អំពី Classes និង Objects​

១. សេចក្តីផ្តើម

តាំងពីដើមមកដល់ពេលនេះ ការបង្កើតកម្មវិធី C++ កន្លងមកពុំមានការប្រើ classes និង Objects ទេដែលជាហេតុនាំឲ្យគេមិនអាចមើលឃើញពីអនុភាពដ៏ខ្លាំងរបស់ C++ ។ Class មួយ គឺជាប្រភេទទិន្នន័យកំណត់ដោយអ្នកប្រើប្រាស់ដែលរួមមាន data និង functions ។ data នៅខាងក្នុង class មួយត្រូវបានហៅថា member data (or data member) ហើយនិង function ត្រូវបានហៅថា member functions ។ member functions ភាគច្រើនប្រើ data ខាងក្នុង class ។ អញ្ញាតនៃ class មួយត្រូវបានហៅថា Objects ឬ instances នៃ class ។

ពាក្យ class ជាពាក្យសំខាន់មួយរបស់ភាសា c++ វាត្រូវបានប្រើប្រាស់សម្រាប់បណ្តុំ data និងការប្រតិបត្តនៃ structure ទៅជារបស់មួយ (entity) ។ ការបង្កើត class មានសភាពខុសពីការបង្កើត struct នៅក្នុងភាសា c ។ ការបង្កើត class ផ្តល់នូវលក្ខណ: data hiding, abstraction, encapsulation, single inheritance, multiple inheritance, polymorphism និង public interface functions (methods) សំរាប់បញ្ជូនព័ត៌មានរវាង objects និង​ objects ។ មុននឹង សិក្សាទៅលើលក្ខណ:ទាំងនេះ យើងគួរយល់ពាក្យមួយចំនួនដូចខាងក្រោម

+ Data abstraction : នៅក្នុង oop , data abstraction ត្រូវបានកំណត់ដូចទៅនឹងការប្រមូលផ្តុំ នៃ data និង​ methods ( functions ) ។​

+ Data hiding : នៅក្នុង C++ ការបង្កើត class អាចឲ្យគេប្រកាស data និង methods ជាលក្ខណ: public, private និង​ protected ។ ការប្រើឲ្យល្អិតល្អន់នូវ class មួយអាចអោយគេបិទបាំង class បាន ។ លក្ខណ:នេះអាចប្រព្រឹត្តទៅបានតាមគោលការណ៏ data hiding ។

+ Data encapsulation : data ខាងក្នុង (member data) នៃ class មួយត្រូវបានគេញែកដាច់ពី ពិភពខាងក្រៅ (class ដែលបានកំណត់) ។ ពេលនោះវាត្រូវបានដាក់ជាមួយនឹង member function នៅក្នុងកន្សោម​(​capsule) ។ គេអាចនិយាយម៉្យាងទៀតថា encapsulation ប្រមូលផ្តុំគ្រប់បំណែកនៃ object មួយទៅជាកញ្ចប់មួយយ៉ាងមាំ។ វាអាចបញ្ជៀសនូវឥទ្ធិពលនៃ member data ដែលមិនចាំបាច់នៅពេលដែលវាបានកំណត់ខាងក្រៅ class ហើយអាចការពារ ពីការប្រើមិនត្រឹមត្រូវរបស់ data ។ Classes គ្រប់គ្រងភាពសាំញ៉ាំរបស់កម្មវិធីធំៗតាមរយ:នៃ ការប្រើ encapsulation នេះដោយមានប្រសិទ្ធភាពខ្ពស់។

+ Inheritance : C អាចឲ្យសរសេរកម្មវិធីបង្កើតលំដាប់ថ្នាក់នៃ class ។ ការទទូលលក្ខណ: បន្តគ្នានៃ class ត្រូវបានប្រើសម្រាប់បង្កើតលំដាប់ថ្នាក់។​ លក្ខណ:ពិសេសរបស់ class (parent classes or basic classes ) ត្រូវបានបញ្ជូនទៅលើ class ផ្សេងទៀត​ (chile classes) ។ នៅក្នុង ការអនុវត្តជាក់ស្តែង គោលការណ៏ inheritance បន្ថយការសរសេរ ព្រោះលក្ខណ:ដែលទទួល បានពី class នោះមិនបាច់សរសេរឡើងវិញម្តងទៀតឡើយ។

+ Polymorphism : នៅក្នុង OOP , Polymorphism ត្រូវបានកំណត់ដូចជារបៀបអនុវត្តនូវជំហាន ដំណើរការផ្សេងគ្នាតាមរយ: function មួយដែលមានព័ត៌មានដូចគ្នា។ Polymorphism ចាត់ទុក objects នៃ​ Classes ដែលមានទំនាក់ទំនងគ្នាតាមលក្ខណ:ទូទៅមួយ។

**២. អំពី Classes**

Class មាន member មួយឬច្រើនដែលផ្តុំគ្នាជារបស់តែមួយ លើសពីនេះវាមិនគ្រាន់តែមាន data elements តែថែមទាំងមាន functions ដែលត្រូវប្រតិបត្តិទៅលើ data elements។ ដោយឡែកនៅក្នុង structure គ្រប់ elements ទាំងអស់មានលក្ខណ:ជា public រីឯ class វិញ មានលក្ខណ:ជា private ។ Data និង functions ដែលបានកំណត់ក្នុង class មួយជាលក្ខណ: private, public, និង​ protected ។ ​Functions ដែលបានកំណត់នៅក្នុង class នោះមានទំនាក់ ទំនងគ្នាពិសេសទៅនឹង member data ហើយ​ និង functions (methods) ។

