អំពី Inheritance

ជំពូកទី ៩

១. សេចក្តីផ្តើម

នៅជំពូកមុននេះ យើងបានសិក្សាអំពី class និង objects ដែលជាលក្ខណៈមួយនៃ OOP ក៏ប៉ុន្តែការប្រើ class ដាច់តែឯង (ទោល) ពុំទាន់មានលក្ខណៈគ្រប់គ្រាន់ ដើម្បីឆ្លុះបញ្ចាំងពី OOP នោះឡើយ។ ដើម្បីរក្សានិងការប្រើជាថ្មីនូវ objects នៃ class ឲ្យមានលក្ខណៈងាយស្រួលនោះវាត្រូវការទទួលលក្ខណៈពី class មួយឬច្រើនទៅឲ្យ class មួយទៀត។ ក្នុងជំពូកនេះ យើងនឹងសិក្សាពីលក្ខណៈសំខាន់នៃ inheritance ដែលជាអានុភាពមួយនៃ OOP។ Inheritance គឺជាដំណើរនៃការបង្កើត classes ថ្មីពី class ដែលមានរួចហើយ។ Class ដែលមានរួចហើយនោះ ហៅថា base class (ភាសា Java ហៅថា superclass) រីឯ class ដែលទើបបង្កើតថ្មីហៅថា derived class (ភាសា Java ហៅថា subclass)។ Derived class ទទួលលក្ខណៈរបស់ base class។ គេអាចបន្ថែមលក្ខណៈច្រើនទៀតទៅឲ្យ derived class នេះ។ Base class ពុំមានការផ្លាស់ប្តូរដោយដំណើរការរបស់ derived class ឡើយ។ ផលប្រយោជន៍នៃលក្ខណៈ inheritance គឺ ៖

 អាចឲ្យគេប្រើ code ជាថ្មី

 បង្កើនភាពអាស្រ័យនៃ code

 បង្កើនលក្ខណៈដល់ base class

២. អំពី Single Inheritance

Single inheritance គឺជាដំណើរនៃការបង្កើត class ថ្មីពី base class ដែលមានរួច ហើយ។ class ដែលមានរួចហើយត្រូវបានហៅថា direct base class ហើយ class ដែលទើបបង្កើតថ្មីត្រូវបានហៅថា singly derived class។

Single inheritance គឺជាលទ្ធភាពនៃ derived class ដើម្បីទទួលលក្ខណៈ member functions និងអញ្ញាតរបស់ base class ដែលមានរួចហើយ។

ការបង្កើត derived class មួយគឺដូចទៅនឹងការបង្កើត class ធម្មតាដែរ។ Derived class មានសមាសធាតុដូចខាងក្រោម ៖

 ពាក្យគន្លឹះ class

 ឈ្មោះ derived class

 សញ្ញា **:** មួយ

 ការបញ្ជាក់លក្ខណៈរបស់ base class ជាប្រភេទ private, public ឬ protected

 ឈ្មោះ base class ឬ parent class

 ការកំណត់លក្ខណៈបន្ថែមទៅឲ្យ derived class

ទម្រង់ទូទៅនៃការប្រកាស derived class មានដូចខាងក្រោម ៖

class *derived\_class\_name* : private / public / protected *base\_class\_name* {

private :

// *data members*

public :

// *data members*

// *methods*

protected :

// *data members*

};

ឧទាហរណ៍ code ខាងក្រោមនេះបង្ហាញនូវការប្រកាស class មួយទទួលលក្ខណៈពី class មួយទៀត។ Base class មានពីរផ្នែក data member និង member function។ Data member មាន name, id និង sex ហើយត្រូវបានកំណត់នៅក្នុងផ្នែក private។ Member functions មាន getdata() និង putdata() ដែលត្រូវបានប្រកាសនៅក្នុង base class។ Members របស់ base class អាចបញ្ជូនទៅឲ្យ derived class។

**class Student {**

**private :**

**char name[20];** // ការតាងអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**​ long int id;** // ការតាងអញ្ញាត id ជាប្រភេទចំនួនគត់ long int

**char sex;** // ការតាងអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**​ void display();** // ការប្រកាសអនុគមន៍ display()

**};** // ទីបញ្ចប់នៃ class Student

**class PhysicalFit:public Student {**

**private :**

**float height;** // ការតាងអញ្ញាត height ជាប្រភេទទសភាគ float

**float weight;** // ការតាងអញ្ញាត weight ជាប្រភេទទសភាគ float

**public :**

**void input();** // ការប្រកាសអនុគមន៍ input()

**void output();** // ការប្រកាសអនុគមន៍ output()

**};** // ទីបញ្ចប់នៃ class PhysicalFit

Derived class បានទទួលលក្ខណៈរបស់ base class រួមមាន data member និង member functions។ PhysicalFit គឺជា derived class ដែលមានសមាសភាពពីរផ្នែក ៖ private និង public។ លើសពីនេះ ក្រៅពី data members ថ្មីរបស់វា ដូចជា height និង weight នោះ វាអាចទទួល data members របស់ base class ទៀត។ Derived class មិនគ្រាន់តែមាន methods របស់ខ្លួនវាប៉ុណ្ណោះទេ ថែមទាំងមាន methods របស់ base class ទៀតផង។

ឧទាហរណ៍ទី ១**៖** កម្មវិធីខាងក្រោមទទួលតម្លៃតាមរយៈ member របស់ Student class ដូចជា name, id និង sex ពី keyboard រួចហើយបង្ហាញតម្លៃរបស់វានៅលើអេក្រង់។ កម្មវិធីនេះពុំមានការប្រើលក្ខណៈ inheritance ឡើយ។

**// Program version 1**

**#include <iostream.h>**

**#include <iomanip.h>**

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**};** // ទីបញ្ចប់នៃ class Student

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃជាអក្សរមួយឃ្លា

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃជាចំនួនគត់

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃជាអក្សរមួយតួ

**}**

**void Student::display(){**

**cout << " " << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void main(){**

**Student ob;** // ការបង្កើត object មួយឈ្មោះ ob ពី Student class

**cout << "Enter the following information \n";**

**ob.getdata();** // ការហៅអនុគមន៍ getdata() មកប្រើ

**cout << "--------------------------------\n";**

**cout << " Name ID Sex \n";**

**cout << "--------------------------------\n";**

**ob.display();** // ការហៅអនុគមន៍ display() មកប្រើ

**cout << endl;**

**cout << "--------------------------------\n";**

**} // end of main program**

ឧទាហរណ៍ទី ២ ៖ កម្មវិធីខាងក្រោមនេះអានតម្លៃបញ្ចូលរបស់ derived class ដូចជា name, id, sex, height និង weight ពី keyboard រួចហើយបង្ហាញតម្លៃរបស់វានៅលើអេក្រង់។ កម្មវិធីនេះបានបង្ហាញនូវការប្រើលក្ខណៈ single inheritance ដែលក្នុងនោះមាន base class និង derived class។

**// Single inheritance**

**#include <iostream.h>**

**#include <iomanip.h>**

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**class PhysicalFit:public Student {**

**private :**

**float height;** // ការតាងអញ្ញាត height ជាប្រភេទទសភាគ float

**float weight;** // ការតាងអញ្ញាត weight ជាប្រភេទទសភាគ float

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃជាអក្សរមួយឃ្លា

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃជាចំនួនគត់

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃជាអក្សរមួយតួ

**}**

**void Student::display(){**

**cout << " " << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void PhysicalFit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() របស់ Student class មកប្រើ

**cout << "Height : ";** // បង្ហាញអក្សរ Height : មកលើអេក្រង់

**cin >> height;** // បញ្ចូលតម្លៃជាចំនួនទសភាគ

**cout << "Weight : ";** // បង្ហាញអក្សរ Weight : មកលើអេក្រង់

**cin >> weight;** // បញ្ចូលតម្លៃជាចំនួនទសភាគ

**}**

**void PhysicalFit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() របស់ Student class មកប្រើ

**cout << setprecision(2);**

**cout << height << " ";** // បង្ហាញតម្លៃ height មកលើអេក្រង់

**cout << weight << " ";** // បង្ហាញតម្លៃ weight មកលើអេក្រង់

**}**

**void main(){**

**PhysicalFit ob;** // ការបង្កើត object មួយឈ្មោះ ob ពី PhysicalFit class

**cout << "Enter the following information \n";**

**ob.getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ PhysicalFit​ object មកប្រើ

**cout << "--------------------------------------------\n";**

**cout << " Name ID no Sex Height Weight \n";**

**cout << "--------------------------------------------\n";**

**ob.display();** // ការហៅអនុគមន៍ display() តាមរយៈ PhysicalFit​ object មកប្រើ

**cout << endl;**

**cout << "----------------------------------------\n";**

**} // end of main program**

៣. ប្រភេទនៃ Base Class

class ណាក៏ដោយ វាអាចប្រើជា base class បានដែរ។ Derived class អាចប្រើជា base class សម្រាប់ class ផ្សេងទៀត។ Base class ត្រូវបានចំណាត់ថ្នាក់ជាពីរប្រភេទ គឺ base class ផ្ទាល់ និង base class មិនផ្ទាល់។

៣.១ អំពី Base Class ផ្ទាល់

Base class មួយជា base class ផ្ទាល់កាលណាវាមានលក្ខណៈដូចបានបង្ហាញក្នុងទម្រង់ខាងក្រោមនេះ។

ឧទាហរណ៍ **៖** ការប្រកាស derived class ខាងក្រោមនេះដែលមានលក្ខណៈត្រឹមត្រូវ។

ទម្រង់ទី ១ ៖

**class baseA {**

**.....**

**.....**

**};**

**class baseB : public baseA {**

**.....**

**.....**

**};**

ដែលក្នុងនេះ baseA ជា base class ផ្ទាល់។

ទម្រង់ទី ២ ៖

**class baseA {**

**.....**

**.....**

**};**

**class baseB {**

**.....**

**.....**

**};**

**class derivedC : public baseA, public baseB {**

**.....**

**.....**

**};**

ដែលក្នុងនេះ baseA និង baseB ជា base class ផ្ទាល់។

ទម្រង់ទី ៣ ៖

**class baseA {**

**.....**

**.....**

**};**

**class baseB {**

**.....**

**.....**

**};**

**class baseC {**

**.....**

**.....**

**};**

**class baseD {**

**.....**

**.....**

**};**

**class derivedE:public baseA, public baseB,**

**public baseC, public baseD {**

**.....**

**.....**

**};**

ការប្រកាសឃ្លាខាងក្រោមនេះពុំមានលក្ខណៈត្រឹមត្រូវឡើយ។

ចូរកត់សម្គាល់ថា class មួយដែលមានតែឈ្មោះ ប៉ុន្តែមិនទាន់បានប្រកាសនៅឡើយនោះ វាមិនអាចជា base class បានឡើយ។

