附录C 基 本 类

本附录提供我们在一些模式的C++示例代码中用到的基本类。我们力求使这些类尽量简短。这些基本类包括：

- List，对象的顺序列表。
- Iterator，顺序存取聚集对象的接口。
- ListIterator，遍历一张List的Iterator。
- Point，一个两维点。
- Rect，一个轴对齐的矩形。

在某些编译器中，一些新的C++标准类型可能还未实现。特别地，如果你的编译器没有定义bool类型，你可以象下面这样手工定义它：

```
typedef int bool;
const int true = 1;
const int false = 0;
```

C.1 List

List模板类是一个用来存储一个对象序列的基本容器。 List存放元素的值，其元素既可以是内置类型也可以是类的对象。例如， List<int>声明了一个整数序列。但在大多数模式中使用它来存储对象指针，比如List<Glyph*>。这样List类就可以用于异质元素列表。

为方便使用，List类也提供了栈形式的操作。这样就可以直接将 List用作栈，而无需再定义新类。

```
template <class Item>
class List {
public:
    List(long size = DEFAULT_LIST_CAPACITY);
    List(List&);
    ~List();
    List& operator=(const List&);

    long Count() const;
    Item& Get(long index) const;
    Item& First() const;
    Item& Last() const;
    bool Includes(const Item&) const;

    void Append(const Item&);
    void Prepend(const Item&);

    void Remove(const Item&);
    void RemoveLast();
    void RemoveFirst();
    void RemoveAll();
```

```
    Item& Top() const;
    void Push(const Item&);
    Item& Pop();
};
```

下面较详细地讨论这些操作。

构造、析构、初始化和赋值

```
List(long size)
```

初始化列表。参数size提示初始元素数目。

```
List(List&)
```

重载缺省拷贝构造函数，以正确地初始化成员数据。

```
~List()
```

释放该列表的内部数据结构的存储空间。但它并不释放其元素的数据。设计者不希望用户继承这个类，因而析构函数不是虚的。

```
List& operator=(const List&)
```

实现列表赋值，以正确赋值各成员数据。

访问

这些操作支持对列表元素的基本存取。

```
long Count() const
```

返回列表中对象的数目。

```
Item& Get(long index) const
```

返回制定下标处的对象。

```
Item& First() const
```

返回列表的第一个对象。

```
Item& Last() const
```

返回列表的最后一个对象。

```
bool Includes(const Item&) const
```

列表是否含有给定元素。本操作要求列表元素类型支持用于比较的 ==操作。

增添

```
void Append(const Item&)
```

在列表尾部添加元素。

```
void Prepend(const Item&)
```

在列表头部插入元素。

删除

```
void Remove(const Item&)
```

从列表中删除给定元素。本操作要求列表元素类型支持用于比较的 ==操作。

```
void RemoveLast()
```

删除最后一个元素。

```
void RemoveFirst()
```

删除第一个元素。

```
void RemoveAll()
```

删除所有元素。

栈接口

```
Item& Top() const
```

返回栈顶元素（将列表视为一个栈）。

```
void Push(const Item&)
```

将该元素压入栈。

```
Item& Pop()
```

弹出栈顶元素。

C.2 Iterator

Iterator是定义了一种遍历对象集合的接口的抽象类。

```
template <class Item>
class Iterator {
public:
    virtual void First() = 0;
    virtual void Next() = 0;
    virtual bool IsDone() const = 0;
    virtual Item CurrentItem() const = 0;
protected:
    Iterator();
};
```

其操作含义为：

```
virtual void First()
```

使本Iterator指向顺序集合中的第一个对象。

```
virtual void Next()
```

使本Iterator指向对象序列的下一个元素。

```
virtual bool IsDone() const
```

当序列中不再有未到达的对象时返回真。

```
virtual Item CurrentItem() const
```

返回序列中当前位置的对象。

C.3 ListIterator

ListIterator实现了遍历列表的Iterator接口。它的构造函数以一个待遍历的列表为参数。

```
template <class Item>
class ListIterator : public Iterator<Item> {
public:
    ListIterator(const List<Item>* aList);

    virtual void First();
    virtual void Next();
    virtual bool IsDone() const;
    virtual Item CurrentItem() const;
};
```

C.4 Point

Point表示两维笛卡儿坐标空间上的一个点。Point支持一些最基本的向量运算。Point的坐标值类型定义为：

```
typedef float Coord;
```

Point的操作含义是自明的。

```
class Point {
public:
    static const Point Zero;

    Point(Coord x = 0.0, Coord y = 0.0);

    Coord X() const;  void X(Coord x);
    Coord Y() const;  void Y(Coord y);

    friend Point operator+(const Point&, const Point&);
    friend Point operator-(const Point&, const Point&);
    friend Point operator*(const Point&, const Point&);
    friend Point operator/(const Point&, const Point&);

    Point& operator+=(const Point&);
    Point& operator-=(const Point&);
    Point& operator*=(const Point&);
    Point& operator/=(const Point&);

    Point operator-();

    friend bool operator==(const Point&, const Point&);
    friend bool operator!=(const Point&, const Point&);

    friend ostream& operator<<(ostream&, const Point&);
    friend istream& operator>>(istream&, Point&);
};
```

静态成员Zero代表Point(0,0)。

C.5 Rect

Rect代表一个轴对齐的矩形。一个矩形用一个原点和一个范围（长度和宽度）来表示。其操作含义也是自明的。

```
class Rect {
public:
    static const Rect Zero;
```

```
    Rect(Coord x, Coord y, Coord w, Coord h);
    Rect(const Point& origin, const Point& extent);

    Coord Width() const;    void Width(Coord);
    Coord Height() const;   void Height(Coord);
    Coord Left() const;     void Left(Coord);
    Coord Bottom() const;   void Bottom(Coord);

    Point& Origin() const; void Origin(const Point&);
    Point& Extent() const; void Extent(const Point&);

    void MoveTo(const Point&);
    void MoveBy(const Point&);

    bool IsEmpty() const;
    bool Contains(const Point&) const;
};
```

静态成员Zero**等于矩形**

```
Rect(point(0, 0)point(0, 0));
```