ទម្រង់ទូទៅរបស់ class មួយគឺ

class user\_defined\_name {

private:

data\_type members

implementation operations

list of friend functions

public:

data\_type members

implementation operations

protected:

data\_type operators

implementation operations

};

class user\_defined\_name variable1, variable2,…….,variableN;

ពាក្យ private, protected និង public ត្រូវបានប្រើសម្រាប់បញ្ជាក់កម្រិតនៃការចូលទៅ ប្រើ hiding data និង function members ខាងក្នុង class ។

+ លក្ខណ: private : Member data អាចចូលទៅប្រើដោយ member functions និង​ friend class​ ។ member functions និង​ friend class ទាំងនេះជានិច្ចកាលអាចអាន ឬសរសេរទៅលើ data members ដែលមានលក្ខណ: private ​។ data members ដែលមានលក្ខណ:ជា private មិនអាចចូលទៅប្រើអ្វីដែលមាននៅក្រៅ class បានឡើយ ។

+ លក្ខណ: protected : Members ដែលត្រូវបានប្រកាសនៅក្នុងផ្នែកនេះអាចចូលប្រើដោយ member functions និង​ friend class ។ functions ទាំងនេះអាចចូលប្រើ ដោយ member functions និង friends ដែលបានទទួលលក្ខណ:ពី class នេះ។ វាមិនអាចចូលប្រើអ្វីដែលនៅក្រៅ class បានឡើយ។

+ លក្ខណ: public : Members ដែលបានប្រកាសនៅក្នុងផ្នែកនេះអាចចូលប្រើដោយ function ណាមួយនៅខាងក្រៅ class ។ ប្រតិបត្តិការដែលមានលក្ខណ: public អាចជា member functions or methods or interfaces ដែលហៅមកប្រើនៅខាង ក្រៅ class ។ function ណាក៏ដោយអាចបញ្ជូនព័ត៌មានទៅឲ្យ object នៃ class នេះតាមរយ: interface functions ទាំងនេះ។

ឧទាហរណ៏ទី១ class sample {

int a ;

float b;

char ch;

}; //A class by default has all its members private

ឧទាហរណ៏ទី២ struct sample {

Int a;

Float b;

Char ch;

}; //A struct by default has all its members public

Class គឺជាពាក្យរបស់ C++ ។ វាជាប្រភេទទិន្នន័យមួយកំណត់ដោយអ្នកប្រើប្រាស់ហើយវាជាការបង្រួមនៃ data hiding និង encapsulation ។ Polymorphism ត្រូវបានប្រើតាមរយ: class ជាមួយនឹង virtual functions ។

៣. ការប្រកាស class

Class មួយជាប្រភេទទិន្នន័យមួយកំណត់ដោយអ្នកប្រើប្រាស់ដែលរួមមានពីរផ្នែក គឺផ្នែក private និង​ protected ដែលផ្ទុក data ហើយនិងផ្នែក public ដែលផ្ទុកការប្រតិបត្តិ interface ។

ការកំណត់ class មួយគឺជាដំណើរការនៃការដាក់ឈ្មោះឲ្យ class មួយនិងអញ្ញាត methods ឬការប្រតិបត្តិ interface របស់ class ។

ការប្រកាស class មួយគឺជាដំណើរការនៃការដាក់ឈ្មោះឲ្យ class មួយនិងអញ្ញាត ប្រតិបត្តិដែលអាចប្រើទៅលើការបង្កើត class មួយគឺ:

Class user\_defined\_name {

Private:

data\_type members

implementation operations

list of friend functions

Public:

data\_type members

implementation operations

protected:

data\_type members

implementation operations

};

class user\_defined\_name variable1, variable2,…….,variableN;

ការប្រកាស class មួយប្រាប់ឲ្យដឹងឈ្មោះ class ក្នុងទំហំ​ scope ដែលវាបានប្រកាសនិងបិទបាំង class , object, function ណាមួយ ឬការប្រកាសផ្សេងទៀតរបស់ឈ្មោះនោះនៅក្នុងទំហំ scope ដែលបានកំណត់។

Class members អាចជា member ក្នុងចំណោម members ដែលបានរាយដូចខាងក្រោម

. data

. functions

. classes

. friends

. data type names

. encapsulations

ឧទាហរណ៏ទី១

class date {

int day ;

int month ;

int year ;

};

class date today ; // today is object of class date

ឧទាហរណ៏ទី២

class student {

private :

long int rollno;

int age ;

char sex ;

float height ;

float weight ;

public :

void getinfo () ;

void disinfo () ;

}; // end of class definition

ការកំណត់លក្ខណ: class អាចឲ្យគេប្រកាសនៅក្នុងការប្រកាស class ខ្លួនវា ។

class date {

public :

int day ;

int month ;

int year ;

} today ; // now the object is created of class date

ពាក្យ class គឺជាពាក្យសំខាន់សម្រាប់កំណត់ប្រភេទទិន្នន័យ class មួយនៅក្នុងភាសា C++ ។ ភាពខុសគ្នារវាង structure និង classes គឺស្ថិតនៅលើកម្រិតប្រើបាននៃទីតាំងរបស់ members ដូចយើងបានដឹងរួចហើយថានៅក្នុង class មួយតាមលក្ខណ: default គឺគ្រប់ members ទាំងអស់មានលក្ខណ:ជា private រីឯនៅក្នុង structure មានលក្ខណ:ជា​ public ។

ឧទាហរណ៏ទី១

class sample {

Int x;

Int y;

}; // by default members are private

ឧទាហរណ៏ទី២

class sample {

private:

int x;

int y;

};

ពាក្យ private ត្រូវបានប្រើសំរាប់ប្រកាស data items នៃ class មួយជាលក្ខណ:បង្ហាញឲ្យឃើញ ច្បាស់ថាជា private ។