ឧទាហរណ៍ ៖ ការប្រកាស class ខាងក្រោមនេះមិនមានលក្ខណៈត្រឹមត្រូវ។

ទម្រង់ទី ១ ៖ Base class មិនបានប្រកាស ហើយឃ្លា error នឹងបង្ហាញ ដោយ compiler។

**class baseA;**

**class derivedB:public baseA {**

**.....**

**..... //** មាន **error** កើតឡើង

**};**

ទម្រង់ទី ២ ៖ class មួយមិនអាចប្រកាសជា base class ផ្ទាល់នៃ derived class ដោយច្រើនជាងមួយឡើយ។ បានន័យថា base class ឈ្មោះ baseA មិនអាចប្រកាសពីរដងធ្វើជា base class របស់ derived class ឈ្មោះ derivedB ឡើយ។

**class baseA {**

**.....**

**.....**

**}**

**class derivedB:public baseA, public baseA {**

**.....**

**..... //** មាន **error** កើតឡើង

**};**

៣.២ អំពី Base Class មិនផ្ទាល់

Derived class អាចប្រើជា base class សម្រាប់ class មួយផ្សេងទៀត ហើយ class ដែលទើបបង្កើតថ្មីនេះបានទទួលលក្ខណៈពី base class របស់វារួមមាន data members និង member functions។ class មួយជា base class មិនផ្ទាល់ បើសិនវាមិនមែនជា base class ផ្ទាល់។

ឧទាហរណ៍ទី ១ ៖ ការប្រកាស class ខាងក្រោមនេះមានលក្ខណៈត្រឹមត្រូវ។ ហើយ derivedB ជា base class នៃ derivedC ដែលត្រូវបានហៅថា base class មិនផ្ទាល់។

**class baseA {**

**.....**

**.....**

**};**

**class derivedB:public baseA {**

**.....**

**.....**

**};**

**class derivedC:public derivedB{**

**.....**

**.....**

**};**

ឧទាហរណ៍ទី ២ ៖ class មួយអាចបញ្ជាក់ជា base class ច្រើនជាងមួយ។

**class baseA {**

**.....**

**.....**

**};**

**class baseB:public baseA {**

**.....**

**.....**

**};**

**class baseC:public baseA{**

**.....**

**.....**

**};**

**class derivedD:public baseB, public baseC {**

**.....**

**.....**

**};**

ឧទាហរណ៍ទី ៣ ៖ កម្មវិធីខាងក្រោមនេះទទួលព័ត៌មាននៃ data members របស់ derived class តាមរយៈ name, id, sex, height និង weight ពី keyboard។ រួចហើយអានតម្លៃជាថ្មីរបស់ data members របស់ derived class ផ្សេងទៀត ដូចជា name, id, sex, course, semester និង rank បន្ទាប់មកបង្ហាញតម្លៃរបស់ class ដែលទើបបង្កើតបាននៅលើអេក្រង់។ កម្មវិធីនេះចង់បង្ហាញពីទស្សនៈដែលមាន base class មួយ និង derived classes ពីរ។ data members និង member functions របស់ base class ត្រូវបានចូលប្រើដោយ derived classes ទាំងពីរនេះ។

**// two derived classes access the same base class**

**// Single inheritance**

**#include <iostream.h>**

**#include <iomanip.h>**

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**class PhysicalFit:public Student {**

**private :**

**float height, weight;** // ការប្រកាសអញ្ញាត height និង weight ជាប្រភេទ float

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**class AcademicFit:public Student {**

**private :**

**char course[20]; //** ប្រកាសអញ្ញាត course ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**char semester[10]; //** ប្រកាសអញ្ញាត semester ជា array ប្រភេទ char ប្រវែង 10 តួអក្សរ

**int rank; //** ប្រកាសអញ្ញាត rank ជាប្រភេទ int

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃ name តាមរយៈ keyboard

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃ id តាមរយៈ keyboard

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃ sex តាមរយៈ keyboard

**}**

**void Student::display(){**

**cout << " " << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void PhysicalFit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ Student​ class មកប្រើ

**cout << "Height : ";** // បង្ហាញអក្សរ Height : មកលើអេក្រង់

**cin >> height;** // បញ្ចូលតម្លៃ height តាមរយៈ keyboard

**cout << "Weight : ";** // បង្ហាញអក្សរ Weight : មកលើអេក្រង់

**cin >> weight;** // បញ្ចូលតម្លៃ weight តាមរយៈ keyboard

**}**

**void PhysicalFit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() តាមរយៈ Student​ class មកប្រើ

**cout << setprecision(2);** // ការកំណត់ទំហំបង្ហាញតម្លៃលេខ

**cout << height << " " << weight << " ";**

**}**

**void AcademicFit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ Student​ class មកប្រើ

**cout << "Course name (BTech/MCA/DCA etc) : ";**

**cin >> course;** // បញ្ចូលតម្លៃ course តាមរយៈ keyboard

**cout << "Semester (first/second etc) : ";**

**cin >> semester;** // បញ្ចូលតម្លៃ semester តាមរយៈ keyboard

**cout << "Rank of the student : ";**

**cin >> rank;** // បញ្ចូលតម្លៃ rank តាមរយៈ keyboard

**}**

**void AcademicFit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() តាមរយៈ Student​ class មកប្រើ

**cout << course<< " "<< semester<< " " << rank<< " ";**

**}**

**void main(){**

**PhysicalFit ob1;** // ការបង្កើត object មួយឈ្មោះ ob1 ពី PhysicalFit class

**AcademicFit ob2;** // ការបង្កើត object មួយឈ្មោះ ob2 ពី AcademicFit class

**cout << "Enter the following information ";**

**cout << " for physical fitness\n";**

**ob1.getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ ob1 នៃ​ PhysicalFit class មកប្រើ

**cout << "Enter the following information";**

**cout << " for academic fitness\n";**

**ob2.getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ ob2 នៃ​ AcademicFit class មកប្រើ

**cout << "Physical Fitness of the student\n";**

**cout << "------------------------------------------\n";**

**cout << " Name ID no Sex Height Weight \n";**

**cout << "------------------------------------------\n";**

**ob1.display();** // ការហៅអនុគមន៍ display() តាមរយៈ ob1 នៃ​ PhysicalFit class មកប្រើ

**cout << endl;**

**cout << "------------------------------------------\n";**

**cout << endl;**

**cout << " Academic performance of the student : \n";**

**cout << "------------------------------------------\n";**

**cout << " Name ID Sex Course Semester Rank \n";**

**cout << "------------------------------------------\n";**

**ob2.display();** // ការហៅអនុគមន៍ display() តាមរយៈ ob2 នៃ​ AcademicFit class មកប្រើ

**cout << endl;**

**cout << "-----------------------------------------\n";**

**} // end of main program**

៤. ប្រភេទនៃការទទួលលក្ខណៈរបស់ Derived class (Types of Derivation)

Inheritance គឺជាដំណើរការបង្កើត class ថ្មីមួយពី class ដែលមានរួចហើយ។ នៅពេលទទួលលក្ខណៈមកឲ្យ class ថ្មីនោះ​ពាក្យបញ្ជាក់លក្ខណៈនៃការត្រួតពិនិត្យពីការចូលប្រើ members ផ្តល់នូវការគ្រប់គ្រងទាំងអស់លើ data members និង methods របស់ base class។ Derived class អាចបង្កើតឡើងដោយមានការប្រើជាមួយពាក្យ private, public និង protected។

៤.១ អំពី Public Inheritance

ប្រភេទសំខាន់បំផុតនៃការចូលប្រើ members គឺ public។

* Member បញ្ជាក់ public នៅក្នុង base class គឺមានលក្ខណៈជា public នៅក្នុង derived class
* Member បញ្ជាក់ protected នៅក្នុង base class គឺមានលក្ខណៈជា protected នៅក្នុង derived class
* Member បញ្ជាក់ private នៅក្នុង base class គឺនៅតែរក្សាលក្ខណៈជា private នៅក្នុង base class ប៉ុណ្ណោះ។

ទម្រង់ទូទៅនៃការទទួលលក្ខណៈជា public គឺ ៖

*class base\_class\_name {*

*. . . . . . . .*

*. . . . . . . .*

*};*

*class derived\_class\_name : public base\_class\_name {*

*. . . . . . . .*

*. . . . . . . .*

*};*

ឧទាហរណ៍ ៖ code ខាងក្រោមនេះបង្ហាញពីរបៀបចូលទៅប្រើ members នៃ base class តាម រយៈ members របស់ derived class។

**class baseA {**

**private :**

**int x; //** ប្រកាសអញ្ញាត x ជាប្រភេទ int កម្រិត private

**protected :**

**int y; //** ប្រកាសអញ្ញាត y ជាប្រភេទ int កម្រិត protected

**public :**

**int z; //** ប្រកាសអញ្ញាត z ជាប្រភេទ int កម្រិត public

**};**

**class derivedD:public baseA { // public base class**

**private :**

**int w; //** ប្រកាសអញ្ញាត w ជាប្រភេទ int កម្រិត private

**};**

class derivedD បានទទួលលក្ខណៈពី base class ឈ្មោះ baseA ហើយការបញ្ជាក់ពី កម្រិតចូលប្រើជាប្រភេទ public។ data members នៃ derivedD គឺ ៖

**int x; // not accessible**

**int y;**

**int z;**

**int w;**

តារាងខាងក្រោមនេះបង្ហាញនូវការបញ្ជាក់កម្រិតចូលប្រើ data member នៃ base class នៅក្នុង derived class។

|  |  |  |
| --- | --- | --- |
| Member នៅក្នុង derivedD | សភាពលក្ខណៈនៃការចូលប្រើ | ប្រភពទទួលបាន |
| x  y  z  w | មិនអាចចូលប្រើបាន  *protected*  *public*  *private* | ពី class baseA  ពី class baseA  ពី class baseA  បន្ថែមលើ class derivedD |

៤.២ អំពី Private Inheritance

Base class មានលក្ខណៈជា private នោះ គឺ៖

- member ដែលបានបញ្ជាក់លក្ខណៈ public នៅក្នុង base class មានលក្ខណៈជា private នៅក្នុង derived class។

- member ដែលបានបញ្ជាក់លក្ខណៈ protected នៅក្នុង base class មានលក្ខណៈជា private នៅក្នុង derived class។

- member បញ្ជាក់លក្ខណៈ private នៅក្នុង base class នៅតែរក្សាលក្ខណៈជា private នៅក្នុង base class ដូចនេះ វាអាចប្រើបានតែនៅក្នុង base class ប៉ុណ្ណោះ។

ទម្រង់ទូទៅនៃ base class មានលក្ខណៈជា private នោះ គឺ ៖

class *base\_class\_name* {

. . . . . .

