

**Advanced Programming in C++**  
Afshin Sepehri - Summer 2002  
**Handout 7 – Polymorphism**

---

**The following program demonstrates Polymorphism and also implements a simple Linked-List. The classes TShape and TCircle implemented in handout 6 are needed:**

```
/*rectangle.h*/

#ifndef RECTANGLE_H
#define RECTANGLE_H
/*****/
#include <iostream>
#include "shape.h"
/*****/
class TRectangle : public TShape {
    int Width;
    int Height;
    static int No;

public:
    TRectangle(const TPoint base=TPoint(0, 0), int width=1, int height=1, char *label=NULL);
    TRectangle(const TRectangle &rectangle);
    ~TRectangle();
    TRectangle &operator=(const TRectangle &rectangle);
    void SetLabel(const char *label);
    float GetPerimeter() const;
    bool IsEquilateral() const;
    ostream &Print(ostream &out) const;
};
/*****/
#endif
```

---

```
/*rectangle.cc*/

#include <string>
#include "shape.h"
#include "rectangle.h"
/*****/
int TRectangle::No = 100;
/*****/
TRectangle::TRectangle(const TPoint base, int width, int height, char *label) :
    TShape(base, label) , Width(width) , Height(height)
{
    MyNo = No++;
    cout << "TRectangle default constructor is called" << endl;
}
/*****/
TRectangle::TRectangle(const TRectangle &rectangle) :
    TShape(rectangle) , Width(rectangle.Width) , Height(rectangle.Height)
{
    MyNo = No++;
    cout << "TRectangle copy constructor is called" << endl;
}
```

```

/*****/
TRectangle::~~TRectangle() {
    cout << "TRectangle destructor is called" << endl;
}
/*****/
TRectangle &TRectangle::operator=(const TRectangle &rectangle) {
    if( this!=&rectangle ) {
        *((TShape*)(this)) = rectangle;
        Height = rectangle.Height;
        Width = rectangle.Width;
    }
    return( *this );
}
/*****/
void TRectangle::SetLabel(const char *label) {
    cout << "TRectangle::SetLabel is called" << endl;
    cout << "Label cannot be changed" << endl;
}
/*****/
float TRectangle::GetPerimeter() const {
    cout << "TRectangle::GetPerimeter is called" << endl;
    return( (Width+Height)*2 );
}
/*****/
bool TRectangle::IsEquilateral() const {
    cout << "TRectangle::IsEquilateral is called" << endl;
    return(Width==Height);
}
/*****/
ostream &TRectangle::Print(ostream &out) const {
    out << "Rectangle Number " << MyNo << "\n";
    TShape::Print(out);
    out << "Width: " << Width << endl;
    out << "Height: " << Height << endl;
    return(out);
}

```

---

```

/*queue.h*/

```

```

#ifndef QUEUE_H
#define QUEUE_H
/*****/
#include <iostream>
#include "shape.h"
/*****/
struct TNode {
    TShape *Data;
    TNode *Next;
    TNode(TShape *data);
};
/*****/
class TQueue {
    TNode *List;
public:

```

```

    TQueue();
    TQueue(const TQueue &queue);
    ~TQueue();
    void Add(TShape *psh);
    TShape *Remove();
};
/*****/
#endif

```

---

```

/*queue.cc*/

```

```

#include "shape.h"
#include "queue.h"
/*****/
TNode::TNode(TShape *data) :
    Data(data) , Next(NULL)
{
}
/*****/
TQueue::TQueue() :
    List(NULL)
{
    cout << "TQueue default constructor is called" << endl;
}
/*****/
TQueue::TQueue(const TQueue &queue) {
    cout << "A queue cannot be copied" << endl;
    exit(1);
}
/*****/
TQueue::~~TQueue() {
    TNode *p;
    while(List) {
        p = List;
        List = List->Next;
        delete p->Data;
        delete p;
    }
    cout << "TQueue destructor is called" << endl;
}
/*****/
void TQueue::Add(TShape *psh) {
    if(!List) {
        List = new TNode(psh);
    }
    else {
        TNode *p = List;
        while(p->Next) {
            p = p->Next;
        }
        p->Next = new TNode(psh);
    }
}
/*****/
TShape *TQueue::Remove() {