ចូរយើងសង្កេតនូវលក្ខណ:ពិសេសមួយចំនួននៃការប្រកាសប្រភេទទិន្នន័យជា class ។ class អាចបង្កើតឡើងដោយគ្មានឈ្មោះ។ class ដែលគ្មានឈ្មោះមិនអាចមាន constructors និង destructores បានឡើយ ហើយក៏មិនអាចបញ្ចូលតម្លៃជា argument ឬឲ្យតម្លៃបានដែរ។ Object នៃ class អាចកំណត់តម្លៃបាន បញ្ជូនតម្លៃជា​ arguments ទៅឲ្យ function​ ឬឲ្យតម្លៃតាម រយ: function ។

នៅក្នុងភាសា C++ គេអាចប្រកាស class ទទេមួយដែលមានទំហំមិនសូន្យ។

Class sample { }; // class with no members

គួរកត់សម្គាល់ដែរថា member មួយមិនអាចប្រកាសពីរដងនៅក្នុង list ឡើយ។

ឧទាហរណ៏ទី១

class sample {

int one;

int two;

int one;

}; // error, the data member ‘ one ‘ has been redeclared

ឧទាហរណ៏ទី២

class item {

private:

float x ;

char ch;

};

Item :: y; //error, no member can be added elsewhere, other than class declaration.

ឧទាហរណ៏ទី៣

class sample {

private:

Int x, y;

public:

void setdata();

void getdata();

void display();

void setdata(); // error

};

វាមានគោលការណ៏ដូចគ្នាទៅនឹងការប្រកាស function ដែរ។

ឧទាហរណ៏ទី៤

class xy {

private:

int funct;

public:

int funct(); // error, same name is used for both data member and function.

};

ឧទាហរណ៏ទី៥

class xy {

private:

int funct();

int (\*funct)();

};

// error , pointer to a function and the function name are same

៤.ការប្រកាស function ជា member របស់ class

Function មួយបានប្រកាសជា member មួយ (ដោយពុំមានបញ្ជាក់លក្ខណ: firend ) នៃ class មួយត្រូវបានហៅថា member function ។ member function ភាគច្រើនផ្តល់នូវលក្ខណ: public ព្រោះវាត្រូវហៅមកប្រើខាងក្រៅ class មិនថានៅក្នុងកម្មវិធីឬនៅក្នុង function ឡើយ។

Member function នៃ class មួយត្រូវរៀបចំឡើងសម្រាប់ធ្វើការប្រតិបត្តិលើប្រភេទទិន្នន័យបីប្រភេទ។ ជាធម្មតា វាអាចចំណាត់ថ្នាក់ជាបីប្រភេទមានដូចខាងក្រោមនេះ

+ Manager functions : ត្រូវបានប្រើសម្រាប់ធ្វើការកំណត់តម្លៃដំបូងនិងបំបាត់កំណកំណើតនៃ class object ។ ឧទាហរណ៏ Manager functions គឺជា constructor និង destructor function ។

+ Accssor functions : គឺជា​ constructor functions ដែលឲ្យព័ត៌មានអំពីសភាពកំពុង ប្រើ object ។ ឧទាហរណ៏ accssor function គឺជា member function មានតម្លៃថេរ។

+ Implementor functions: គឺជា function ដែលធ្វើការកែប្រែទៅលើ data members។

Functions ទាំងនេះត្រូវបានហៅថា mutators ។

ឧទាហរណ៏

class sample {

private :

int x;

int y;

public :

int sum () {

return (x + y );

}

int diff () {

return ( x – y );

}

}; //end of class definition

ចូរសង្កេតនួវការកំណត់ member function មួយរបស់ class ខាងក្រៅទំហំ scope របស់វា។ នៅក្នុងភាសា C++ វាអាចអោយគេប្រកាស member function ទោះជានៅខាងក្នុង class ឬ ខាងក្រៅ class ក៏ដោយ។ Member function នៃ class មួយត្រូវបានបង្កើតឡើងដោយប្រើសញ្ញា :: ហៅថា scoping operator ។

ទម្រង់ទូទៅរបស់ member function នៃ class មួយខាងក្រៅ scope របស់វាគឺ

Return\_type class\_name::member\_function (argument1,2,….,n)

គួរកត់សម្គាល់ថា ប្រភេទទិន្នន័យរបស់ member function arguments ត្រូវតែស៊ីគ្នាជាមួយនឹងប្រភេទទិន្នន័យដែលបានប្រកាសនៅក្នុង class របស់ class\_name ។

ចំណុចសំខាន់ដែលគួរឲ្យកត់សម្គាល់គឺការប្រើទំហំ resolution operator (::) ជាមួយនឹង ឈ្មោះ class នៅខាងដើមនៃ function ។ មានតែ scope operator ទេដែលអាចសម្គាល់បាននូវ function ជា​ member របស់ class មួយ។ បើគ្មាន scope operator នេះ ការបង្កើត function នឹង ក្លាយទៅជា function ធម្មតា ហើយត្រូវគោរពតាមគោលការណ៏ function នៅពេលហៅមកប្រើ និង ទំហំ scope ដែលអាចប្រើបាន។

ឧទាហរណ៏ទី១

class sample {

private :

int x;

int y;

public :

int sum ();

int diff () ;

};

int sample :: sum () {

return ( x + y );

}

int sample :: diff () {

return ( x – y ) ;

}

ឧទាហរណ៏ទី២

class first {

private :

int x;

int y;

public :

int sum ();

int diff ();

};

first one ;

first two;

int sum () {

// error, scope of the member function is not defined

return ( x + y);

