int *array1,*array2;
   int dim1, dim2;

   /* read numbers from screen for array1 and array2      */
   /* enter -1 to end input                               */
   /* return the dimension of each array                  */


   Vec vec1(array1,dim1);
   Vec vec2;
   vec2.assign(array2,dim2);

   cout << "vec1(sorted): ";
   vec1.sort();
```

```
    vec1.printVec();

    cout << "vec2(sorted): ";
    vec2.sort();
    vec2.printVec();

    bool isEqual = vec1.isEqual(vec2);

    /* print out the message if vec1 and vec2 are the same or not*/

    Vec vec3;
    vec3.unionSet(vec1,vec2);    // vec3 is union set of vec1 and vec2

    cout << "vec3: ";
    vec3.printVec();

    vec3.sort();
    cout << "vec3(sorted): ";
    vec3.printVec();

    cout << "Min in vec1 and vec2: " << vec3.min() << endl;
    cout << "Max in vec1 and vec2: " << vec3.max() << endl;

    int target = 10;
    bool findInVec = vec3.find(target);

    /* print out the message if target is found or not.  */

    Vec vec4;
    vec4 = vec3.inpendetSet();
    cout << "vec4: ";
    vec4.printVec();

    delete [] array1;
    delete [] array2;
    return 0;
}
```

✓ The sample output is

```
Enter the elements for array (-1 to end):
8 3 2 19 2 3 4 5 -1
```

National Chiao Tung University   Laboratory Manual 05
Department of Electrical and Computer Engineering   Match 26, 2012
Computer Intelligence on Automation(C.I.A.) Lab   Prof. Hung-Pin (Charles) Wen

```
Enter the elements for array (-1 to end):
12 8 6 2 9 2 4 7 -1
vec1(sorted): 2 2 3 3 4 5 8 19
vec2(sorted): 2 2 4 6 7 8 9 12
vec1 is not equal to vec2
vec3: 2 2 3 3 4 5 8 19 2 2 4 6 7 8 9 12
vec3(sorted): 2 2 2 2 3 3 4 4 5 6 7 8 8 9 12 19
Min in vec1 and vec2: 2
Max in vec1 and vec2: 19
Target 10 cannot be found in vec1 or vec2
vec4: 2 3 4 5 6 7 8 9 12 19
```