. . . . . .

};

class *derived\_class\_name* : private *base\_class\_name* {

. . . . . .

. . . . . .

}

ឧទាហរណ៍ ៖ code ខាងក្រោមនេះបង្ហាញពីរបៀបចូលទៅប្រើ member និមួយៗរបស់ base class តាមរយៈ members នៃ derived class។

**class baseA {**

**private :**

**int x; //** ប្រកាសអញ្ញាត x ជាប្រភេទ int កម្រិត private

**protected :**

**int y; //** ប្រកាសអញ្ញាត y ជាប្រភេទ int កម្រិត protected

**public :**

**int z; //** ប្រកាសអញ្ញាត z ជាប្រភេទ int កម្រិត public

**};**

**class derivedD:private baseA { // private base class**

**private :**

**int w; //** ប្រកាសអញ្ញាត w ជាប្រភេទ int កម្រិត private

**};**

class derivedD បានទទួលលក្ខណៈពី base class ឈ្មោះ baseA ហើយការបញ្ជាក់ពីកម្រិតចូលប្រើគឺ public។ data members នៃ derivedD គឺ ៖

**int x; // not accessible**

**int y;**

**int z;**

**int w;**

តារាងខាងក្រោមនេះបង្ហាញនូវការបញ្ជាក់កម្រិតចូលប្រើ data member នៃ base class នៅក្នុង derived class។

|  |  |  |
| --- | --- | --- |
| Member នៅក្នុង derivedD | សភាពលក្ខណៈនៃការចូលប្រើ | ប្រភពទទួលបាន |
| x  y  z  w | មិនអាចចូលប្រើបាន  *private*  *private*  *private* | ពី class baseA  ពី class baseA  ពី class baseA  បន្ថែមលើ class derivedD |

៤.៣ អំពី Protected Inheritance

Base class មានលក្ខណៈជា protected នោះ គឺ៖

- member ដែលបានបញ្ជាក់លក្ខណៈ public នៅក្នុង base class មានលក្ខណៈជា protected នៅក្នុង derived class។

- member ដែលបានបញ្ជាក់លក្ខណៈ protected នៅក្នុង base class មានលក្ខណៈជា protected នៅក្នុង derived class។

- member ដែលបានបញ្ជាក់លក្ខណៈ private នៅក្នុង base class នៅតែរក្សាលក្ខណៈជា private នៅក្នុង base class ដូចនេះ វាអាចប្រើបានតែនៅក្នុង base class ប៉ុណ្ណោះ។

ទម្រង់ទូទៅនៃ base class មានលក្ខណៈជា protected នោះ គឺ ៖

class *base\_class\_name* {

. . . . . .

. . . . . .

};

class *derived\_class\_name* : protected *base\_class\_name* {

. . . . . .

. . . . . .

}

ឧទាហរណ៍**៖** code ខាងក្រោមនេះបង្ហាញនូវរបៀបចូលប្រើ member និមួយៗរបស់ base class តាមរយៈ members នៃ derived class។

**class baseA {**

**private :**

**int x; //** ប្រកាសអញ្ញាត x ជាប្រភេទ int កម្រិត private

**protected :**

**int y; //** ប្រកាសអញ្ញាត y ជាប្រភេទ int កម្រិត protected

**public :**

**int z; //** ប្រកាសអញ្ញាត z ជាប្រភេទ int កម្រិត public

**};**

**class derivedD:protected baseA {// protected base class**

**private :**

**int w; //** ប្រកាសអញ្ញាត w ជាប្រភេទ int កម្រិត private

**};**

class derivedD បានទទួលលក្ខណៈពី base class ឈ្មោះ baseA ហើយការបញ្ជាក់ពីកម្រិតចូលប្រើគឺ protected។ data members នៃ derivedD គឺ ៖

**int x; // not accessible**

**int y;**

**int z;**

**int w;**

តារាងខាងក្រោមនេះបង្ហាញនូវការបញ្ជាក់កម្រិតចូលប្រើ data member នៃ base class នៅក្នុង derived class។

|  |  |  |
| --- | --- | --- |
| Member នៅក្នុង derivedD | សភាពលក្ខណៈនៃការចូលប្រើ | ប្រភពទទួលបាន |
| x  y  z  w | មិនអាចចូលប្រើបាន  *protected*  *protected*  *private* | ពី class baseA  ពី class baseA  ពី class baseA  បន្ថែមលើ class derivedD |

៥. ភាពមិនច្បាស់លាស់នៅក្នុង Single Inheritance (Ambiguity in Single Inheritance)

កាលណា data member និង member function បានកំណត់ដោយមានឈ្មោះដូចគ្នាទាំងនៅក្នុង base class និង derived class នោះឈ្មោះទាំងនេះត្រូវតែបញ្ជាក់ឲ្យច្បាស់។ សញ្ញា scope resolution operator (::) អាចប្រើសម្រាប់បញ្ជាក់ទៅឲ្យ member របស់ base class បានជាក់លាក់។ វិធីនេះអាចឲ្យគេចូលប្រើឈ្មោះមួយដែលបានកំណត់ជាថ្មីនៅក្នុង derived class។

ឧទាហរណ៍ទី ១ ៖ code ខាងក្រោមនេះបង្ហាញនូវភាពមិនច្បាស់លាស់កើតឡើងនៅពេល getdata() member function ត្រូវបានគេចូលប្រើតាមរយៈ main()។

**class baseA {**

**public :**

**int i; //** ប្រកាសអញ្ញាត i ជាប្រភេទ int កម្រិត public

**getdata(); //** ប្រកាសអនុគមន៍ getdata()

**};**

**class baseB{**

**public :**

**int i; //** ប្រកាសអញ្ញាត i ជាប្រភេទ int កម្រិត public

**getdata(); //** ប្រកាសអនុគមន៍ getdata()

**};**

**class derivedC : public baseA, public baseB {**

**public :**

**int i; //** ប្រកាសអញ្ញាត i ជាប្រភេទ int កម្រិត public

**getdata(); //** ប្រកាសអនុគមន៍ getdata()

**};**

**void main(){**

**derivedC ob; //** ការបង្កើត ob នៃ derivedC class

**ob.getdata();**// ការហៅអនុគមន៍ getdata() តាមរយៈ ob នៃ​ derivedC class មកប្រើ

**}**

members របស់វាពុំមានលក្ខណៈច្បាស់លាស់ទេ ព្រោះមិនបានប្រើ scope operators។ នៅពេល member function ឈ្មោះ getdata() ត្រូវបានចូលប្រើដោយ class object តាមធម្មតា compiler មិនអាចញែកភាពខុសគ្នារវាង member function នៃ class baseA និង baseB បានឡើយ។ ដូច្នេះវាត្រូវតែមានការប្រកាសប្រើជាមួយនឹង scope operator ដោយបញ្ជាក់ឲ្យបានច្បាស់ដើម្បីហៅ member របស់ base class ដូចបង្ហាញក្នុងឃ្លាខាងក្រោម ៖

**ob.baseA::getdata();**

**ob.baseB::getdata();**

ឧទាហរណ៍ទី ២ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវភាពមិនច្បាស់លាស់ដែលត្រូវជៀសវាងនៅក្នុង single inheritance ដោយប្រើ scope resolution operator។

**// Ambiguity in the single inheritance**

**#include <iostream.h>**

**class baseA {**

**private :**

**int i; //** ប្រកាសអញ្ញាត i ជាប្រភេទ int កម្រិត private

**public :**

**void getdata(int x); //** ប្រកាសអនុគមន៍ getdata() កម្រិត public

**void display(); //** ប្រកាសអនុគមន៍ display() កម្រិត public

**}; // end of class definition**

**class baseB {**

**private :**

**int j; //** ប្រកាសអញ្ញាត j ជាប្រភេទ int កម្រិត public

**public :**

**void getdata(int y); //** ប្រកាសអនុគមន៍ getdata() កម្រិត public

**void display(); //** ប្រកាសអនុគមន៍ display() កម្រិត public

**}; // end of class definition**

**class derivedC:public baseA, public baseB {**

**};**

**void baseA::getdata(int x) {**

**i = x;**

**}**

**void baseA::display(){**

**cout << "Value of i = " << i << endl;**

**}**

**void baseB::getdata(int y){**

**j = y;**

**}**

**void baseB::display(){**

**cout << "Value of j = " << j << endl;**

**}**

**void main(){**

**derivedC ob; //** ការបង្កើត ob នៃ derivedC class

**int x, y; //** ប្រកាសអញ្ញាត x, y ជាប្រភេទ int

**cout << "Enter a value for i : ";**

**cin >> x;** // បញ្ចូលតម្លៃ x តាមរយៈ keyboard

**ob.baseA::getdata(x);**

**// member is ambiguous without scope**

**cout << "Enter a value for j : ";**

**cin >> y;** // បញ្ចូលតម្លៃ y តាមរយៈ keyboard

**ob.baseB::getdata(y);** // ការហៅអនុគមន៍ getdata() តាមរយៈ ob នៃ baseB​ class មកប្រើ

**ob.baseA::display();** // ការហៅអនុគមន៍ display() តាមរយៈ ob នៃ baseA​ class មកប្រើ

**ob.baseB::display();** // ការហៅអនុគមន៍ display() តាមរយៈ ob នៃ baseB class មកប្រើ

**} // end of main program**

៦. ការប្រើ arrays នៃ object របស់ class និង single inheritance

នៅក្នុងផ្នែកនេះ យើងសង្កត់ធ្ងន់ទៅលើរបៀបប្រើ array នៃ objects របស់ class ដែលអាចទទួលលក្ខណៈពី base class មួយ។ កាលណា derived class បានបង្កើតឡើងនោះ ការចូលប្រើ member នៃ class មួយរបស់ array នៃ class objects មានលក្ខណៈដូចគ្នាទៅនឹង class ធម្មតាដែរ។

ទម្រង់ទូទៅនៃ array របស់ class objects នៅក្នុង derived class តែមួយ នោះគឺ ៖

class *baseA* {

*private* :

. . . . . .