```

```

TNode *p;
TShape *data;
if(!List) {
    return(NULL);
}
else {
    p = List;
    List = p->Next;
    data = p->Data;
    delete p;
    return(data);
}
}

```

---

```

/*main.cc*/

```

```

#include <iostream>
#include "shape.h"
#include "circle.h"
#include "rectangle.h"
#include "queue.h"
/*****/
int main() {
    cout << "***** 0" << endl;
    TQueue queue;
    cout << "***** 1" << endl;
    queue.Add( new TCircle(TPoint(2,3), 5, "Sun" ) );
    queue.Add( new TRectangle(TPoint(0,0), 3, 4, "Stone" ) );
    cout << "***** 2" << endl;
    queue.Remove()->Print(cout);
    cout << "***** 3" << endl;
    queue.Add( new TCircle(TPoint(1,4), 1, "Moon" ) );
    queue.Add( new TRectangle(TPoint(2,6), 2, 2, "Cloud" ) );
    cout << "***** 4" << endl;
    TShape *psh;
    while( (psh=queue.Remove()) ) {
        psh->Print(cout);
        cout << "-----\n";
        cout << (( psh->IsEquilateral() ) ? "Equilateral" : "Not Equilateral") << endl;

        cout << "-----\n";
        cout << psh->GetPerimeter() << endl;
        cout << "-----\n";
        delete psh;
        cout << "=====\n";
    }
    cout << "***** 5" << endl;
    TRectangle r(TPoint(0,0), 1, 1, "Old Label");
    r.SetLabel("New Label");
    r.Print(cout);
    cout << "***** 6" << endl;
    ((TShape *)(&r))->SetLabel("New Label");
    r.Print(cout);
    cout << "***** 7" << endl;
    return(0);
}

```

```
}
```

---

```
/*makefile*/
```

```
project: shape.o circle.o rectangle.o queue.o main.o
    g++ -Wall shape.o circle.o rectangle.o queue.o main.o -o project
shape.o: shape.cc shape.h
    g++ -Wall -c shape.cc
circle.o: circle.cc shape.h circle.h
    g++ -Wall -c circle.cc
rectangle.o: rectangle.cc shape.h rectangle.h
    g++ -Wall -c rectangle.cc
queue.o: queue.cc shape.h queue.h
    g++ -Wall -c queue.cc
main.o: main.cc shape.h circle.h rectangle.h queue.h
    g++ -Wall -c main.cc
clean:
    rm *.o project
systar:
    tar -xvf project.tar shape.h shape.cc circle.h circle.cc \
    rectangle.h rectangle.cc queue.h queue.cc main.cc makefile
```

---

```
/*Program Output*/
```

```
***** 0
TQueue default constructor is called
***** 1
TShape default constructor is called
TCircle default constructor is called
TShape default constructor is called
TRectangle default constructor is called
***** 2
Circle Number 1:
Label: Sun
Base: 2 , 3
Radius: 5
***** 3
TShape default constructor is called
TCircle default constructor is called
TShape default constructor is called
TRectangle default constructor is called
***** 4
Rectangle Number 100:
Label: Stone
Base: 0 , 0
Width: 3
Height: 4
-----
TRectangle::IsEquilateral is called
Not Equilateral
-----
TRectangle::GetPerimeter is called
14
```

```
-----
TRectangle destructor is called
TShape destructor is called
=====
Circle Number 2:
Label: Moon
Base: 1 , 4
Radius: 1
-----
TShape::IsEquilateral is called
Equilateral
-----
TCircle::GetPerimeter is called
6.28312
-----
TCircle destructor is called
TShape destructor is called
=====
Rectangle Number 101:
Label: Cloud
Base: 2 , 6
Width: 2
Height: 2
-----
TRectangle::IsEquilateral is called
Equilateral
-----
TRectangle::GetPerimeter is called
8
-----
TRectangle destructor is called
TShape destructor is called
=====
***** 5
TShape default constructor is called
TRectangle default constructor is called
TRectangle::SetLabel is called
Label cannot be changed
Rectangle Number 102:
Label: Old Label
Base: 0 , 0
Width: 1
Height: 1
***** 6
Rectangle Number 102:
Label: New Label
Base: 0 , 0
Width: 1
Height: 1
***** 7
TRectangle destructor is called
TShape destructor is called
TQueue destructor is called
```

This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.