Chapter 9

More on Classes and Objects

Chapter 9 Topics (Part 2)

- **Object Pointer**
  - Object Pointer Declaration
  - Pointer `this`
- **`const` Declaration**
  - `const` Data Member
  - `const` Member Function
  - `const` Object
  - Pointers with `const` Declaration
- **`new` and `delete` Operators**
- **Assignment and Copy**
  - Meaning of Assignment
  - Copy Constructor

Object Pointer Declaration

SYNTAX

```
ClassName *PointerName;
```

The pointer variable will hold the beginning address of an object in memory space.

Accessing Members of an Individual Object

- point to a created object
- then, access members of that object

SYNTAX

```
ClassName *PointerName;
PointerName=&ObjectName;
PointerName->MemberName;
```

Accessing Array of Objects

- point to a created array of objects
- then, access members of elemental objects

SYNTAX

```
ClassName *PointerName;
PointerName=ArrayName;
PointerName->MemberName;
```

EXAMPLES

```
PlayingCard *pCard, *pCardArray;
PlayingCard cardOne;
pCard=& cardOne;
PlayingCard cardArray[52];
pCardArray= cardArray;
```

```
cout<<"The rank of cardOne is: "<<pCard ->rank( )<<endl;
```

```
for(i=0;i<52;i++)
{
   cout<<"The suit of element "<<i<<"is: "<<cardArray[i].suit( )<<endl;
   cout<<"The rank of element "<<i<<"is: "<<pCardArray ->rank( )<<endl;
   pCardArray++;
}
```

Pointer `this`

Every member function includes a hidden pointer with a fixed name: `this`, which holds the beginning address of the object that the member function belongs to.

EXAMPLE

```
int PlayingCard ::rank ( )
{ return rankValue; }
```

```
int PlayingCard ::rank ( PlayingCard *this )
{ return this->rankValue; }
```

Chapter 9 Topics (Part 2)

- Object Pointer
  - Object Pointer Declaration
  - Pointer `this`
- `const` Declaration
  - `const` Data Member
  - `const` Member Function
  - `const` Object
  - Pointers with `const` Declaration
- `new` and `delete` Operators
- Assignment and Copy
  - Meaning of Assignment
  - Copy Constructor

const Data Member

SYNTAX

```
const DataType MemberName;
```

- The value of const data members can not be modified.
- Only the initializer can be used to specify the initial value for const dada members.

const Member Function

SYNTAX

```
DataType FunctionName(Parameter List) const;
```

- The const member function can access the data members in its class, but can not modify their values.
- The const member function can not invoke non-const member functions.

const Object

SYNTAX

```
const ClassName ObjectName[(Augument List)];//or
ClassName const ObjectName [(Augument List)];
```

- All the data members of a const object are const data members.
- Only the const member function of a const object can be accessed.

const Pointer Variable

SYNTAX

```
ClassName ObjectName;
ClassName * const PointerName;
PointerName =& ObjectName;
```

- The value of const pointer variable can not be changed after initialization.
- But, the value of pointed object can be changed.

Pointing to const Object

SYNTAX

```
ClassName ObjectName;
const ClassName * PointerName =& ObjectName;
```

- The value of pointed object can not be changed after initialization.
- But, the value of pointer variable can be changed.

const Reference of an Object

SYNTAX

```
DataType FunctionName(const ClassName & ReferenceName);
```

- The value of referenced object can not be changed within the function.

Chapter 9 Topics (Part 2)

- Object Pointer
  - Object Pointer Declaration
  - Pointer `this`
- `const` Declaration
  - `const` Data Member
  - `const` Member Function
  - `const` Object
  - Pointers with `const` Declaration
- `new` and `delete` Operators
- Assignment and Copy
  - Meaning of Assignment
  - Copy Constructor

Memory Allocation

- *Stack-based* allocation. Amount of space required is determined at compile time, based on static types of variables.
- *Heap-based* allocation. Amount of space used can be determined at run-time, based upon dynamic considerations.