. . . . . .

*public* :

. . . . . .

. . . . . .

};

class *derivedB* : public *baseA* {

*private* :

. . . . . .

. . . . . .

*public* :

. . . . . .

. . . . . .

};

void main(){

*derivedB* obj[100]; *// array of class objects of the derived class*

. . . . . .

. . . . . .

}

ឧទាហរណ៍ ៖ កម្មវិធីខាងក្រោមនេះមាន base class មួយ និង derived class មួយ។ data member របស់ base class រួមមាន ៖ name, id និង sex។ data member របស់ derived class គឺ height និង weight។ derived class ត្រូវបានប្រកាសដូចទៅនឹង array នៃ class objects។ Member functions ត្រូវបានប្រើសម្រាប់ទទួលយកព័ត៌មាននៅលើ derived class តាមរយៈ keyboard រួចហើយបង្ហាញតម្លៃនៃ array របស់ class objects នៅលើអេក្រង់។

កម្មវិធីនេះបង្ហាញពីរបៀបកំណត់ array នៃ class objects ដោយប្រើលក្ខណៈ single inheritance។

**// single inheritance using array of objects**

**#include <iostream.h>**

**#include <iomanip.h>**

**const int max = 100; //** ប្រកាសអញ្ញាត max ជាប្រភេទ int មានតម្លៃថេរ

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**class PhysicalFit:public Student {**

**private :**

**float height; //** ប្រកាសអញ្ញាត height ជាប្រភេទ float

**float weight; //** ប្រកាសអញ្ញាត weight ជាប្រភេទ float

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃ name តាមរយៈ keyboard

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃ id តាមរយៈ keyboard

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃ sex តាមរយៈ keyboard

**}**

**void Student::display(){**

**cout << " " << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void PhysicalFit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ Student​ class មកប្រើ

**cout << "Height : ";** // បង្ហាញអក្សរ Height : មកលើអេក្រង់

**cin >> height;** // បញ្ចូលតម្លៃ height តាមរយៈ keyboard

**cout << "Weight : ";** // បង្ហាញអក្សរ Weight : មកលើអេក្រង់

**cin >> weight;** // បញ្ចូលតម្លៃ weight តាមរយៈ keyboard

**}**

**void PhysicalFit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() តាមរយៈ Student​ class មកប្រើ

**cout << setprecision(2);**

**cout << height << " ";** // បង្ហាញតម្លៃ height មកលើអេក្រង់

**cout << weight << " ";** // បង្ហាញតម្លៃ weight មកលើអេក្រង់

**}**

**void main(){**

**PhysicalFit a[max]; //** ការបង្កើត object នៃ PhysicalFit class ចំនួន max

**int i, n; //** ប្រកាសអញ្ញាត i, n ជាប្រភេទ int

**cout << "How many students ? \n";**

**cin >> n;** // បញ្ចូលតម្លៃ n តាមរយៈ keyboard

**cout << "Enter the following information \n";**

**for (i = 0; i < n; i++){**

**cout << "Record : " << (i+1) << endl;**

**a[i].getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ PhysicalFit​ class មកប្រើ

**}**

**cout << endl;**

**cout << "------------------------------------------\n";**

**cout << " Name ID no Sex Height Weight \n";**

**cout << "------------------------------------------\n";**

**for (i = 0; i < n; i++){**

**a[i].display();** // ការហៅអនុគមន៍ display() តាមរយៈ object នៃ PhysicalFit​ class មកប្រើ

**cout << endl;**

**}**

**cout << endl;**

**cout << "------------------------------------------\n";**

**} // end of main program**

៧. អំពី Multiple Inheritance

នៅដើមជំពូកនេះ យើងបានសិក្សារួចហើយអំពី derived class មួយអាចទទួលលក្ខណៈបន្ថែមពី base class។ ប្រភពដើមនៃភាសា C++ គឺ derived class មួយអាចទទួលលក្ខណៈពី base class តែមួយគត់។ ទោះជាយ៉ាងណាក៏ដោយលំនាំ OOP គឺជាមធ្យោបាយសរសេរកម្មវិធីដែលអានុភាពខ្លាំង និងអាចសម្របតាមកាលៈទេសៈបាន។ version ចុងក្រោយរបស់ C++ compiler បានអនុវត្តនូវការទទួលលក្ខណៈពីប្រភពជាច្រើនបាន (multiple inheritance)។ នៅក្នុងផ្នែកនេះ យើងនឹងសិក្សានូវរបៀប class មួយទទួលលក្ខណៈពី base class លើសពីមួយ។

Multiple inheritance គឺជាដំណើរនៃការបង្កើត class ថ្មីមួយពី base class ចំនួនច្រើន។ ទម្រង់ multiple inheritance ស្រដៀងទៅនឹង single inheritance ដែរ។

**ឧទាហរណ៍ ៖**

class *baseA* {

. . . . . .

. . . . . .

};

class *baseB* {

. . . . . .

. . . . . .

};

class *derivedC* : public *baseA*, public *baseB* {

. . . . . .

. . . . . .

};

class derivedC បានទទួលលក្ខណៈពី baseA និង baseB។

Multiple inheritance គឺជា derived class ដែលបានប្រកាសឡើងដើម្បីទទួលលក្ខណៈរបស់ base class ពីរឬច្រើន។ Multiple inheritance អាចបន្សំលក្ខណៈរបស់ base class ជាច្រើនទៅឲ្យ derived class តែមួយ។ គោលការណ៍នៃ inheritance និងការចូលប្រើអ្វីៗពុំមានការផ្លាស់ប្តូរឡើយពីលំដាប់ថ្នាក់ single inheritance ទៅ multiple inheritance។ Derived class មួយទទួលលក្ខណៈ data members និង methods ពី base class ទាំងអស់របស់វា មិនថាជា private, protected ឬ public ឡើយ។

ករណីទី ១**៖** Multiple inheritance ជាមួយនឹងការទទួលលក្ខណៈប្រភេទ public ទាំងអស់។

class *baseA* { // base class 1

. . . . . .

. . . . . .

};

class *baseB* { // base class 2

. . . . . .

. . . . . .

};

class *baseC* { // base class 3

. . . . . .

. . . . . .

};

class *derivedD* : public *baseA*, public *baseB*, public *baseC* {

. . . . . .

. . . . . .

};

ករណីទី ២ ៖ Multiple inheritance ជាមួយនឹងការទទួលលក្ខណៈប្រភេទ private ទាំងអស់។

class *baseA* { // base class 1

. . . . . .

. . . . . .

};

class *baseB* { // base class 2

. . . . . .

. . . . . .

};

class *baseC* { // base class 3

. . . . . .

. . . . . .

};

class *derivedD* : private *baseA*, private *baseB*, private *baseC* {

. . . . . .

. . . . . .

};

ករណីទី ៣**៖** Multiple inheritance ជាមួយនឹងការទទួលលក្ខណៈប្រភេទផ្សេងៗគ្នា។

class *baseA* { // base class 1

. . . . . .

. . . . . .

};

class *baseB* { // base class 2

. . . . . .

. . . . . .

};

class *baseC* { // base class 3

. . . . . .

. . . . . .

};

class *derivedD* : private *baseA*, public *baseB*, private *baseC* {

. . . . . .

. . . . . .

};

ឧទាហរណ៍ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីរបៀប multiple inheritance បានប្រកាសឡើង ហើយបានកំណត់នៅក្នុងកម្មវិធីមួយ។ កម្មវិធីនេះរួមមាន base classes ចំនួនពីរនិង derived class មួយ។ base class ឈ្មោះ Student មាន data members ដូចជា ៖ name, id និង sex។ រីឯ base class ឈ្មោះ AcademicFit មាន data members ដូចជា ៖ course, semester និង rank។ derived class ឈ្មោះ FinancialAssit មាន data member មួយគឺ amount ក្រៅពី data members របស់ base class។ Member functions ត្រូវបានប្រើសម្រាប់ទទួលព័ត៌មានរបស់ derived class ពី keyboard ហើយបង្ហាញតម្លៃនៃ class objects នៅលើអេក្រង់។

**// Multiple inheritance**

**#include <iostream.h>**

**#include <iomanip.h>**

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata()

**void display();** // ការប្រកាសអនុគមន៍ display()

**}; // end of class definition**

**class AcademicFit {**

**private :**

**char course[20]; //** ប្រកាសអញ្ញាត course ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**char semester[10];//** ប្រកាសអញ្ញាត semester ជា array ប្រភេទ char ប្រវែង 10 តួអក្សរ

