EEEC10008(515169) S23: Object-Oriented Programming

Template

What you will learn from Lab 10

In this laboratory, you will learn how to use function template and class template.

Task 10-1 Function Template

✔ A function template defines a function that takes type parameters. Please execute `lab10-1`. Here is an example to maintain memory allocation for different types.

```
// lab10-1.cpp
#include <iostream>
#include <cassert>
#include <iomanip>
using namespace std;

template <class T>
T *new1D(int n, T k)
{
    T *vec = new T [n];
    for (int i=0;i<n;i++)
        vec[i] = k;
    return vec;
}

template <class T>
void delete1D(T *vec)
{
    assert(vec != NULL);
    delete [] vec;
}

template <class T>
void display1D(T *vec, int n)
{
    for (int i=0;i<n;i++)
        cout << vec[i] << " ";
    cout << endl;
}

int main()
{
    int *ivec = new1D<int>(10,1);
    display1D<int>(ivec,10);
    delete1D<int>(ivec);

    double *dvec = new1D<double>(10,3.2);
    display1D<double>(dvec,10);
    delete1D<double>(dvec);
    return 0;
}
```

TASK 10-2 FUNCTION TEMPLATE: SPECIALIZATION

✔ In program `lab10-1`, you can maintain a specific version of `display1D()` for `double`. Please add this specialization of `display1D<T>` to `lab10-1` as `lab10-2` and execute the program again.

```
template <>
void display1D(double *vec, int n)
{
    cout << fixed << setprecision(2);
    for (int i=0;i<n;i++)
        cout << vec[i] << " ";
    cout << endl;
}
```

TASK 10-3 CLASS TEMPLATE

✔ You can also define a class template by adding prefix `template<class T>`.

```
// lab10-3.cpp
#include <iostream>
#include <iomanip>
using namespace std;

template <class T>
class Point2D
{
private:
    T x;
    T y;
public:
    Point2D():x(T(0)),y(T(0)){}
    Point2D(T a, T b):x(a),y(b){}
    void display() const;
};

template <class T>
void Point2D<T>::display() const
{
    cout << x << " " << y << endl;
}

int main()
{
    Point2D<int> p1;
    p1.display();

    Point2D<double> p2(1.9,3.4);
    p2.display();

    return 0;
}
```

TASK 10-4 CLASS TEMPLATE: SPECIALIZATION

✔ Here define a specialization of the template class `Point2D<T>` when its elements are complex numbers.

```
// lab10-4.cpp
#include <iostream>
using namespace std;

class Complex
{
private:
    double real;
    double image;
public:
    Complex(const double a, const double b):real(a), image(b){}
    Complex(const Complex &c):real(c.real), image(c.image){}
    void display() const
    {
        cout << real << " " << image << endl;
    }
};
// template Point2D defined in lab10-3

template <>
class Point2D<Complex>
{
private:
    Complex x;
    Complex y;
public:
    Point2D(const Complex &a, const Complex &b):x(a),y(b){}
    void display() const;
};

void Point2D<Complex>::display() const
{
    x.display();
    y.display();
}
int main()
{
    Complex c1(1.9,3.4);
    Complex c2(2.0,1.3);

    Point2D<Complex> pc(c1,c2);
    pc.display();

    return 0;
}
```

EXERCISE 10-1: STATISTICAL ANALYSIS

✔ Please finish the undefined function template in `ex10-1`.

✔ The main function of the program is shown as follows. You could get it from directory `/home/share/lab10/ex1`.

```
// ex10-1.cpp
class Point2D
{
private:
    int x;
    int y;
public:
    // add any member if necessary
};
template<class T>
void analysis(int n)
{
    T *vec = new1D<T>(n);
    rand1D<T>(vec, n);
    // for int, 0~9
    // for double, 0.00~9.99, i.e., rand()%1000/100.0
    // for char, A~Z
    // for Point2D, x: 0~9 y: 0~9
    display1D<T>(vec, n);
    // for double, set the precision with 2
    sort1D<T>(vec, n);
    display1D<T>(vec, n);
}

int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;

    srand(1);

    analysis<int>(n);
    analysis<double>(n);
    analysis<char>(n);
    analysis<Point2D>(n);

    return 0;
}
```

✔ For Point2D, sort according to the x-coordinate of Point2D ascendingly, if the x-coordinates of Point2D are same, then sort according to the y-coordinate of Point2D ascendingly.