Dynamic Memory Allocation

Requires user to explicitly allocate new objects and free no longer used storage.

SYNTAX

```
ClassName *PointerName;

PointerName=new ClassName;//or

PointerName=new ClassName(Argument List);

...

delete PointerName;
```

EXAMPLE

```
PlayingCard *pCard;
pCard= new PlayingCard(PlayingCard::Heart,1);
//invoke the constructor with two parameters
...
delete pCard;
//invoke the destructor
```

Chapter 9 Topics (Part 2)

- Object Pointer
  - Object Pointer Declaration
  - Pointer `this`
- `const` Declaration
  - `const` Data Member
  - `const` Member Function
  - `const` Object
  - Pointers with `const` Declaration
- `new` and `delete` Operators
- Assignment and Copy
  - Meaning of Assignment
  - Copy Constructor

Meaning of Assignment

- Copy semantics（语义）

  x and y are independent of each other, a change in one has no effect on the other.

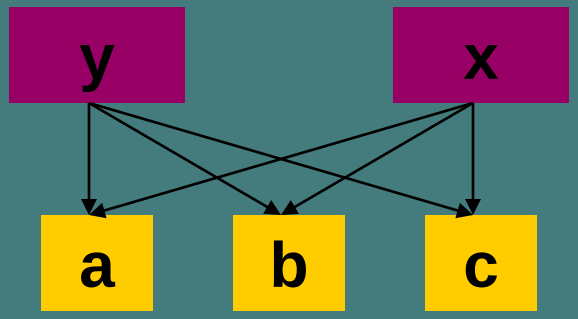
- Pointer semantics

  x and y refer to the same object, and hence a change in one will alter（改变） the other.

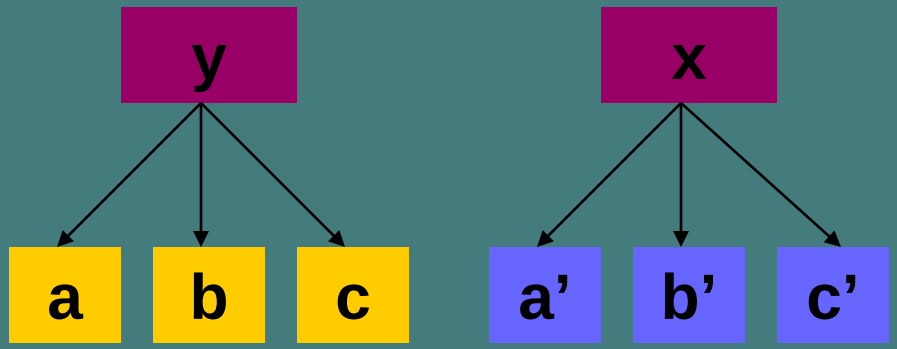
Shallow Copy versus Deep Copy

A shallow copy shares instance variables with the original.

A deep copy creates new copies of the instance variables.



shallow copy



deep copy

Assignment in C++

SYNTAX

```
ClassName ObjectName1,ObjectName2;
ObjectName1=ObjectName2;
```

EXAMPLE

```
PlayingCard cardOne, cardTwo;
...
cardOne=cardTwo;
```

Copy Constructor Declaration

SYNTAX

```
ClassName:: ClassName(const ClassName & ObjectName)
{
  ...
 }
```

INVOKING

```
ClassName ObjectName1;
...
ClassName ObjectName2(ObjectName1);//or
ClassName ObjectName3=ObjectName1;
```

Copy Constructor for class PlayingCard

```
class PlayingCard {
public:
 PlayingCard(const PlayingCard &source)
 {
   suitValue =source. suitValue;
   rankValue =source. rankValue;
  }
  …
private:
        Suits suitValue;
        int rankValue;
};
```

```
PlayingCard cardOne(PlayingCard::Club,6);
PlayingCard cardTwo(cardOne);
PlayingCard cardThree=cardOne;
```