**int rank; //** ប្រកាសអញ្ញាត rank ជាប្រភេទ int

**public :**

**void getdata(); //** ប្រកាសអនុគមន៍ getdata() កម្រិត public

**void display(); //** ប្រកាសអនុគមន៍ display() កម្រិត public

**}; // end of class definition**

**class FinancialAssit:private Student,**

**private AcademicFit {**

**private :**

**float amount; //** ប្រកាសអញ្ញាត amount ជាប្រភេទ float

**public :**

**void getdata(); //** ប្រកាសអនុគមន៍ getdata() កម្រិត public

**void display(); //** ប្រកាសអនុគមន៍ display() កម្រិត public

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃ name តាមរយៈ keyboard

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃ id តាមរយៈ keyboard

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃ sex តាមរយៈ keyboard

**}**

**void Student::display(){**

**cout << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void AcademicFit::getdata(){**

**cout << "Course name (BTech/MCA/DCA etc) : ";**

**cin >> course;** // បញ្ចូលតម្លៃ course តាមរយៈ keyboard

**cout << "Semester (first/second etc) : ";**

**cin >> semester;** // បញ្ចូលតម្លៃ semester តាមរយៈ keyboard

**cout << "Rank of the student : ";**

**cin >> rank;** // បញ្ចូលតម្លៃ rank តាមរយៈ keyboard

**}**

**void AcademicFit::display(){**

**cout << course<< " " << semester<< " " << rank << " ";**

**}**

**void FinancialAssit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ Student​ class មកប្រើ

**AcademicFit::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ AcademicFit​ class មកប្រើ

**cout << "Amount in $ : ";** // បង្ហាញអក្សរ Amount in $ : មកលើអេក្រង់

**cin >> amount;** // បញ្ចូលតម្លៃ amount តាមរយៈ keyboard

**}**

**void FinancialAssit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() តាមរយៈ Student​ class មកប្រើ

**AcademicFit::display();**// ការហៅអនុគមន៍ display() តាមរយៈ AcademicFit​ class មកប្រើ

**cout << setprecision(2);**

**cout << amount << " ";** // បង្ហាញតម្លៃ amount មកលើអេក្រង់

**}**

**void main(){**

**FinancialAssit f;** // ការបង្កើត object នៃ FinancialAssist class ឈ្មោះ f

**cout << "Enter the following information ";**

**cout << "for financial assistance\n";**

**f.getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ FinancialAssist​ class មកប្រើ

**cout << endl;**

**cout << "** **Academic Performance for Financial Assistance \n";**

**cout << "------------------------------------------\n";**

**cout << "Name ID Sex Course Semester Rank Amount\n";**

**cout << "------------------------------------------\n";**

**f.display();** // ការហៅអនុគមន៍ display() តាមរយៈ object នៃ FinancialAssist​ class មកប្រើ

**cout << endl;**

**cout << "------------------------------------------\n";**

**} // end of main program**

៧.១ អំពី array នៃ class objects និង Multiple inheritance

នៅក្នុងផ្នែកនេះ យើងនឹងលើកយក array នៃ class objects ដែលអាចទទួលលក្ខណៈពី base classes ជាច្រើន។ ជួនកាល derived class ដែលបានបង្កើតឡើងមានការចូលប្រើ class members នៃ array របស់ class objects គឺដូចគ្នាទៅនឹងប្រភេទ class ធម្មតាដែរ។

ទម្រង់ទូទៅនៃ array របស់ class objects នៅក្នុង multiple inheritance មានដូចខាងក្រោម ៖

class *baseA* {

. . . . . .

. . . . . .

};

class *baseB* {

. . . . . .

. . . . . .

};

class *derivedC* : public *baseA*, public *baseB* {

. . . . . .

. . . . . .

};

void main(){

derivedC ob[100]; // array of class objects of the derived class

. . . . . .

. . . . . .

}

ឧទាហរណ៍ ៖ កម្មវិធីខាងក្រោមនេះរួមមាន base classes ពីរ និង derived class មួយ។ base class ឈ្មោះ Student មាន data members ដូចជា ៖ name, id និង sex។ រីឯ base class ឈ្មោះ AcademicFit មាន data members ដូចជា ៖ course, semester និង rank។ derived class ឈ្មោះ FinancialAssit មាន data member មួយគឺ amount ក្រៅពី data members របស់ base class។ Member functions ត្រូវបានប្រើសម្រាប់ទទួលព័ត៌មានរបស់ derived class ពី keyboard ហើយបង្ហាញតម្លៃនៃ array របស់ class objects នៅលើអេក្រង់។

**// Multiple inheritance using array of objects**

**#include <iostream.h>**

**#include <iomanip.h>**

**const int max = 10; //** ប្រកាសអញ្ញាត max ជាចំនួនថេរប្រភេទ int

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata() មានលក្ខណៈ public

**void display();** // ការប្រកាសអនុគមន៍ display() មានលក្ខណៈ public

**}; // end of class definition**

**class AcademicFit {**

**private :**

**char course[20]; //** ប្រកាសអញ្ញាត course ជា array ប្រភេទ char ប្រវែង 20 តួអក្សរ

**char semester[10]; //** ប្រកាសអញ្ញាត semester ជា array ប្រភេទ char ប្រវែង 10 តួអក្សរ

**int rank; //** ប្រកាសអញ្ញាត rank ជាប្រភេទ int

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata() មានលក្ខណៈ public

**void display();** // ការប្រកាសអនុគមន៍ display() មានលក្ខណៈ public

**}; // end of class definition**

**class FinancialAssit:private Student, private AcademicFit {**

**private :**

**float amount; //** ប្រកាសអញ្ញាត amount ជាប្រភេទ float

**public :**

**void getdata();** // ការប្រកាសអនុគមន៍ getdata() មានលក្ខណៈ public

**void display();** // ការប្រកាសអនុគមន៍ display() មានលក្ខណៈ public

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃ name តាមរយៈ keyboard

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃ id តាមរយៈ keyboard

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃ sex តាមរយៈ keyboard

**}**

**void Student::display(){**

**cout << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void AcademicFit::getdata(){**

**cout << "Course name (BTech/MCA/DCA etc) : ";**

**cin >> course;** // បញ្ចូលតម្លៃ course តាមរយៈ keyboard

**cout << "Semester (first/second etc) : ";**

**cin >> semester;** // បញ្ចូលតម្លៃ semester តាមរយៈ keyboard

**cout << "Rank of the student: ";**

// បង្ហាញអក្សរ Rank of the student : មកលើអេក្រង់

**cin >> rank;** // បញ្ចូលតម្លៃ rank តាមរយៈ keyboard

**}**

**void AcademicFit::display(){**

**cout << course << " ";** // បង្ហាញតម្លៃ course មកលើអេក្រង់

**cout << semester << " ";** // បង្ហាញតម្លៃ semester មកលើអេក្រង់

**cout << rank << " ";** // បង្ហាញតម្លៃ rank មកលើអេក្រង់

**}**

**void FinancialAssit::getdata(){**

**Student::getdata();** // ការហៅអនុគមន៍ getdata() តាមរយៈ Student​ class មកប្រើ

**AcademicFit::getdata();**// ការហៅអនុគមន៍ getdata() តាមរយៈ AcademicFit​ class មកប្រើ

**cout << "Amount in $ : ";** // បង្ហាញអក្សរ Amount in $ : មកលើអេក្រង់

**cin >> amount;** // បញ្ចូលតម្លៃ amount តាមរយៈ keyboard

**}**

**void FinancialAssit::display(){**

**Student::display();** // ការហៅអនុគមន៍ display() តាមរយៈ Student​ class មកប្រើ

**AcademicFit::display();**// ការហៅអនុគមន៍ display() តាមរយៈ AcademicFit​ class មកប្រើ

**cout << setprecision(2);**

**cout << amount << " ";** // បង្ហាញតម្លៃ amount មកលើអេក្រង់

**}**

**void main(){**

**FinancialAssit f[max];** // ការបង្កើត object នៃ FinancialAssist class ឈ្មោះ f ចំនួន max

**int n; //** ប្រកាសអញ្ញាត n ជាប្រភេទ int

**cout << "How many students ? \n";**

**cin >> n;** // បញ្ចូលតម្លៃ n តាមរយៈ keyboard

**cout << "Enter the following information ";**

**cout << "for financial assistance\n";**

**for (int i = 0; i < n; i++){**

**cout << "ID : " << (i+1) << endl;**

**f[i].getdata();**// ការហៅ getdata() តាមរយៈ object នៃ FinancialAssist ​ class មកប្រើ

**cout << endl;**

**}**

**cout << endl;**

**cout << "Academic Performance for Financial Assistance \n";**

**cout << "----------------------------------------------\n";**

**cout << "Name ID Sex Course Semester Rank Amount \n";**

**cout << "----------------------------------------------\n";**

**for (i = 0; i < n; i++){**

**f[i].display();** // ការហៅ display() តាមរយៈ object នៃ FinancialAssist ​ class មកប្រើ

**cout << endl;**

**}**

**cout << "------------------------------------------\n";**

**} // end of main program**

៧.២ ភាពមិនច្បាស់លាស់នៅក្នុង multiple inheritance

ដើម្បីជៀសវាងនូវភាពមិនច្បាស់លាស់រវាង derived class ហើយនិង base class មួយឬ រវាង base class និង base class នោះ គេគួរតែប្រើ scope resolution operator (::) ជាមួយនឹង data members និង methods។

ឧទាហរណ៍ ៖ code ខាងក្រោមនេះបង្ហាញនូវភាពមិនច្បាស់លាស់លេចឡើងនៅក្នុង base class និង derived class។

**// ambiguity in multiple inheritance**

**#include <iostream.h>**

**class baseA {**

**public :**

**​ int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseA class

**};**

**class baseB {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseB class

**};**

**class baseC {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseC class

**};**

**class derivedD:public baseA, public baseB, public baseC {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង derivedD class

**};**

**void main(){**

**derivedD ob; //** ការបង្កើត object នៃ derivedD class

**ob.a = 10; // local to the derived class**

**}**

ឧបមាថា គេចង់ចូលប្រើ data members របស់ base classes នោះ វានឹងមានការកើតឡើង ដោយសារប៉ះទង្គិចគ្នារវាង base class និង base class ហើយ compiler មិនអាចញែកសម្គាល់បានរវាងការហៅ data members នោះមកប្រើ។ ដើម្បីឲ្យវាញែកសម្គាល់បាននោះ គេត្រូវប្រើ scope operator។

**void main(){**

**derivedD ob; //** ការបង្កើត object នៃ derivedD class

**ob.a = 10; //** ការកំណត់តម្លៃ 10 ទៅឲ្យអញ្ញាត a នៃ object របស់ derivedD class

**ob.baseA::a = 20; // accessing the base class A member**

**ob.baseB::a = 30; // accessing the base class B member**

**ob.baseC::a = 40; // accessing the base class C member**

**}**

ឧទាហរណ៍ ៖កម្មវិធីខាងក្រោមនេះបង្ហាញនូវភាពមិនច្បាស់លាស់ដែលត្រូវជៀសវាងនៅក្នុង multiple inheritance ដោយប្រើ scope resolution operator។

**// Ambiguity in multiple inheritance**

**// without scope operator**

**#include <iostream.h>**

**class baseA {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseA class

**};**

**class baseB {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseB class

**};**

**class baseC {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង baseC class

**};**

**class derivedD:public baseA, public baseB, public baseC {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int នៅក្នុង derivedD class

