C++

程式語言（二）

Introduction to Programming (II)

Inheritance

Joseph Chuang-Chieh Lin

Dept. CSE, NTOU

Platform/IDE

- Dev-C++

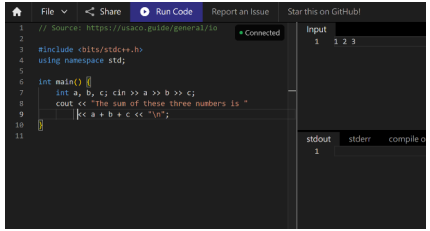
https://www.pngegg.com/en/search?q=Dev-C

- Codeblocks

https://icons8.com/icons/set/code-blocks

- OnlineGDB (https://www.onlinegdb.com/)

- Real-Time Collaborative Online IDE (https://ide.usaco.guide/)



Textbooks (We focusing on C++11)

- ***Learn C++ Programming by Refactoring* (由重構學習 C++ 程式設計). Pang-Feng Liu (劉邦鋒). NTU Press. 2023.**
- ***C++ Primer. 5th Edition.* Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. 2019.**
- *Effective C++*. Scott Meyers. O'Reilly. 2016.
- *Thinking in C++. Vol. 1: Introducing to Standard C++*. 2nd Edition. Bruce Eckel. Prentice Hall PTR. 2000.

Useful Resources

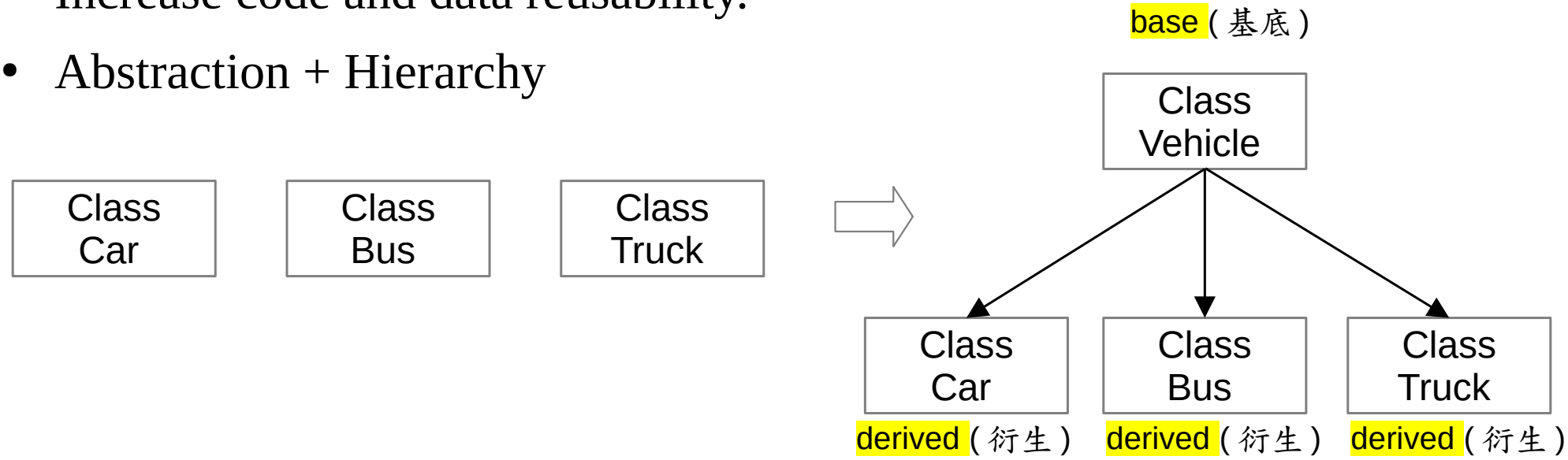
- Tutorialspoint
  - https://www.tutorialspoint.com/cplusplus/index.htm
  - Online C++ Compiler
- Programiz
  - https://www.programiz.com/cpp-programming
- LEARN C++
  - https://www.learncpp.com/
- MIT OpenCourseWare - Introduction to C++
  - https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/pages/lecture-notes/
- Learning C++ Programming
  - https://www.programiz.com/cpp-programming
- GeeksforGeeks
  - https://www.geeksforgeeks.org/c-plus-plus/

Inheritance

Inheritance

https://www.geeksforgeeks.org/inheritance-in-c/?ref=lbp

- Get rid of duplication of the same codes.
- Decrease the chance of error.
- Increase code and data reusability.
- Abstraction + Hierarchy



An Easy Illustrating Example

```
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A
{
    // x is public
    // y is protected
    // z is not accessible from B
};
```

access mode

```
class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A
// 'private' is default for classes
{
    // x is private
    // y is private
    // z is not accessible from D
};
```

Modes of Inheritance

Just like going through a mask...

