Advanced Programming

Operator Overloading

Topics

- Dynamic memory allocation
- Copy constructor
- Overloading Operators
  - Overloading assignment operator
  - Overloading function call operator
  - Overloading new
- Examples
- Bitwise operations

Overloading ()

- The function call operator () can be overloaded for objects of class type.
- When () are overload, a new way to call a function is not created. Rather, an operator function is created that can be passed an arbitrary number of parameters.

Example

```
class Point
{
   int x, y;
   public:
   Point() {x=0;  y=0; }
   Point& operator()(int dx, int dy)
   {
       x += dx;
       y += dy;
      return *this;
   }
};
 void main()
 {
     Point pt;
     // Offset this coordinate x with 3 points
    // and coordinate y with 2 points.
    pt(3, 2);
 }
```

Overloading ()

- Overloaded () operator can be used to
  - Access multidimensional array elements
  - Initialize or erase all elements in a matrix

Dynamic Memory Allocation

- new is used for dynamic allocation of memory
- new can allocate an array of items
- If an array of items is created using new operator, there should be constructor with no parameter.
- delete operator is used to return the allocated memory to the system

Overloading new

- The memory management operators can be overloaded to customize allocation and de-allocation.
- **new** should return a pointer to a newly allocated object on the heap, **delete** should de-allocate memory, ignoring a NULL argument.
- To overload **new**, several rules must be followed:
  - **new** must be a member function
  - the return type must be *void**
  - the first explicit parameter must be a *size_t* value
- To overload **delete** there are also conditions:
  - **delete** must be a member function
  - the return type must be *void*

Overloading ->, *

- Reference to memory locations using pointers can be diverted to user defined functions by overloading ->, and * operators.
- class X

```
{
    public:
    Y* operator->();
}
```

Example

- Assume the records of a library are given in a raw format where
  - each field starts with % and a letter followed by the value.
  - The values of each field end with null character
  - The order of the fields are not preserved
  - Some fields may not be present in the record
- Write a book and a rawBook class and overload -> operator to access the book data fields

Overloading =

- Must be a member function
- When a class contains references or pointers to outside resources, the assignment operator should be overloaded
- Assigning an object to itself should not cause any problem

Example

- Assume a class is used to represent a student. Define the class dynamically allocating the space for name and address fields. Overload the assignment operator.

Object Copying

- To copy an object either:
  - Overload = operator
  - Use a constructor which gets an object of the same class as argument X::X(X& X). This constructor is called copy constructor

Copy Constructor

- The **copy constructor** is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is also used to:
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

Copy Constructor

- Copy constructors are necessary when objects include pointers.
- Without a copy constructor there can be problems such as:
  - Objects referring to the same locations
  - Destructor function is called more than once

Example

- Define a class to represent a one-dimensional integer vector. The vector is created dynamically using the length argument by the constructor function.
- Write a non-member function to display the vector.

Function Pointers

- It is possible to store the address of a function in a pointer.
  - E.g. int (*MyFunction)(int&, int&);
  - MyFunction = &swap;
- Function pointers are used as arguments to functions to create different versions of it

Example

- Write a function named BinarySearch to locate an item in an array. To compare elements of the array use a pointer to the compare function.

Example

- Persistent arrays
  - Assume you are using an array in your program. The array stores some data. You want to have the same data loaded into the array when you run the program next time.

Example

- Assume an array is created with an initial size. However, if the index range is larger than the array size, the array is enlarged to include the given index. Define the class and overload necessary operators

Bitwise Operations

- The smallest addressable unit in memory is byte. However, it is possible to access and modify bits.
- Bitwise operators are :
  - & bitwise AND
  - | bitwise OR
  - ^ bitwise XOR
  - ~ bitwise NOT
  - >> shift right
  - << shift left

Example

- Define a Byte class where each bit is accessible through the [] operator.

Streams

- A stream is defined as a sequence of bytes
- We consider input and output as stream of bytes being inserted or extracted from a stream, respectively.
- Two main stream classes are:
  - istream
  - ostream
  - iostream has been driven from both istream and ostream

Streams

- cin, cout, and cerr are object instances from istream and ostream
- >> and << operators have been overloaded for input and output operations.

Overloading >>

- >> is overloaded using a global function
- ostream& operator>> (ostream&, classType)

File Streams

- Files are also treated as streams.
- ifstream, ofstream, and fstream are used for file I/O
- Example: fstream f(“MyFile.txt”, ios::out|ios::in)