**};**

**void main(){**

**derivedD ob; //** ការបង្កើត object នៃ derivedD class

**ob.a = 10; //** ការកំណត់តម្លៃ 10 ទៅឲ្យអញ្ញាត a នៃ object របស់ derivedD class

**ob.baseA::a = 20; //** ការកំណត់តម្លៃ 20 ទៅឲ្យអញ្ញាត a នៃ object របស់ baseA class

**ob.baseB::a = 30; //** ការកំណត់តម្លៃ 30 ទៅឲ្យអញ្ញាត a នៃ object របស់ baseB class

**ob.baseC::a = 40; //** ការកំណត់តម្លៃ 40 ទៅឲ្យអញ្ញាត a នៃ object របស់ baseC class

**cout << "Value of a in the derived class = " << ob.a**

**<< endl;**

**cout << "Value of a in the baseA = " << ob.baseA::a << endl;**

**cout << "Value of a in the baseB = " << ob.baseB::a << endl;**

**cout << "Value of a in the baseC = " << ob.baseC::a << endl;**

**} // end of main program**

៨. អំពី Container class

នៅជំពូកមុននេះ យើងបានយល់អំពីការប្រកាស class មួយនៅក្នុង class មួយទៀត។ នៅក្នុងផ្នែកនេះ យើងនឹងលើកយកពីរបៀប container class អាចប្រកាសនិង កំណត់នៅក្នុងកម្មវិធី។ C++ អាចឲ្យគេប្រកាស object នៃ class មួយជា member នៃ class មួយទៀតបាន។ នៅពេល object នៃ class មួយបានប្រកាសជា member នៃ class មួយទៀតនោះ វាត្រូវបានគេហៅថា container class។ ឧទាហរណ៍ខ្លះៗចំពោះ container class មានដូចជា arrays, linked lists, stacks និង queues។

ទម្រង់ទូទៅ ចំពោះការប្រកាស container class គឺ ៖

class *user\_defined\_name\_1* {

… … … … … …

… … … … … …

};

class *user\_defined\_name\_2* {

… … … … … …

… … … … … …

};

class *user\_defined\_name\_n* {

… … … … … …

… … … … … …

};

class *derived\_class* {

*user\_defined\_name\_1* ob1; // object of the class one

*user\_defined\_name\_2* ob2; // object of the class two

… … … … … …

… … … … … …

*user\_defined\_name\_n* obn; // object of the class *n*

};

ឧទាហរណ៍ ៖ code ខាងក្រោមនេះបង្ហាញនូវរបៀបប្រកាស container class។

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ចំនួន 20 តួអក្សរ

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង Student class

**}; // end of class definition**

**class AcademicFit {**

**private :**

**int rank; //** ប្រកាសអញ្ញាត rank ជាប្រភេទ int

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង AcademicFit class

**}; // end of class definition**

**class FinancialAssit {**

**private :**

**Student bdata; //** ការបង្កើត object នៃ Student class ឈ្មោះ bdata

**AcademicFit acd; //** ការបង្កើត object នៃ AcademicFit class ឈ្មោះ acd

**float amount; //** ប្រកាសអញ្ញាត amount ជាប្រភេទ float

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង FinancialAssit class

**void display(); //** ការប្រកាសអនុគមន៍ display() ក្នុង FinancialAssit class

**}; // end of class definition**

**void main(){**

**FinancialAssit ob; //** ការបង្កើត object នៃ FinancialAssit class

**.........**

**.........**

**}**

ឧទាហរណ៍ **៖**កម្មវិធីខាងក្រោមនេះបង្ហាញនូវ container class មួយដែលបានប្រកាស និងកំណត់នៅក្នុងកម្មវិធី។ កម្មវិធីរួមមាន base classes ពីរ និង derived class មួយ។ base class ឈ្មោះ Student មាន data members ដូចជា name, id និង sex។ ហើយ base class មួយទៀតឈ្មោះ AcademicFit ដែលមាន data members ដូចជា course, semester និង rank។ derived class ឈ្មោះ FinancialAssit មាន data member មួយ គឺ amount ក្រៅពី data members របស់ base classes។ objects នៃ base classes ទាំងពីរនេះត្រូវបានកំណត់ជា members នៃ derived class។ Member functions ត្រូវបានប្រើសម្រាប់ទទួលព័ត៌មានរបស់ derived class ពី keyboard ហើយបង្ហាញតម្លៃនៃ class objects នៅលើអេក្រង់។

**// Demonstration of container class**

**#include <iostream.h>**

**#include <iomanip.h>**

**class Student {**

**private :**

**char name[20]; //** ប្រកាសអញ្ញាត name ជា array ប្រភេទ char ចំនួន 20 តួអក្សរ

**long int id; //** ប្រកាសអញ្ញាត id ជាប្រភេទ long int

**char sex; //** ប្រកាសអញ្ញាត sex ជាប្រភេទ char

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង Student class

**void display(); //** ការប្រកាសអនុគមន៍ display() ក្នុង Student class

**}; // end of class definition**

**class AcademicFit {**

**private :**

**char course[20]; //** ប្រកាសអញ្ញាត course ជា array ប្រភេទ char ចំនួន 20 តួអក្សរ

**char semester[10]; //** ប្រកាសអញ្ញាត semester ជា array ប្រភេទ char ចំនួន 10 តួអក្សរ

**int rank; //** ប្រកាសអញ្ញាត rank ជាប្រភេទ int

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង AcademicFit class

**void display(); //** ការប្រកាសអនុគមន៍ display() ក្នុង AcademicFit class

**}; // end of class definition**

**class FinancialAssit {**

**private :**

**Student bdata; //** ការបង្កើត object នៃ Student class ឈ្មោះ bdata

**AcademicFit acd; //** ការបង្កើត object នៃ AcademicFit class ឈ្មោះ acd

**float amount; //** ប្រកាសអញ្ញាត amount ជាប្រភេទ float

**public :**

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata() ក្នុង FinancialAssit class

**void display(); //** ការប្រកាសអនុគមន៍ display() ក្នុង FinancialAssit class

**}; // end of class definition**

**void Student::getdata() {**

**cout << "Enter a name : ";** // បង្ហាញអក្សរ Enter a name : មកលើអេក្រង់

**cin >> name;** // បញ្ចូលតម្លៃ name តាមរយៈ keyboard

**cout << "ID : ";** // បង្ហាញអក្សរ ID : មកលើអេក្រង់

**cin >> id;** // បញ្ចូលតម្លៃ id តាមរយៈ keyboard

**cout << "Sex : ";** // បង្ហាញអក្សរ Sex : មកលើអេក្រង់

**cin >> sex;** // បញ្ចូលតម្លៃ sex តាមរយៈ keyboard

**}**

**void Student::display(){**

**cout << name << " ";** // បង្ហាញតម្លៃ name មកលើអេក្រង់

**cout << id << " ";** // បង្ហាញតម្លៃ id មកលើអេក្រង់

**cout << sex << " ";** // បង្ហាញតម្លៃ sex មកលើអេក្រង់

**}**

**void AcademicFit::getdata(){**

**cout << "Course name (BTech/MCA/DCA etc) : ";**

**cin >> course;** // បញ្ចូលតម្លៃ course តាមរយៈ keyboard

**cout << "Semester (first/second etc) : ";**

**cin >> semester;** // បញ្ចូលតម្លៃ semester តាមរយៈ keyboard

**cout <<"Rank of the student :";**// បង្ហាញអក្សរ Rank of the student : មកលើអេក្រង់

**cin >> rank;** // បញ្ចូលតម្លៃ rank តាមរយៈ keyboard

**}**

**void AcademicFit::display(){**

**cout << course << " ";** // បង្ហាញតម្លៃ course មកលើអេក្រង់

**cout << semester << " ";** // បង្ហាញតម្លៃ semester មកលើអេក្រង់

**cout << rank << " ";** // បង្ហាញតម្លៃ rank មកលើអេក្រង់

**}**

**void FinancialAssit::getdata(){**

**bdata.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ Student class

**acd.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ AcademicFit class

**cout << "Amount in $ : ";** // បង្ហាញអក្សរ Amount in $ : មកលើអេក្រង់

**cin >> amount;** // បញ្ចូលតម្លៃ amount តាមរយៈ keyboard

**}**

**void FinancialAssit::display(){**

**bdata.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ Student class

**acd.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ AcademicFit class

**cout << setprecision(5);**

**cout << amount << " ";** // បង្ហាញតម្លៃ amount មកលើអេក្រង់

**}**

**void main(){**

**FinancialAssit f; //** ការបង្កើត object នៃ FinancialAssit class

**cout << "Enter the following information : \n";**

**f.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ FinancialAssit class

**cout << endl;**

**cout << " Academic Performance for Financial Assistance\n";**

**cout << "----------------------------------------------\n";**

**cout << " Name ID no Sex Course Semester Rank Amount \n";**

**cout << "----------------------------------------------\n";**

**f.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ FinancialAssit class

**cout << endl;**

**cout << "---------------------------------------------\n";**

**} // end of main program**

៩. ការចូលទៅប្រើ Member របស់ class

នៅក្នុង C++, class បានបង្កើតនូវមជ្ឈដ្ឋានមួយដែលអ្នកសរសេរកម្មវិធីមិនគ្រាន់តែអាចបង្កើត objects ដែលមាន methods របស់វានៅក្នុងទម្រង់នៃ member functions ប៉ុណ្ណោះទេ ថែមទាំងអាចធ្វើការគ្រប់គ្រងស្ទើរតែទាំងអស់ទៅលើការចូលប្រើ members របស់ class។ នៅក្នុងផ្នែកនេះ យើងនឹងលើកយកមកសិក្សានូវការចូលប្រើ members របស់ class ដែលនៅដោយឡែកៗពីគ្នា និងនៅក្នុង derived class ផងដែរ។

៩.១ ការចូលទៅប្រើ data member ប្រភេទ public

Members នៃ class ប្រភេទ public អាចប្រើបានដូចខាងក្រោម ៖

 Member functions នៃ class

 Non-member functions នៃ class

 Member function នៃ friend class

 Member function នៃ derived class ប្រសិនបើវាបានទទួលលក្ខណៈជា public។

ឧទាហរណ៍ **៖** code ខាងក្រោមនេះបង្ហាញពីការប្រកាសបានត្រឹមត្រូវចំពោះការចូលប្រើ data member នៃ class។