Example: https://onlinegdb.com/Z7tf4BU0x

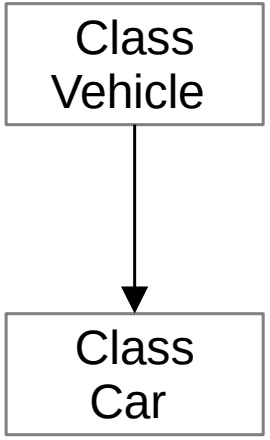
- `Public`
  - public member of the base class => **public** in the derived class.
  - protected members of the base class => **protected** in derived class.
  - **private** members of the base class => not accessible.
- `Protected`
  - public member of the base class => **protected** in the derived class.
  - protected members of the base class => **protected** in derived class.
  - **private** members of the base class => not accessible.
- `Private`
  - public member of the base class => **private** in the derived class.
  - protected members of the base class => **private** in derived class.
  - **private** members of the base class => **NOT** accessible.

Single Inheritance

```
#include<iostream>
using namespace std;

class Vehicle {
public:
    Vehicle() {
      cout << "This is a Vehicle. " << endl;
    }
};

class Car : public Vehicle {
// nothing to do here so far...
};
```




```
int main()
{
    // invoke the constructors
    Car obj;
    return 0;
}
```

Output:

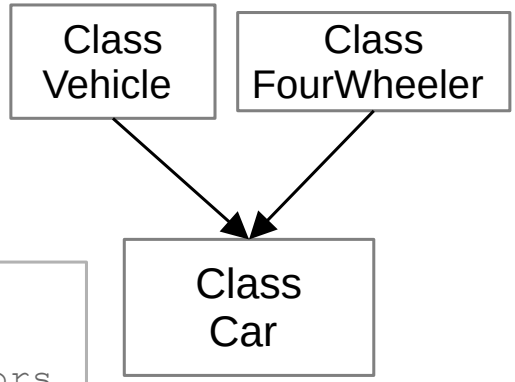
```
This is a Vehicle.
```

Multiple Inheritance

```
#include<iostream>
using namespace std;

class Vehicle {
public:
    Vehicle() {
      cout << "This is a Vehicle."
           << endl;
    }
};
class FourWheeler {
public:
    FourWheeler() {
         cout << "This is a 4 wheeler
                  Vehicle. " << endl;
    }
};
```

```
class Car : public Vehicle, public FourWheeler {
    // nothing to do here so far...
};
```




```
int main()
{
    // invoke the constructors
    Car obj;
    return 0;
}
```

Output:

```
This is a Vehicle.
This is a 4 wheeler Vehicle.
```

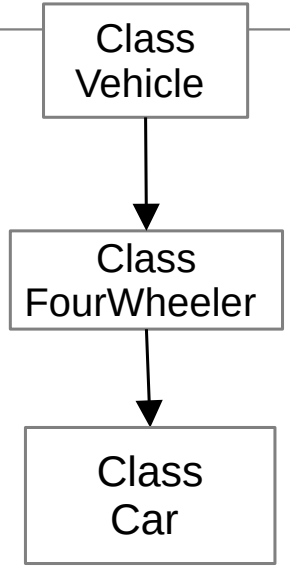
Multilevel Inheritance

```
#include<iostream>
using namespace std;

class Vehicle {
public:
    Vehicle() {
      cout << "This is a Vehicle."
           << endl;
    }
};
class FourWheeler: public Vehicle {
public:
    FourWheeler() {
         cout << "A 4 wheeler Vehicle."
              << endl;
    }
};
```

```
class Car: public FourWheeler {
public:
     Car() {
       cout << "A Car has 4 Wheels." << endl;
     }
};
```

```
int main()
{
    // invoke the constructors
    Car obj;
    return 0;
}
```



Output:

```
This is a Vehicle.
A 4 wheeler Vehicle.
A Car has 4 Wheels.
```

More Details in Examples

- https://www.programiz.com/cpp-programming/public-protected-private-inheritance

Exercise

```
class Shape {
public:
    string type;
protected:
    double parameter;
};
```

```
class Circle : protected Shape {
private:
    double area = 0.0;
public:
    void compute_area()     {
/* please implement this member function */
    }
    void setRadius() {
/* please implement this member function */
    }
    double getArea() {
/* please implement this member function */
    }
};
```

```
int main()
{
    Circle obj;
    obj.setRadius();
    obj.compute_area();
    cout << "Area: " << obj.getArea();
    return 0;
}
```

Sample Input & Output:

```
3.2
Area: 32.1699
```

Exercise

```
class A {
public:
     int x = 0;
     int get_pvt() { return z; }
protected:
     int y = 1;
private:
     int z = 2;
};

class B : public A {
    // x is public
    // y is protected
    // z is not accessible from B
};
```

```
class C : protected A {
    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A {
// 'private' is default for classes
    // x is private
    // y is private
    // z is not accessible from D
};
```

Please modify the code here by "adding appropriate member functions" in the the classes B, C, and D.

```
int main (){
     B obj1;
     C obj2;
     D obj3;
     cout << obj1.x << obj2.y << obj3.y;
     // try to print these values
     // by adding appropriate member
     // functions
}
```

Discussions & Questions