}

ចំពោះ code នៃកម្មវិធីខាងលើ ​classes ទាំងពីរត្រូវបានបង្កើតឡើងដោយមានឈ្មោះនៃ member function ដូចគ្នា ហើយនៅពេលចូលប្រើ ​member function ទាំងនេះ វានឹងមាន error កើតឡើង។ ទំហំ scope នៃ ​member function ឈ្មោះ sum() មិនបានបញ្ជាក់ឡើយ។ កាលណាចូលប្រើ member function ដែលឈ្មោះ sum() នេះ ការត្រូតពិនិត្យនឹងត្រូវបញ្ជួនទៅកាន់ classes ទាំងពីរគឺ one និង​ two ។ ដូចនេះសញ្ញា scopint operator ត្រូវការជាចាំបាច់ដើម្បី កំណត់នូវ member functions នៅខាងក្រៅ class ។

int one:: sum() {

return (x + y);

}

int two:: sum () {

Return (x – y );

}

៥. ការកំណត់ object នៃ class មួយ

ពាក្យ object និង​ class ត្រូវបានប្រើជាញឹកញាប់នៅក្នុងផ្នែកមុននេះ។ ជាទូទៅ class មួយគឺជាប្រភេទទិន្នន័យដែលបានកំណត់ដោយអ្នកប្រើប្រាស់ រីឯ object ជាអ្វីដែលកកើតពី ទម្រង់ class មួយ។ class មួយផ្តល់នូវទម្រង់ទូទៅមួយដែលកំណត់​ member functions និងអញ្ញាតដែលជា object នៃប្រភេទ class នេះតាមត្រូវការ។ class មួយត្រូវតែកំណត់លក្ខណ: ជាមុន មុននឹងប្រកាសវា។

ទម្រង់ទូទៅសម្រាប់ការកំណត់នូវ objects នៃ class មួយគឺ

class user\_defined\_name {

private :

data

methods

public :

methods

protected :

data

};

User\_defined\_name object1, object2,…..objectN;

ឧទាហរណ៏ទី១

class student {

private :

long int rollno;

int age;

char sex;

float height;

float weight;

public :

void getinfo ();

void disinfo ();

void process ();

void personal ();

};

student obj;

ឧទាហរណ៏ទី២

class hospital\_info {

private :

char patient\_name[20];

char sex;

int age;

char parents\_name[20];

char address[30];

char illness[30];

int bedno;

int wardno;

public :

void getinfo ();

void display\_info ();

void payment ();

void operation ();

};

hospital\_info obj1, obj2;

ឧទាហរណ៏ទី៣

class employ {

private :

char name [20];

int code;

char designation[20];

char address[30];

float salary;

int age;

public :

void salary\_payment ();

void saving ();

void tax\_payment ();

void getinfo ();

void display ();

};

Employ ob1, ob2;

៦. ការចូលប្រើ member របស់ class

វាមានពីរវិធីដែលអាចចូលប្រើ member នៃ class មួយស្រដៀងទៅនឹងការចូលប្រើ member function នៃ structure ដែរ។ Data ឬ function member មួយអាចចូលប្រើតាមរយ:សញ្ញាចំណុច។

ទម្រង់ទូទៅនៃការចូលប្រើ member មួយរបស់ class គឺ

class\_object.data\_member

class\_object.function\_member

ឧទាហរណ៏

class sample {

private :

int x;

int y;

public :

int sum ();

int diff ();

};

void main () {

sample one;

one.sum ();

one.diff ();

……….

}

ឧទាហរណ៏ទី១

#include<iostream.h>

#include<conio.h>

class date {

public :

int day;

int month;

int year;

};

void main () {

class date today;

today.day = 15;

today.month = 10;

today.year = 2009;

cout << “Today ‘s date is = “ << today.day << “/”;

cout << today.month << “/ “<< today.year << endl;

getch();

}

ឧទាហរណ៏ទី២

កម្មវិធីខាងក្រោមនេះបង្ហាញនួវការចូលទៅប្រើ members របស់ class ។​ កម្មវិធីនេះមាន error កើតឡើងដោយសារតែ members របស់ class​ ពុំបានបញ្ជាក់កម្រិតចូលប្រើជាអ្វីទេ។ ដូចនេះតាមធម្មតាជា private ។

#include<iostream.h>

#include<conio.h>

class date {

int day;

int month;

int year;

};

void main () {

class date today;

today.day = 15;

today.month = 10;

today.year = 2009;

cout << “Today ‘s date is = “ << today.day << “/”;

cout << today.month << “/ “<< today.year << endl;

getch();

}

ឧទាហរណ៏ទី៣

// class with data and member function

class date {

private :

int day;

int month;

int year;

public :

void getdata ( int d, int m, int y ) {

day =d;

month =m;

year = y;

}

void display () {

cout << “Today’s date is = “<< day << “ / “ ;

cout << month << “/ “ << year << endl;

}

void main () {

class date today;

int d1, m1,y1;

d1= 15;

m1=10;

y1=2009;

today.getdata( d1, m1, y1);

today.display ();

getch();

}

ឧទាហរណ៏ទី៤

class date {

private :

int day;

int month;

int year;

public :

void getdata ( ) {

cout << “Enter the date ( dd- mm- yy) “<<endl;

cin >> day >> month >> year;

}

void display () {

cout << “Today’s date is = “<< day << “ / “ ;

cout << month << “/ “ << year << endl;

}

void main () {

class date today;

today.getdata( );

today.display ();

getch();

}

ឧទាហរណ៏ទី៥

//member function are defined within the class definition

#include<iostream.h>

#include<conio.h>

class sample {

private :

int x;

int y;

public :

void getinfo () {

cout << “Enter any two numbers ? “ << endl;

cin >> x >> y ;

}

void display () {

cout << “ x = “ << x << endl;

cout << “ y = “ << y << endl;

cout << “ sum = “<< sum() << endl;

cout <<” sub = “<< sub() << endl;

cout << “ mul = “ << mul() << endl;

cout << “ div =” << div() << endl;

