Module 4: Review

Prof. Tran Minh Triet

Acknowledgement

- Slides
  - Course CS202: Programming Systems

    Instructor: MSc. Karla Fant,

    Portland State University
  - Course CS202: Programming Systems

    Instructor: Dr. Dinh Ba Tien,

    University of Science, VNU-HCMC
  - Course DEV275: Essentials of Visual Modeling with UML 2.0

    IBM Software Group

Outline

- OOP: class/object, data hiding, encapsulation...
- Constructors, destructor
- Copy constructor and assignment operator
- Operators overloading

Object-oriented programming

- Class/object?
- Data hiding
- Encapsulation
- Inheritance
- …

➔ portable, mantainable, extensible code

Object-oriented programming

- Class: an abstraction representing a user defined data type that defines structure and behavior.
- Object: an instance of a class that gets allocated and deallocated.

Class members

- Member: data and operations that are defined as part of a class.
- Data members: the data that is specified as part of a class.
- Member functions: the operations (or methods) that can be performed on each object.

Class interface & implementation

- Class interface: declares the data and operations for a user defined type. Also known as a class definition.
- Class implementation: implements the operations of the class.

Constructors

- Syntax? return type?
- When is it invoked?
- Overloading?
- Default constructor
- Initialization list
  - Better or worse?
  - In which case, do we have to use initialization list?
  - The order of being initialized.

Constructor

- Constructors: implicitly invoked when the lifetime of an object begins; typically they are used to initialize data members.
- Default constructor: a constructor with no arguments or with arguments that all have default values.
- Initialization list: supplies the initial values for data members.

Destructor

- Syntax?
- When do we have to define?
- Anything else?

Destructor

- Destructors: implicitly invoked when the lifetime of an object ends; typically they are used to release any resources set aside for the object.

Copy constructor

- Do we have default copy constructor generated by the compiler?
- What does it do?
- Shallow copy VS deep copy
- When do we need to write a copy constructor?
- Syntax?
- Anything else?

Assignment operator

- Syntax?
- Default assignment operator?
- When do we need to write our own assignment operator?
- What is the difference between copy constructor and assignment operator?

Copy constructor and assignment op.

- Copy constructor: makes a copy of an object upon an object's creation.
- Assignment operator: makes a copy of an object when one object is assigned to another.

Shallow vs Deep copy

- Memberwise copy: the data members of one object are copied into the data members of another object.
- Shallow copy: the data members of one object are copied into the data members of another object without taking any dynamic memory pointed to by those data members into consideration.
- Deep copy: any dynamic memory pointed to by the data members is duplicated and the contents of that memory is copied.

Operator overloading

- Guidelines?
- Non-member functions?
- Overload insertion and extraction operators
- Subscript operator

Operator overloading

- Operator overloading: defining the behavior of C++ operators when one or more of the operands are objects of a class.

Operator overloading restriction

- Operators must come from the built-in operators. We cannot define our own operators.
- Operators maintain their precedence and associativity. We cannot alter it for our own classes.
- Operators must be overloaded to expect the correct number of operands. Unary operators expect one operand and binary operators expect two operands.
- Operators can only be overloaded when at least one of the operands is a object of a class. We cannot redefine the operation of operators on built-in types.
- Operators cannot have default arguments.

Guidelines

- Determine if any of the class operations should be implemented as overloaded operators instead of member functions.
- Be sure that the operators provided for our class make sense to the abstraction being defined.
- Be consistent with how the operators work with the built-in types. Understand what data types are allowed as operands, what conversions can be applied to the operands, whether or not the operands are modified, what data type is returned...
- Provide a complete set of overloaded operators for each class.

Operators that cannot be overloaded

- `::` scope resolution operator
- `.` direct member access operator
- `.*` direct pointer to member access operator
- `?:` conditional operator
- `sizeof` size of object operator
- `new` memory allocation operator
- `delete` memory deallocation operator
- `static_cast` cast operator
- `const_cast` cast operator
- `reinterpret_cast` cast operator
- `dynamic_cast` run time type identification cast operator
- `typeid` run time type identification type operator

Operators must be overloaded as members

- `=` simple assignment operator
- `[]` subscript operator
- `()` function call operator
- `->` indirect member access operator
- `->*` indirect pointer to member access operator

Unary operators that should be overloaded as members

- `&` address of operator
- `*` dereference operator
- `++` increment operator (both prefix and postfix)
- `--` decrement operator (both prefix and postfix)
- `+` plus operator
- `-` minus operator
- `~` bitwise negation operator (complement)
- `!` logical negation operator

Binary operators that should be overloaded as members

- `*= /= %= += -=`
  arithmetic assignment operators

- `<<= >>= &= ^= |=`
  bitwise assignment operators

Operators that should be overloaded as non-members

- `* / % + -` arithmetic operators
- `<< >> & ^ |` bitwise operators
- `< <= > >=` relational operators
- `== !=` equality operators

Operators that should not be overloaded

- `&&` `||` logical operators
- `,` comma operator

Operators that work best in pairs

- `-> ->*` indirect member and pointer to member access operators
- `& *` address of and dereference operators
- `++ --` increment and decrement operators (both prefix and postfix)
- `+ -` plus and minus operators
- `* /` multiplication and division operators
- `+ -` addition and subtraction operators
- **`new delete`** dynamic allocation/deallocation
- **`new[] delete[]`** dynamic array allocation/deallocation