

C++ struct Examples

A C structure:

```
struct bx {  
    char buf[MAXBUF+1];  
    int size;  
    int front;  
    int rear;  
};
```

C++ allows functions to be structure members. This example shows a special kind of function, a constructor, used to initialize an object upon creation:

```
struct by {  
    by() { size = MAXBUF+1; front = rear = 0; }  
    char buf[MAXBUF+1];  
    int size;  
    int front;  
    int rear;  
};
```

Structures and Classes

```
struct bz {
    bz() { size = MAXBUF+1; front = rear = 0; }
    int succ(int i) { return (i + 1) % size; }
    int enter(char);
    char leave();
    char buf[MAXBUF+1];
    int size;
    int front;
    int rear;
};
```

Function bodies defined separately:

```
int bz::enter(char x) { /* body of enter */ }
char bz::leave() { /* body of leave */ }

struct x { <member-declarations> };
```

is just shorthand for

```
class x { public: <member-declarations> };
```

Classes Support Information Hiding

The following class implements an abstract buffer type:

```
class buffer {
public:
    buffer() { size = MAXBUF+1; front = rear = 0; }
    int  enter(char);
    char leave();
private:
    char buf[MAXBUF+1];
    int  size, front, rear;
    int  succ(int i) { return (i + 1) % size; }
};
```

Derived Structures

```
struct <derived> : <base> {  
    <added-members>  
};
```

A derived structure inherits all of the members of its base structure except the base constructor.

```
struct B {  
    int x;  
    char f();  
    B() { x = 1; }  
};  
struct D : B {  
    int g();  
};
```

declaration of structure B
public data member
public member function

D derived from B
added member function

Structure D has three members: x, f and g.

Derived Classes - Controlling Visibility of Names

```
class <derived> : public <base> {  
    <member-declarations>  
};
```

All inherited members are visible to outside code when `public` is used. This is just as is the case for derived structures.

```
class <derived> : private <base> {  
    <member-declarations>  
};
```

When `private` is used, inherited members are only seen in the derived class.

protected can also be used inside of a class to define members that are only visible to a derived class, even when `public` is used in the derivation.

Defining a Base Class for Deriving Classes

```
class circlist {
public:
    int  empty();
protected:
    circlist();
    void push(int);
    int  pop();
    void enter(int);
private:
    cell *rear;
};
```

```
class queue : private circlist {
public:
    queue();
    void enterq(int);
    int  leaveq();
    circlist::empty;      inherited and explicitly made public
};
```

Implementing Derived Classes

```
class queue : private circlist {
public:
    queue() {}
    void enterq(int x) { enter(x); }
    int leaveq()      { return pop(); }
    circlist::empty;
};

class stack : private circlist {
public:
    stack {}
    void push(int x) { circlist::push(x); }
    int pop()       { return circlist::pop(); }
    circlist::empty;
};
```

These derived classes provide an example of
implementation inheritance.