}

int sum () {

return (x +y);

}

int sub () {

return (x -y);

}

int mul () {

return (x \*y);

}

int div () {

return (x /y);

}

}; // end of class definition

void main() {

sample obj1;

obj1.getinfo ();

obj1.display ();

obj1.sum ();

obj1.sub ();

obj1.mul ();

obj1.div ();

getch();

}

ឧទាហរណ៏ទី៦

#include<iostream.h>

#include<conio.h>

class sample {

private :

int x;

int y;

public :

void getinfo();

void display();

int sum();

int sub();

int mul();

int div();

}; // endl class

void sample::getinfo () {

cout << “Enter any two numbers ? “ << endl;

cin >> x >> y ;

}

void sample::display () {

cout << “ x = “ << x << endl;

cout << “ y = “ << y << endl;

cout << “ sum = “<< sum() << endl;

cout <<” sub = “<< sub() << endl;

cout << “ mul = “ << mul() << endl;

cout << “ div =” << div() << endl;

}

int sample::sum () {

return (x +y);

}

int sample::sub () {

return (x -y);

}

int sample::mul () {

return (x \*y);

}

int sample::div () {

return (x /y);

}

void main() {

sample obj1;

obj1.getinfo ();

obj1.display ();

obj1.sum ();

obj1.sub ();

obj1.mul ();

obj1.div ();

getch();

}

ឧទាហរណ៏ទី៧

// solution of quadratic equation using oop

#include<iostream.h>

#include<ocnio.h>

class equation {

private:

float a;

float b;

float c;

public:

void getinfo(float a, float b, float c);

void display();

void equal(float a, float b );

void imag();

void real(float a,float b, float det);

};

void equation::display() {

cout<< endl;

cout << “a = “<< a << ‘\t’;

cout << “b = “<< b << ‘\t’;

cout << “c = “<< c << endl;

}

void equation::equal(float a, float b) {

float x;

x = -b/(2\*a);

cout <<”Roots are equal = “<< x << endl;

}

void equation::imag() {

cout<< “Roots are imaginary \n”;

}

void equation::real(float a, float b,float det) {

float x1,x2,temp;

temp = sqrt(det);

x1 = (-b+temp)/(2\*a);

x2 = (-b-temp)/(2\*a);

cout << “Root are real \n”;

cout << “x1 = “ << x1 << endl;

cout << “x2 =” << x2 << endl;

}

void main () {

class equation equ;

float aa , bb , cc;

cout << “ Enter three number \n”;

cin >> aa >> bb >> cc;

equ.getinfo(aa, bb ,cc);

equ.display();

if(aa==0) {

float temp;

temp = cc/bb;

cout << “Linear roots = “ << temp <<endl;

}

else {

float det;

det = bb \*bb - 4\*aa\*cc;

if(det == 0 )

equ.equal(aa,bb);

else if (det < 0)

equ.imag();

else

equ.real(aa,bb,det);

}

getch();

} // end of main program

៧.​ អំពី array នៃ class Object

ការប្រកាស​ array នៃ class objects គឺមានលក្ខណ:ស្រដៀងទៅនឹងការប្រកាស array នៃ structure នៅក្នុង C++ ដែរ។

ទម្រង់ទូទៅរបស់វា​ គឺ

class user\_defined\_name {

private:

data

methods

public:

methods

protected:

data

};

class user\_defined\_name object[Max];

ដែល Max ជាទំហំ array របស់ class objects ។

ឧទាហរណ៏ទី១

const int Max 20;

class employee {

private:

………

………

public:

………

……….

} ;

Class employee obj[Max];

ឧទាហរណ៏ទី២

class library {

private:

……….

public:

……….

} obj [10];

ឧទាហរណ៏ទី៣ :​ កម្មវិធីខាងក្រោមនេះអានតម្លៃជាលក្ខណ:របស់និស្សិត ដូចជា id, name,sex, age,avg ពី keyboard រួចហើយបង្ហាញតម្លៃរបស់ class នៅលើអេក្រង់។ class ឈ្មោះ student ត្រូវបានកំណត់ជា array នៃ class objects។ វាបង្ហាញនូវការបង្កើត array នៃ class objects ហើយនិងរបៀបចូចប្រើ data member និង member functions ទាំងនោះ។

#include<iostream.h>

const int Max 20;

class student {

private:

int id;

char name[20];

char sex;

int age;

float avg;

public:

void getinfo();

void disinfo();

char grade();

}; // end of class

void getinfo() {

cout<<” Enter id =” ; cin >> id;

cout << “Enter name =”; cin.seekg(0);

cin.get(name,20);

cout<< “Enter sex = “ ; cin >> sex;

cout<< “Enter age =” ; cin >> age;

cout<< “Enter avg = “; cin >> avg;

}

void disinfo() {

cout << “ id = “ << id << endl;

cout<< “ Name =” << name<<endl;

cout<< “Sex = “<<sex <<endl;

cout<<” Avg = “<<avg<<endl;

cout<< “Grade = “<< grade()<<endl;

}

char grade() {

char g;

if (avg < 50)

g=’F’;

else if ( avg <65)

g=’E’;

else if ( avg < 75)

g=’D’;

else if ( avg <85)

g=’C’;

else if ( avg <95)

g=’B’;

else

g= ‘A’;

return g;

}

void main () {

student stu[Max];

int n;

cout<< “Enter n =” ; cin >> n;

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

stu[i].getinfo();

cout<< “Contents of class \n”;

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

stu[i].disinfo();

