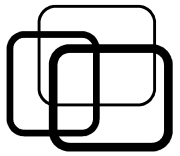
Object and Class

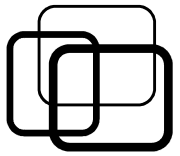
Inst. Nguyễn Minh Huy

Contents



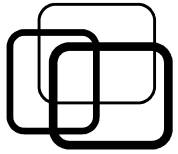
- Basic concepts.
- Object usage.
- Access control.

Contents



- **Basic concepts.**
- Object usage.
- Access control.

Basic concepts



- Procedural vs. object oriented:
  - Cooking: món thịt kho trứng + rau muống xào.

Action
Lặt
Luộc
Ướp
Kho
Xào
Bóc vỏ

Procedural
Ướp (Thịt)
Luộc (Trứng)
Lặt (Rau)
Bóc vỏ (Trứng)
Kho (Thịt, Trứng)
Xào (Rau)

Object Oriented
Trứng. Luộc( )
Trứng. Bóc vỏ( )
Rau. Lặt( )
Rau. Xào( )
Thịt. Ướp( )
Thịt. Kho(Trứng)

Materials
Thịt
Trứng
Rau

Procedural:

- Action first.

- Function + Data.
(Verb) + (Object)

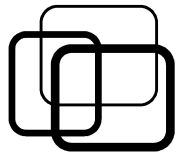
Object Oriented:

- Data first.

- Data triggers function.
(Object) does (Verb)

→ Change your thinking!!

Basic concepts



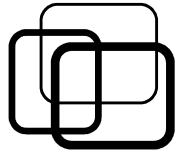
- What is object?
  - Program is a complex machine.
  - Compose of units:
    - Basic units: variables, functions.
    - Procedural: function + variables.

      ➔ **Not easy to create abstract program!**
    - Object Oriented: variable triggers functions.

      ➔ **Need a new kind of unit.**

Special unit: Object!!

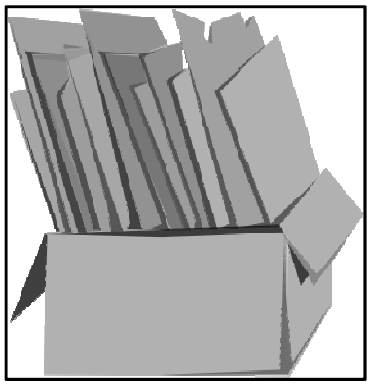
Basic concepts



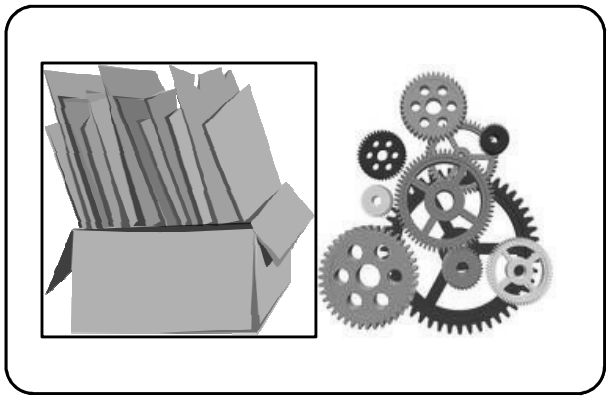
- Object characteristics:
  - A special variable.
  - Contain data and trigger functions:
    - Attribute: data of object.
    - Method: functions of object.

Function

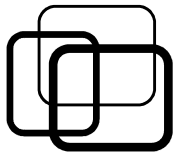
Variable/Struct



Object

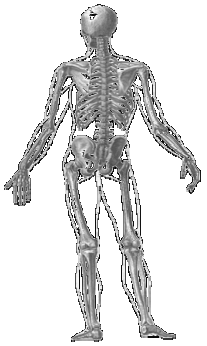


Basic concepts



- What is class?
  - Variable ~ Type.
  - Struct variable ~ Struct type.
  - Object ~ Class:
    - Class is object type.
    - A description of:
      - Attributes.
      - Methods.

Person:



- Name.
- Age.
- Hair Color.
- Eat( ).
- Work( ).

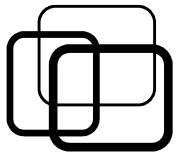
Person1:

- Name: Peter.
- Age: 25.
- Hair Color: Brown.
- Eat().
- Work().

Person2:

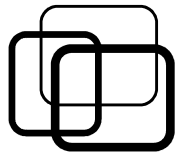
- Name: Thomas.
- Age: 50.
- Hair Color: White.
- Eat( ).
- Work( ).

Contents



- Basic concepts.
- **Object usage.**
- Scope.

Object usage

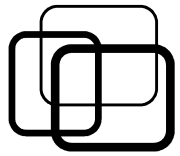


- Object in C++:
  - Same as struct variable.
  - Declare class (file .h):

```
class <Class name>
{
    <Attribtes>;
    <Methods>;
};
```

  - Implement class (file .cpp):
    - Implement methods same as functions.
  - Create object from class:
    - Declare variables from class.

