UNIVERSITY
OF WOLLONGONG
AUSTRALIA

CSCI251
Advanced Programming
OOP in C++

Outline

Object Oriented Programming in C++

Copy operations

1. Copy constructor
2. Copy assignment

Friend Functions

3. What is friend function?
4. Practice 1

this Pointer

5. What is this Pointer?
6. Practice 2

Overloading Operator

7. What is Operator Overloaded
8. Practice 3

Static Member

9. What is static member?

Pointer to a Class

Copy Constructor

```
X (const X &cp);
```

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to

- Initialize one object from another of the same type.
- Pass the object to be copied as an argument to a function.

Code – Demo

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

class Cat
{
private:
   string name;
   string breed;
   int age;
   static constexpr double licenseFee = 10;

public:
   Cat();
   ~Cat();
   Cat(string name, string breed, int age);
   Cat(const Cat &);
   void setCatData(string, string, int);
   void showCat();
   void printAddresses();
   static const void showFee();
};
//const double Cat::licenseFee = 10;
```

```cpp
//const double Cat::licenseFee = 10;
Cat::~Cat(){
    cout << "Destructor remove memory"<< endl;
}
Cat::Cat(string _name, string _breed, int _age){
    name = _name;
    breed = _breed;
    age = _age;
}
Cat::Cat(const Cat & copyCat){
    name = copyCat.name;
    breed = copyCat.breed;
    age = copyCat.age;
}
void Cat::setCatData(string catName, string catBreed, int catAge
)
{
   name = catName;
   breed = catBreed;
   age = catAge;
}
void Cat::showCat()
{
  cout << "Cat: " << name << " is a " << breed << endl;
  cout << "The cat's age is " << age << endl;
  cout << "License fee: $" << licenseFee << endl;
}
```

Code – Demo

```
Cat: Tigger is a Fluffy unit
The cat's age is 3
License fee: $10
-------------
Cat: Tigger is a Fluffy unit
The cat's age is 3
License fee: $10
--------------
Cat     :0x7ffef2dc3120
Name    :0x7ffef2dc3120
Breed   :0x7ffef2dc3140
Age     :0x7ffef2dc3160
--------------
Cat     :0x7ffef2dc3098
Name    :0x7ffef2dc3098
Breed   :0x7ffef2dc30b8
Age     :0x7ffef2dc30d8
---------------
Fee     :10
----------------
Destructor remove memory
Destructor remove memory
```

```cpp
void Cat::printAddresses(){
    cout << "Cat     :" << this << endl;
    cout << "Name    :" << &name << endl;
    cout << "Breed   :" << &breed << endl;
    cout << "Age     :" << &age << endl;
}

const void Cat::showFee(){
    cout << "Fee     :" << licenseFee << endl;
}

int main()
{
     Cat myCat;
     myCat.setCatData("Tigger", "Fluffy unit", 3)
;
     //Cat secondCat(myCat);
     Cat secondCat = myCat;
     myCat.showCat();
     secondCat.showCat();
     myCat.printAddresses();
     secondCat.printAddresses();
     Cat::showFee();
     return 0;
}
```

Copy Assignment

```
X& operator=(const X &cp);
```

The copy assignment which assigns values of one object to another existing object.

```
Cat& operator=(const Cat &cp){
        return *this;}
```

```
Cat cat2;
cat2=cat1;
```

Practice 0

2. Friend Function

What is Friend Function?

- A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.
- Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

What is Friend Function?

- A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.
- To declare a function as a friend of a class, precede the function prototype in the class definition with keyword friend.

How to use Friend Function.

- Syntax is:

```
friend return_type function_name(Class & objClass){…}
```

  - Declare inside the class then define friend function outside the class

Code – Demo

```
The number is: 8
--------------
The number is: 14
```

```cpp
#include <iostream>
using namespace std;
class Number{
    private:
        int num;
    public:
        Number(){num=8;};
        void printNum(){
            cout<<"The number is: "<<num<<endl;
        }
        friend void addNum(Number &n); //Pass by ?
};
void addNum(Number &n){
    n.num = n.num + 6;
}
int main(){
    Number n1;
    n1.printNum();
    cout<<"--------------"<<endl;
    // Call friend function
    addNum(n1);
    n1.printNum();
    return 0;
}
```

Practice 1: friend Function

- Do in Code:Block
- Copy the code from lecture
- Define the friend function minusNum() that decrease the num by 2
  - n.num = n.num - 2
- In main() function:
  - After calling friend addNum(n1) and printing the num
  - Call the friend function minusNum(n1)
  - Access to the method printNum()
- Explain why we have the new number

Code – Demo