} // end of main

៨. Pointer និង classes

យើងបានសិក្សារួចមកហើយអំពី pointer ដែលមានការប្រើនៅលើផ្នែកផ្សេងៗជាច្រើនដូចជា អញាត្ត function និង​ structure ជាដើម។ នៅក្នុងផ្នែកនេះ យើងនឹងឃើញអំពីការប្រើ pointer ជា object នៃ class មួយ។

ឧទាហរណ៏ទី១

class sample {

private:

int x;

int y;

char s;

public:

void getdata();

void display();

};

Sample \*ptr;

ក្នុងនេះ ptr ជាអញាត្ត pointer ផ្ទុកតម្លៃជាអាស័យដ្ឋាននៃ object របស់ sample class ដែលវាមាន data members ចំនួនបី ដូចជា int x, float y និង​ char s ហើយនិងមាន member functions ដូចជា getdata() និង​ display() ។

ការប្រើ pointer ទៅលើនៃ Class អាចចូលទៅប្រើ និង ដំណើរការតាមទម្រង់ខាងក្រោម

(\*object name).member name=variable;

សញ្ញាវង់ក្រចកជាសញ្ញាសំខាន់ដោយ member របស់ class មានប្រើសញ្ញាចំណុច (.) ដែលមានអទិភាពខ្ពស់ជាង indirection operator(\*)។ រីឯ pointer ទៅលើ member នៃ class មួយអាច ប្រើបានដោយសញ្ញាគំនូស(-) ដែលមានសញ្ញាធំជាង​ (>) នៅពីក្រោយ។

Obect name ->member name = variable;

ករណីទី១:

class student {

private:

…………

…………

public:

………..

………..

}; // end of class

void main() {

student \*ptr;

(\*ptr).data\_member;

(\*ptr).member\_function();

}

ករណីទី២:

class student {

private:

…………

…………

public:

………..

………..

}; // end of class

void main() {

student \*ptr;

\*ptr​->data\_member;

\*ptr->member\_function();

}

ឧទាហរណ៏ទី២

#include<iostream.h>

const int Max 20;

class student {

private:

int id;

char name[20];

char sex;

int age;

float avg;

public:

void getinfo();

void disinfo();

char grade();

}; // end of class

void getinfo() {

cout<<” Enter id =” ; cin >> id;

cout << “Enter name =”; cin.seekg(0);

cin.get(name,20);

cout<< “Enter sex = “ ; cin >> sex;

cout<< “Enter age =” ; cin >> age;

cout<< “Enter avg = “; cin >> avg;

}

void disinfo() {

cout << “ id = “ << id << endl;

cout<< “ Name =” << name<<endl;

cout<< “Sex = “<<sex <<endl;

cout<<” Avg = “<<avg<<endl;

cout<< “Grade = “<< grade()<<endl;

}

char grade() {

char g;

if (avg < 50)

g=’F’;

else if ( avg <65)

g=’E’;

else if ( avg < 75)

g=’D’;

else if ( avg <85)

g=’C’;

else if ( avg <95)

g=’B’;

else

g= ‘A’;

return g;

}

1. void main () {

student \*ptr;

ptr->getinfo();

cout<< “Contents of class \n”;

ptr->disinfo();

} // end of main

1. void main () {

student \*ptr;

(\*ptr).getinfo();

cout << “Contens of class \n”;

(\*ptr).disinfo();

getch();

}

៩. class មួយនៅក្នុង class មួយទៀត (Nested class)

ភាសា c++ អនុញ្ញាតឲ្យ class មួយប្រកាសក្នុង class មួយទៀត។ ឈ្មោះនៃ class ខាងក្នុងមានលក្ខណ: local ។ ទម្រង់ទូទៅនៃការប្រកាស class មួយនៅក្នុង class មួយទៀតគឺ

class outer\_class\_name {

private:

//data

protected:

// data

public:

//methods

class inner\_class\_name {

private:

//data of inner class

public:

//methods of inner class

}; // end of inner class declaration

}; // end of outer class declaration

outer\_class\_name object1;

outer\_class\_name::inner\_class\_name object2;

ចូរកត់សម្គាល់ថាការប្រកាស​ class មួយនៅក្នុង class មួយទៀតមិនមែនមានន័យថា​ class នៅខាងក្រៅបានផ្ទុក object នៃ class ក្នុងឡើយ។

ឧទាហរណ៏ទី១

class student\_info() {

private:

char name[20];

char sex;

int age;

public:

student\_info(char \*na,char s,int a);

void display();

class date {

private:

int day,month,year;

public:

date(int dy,int mh,int yr);

void show\_date();

}; //end of inner class

}; // end of outer class

Member functions នៃ class ក្នុងពុំមានលក្ខណ:ពិសេសចូលប្រើ members នៃ class ក្រៅឡើយ​។​ រីឯ member functions នៃ class ក្រៅក៏ពុំមានលក្ខណ:ពិសេសចូលប្រើ members របស់ class ក្នុងដែរ។

ឧទហរណ៏ទី២

class outer {

int a;

void outer\_funct(int b);

class inner {

int x;

void inner\_funct(int y);

};

};

outer obj1;

outer::inner obj2;

obj1.outer\_funct();

obj2.inner\_funct();

នៅពេលដែល class មួយបានប្រកាសជា member នៃ​ class មួយទៀតនោះវាមានទំហំ scopeត្រឹម class ក្រៅប៉ុណ្ណោះ។ object នៃ class ក្រៅពុំបានផ្ទុក object នៃ class ក្នុងឡើយ។