✔ The output of the program should be following.

```
$ ./ex10-1   ↵
Enter n: 10
3 6 7 5 3 5 6 2 9 1
1 2 3 3 5 5 6 6 7 9
3.62 0.27 6.90 0.59 7.63 9.26 5.40 4.26 1.72 7.36
```

```
0.27 0.59 1.72 3.62 4.26 5.40 6.90 7.36 7.63 9.26
H I D D Q S C D X R
C D D D H I Q R S X
(9, 2) (2, 8) (9, 7) (3, 6) (1, 2) (9, 3) (1, 9) (4, 7) (8, 4) (5, 0)
(1, 2) (1, 9) (2, 8) (3, 6) (4, 7) (5, 0) (8, 4) (9, 2) (9, 3) (9, 7)
```

EXERCISE 10-2: VECTOR

✔ Please finish the undefined function template in `ex10-2.cpp`. The main function is shown below.

```
// ex10-2.cpp
int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;

    Vector<double> dvec(n,1);

    double *b = new double[n];
    for (int i=0;i<n;i++) b[i] = i;
    Vector<double> dvec2(n,b);

    cout << "dvec = "; dvec.display();
    cout << "dvec2 = "; dvec2.display();
    dvec2 += dvec;
    cout << "new dvec2 = "; dvec2.display();

    double c = dot(dvec, dvec2); //dot operation
    cout << "dot(dvec, dvec2) = " << c << endl << endl;

    srand(1);

    Vector<Point2D> vp1(n, Point2D());// Point2D() (x,y) = (1,1)
    Point2D *v = new Point2D[n];
    rand1D<Point2D>(v,n);   //0~9
    Vector<Point2D> vp2(n,v);

    cout << "vp1 = "; vp1.display();
    cout << "vp2 = "; vp2.display();

    vp2 += vp1;

    cout << "new vp2 = "; vp2.display();

    Point2D d = dot(vp1, vp2);
    cout << "dot(vp1, vp2) = " << d << endl;

    return 0;
}
```

✔ The execution results are following.

```
$ ./ex10-2    ↵
```

```
Enter n: 5
dvec = 1 1 1 1 1
dvec2 = 0 1 2 3 4
new dvec2 = 1 2 3 4 5
dot(dvec, dvec2) = 15

vp1 = (1, 1) (1, 1) (1, 1) (1, 1) (1, 1)
vp2 = (3, 6) (7, 5) (3, 5) (6, 2) (9, 1)
new vp2 = (4, 7) (8, 6) (4, 6) (7, 3) (10, 2)
dot(vp1, vp2) = (33, 24)
```

- Hint: dot operation for two vectors is defined as $\sum_{i=0}^{n-1} v_1(i) \cdot v_2(i)$, and multiplication for two `Point2D`s is written as `Point2D(p1.x*p2.x, p1.y*p2.y);`

✔ Please declare class `Point2D` in `Point2D.h` and define its functionality in `Point2D.cpp`.

✔ Please declare the template class `Vector` and finish its definition in `Vector.h`.

```
// Vector.h
template <class T>
class Vector
{
private:
    int len;
    T* vec;
public:
    // add any member if necessary
    template<class S>
    friend S dot (const Vector<S> &, const Vector<S> &);
};
```

✔ You could get `Point2D.h, Vector.h and ex10-2.cpp` from directory `/home/share/lab10/ex2.`

✔ **Please notice that in this case, the initial value of Point2D x, y is (1, 1), not (0, 0), i.e., Point2D():x(1),y(1){}**

✔ Please write a "Makefile" in your exercise directory to compile your code. Your code can be compiled by typing "`make`", and the name of your program should be `ex10-2`.