ករណីទី ១ ៖ member function នៃ class អាចចូលប្រើ data member ជា public។

**class sample {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata()

**void display(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ករណីទី ២ ៖ Non-member function នៃ class អាចចូលប្រើ data member ជា public។

**class sample {**

**public :**

**int a; //** ប្រកាសអញ្ញាត a ជាប្រភេទ int

**};**

**void main(){**

**sample ob; //** ការបង្កើត object នៃ sample class

**ob.a++; //** មានលក្ខណៈត្រឹមត្រូវ

**};**

ករណីទី ៣ ៖ member function នៃ derived class អាចចូលប្រើ data member របស់ base class ប្រភេទ public ប្រសិនបើវាបានទទួលលក្ខណៈជា public។

**class base {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata()

**};**

**class derived:public base {**

**public :**

**void display(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

code ខាងក្រោមនេះបង្ហាញពីការប្រកាស ចំពោះការចូលប្រើ data member ជា public ដែលមិនមានលក្ខណៈត្រឹមត្រូវ ព្រោះវាបានប្រកាសទទួលលក្ខណៈជា private។

**class base { // error**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**void getdata(); //** ការប្រកាសអនុគមន៍ getdata()

**};**

**class derived:private base {**

**public :**

**void display(){**

**++value; //** មានលក្ខណៈមិនត្រឹមត្រូវ

**}**

**};**

ឧទាហរណ៍ទី ១ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវ data member នៃ class មួយដែលត្រូវបានចូលប្រើដោយ member function។

**// Accessing public data by member function**

**#include <iostream.h>**

**class sample {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**inline void getdata(){**

**cout << "Enter a number :" << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**inline void display(){**

**cout << "Enter number is = " << value << endl;**

**}**

**};**

**void main(){**

**sample ob; //** ការបង្កើត object នៃ sample class

**ob.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ sample class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ sample class

**}**

ឧទាហរណ៍ទី ២ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវ data member នៃ class មួយជាប្រភេទ public ដែល ត្រូវបានចូលប្រើដោយ non-member function នៃ class។

**// Accessing public data by non-member function**

**#include <iostream.h>**

**class sample {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**inline void getdata(){**

**cout << "Enter a number : " << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**};**

**void main(){**

**sample ob; //** ការបង្កើត object នៃ sample class

**ob.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ sample class

**cout << "Enter number is = " << ob.value << endl;**

**}**

ឧទាហរណ៍ទី ២​ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវ member function មួយនៃ derived class អាចចូលប្រើទិន្នន័យរបស់ base class ប្រភេទ public ដែល derived class បានទទួលលក្ខណៈជា public។

**// Accessing public data by derived class**

**#include <iostream.h>**

**class base {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**inline void getdata(){**

**cout << "Enter a number :" << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derived:public base {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**}**

**}; // end of class definition**

**void main(){**

**derived ob; //** ការបង្កើត object នៃ derived class

**ob.getdata(); //** ការហៅអនុគមន៍ getdata() តាមរយៈ object នៃ derived class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derived class

**cout <<"Value in derived class = "<< ob.value<< endl;**

**}**

ឧទាហរណ៍ទី ៤ ៖ កម្មវិធីខាងក្រោមនេះមិនអាច compiled បានទេ ទោះជា member function នៃ derived class ព្យាយាមចូលប្រើទិន្នន័យរបស់ base class ប្រភេទ public ក៏ដោយ ព្រោះ derived class បានទទួលលក្ខណៈជា private។

**// Accessing public data by derived class**

**// private derivation**

**#include <iostream.h>**

**class base {**

**public :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int

**inline void getdata(){**

**cout << "Enter a number : " << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derived:private base {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**}**

**}; // end of class definition**

**void main(){**

**derived ob; //** ការបង្កើត object នៃ derived class

**ob.getdata(); //** មិនអាចចូលប្រើ getdata() របស់ base class បានឡើយ

**ob.display(); //** មិនអាចចូលប្រើ value របស់ base class បានឡើយ

**cout <<"Value in derived class = "<< ob.value<< endl;**

**}**

៩.២ ការចូលទៅប្រើ data member ប្រភេទ private

Members នៃ class ប្រភេទ private អាចប្រើបានដូចខាងក្រោម ៖

 Member functions នៃ class

 Member functions នៃ friend class ដែលវាបានប្រកាសក្នុងនោះ

គេអាចនិយាយម៉្យាងទៀត member function នៃ derived class ដែលបញ្ជាក់ public មិនអាចចូលប្រើ data members នៃ base class ប្រភេទ private មិនថាទោះជា derived class បានទទួលលក្ខណៈជា public ឬ private ឡើយ។

ឧទាហរណ៍ code នៃកម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រកាសចូលប្រើ data members នៃ base class ប្រភេទ private តាមរយៈ members នៃ derived class ប្រភេទ public ដែលមាន លក្ខណៈមិនត្រឹមត្រូវ ៖

ករណីទី ១**៖** derived class ជាមួយនឹងការទទួលលក្ខណៈប្រភេទ private។

**class baseA {**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:private baseA {**

**public :**

**void f(){**

**++value; // error,** ដោយ **baseA::value** មិនអាចចូលប្រើបាន

**​​​​ }**

**};**

ករណីទី ២ ៖ ទោះជា derived class បានទទួលលក្ខណៈជា public ពី base class ក៏ដោយ member នៃ derived class ប្រភេទ public មិនអាចចូលប្រើ member នៃ base class ប្រភេទ private ឡើយ។

**class baseA {**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:public baseA {**

**public :**

**void f(){**

**++value; // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

**};**

ឧទាហរណ៍ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីរបៀបដែល data member ប្រភេទ private មិនអាចចូលប្រើដោយ member function នៃ derived class ប្រភេទ public ឡើយ ទោះបីជា derived class បាន ទទួលលក្ខណៈជា public ក៏ដោយ។

**// Demonstration of private member**

**#include <iostream.h>**

**class baseA {**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**public :**

**baseA(){**

**cout << "Enter a number :" << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:public baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedB ob; //** ការបង្កើត object នៃ derivedB class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedB class

**}**

ចូរកត់សម្គាល់ថា កម្មវិធីខាងលើនេះបង្ហាញនូវសារ error មួយដោយបញ្ជាក់ថា baseA::value is not accessible។

៩.៣ ការចូលទៅប្រើ data member ប្រភេទ protected

Data members នៃ class ប្រភេទ protected អាចប្រើបានដូចខាងក្រោម ៖

 Member functions នៃ class

 friends នៃ class ដែលបានប្រកាស

 Member functions នៃ derived class មិនថា derived class បានទទួលលក្ខណៈជា private ឬ public ឡើយ។

ឧទាហរណ៍ នៅក្នុង code នៃកម្មវិធីខាងក្រោមនេះបង្ហាញពី data members នៃ base class ប្រភេទ protected ត្រូវបានចូលប្រើដោយ member function នៃ derived class ប្រភេទ public។

ករណីទី ១ ៖ data member នៃ class មួយប្រភេទ protected អាចចូលប្រើដោយ member function ប្រភេទ public ទោះជាវាបានទទួលលក្ខណៈជា private ក៏ដោយ បានផ្តល់នូវ derived class ដែលមាន base class ផ្ទាល់។

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**};**

**class derivedB:private baseA {**

**public :**

**void f(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ករណីទី ២ **៖** data member នៃ class មួយប្រភេទ protected មិនអាចចូលប្រើដោយ member function ប្រភេទ public របស់ derivedD បានទេ ពីព្រោះ derivedD មិនបានទទួលលក្ខណៈពី base class ផ្ទាល់របស់ baseA។

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**};**

**class derivedB:private baseA {**

**......**

**};**

**class derivedC:private derivedB {**

**......**

**};**

**class derivedD:private derivedC {**

**public :**

**void f(){**

**++value; // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

**};**

ករណីទី ៣**៖** data member នៃ base class មួយប្រភេទ protected អាចចូលប្រើ member function ប្រភេទ public របស់ derived class មួយឆ្លងតាមរយៈការទទួលលក្ខណៈ public។

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**};**

**class derivedB:public baseA {**

**public :**

**void f(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ករណីទី ៤ ៖ data member នៃ base class មួយប្រភេទ protected អាចចូលប្រើដោយ member functions ប្រភេទ public របស់ derived class ទោះជាវាបានទទួលលក្ខណៈពី base class ផ្ទាល់ ឬមិនផ្ទាល់ដែលផ្តល់ឲ្យវាឆ្លងតាមរយៈការទទួលលក្ខណៈជា public។

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**};**

**class derivedB:public baseA {**

**......**

**......**

**};**

**class derivedC:public derivedB {**

**......**

**......**

**};**

**class derivedD:public derivedC {**

**public :**

**void f(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ឧទាហរណ៍ទី ១ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពី data member នៃ base class មួយប្រភេទ protected ត្រូវបានចូលប្រើដោយ member function នៃ derived class ប្រភេទ public ប្រសិនបើវាបានទទួលលក្ខណៈជា private។

**// Demonstration of accessing a protected member**

**#include <iostream.h>**

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**public :**

**baseA(){**

**cout << "Enter a number : " << endl;**

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:private baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Content of value = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedB ob; //** ការបង្កើត object នៃ derivedB class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedB class

**}**

ឧទាហរណ៍ទី ២ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពី data member នៃ base class ប្រភេទ protected ត្រូវបានចូលប្រើដោយ member function នៃ derived class ប្រភេទ public មិនថាទោះជាវាមាន base class ផ្ទាល់ ឬមិនផ្ទាល់ ហើយបានទទួលលក្ខណៈជា public។

**// Demonstration of accessing a protected data**

**#include <iostream.h>**

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:public baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:public derivedB {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedD:public derivedC {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedD = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedD ob; //** ការបង្កើត object នៃ derivedD class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedD class

**}**

ឧទាហរណ៍ទី ៣ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវសារ error មួយកើតឡើង ហើយវាមិនធ្វើការ compiled ឡើយ។ data member នៃ base class មួយប្រភេទ protected មិនអាចចូលប្រើ member function នៃ derived class អាស្រ័យដោយ base class មិនផ្ទាល់ឆ្លងតាមរយៈការទទួលលក្ខណៈជា private។