Member function នៃ class ក្រៅអាចកំណត់បានជា

void outer::outer\_funct(int b){

//methods

}

Member function នៃ class ក្នុងអាចកំណត់ជា

void outer::inner::inner\_funct(int y) {

//methods

}

ការប្រកាសឃ្លាខាងក្រោមនេះពុំមានលក្ខណ:ត្រឹមត្រូវឡើយ ចំពោះការហៅ member function របស់ class ក្នុងមកប្រើ។

outer obj1;

outer::inner obj2;

obj1::obj2::inner\_funct(); // error

ឧទាហរណ៏ទី៤

#include<iostream.h>

#include<conio.h>

#include<string.h>

class student\_info() {

private:

char name[20];

char sex;

int age;

public:

student\_info(char \*na,char s,int a);

void display();

class date {

private:

int day,month,year;

public:

date(int dy,int mh,int yr);

void show\_date();

};

};

student\_info::student\_info(char \*na,char s,int a) {

strcpy(name,na);

sex=s;

age =a;

}

student\_info::date::date(int dy, int mh, int yr) {

day =dy;

moth=mh;

year =yr;

}

void student\_info::display() {

cout<<”===========================”<<endl;

cout<< name << “ “ << ‘\t’ ;

cout<< sex << “ “ << ‘\t’ ;

cout<< age << “ “ << ‘\t’;

}

void student\_info()::date::show\_date() {

cout << day << ‘/’ << month << ‘/’ << year << endl;

cout << “==========================” <<endl;

}

void main() {

student\_info obj1(“Dalya “, ‘F’, 22);

student\_info::date obj2(15, 10, 96);

obj1.display();

obj2.show\_date();

getch();

} // end of program

ឧទាហរណ៏ទី៥

#include<iostream.h>

#include<conio.h>

#include<string.h>

class student\_info() {

private:

char name[20];

char sex;

int age;

public:

void getstudent();

void display();

class date {

private:

int day,month,year;

public:

void getdate();

void show\_date();

};

};

void student\_info::getstudent() {

cout<<” Enter name =”; cin.seekg(0);

cin.get(name,20);

cout<<”Enter sex =”; cin>>sex;

cout<<” Enter age =” ; cin>>age;

}

void student\_info::date::getdate() {

cout << “ Enter day =” ; cin>>day;

cout << “ Enter month=” ; cin>>month;

cout << “ Enter year =” ; cin>>year;

}

void student\_info::display() {

cout<<”===========================”<<endl;

cout<< name << “ “ << ‘\t’ ;

cout<< sex << “ “ << ‘\t’ ;

cout<< age << “ “ << ‘\t’;

}

void student\_info()::date::show\_date() {

cout << day << ‘/’ << month << ‘/’ << year << endl;

cout << “==========================” <<endl;

}

void main() {

student\_info obj1[10];

student\_info::date obj2[10];

int i,n;

cout << “How mony students ? \n”;

cin >> n;

cout << “ Enter the following information \n”;

for ( i = 0; i<= n-1; i++) {

int j = i + 1;

cout << “ \n Object : “ << j << endl;

obj1[i].getstudent();

obj2[i].getdate();

}

cout << “ student name sex age date of birth “;

cout<< “ ===========================” ;

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

int j = i+1;

cout << “ \n Object : “ << j << endl;

obj1[i].display();

obj2[i].show\_date();

} // end of main program

១០.អំពី Constructors

Constructor គឺជា member function ពិសេសសម្រាប់កំណត់តម្លៃដំបូងទៅឲ្យ object ដោយស្វ័យប្រវត្ត។​ នៅពេលដែល object មួយបានកកើតឡើងនោះ constructor នឹងចាប់ផ្តើម ប្រតិបត្តិដោយស្វ័យប្រវត្ត។ Constructor function គឺខុសពី member functions ដទៃទៀតដែល មានលក្ខណ:ជា non\_static នៅក្នុង class មួយ ព្រោះវាត្រូវបានប្រើសម្រាប់កំណត់តម្លៃដំបូង អោយអញ្ញាត ពេលដែល object បានកកើតឡើង។ គួរកត់សម្គាល់ថា constructor function អាចមានលក្ខណ: overloaded ដើម្បីគេអាចប្រើបានច្រើននួវទម្រង់នៃការកំណត់តម្លៃដំបូង។

គោលការណ៏សម្រាប់សរសេរ constructor function មួយមានដូចខាងក្រោម

+ ឈ្មោះ constructor ត្រូវតែដូចគ្នាទៅនឹងឈ្មោះ class របស់វា

+ វាប្រកាសឡើងដោយគ្មាន return type (ហើយក៏មិនប្រើ void ​ដែរ)

+ វាមិនអាចជា static

+ វាមិនអាចជា virtual

+ វាអាចបញ្ជាក់កម្រិតចូលប្រើបានគឺ public or protected នៅក្នុង class ហើយនៅក្នុងកាល:ទេស:ខ្លះក៏អាចប្រកាសជា​ private ដែរ​ តែជាករណីកម្រមាន។

ទម្រង់ទូទៅនៃការប្រកាស​​ constructor function មួយ

class user\_name {

private:

…………

…………

public:

user\_name(); // constructor

…………

…………

protected:

……………

……………

};

user\_name::user\_name() {

……………

}

ឧទាហរណ៏ទី១

class employee {

private:

char name[20];

int code;

char add[20];

public:

employee(); //constructor

void getdata();

void display();

};

employee::employee() {

………….

………….

}

ឧទាហរណ៏ទី២

class account {

private:

float balance;

float rate;

public:

account() { // constructor

…………..