Object usage



- Example: object oriented vs. procedural.

```
// Declare class, file Fraction.h
class Fraction
{
private:
        int   m_num;
        int   m_den;
public:
        Fraction add(Fraction p);
};
```

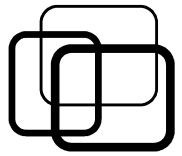
```
// Declare struct, file Fraction.h
struct Fraction
{
        int   m_num;
        int   m_den;
};

Fraction add(Fraction p1, Fraction p2);
```

```
// Implement class, file Fraction.cpp
Fraction Fraction::add(Fraction p)
{
        // …
}
```

```
// Implement add, file Fraction.cpp
Fraction add(Fraction p1, Fraction p2)
{
        // …
}
```

Object usage



- Example: object oriented vs. procedural.

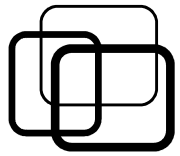
```
// Use object, file main.cpp
int main()
{
        Fraction   p1;
        Fraction   p2;

        p1.add( p2 );
}
```

```
// Use struct, file main.cpp
int main()
{
        Fraction   p1;
        Fraction   p2;

        add( p1, p2 );
}
```

Object usage

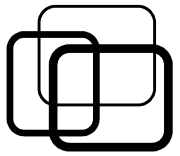


- Example: implement methods.

```
class Fraction
{
private:
        int   m_num;
        int   m_den;
public:
        Fraction add(Fraction p)
        {
            // Implement inside class.
        }
};
```

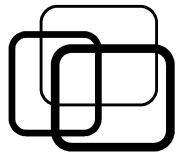
```
Fraction Fraction::add(Fraction p)
{
    // Implement outside class.
}
```

Contents



- Basic concepts.
- Object usage.
- **Access control.**

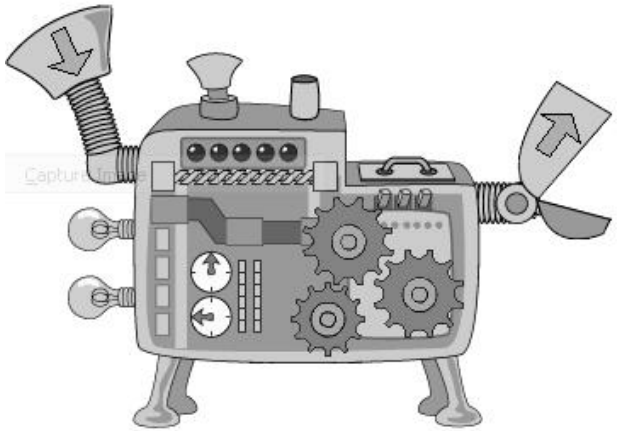
Access control



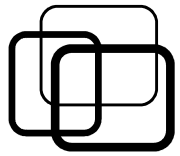
- How to show/hide object members?
  - Class can define which members are shown/hidden.

```
class Fraction {
    // To be shown members.
    int  num;
    int  den;
    Fraction add( Fraction p );

    // To be hidden members.
    int  max;
    int  min;
    int  gcd;
    int  findGCD( );
};
```



Access control



- C++ access specifiers:

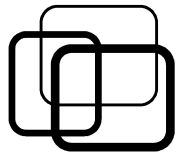
Keywords	Scope
private	Inside class only (class methods).
public	Inside and outside class (every functions).
protected	Inside class and children (class and children methods).

  - Struct: default public access.
  - Class: default private access.

```
struct Fraction {
    // Default public.
    int  num;
    int  den;
};
```

```
class Fraction {
    // Default private.
    int  num;
    int  den;
};
```

Access control



- Example: private, public, default.

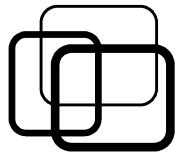
```
class A
{
        int   x;     // Default private.
public:
        int   y;
private:
        int   z;
        void calculate( );
public:
        int getX( );
};
```

```
int main()
{
        A   obj;

        int   u = obj.x;       // Wrong
        int   v = obj.y;       // Right
        int   t  = obj.z;      // Wrong

        obj.calculate( );      // Wrong
        int   w = obj.getX( ); // Right
}
```

Access control



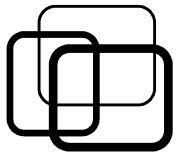
- Dr. Guru advises:
  - Rule of Black Box:
    - Attributes: use **private** to hide inside.
    - Methods: use **public** to provide functions.

```
class Fraction
{
private:
    int   m_num;
    int   m_den;
public:
    Fraction add( Fraction p );
    Fraction reduce( );
};
```

Attributes

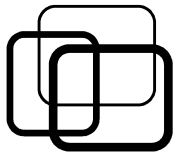
Methods

Summary



- Basic concepts:
  - Object is a new type of variable.
  - Object contains both data and functions.
  - Class is an object type.
- Object usage:
  - Declare class (file .h).
  - Implement class methods (file .cpp).
  - Declare objects from class.
- Access control:
  - Class/struct can show/hide it members.
  - Keywords: private, public, protected.