```cpp
#include <iostream>
#include "Library.h"
using namespace std;

Money::Money(string currency, double numbe
r, double rate):country(currency), amount(
number), exchangeRate(rate){}

City::City(string cityName, string country
Name, double NorthSouth, double EastWest):
name(cityName), country(countryName), lati
tude(NorthSouth), longitude(EastWest){}

bool compare(Money& m, City& c)
{
  return (m.country == c.country);
}
```

```cpp
#include <iostream>
class City;
class Money
{
  friend bool compare(Money&, City&);
  private:
    string country;
    double amount;
    double exchangeRate;
  public:
    Money(string, double, double);
};
class City
{
  friend bool compare(Money&, City&);
  private:
    string name, country;
    double latitude, longitude;
  public:
    City(string, string, double, double);
};
```

Code – Demo

```
#include <iostream>
#include "Library.h"
using namespace std;

int main()
{
  Money nzMoney("NZ", 1000, 0.70);

  City C1("Sydney", "Australia", -33.53, 151.10);
  City C2(“Auckland", “NZ", -43.33, 172.47);

  if (compare(nzMoney, C1))
  cout << "Different associated countries." << endl;
  else
  cout << "They are associated with the same country." << endl;

  if (compare(nzMoney, C2))
  cout << "Different associated countries." << endl;
  else
  cout << "They are associated with the same country." << endl;
}
```

3. “this” Pointer

What is “this”?

Concept: Object in C++ has access to its own address through an important pointer called `this` pointer. The `this` pointer is an implicit parameter to all member functions.

Important note: Friend functions do not have a `this` pointer, because friends are not members of a class. Only member functions have a `this` pointer.

How to use “this” Pointer

- Syntax is:

```
this -> class_component
```

How to use "`this`" Pointer

Demo:

- Create a class Square
  - Attribute: the length of square
  - Method: area of square

Do:

Create a function that compares the area between two square using `this` Pointer

Code – Demo

We can compare by the length of the square. But for the demo, we add function area

```
Constructor created.
Constructor created.
Two are NOT equal
```

```cpp
#include <iostream>
using namespace std;
class Square {
    private:
        int length;     // Length of a square
    public:
    // Constructor non-default
    Square(int l) {
        cout <<"Constructor created." << endl;
        length = l;
    }
    int area() {
        return length * length;
    }
    bool compare(Square sq) {
        return this->area() == sq.area();
    }
};
int main() {
    Square Square1(3);    // Create Square1
    Square Square2(4);    // Create Square2
    if(Square1.compare(Square2)) {
        cout << "Two are equal" <<endl;
    } else {
        cout << "Two are NOT equal" <<endl;
    }
    return 0;
}
```

Practice 2: `this` Pointer

- Do in Code:Block
- Copy the code from lecture
- Create a function to add the areas of two square
  - Int addArea(Square sq)
- In main() function:
  - Create two objects from class Square with constructor
  - Print to screen the total area of two squares

4. Overloading Operator

What is “Operator Overloading”?

Concept: Overloaded operators are functions with special names: the keyword "`operator`" followed by the symbol for the operator being defined.

Like any other function, an overloaded operator has a return type and a parameter list.

How to use “Operator Overloading”

- Syntax is:

```
Class operatorsymbol”+” (const Class & obj){}
```

How to use “Operator Overloading”

Demo:

- Create a class Distance
  - Attribute: the length
  - Method: setter and getter

Do:

Create a Operator+ overloading function to add two objects. In the main, create three objects and call operator overloading

Code – Demo

```
Distance 1 is: 3
Distance 2 is: 4
----------------
The total of 2 distance is: 7
```