}

void create\_acct();

};

Constructor and Destructor អាចហៅមកប្រើដោយបញ្ញាក់ឲ្យបានច្បាស់លាស់។ constructor ត្រូវបានបង្កើតឡើងដោយស្វ័យប្រវត្តនៅពេល object មួយចាប់ផ្តើមកកើត។ Constructor function មួយត្រូវបានបង្កើតឡើងស្វ័យប្រវត្តដោយ c++ compiler តាមលក្ខណ:ដូចខាងក្រោម

+ constructor ត្រូវបានហៅមកប្រើមុន main() ចាប់ផ្តើមដំណើរការប្រតិបត្តិ។

+ នៅពេល object ត្រូវបានបង្កើតឡើងតាមវិធីមួយក្នុងចំណោមវិធីដែលបានបញ្ញាក់ខាងក្រោមនេះ

* ជាអញ្ញាតប្រភេទ globle
* ជាអញ្ញាតប្រភេទ​ local
* ជាអញ្ញាតប្រភេទ static

+ អញ្ញាតប្រភេទ auto នៃ class ត្រូវកំណត់ក្នុង block និងទីតាំងនៃការកំណត់លក្ខណ: របស់វាបានដំណើរការមកដល់។

+ instance នៃ​ class មួយត្រូវបានបង្កើតឡើងជាបណ្តោះអាសន្ន។

+ នៅពេលប្រើ new operator

ឧទាហរណ៏ទី១

class account {

private:

float balance;

float rate;

public:

account() { // constructor

…………..

}

Void create\_acct();

};

void account::account () { // error

…………….

}

ឧទាហរណ៏ទី២

#include<iostream.h>

class Fibonacci {

private:

long int f0, f1, f2, fib;

public:

fibonacci () { // constructor

f0 = 0;

f1 = 1;

fib = f0 + f1;

}

void increment () {

f0 = f1;

f1 = fib;

fib = f0 + f1;

}

void display () {

cout << fib << “ “ ;

}

};

void main() {

fibonacci number;

for (int i = 0; i < =10; i++ ) {

number.display() ;

number.increment();

}

} // end of main program

ឧទាហរណ៏ទី៣

#include<iostream.h>

class Fibonacci {

private:

long int f0, f1, f2, fib;

public:

fibonacci ();

void increment ();

void display ();

}; // end of class

fibnacci::fibonacci () { // constructor

f0 = 0;

f1 = 1;

fib = f0 + f1;

}

void fibonacci::increment () {

f0 = f1;

f1 = fib;

fib = f0 + f1;

}

void fibonacci::display () {

cout << fib << “ “ ;

}

void main() {

fibonacci number;

for (int i = 0; i < =10; i++ ) {

number.display() ;

number.increment();

}

} // end of main program

ឧទាហរណ៏ទី៤​ កម្មវិធីខាងក្រោមនេះមានការប្រើប្រព័ន្ធធនាគារកំណត់នូវប្រាក់កក់ដំបូងនិង​ តម្លៃការប្រាក់ដែលត្រូវបញ្ចូលតាម keyboard ហើយតម្លៃទាំងនេះត្រូវកំណត់រតាមរយ:ការប្រើ constructor function។​ កម្មវិធីនេះមាន methods ដូចខាងក្រោម

#include<iostream.h>

#include<conio.h>

class account {

private:

float balance;

flaot rate;

public:

account(); // constructor

void deposit();

void withdraw();

void getBalance();

void compound();

void menu();

}; // end of class

account::account() {

cout<<”Enter the initial balance\n”;

cin >> balance;

cout <<” Interest rate \n”;

cin >> rate;

}

void account ::deposit() {

float amount;

cout << “Enter the amount =”;

cin >> amount;

balance = balance+amount;

}

void account:: withdraw() {

float amount;

cout << “How much to withdraw?\n”;

cin >>amount;

if ( amount <=balance) {

balance= balance-amount;

cout << “ Amount drawn = “ << amount << endl;

cout << “ Current balance = “ << balance << endl;

} else

cout << o;

}

void account::compound() {

float interest;

interest = balance\*rate;

balance = balance+interest;

cout << “Interest amount = “<< interest <<endl;

cout << “ Total amount = “ << balance << endl;

}

void account ::getbalance() {

cout << “ d -> deposit “ << endl;

cout << “ w -> withdraw “ << endl;

cout << “ c -> compound “ << endl;

cout << “ g -> get balance “ << endl;

cout << “ q -> quit “ << endl;

cout << “ option, please ?\n” ;

}

void main() {

class account acct;

char ch;

acct.menu();

while((ch = getchar()) !=’q’) {

switch (ch) {

case ‘d’: acct.deposit(); bread;

case ‘w’: acct.withdraw(); break;

case ‘c’: acct.compound(); break;

case ‘g’: acct.getbalance(); break;

}

}

getch();

}

១០.១ អំពី Copy Constructors

Copy Constructors ត្រូវបានគេប្រើនៅពេលដែល compiler បង្កើត object នៃ class មួយជាបណ្តោះអាសន្ន។ Copy Constructors ត្រូវបានប្រើនៅពេល

+ ការកំណត់តម្លៃដំបូងរបស់ object មួយតាមរយ: object មួយផ្សេងទៀតនៃ class តែមួយប្រភេទ

+ ឲ្យតម្លៃជា​ object ដូចជាតម្លៃ function

+ ការប្រើ object ជាតម្លៃ parameters នៃ​ function មួយ

ទម្រង់ទូទៅនៃ copy constructor គឺ

class\_name::class\_name(class\_name &ptr)

ការតាងនិមិត្តសញ្ញានៃទម្រង់ខាងលើនេះ​ គឺ

X::X(X &ptr)

ដែល X ជាឈ្មោះ Class កំណត់ដោយអ្នកប្រើប្រាស់ ហើយ ptr គឺជា pointer សម្រាប់ object នៃ X។

Copy constructor អាចប្រើបានតាមទម្រង់ខាងក្រោមដោយប្រើពាក្យ const ។

Class\_name::class\_name(const class\_name & ptr)

ការតាងនិមិត្តសញ្ញានៃទម្រង់ខាងលើនេះ​ គឺ

X::X(const X &ptr)