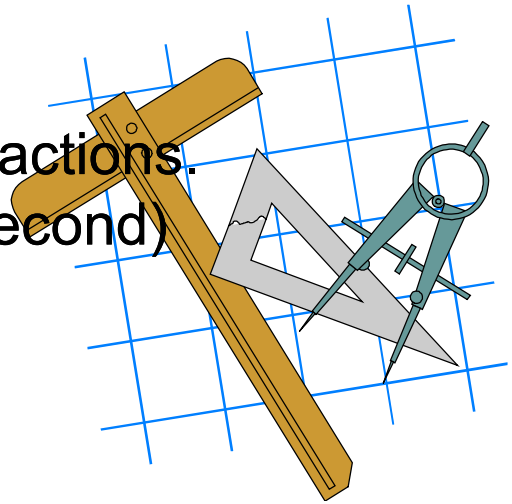
Practice



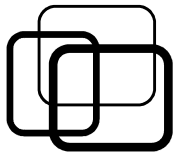
- Practice 2.1:

  Construct class **Fraction** with the following methods:

  - **input**: enter fraction from keyboard.
  - **output**: print fraction to screen.
  - **getNum/setNum**: get/update numerator of fraction.
  - **getDenom/setDenom**: get/update denominator of fraction.
  - **reduce**: return the reduction of fraction.
  - **inverse**: return the inversion of fraction.
  - **add**: return the sum of two fractions.
  - **compare**: return the comparison result of two fractions.
    (0: first = second, -1: first < second , +1: first > second)



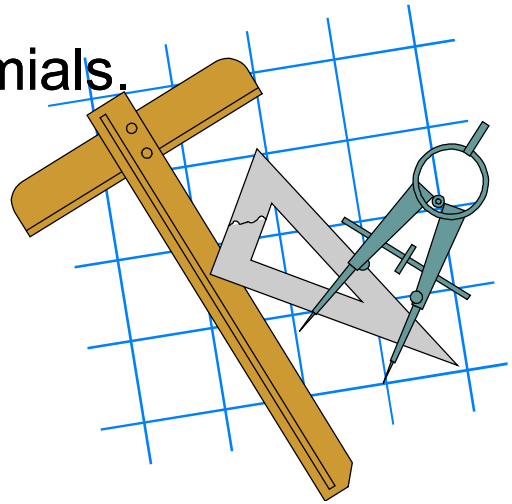
Practice



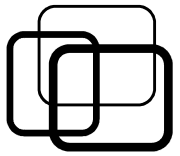
- Practice 2.2:

  Construct class **Monomial** with the following methods:

  - **input**: enter monomial from keyboard.
  - **output**: print monomial to screen.
  - **getCoef/setCoef**: get/update coefficient of monomial.
  - **getExpo/setExpo**: get/update exponent of monomial.
  - **evaluate**: return result of evaluating monomial with float number.
  - **derive**: return the derivative of monomial.
  - **mul**: return the product of multiplying two monomials.



Practice

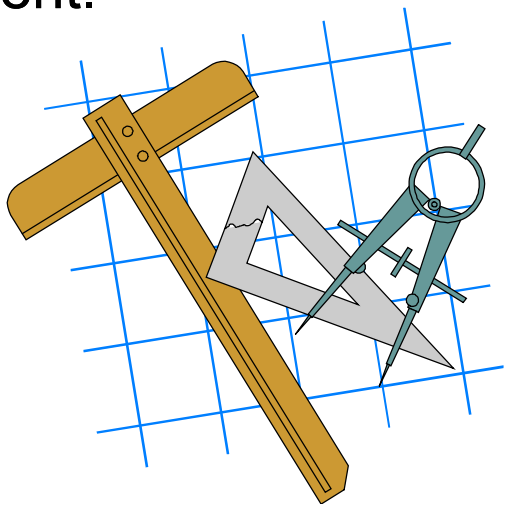


- Practice 2.3:

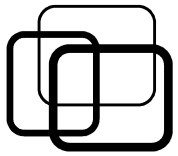
  Student information: name, literature and math points.

  Construct class **Student** with the following methods:

  - **input**: enter student information from keyboard.
  - **output**: print student information to screen.
  - **getName/setName**: get/update name of student.
  - **getLit/setLit**: get/update literature point of student.
  - **getMath/setMath**: get/update math point of student.
  - **calculateGPA**: return GPA of student.
    (GPA = (literature + math) / 2)
  - **grade**: return student grade
    - A (GPA >= 9.0), B (GPA >= 7.0).
    - C (GPA >= 5.0), D (GPA < 5).



Practice



- Practice 2.4:

  Construct class **Array** of integers with the following methods:

  - **input**: enter array size and elements from keyboard.
  - **output**: print array elements to screen.
  - **getSize/setSize**: get/update size of array.
  - **getElement/setElement**: get/update element at specified index.
  - **find**: look for an element and return found index (-1 if not found).
  - **sort**: sort array, the sort criteria can be customized.