**// Demonstration of public access**

**#include <iostream.h>**

**class baseA {**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:private baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:private derivedB {**

**public :**

**void display(){**

**// error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedC ob; //** ការបង្កើត object នៃ derivedC class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedC class

**}**

៩.៤ ការចូលទៅប្រើ data member ប្រភេទ private ដោយ friend class

member function ប្រភេទ public នៃ friend class អាចចូលប្រើ members នៃ base class ប្រភេទ private ដោយមិនគិតថា friend function បានទទួលលក្ខណៈជា private ឬ public ឡើយ។

ករណីទី ១ ៖ friend class បានទទួលលក្ខណៈជា private។

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:private baseA {**

**public :**

**void f(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ករណីទី ២ ៖ friend class បានទទួលលក្ខណៈជា public។

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:public baseA {**

**public :**

**void f(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

ករណីទី ៣ ៖ friend class បានទទួលលក្ខណៈជា private។

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:private baseA {**

**public :**

**void g(){**

**++value; //** មានលក្ខណៈត្រឹមត្រូវ

**}**

**};**

**class derivedC:public baseA {**

**public :**

**void f(){**

**++value; // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

**};**

វាពុំមាន member function ប្រភេទ public អាចចូលប្រើ members នៃ base class ប្រភេទ private ឡើយ ដរាបណាវាជា member function នៃ friend class ប្រភេទ public។ ចូរកត់សម្គាល់ថា derived class ឈ្មោះ derivedC មិនមែនជា friend class នៃ baseA ហេតុនេះហើយវាមិនអាចចូលប្រើ data នៃ base class ប្រភេទ private ទោះជាវាបានទទួលលក្ខណៈជា public ក៏ដោយ។

ករណីទី ៤ ៖ friend class បានទទួលលក្ខណៈជា private។

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:private baseA {**

**.........**

**.........**

**};**

**class derivedC:private derivedB {**

**public :**

**void f(){**

**++value; // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

**};**

ចូរកត់សម្គាល់ថា derived class ឈ្មោះ derivedC មិនអាចចូលប្រើ data នៃ base class ប្រភេទ private ដោយសារវាបានទទួលលក្ខណៈតាមរយៈ derivedB មិនផ្ទាល់។ ហើយនិងដោយសារ baseA ពង្រីកបន្ថែមលក្ខណៈ friend របស់វា តែជាមួយនឹង derivedB មិនមែនជាមួយនឹង class ផ្សេងឡើយ។

ករណីទី ៥ ៖ វាពុំមាន member function ប្រភេទ public អាចចូលប្រើ members នៃ base class ប្រភេទ private ឡើយ ដរាបណាវាជា member function នៃ friend class ប្រភេទ public ទោះជា derived class បានទទួលលក្ខណៈជា public ក៏ដោយ។

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**};**

**class derivedB:public baseA {**

**... ... ...**

**... ... ...**

**};**

**class derivedC:public derivedB {**

**public :**

**void g(){**

**++value; // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

**};**

ឧទាហរណ៍ទី ១ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវទិន្នន័យនៃ base class ប្រភេទ private ត្រូវបានចូលប្រើដោយ member function នៃ derived class ប្រភេទ public តាមរយៈការប្រកាស friend class។ derived class បានទទួលលក្ខណៈតាមរយៈ public inheritance។

**// Demonstration of friend class**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:public baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedB ob; //** ការបង្កើត object នៃ derivedB class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedB class

**}**

ឧទាហរណ៍ទី ២ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវទិន្នន័យនៃ base class ប្រភេទ private ត្រូវបានចូលប្រើដោយ member function នៃ derived class ប្រភេទ public តាមរយៈការប្រកាស friend class។ derived class បានទទួលលក្ខណៈតាមរយៈ private inheritance។

**// Demonstration of friend class**

**// private inheritance**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:private baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedB ob; //** ការបង្កើត object នៃ derivedB class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedB class

**}**

ឧទាហរណ៍ទី ៣ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវសារ error អាស្រ័យមកពីការចូលប្រើទិន្នន័យប្រភេទ private របស់ base class ដោយ member function នៃ derived class ប្រភេទ public តាមរយៈ private inheritance។ ដោយសារការទទួលលក្ខណៈជា private នោះ members របស់ base class មិនអាចចូលប្រើដោយ members របស់ derived class ប្រភេទ public ឡើយ។

**// error baseA::value is not accessible**

**// Demonstration of friend class**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**public :**

**baseA(){**

**cout << "Enter a number: ";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of class definition**

**class derivedB:private baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:private derivedB {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedB ob; //** ការបង្កើត object នៃ derivedB class

**ob.display(); // error** ដោយbaseA::valueមិនអាចចូលប្រើបាន

**}**

ឧទាហរណ៍ទី ៤ ៖ កម្មវិធីខាងក្រោមនេះព្យាយាមចូលប្រើទិន្នន័យនៃ base class ប្រភេទ private ដោយ member function របស់ derived class ប្រភេទ public ឆ្លងតាម public inheritance តាមរយៈ friendship។

**// Demonstration of friend class**

**// Error message will appear**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**private :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ private

**public :**

**baseA(){**

**cout << "Enter a number:";**// បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of base declaration**

**class derivedB:public baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:public derivedB {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedC ob; //** ការបង្កើត object នៃ derivedC class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedC class

**}**

ឧទាហរណ៍ទី ៥ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញទិន្នន័យរបស់ base class ប្រភេទ protected ត្រូវបានចូលប្រើដោយ member function របស់ derived class ប្រភេទ public តាម​​រយៈ​​ការ

ប្រកាស class ជា friend។ derived class បានទទួលលក្ខណៈតាមរយៈ public inheritance។

**// Demonstration of friend class**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**protected :**

**​​​​​​​ int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of base declaration**

**class derivedB:public baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:public derivedB {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedC ob; //** ការបង្កើត object នៃ derivedC class

**ob.display(); //** ការហៅអនុគមន៍ display() តាមរយៈ object នៃ derivedC class

**}**

ឧទាហរណ៍ទី ៦ ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញថា error មួយបានប្រទះឃើញនៅពេល លទ្ធផលនៃការព្យាយាមចូលប្រើទិន្នន័យរបស់ base class ប្រភេទ protected ដោយ member function របស់ derived class ប្រភេទ public តាមរយៈ private inheritance។ ដោយសារការទទួលលក្ខណៈប្រភេទ private នោះ members របស់ base class មិនអាចចូលប្រើដោយ members របស់ derived class ប្រភេទ public ទោះជាវាមានលក្ខណៈជា friend ជាមួយនឹង base class ក៏ដោយ។

**// Demonstration of friend class**

**// Error baseA::value is not accessible**

**#include <iostream.h>**

**class baseA {**

**friend class derivedB;**

**protected :**

**int value; //** ប្រកាសអញ្ញាត value ជាប្រភេទ int មានលក្ខណៈ protected

**public :**

**baseA(){**

**cout << "Enter a number:";** // បង្ហាញអក្សរ Enter a number : មកលើអេក្រង់

**cin >> value; //** បញ្ចូលតម្លៃ value តាមរយៈ keyboard

**}**

**}; // end of base declaration**

**class derivedB:private baseA {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedB = " << value << endl;**

**}**

**}; // end of derived class definition**

**class derivedC:private derivedB {**

**public :**

**void display(){**

**++value; //** ការបង្កើនតម្លៃមួយឯកតាឲ្យអញ្ញាត value

**cout << "Value in derivedC = " << value << endl;**

**}**

**}; // end of derived class definition**

**void main(){**

**derivedC ob; //** ការបង្កើត object នៃ derivedC class

**ob.display(); // error** កើតឡើង

**}**

៩.៥ តារាងសង្ខេបនៃកម្រិតចូលប្រើ data members នៅក្នុង inheritance

|  |  |  |  |
| --- | --- | --- | --- |
| ការបញ្ជាក់លក្ខណៈរបស់ class | អាចចូលប្រើបានតែក្នុង class របស់ខ្លួន | អាចចូលប្រើបានពី Derived class | អាចចូលប្រើបានតាម object |
| public  protected  private | yes  yes  yes | yes  yes  no | yes  no  no |

សំនួរនិងលំហាត់

១ - ចូរនិយាយលក្ខណៈ inheritance ក្នុងភាសា C++។

២ - តើគេចែកលក្ខណៈ inheritance នៅក្នុងភាសា C++ ប៉ុន្មានប្រភេទ? គឺអ្វីខ្លះ?

៣ - ដូចម្តេចហៅថា base class ? ដូចម្តេចហៅថា derived class?

៤ - អ្វីទៅ base class ផ្ទាល់ និង base class មិនផ្ទាល់?

៥ - ចូរនិយាយពីគុណសម្បត្តិ និងគុណវិបត្តិនៃការប្រើលក្ខណៈ multiple inheritance។

៦ - អ្វីទៅភាពមិនច្បាស់លាស់ក្នុង inheritance ? ដើមី្បបញ្ចៀសបញ្ហានេះ តើយើងត្រូវប្រើ

រូបភាពបែបណា? ចូរឲ្យឧទាហរណ៍ជា code មកបង្ហាញ។

៧ - ​ចូរនិយាយពីកម្រិតចូលប្រើ members របស់ base class គឺ private, protected និង public

នៅក្នុង derived class ដែល​វាបានទទួលលក្ខណៈ inheritance កម្រិត private។

៨ - ​ចូរនិយាយពីកម្រិតចូលប្រើ members របស់ base class គឺ private, protected និង public

នៅក្នុង derived class ដែល​វាបានទទួលលក្ខណៈ inheritance កម្រិត protected។

៩ - ​ចូរនិយាយពីកម្រិតចូលប្រើ members របស់ base class គឺ private, protected និង public

នៅក្នុង derived class ដែល​វាបានទទួលលក្ខណៈ inheritance កម្រិត public។

១០​- ចូរសរសេរ code បង្ហាញពីការប្រើលក្ខណៈ single inheritance ។

១១​- ចូរសរសេរ code បង្ហាញពីការប្រើលក្ខណៈ multiple inheritance ។

១២​- អ្វីទៅ Container class ? ចូរឲ្យឧទាហរណ៍ជា code។