```cpp
#include <iostream>
using namespace std;
class Distance {
    private: int length;     // Length of a square
    public:
    void setLength(int l){length = l;}
    int getLength()const {return length;}
    Distance operator+(const Distance d) {
        Distance dTmp; // Create of temporary Distance
        dTmp.length = this->length + d.length;
        return dTmp;
    }
};
int main() {
    Distance d1;    // Create Distance 1
    Distance d2;    // Create Distance 2
    Distance d3;
    d1.setLength(3);
    d2.setLength(4);
    cout << "Distance 1 is: "<<d1.getLength()<<endl;
    cout << "Distance 2 is: "<<d2.getLength()<<endl;
    // Add two objects
    d3 = d1 + d2;
    cout <<"----------------"<<endl;
    cout << "The total of 2 distance is: "<<d3.getLength()<<e
ndl;
    return 0;
}
```

Practice 3: operator Overloading

- Do in Code:Block
- Copy the code from lecture
- Create an operator- to minus two Distance object
  - Distance operator- (const Distance d)
- In main() function:
  - Create three objects from class Distance
  - Implement over the object to compute the deduction of two distance objects

List of operators which can be overloaded

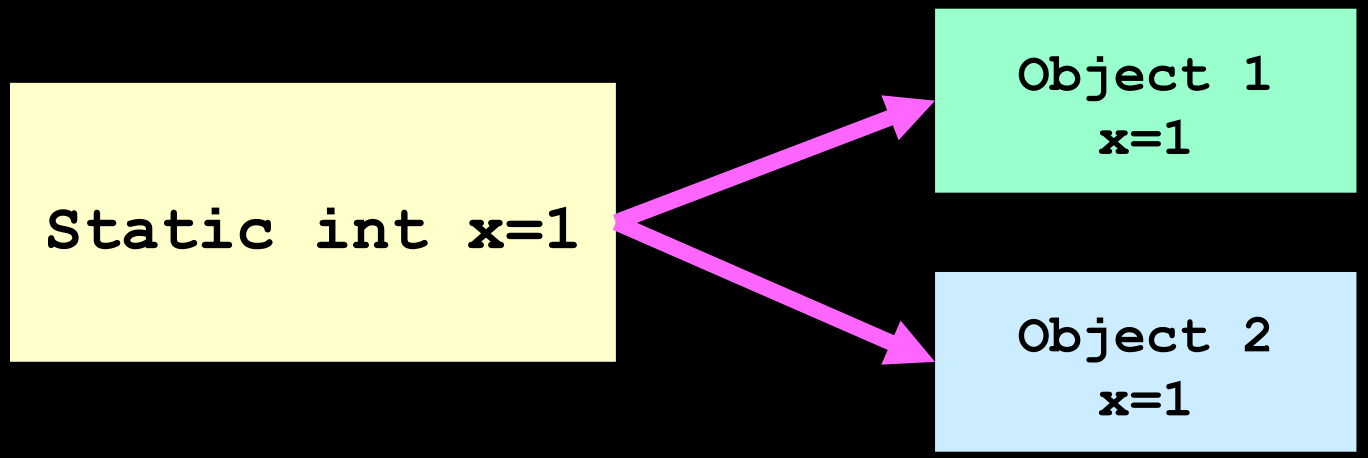
+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

https://www.tutorialspoint.com/cplusplus/cpp_overloading.htm

5. Static Members

Static Data vs Function Member
What is Static Members?

Concept: When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.



Static Data vs Function Member
What is Static Members?

- A static member is shared by all objects of the class
- We cannot put in the class definition but it can be initialized outside the class using scope resolution (::) to identify which class it belongs.

How to use “static data member”

- Inside the class A

```
static type variable;
```

- Outside the class

```
type A::variable=init;
```

How to use “static data member”

Demo:

- Create a class A
  - Declare static data member (count) in public modifier
  - Create a default constructor
  - Define static data member outside the class

Do:

In the main function, generate a number of constructors then count the constructor created by using static data member

Code – Demo

```
#include <iostream>
using namespace std;
class A {
    public:
    static int count;
    A(){
        cout << "Constructor is created."<<endl;
        count++;
    }
};
// Initialising static data
int A::count=0;
int main() {
    A a1, a2;
    cout <<"Number of create constructors: "<<A::coun
t;
    return 0;
}
```

```
Constructor is created.
Constructor is created.
Number of create constructors: 2
```

Static Function Member

- By declaring a function member as static, you make it independent of any particular object of the class. A static member function can be called even if no objects of the class exist and the static functions are accessed using only the class name and the scope resolution operator ::
- A static member function can only access static data member, other static member functions and any other functions from outside the class.
- Static member functions have a class scope and they do not have access to the this pointer of the class.

Code – Demo

```
Init value: 0
Number of create constructors: 2
```

Practice 4

```cpp
#include <iostream>
using namespace std;
class A {
    private:
    int x;
    static int count;
    public:
    A(){
        count++;
    }
    static int getCount(){
        return count;
    }
};
// Initialising static data
int A::count=0;
int main() {
    cout<<"Init value: "<<A::getCount()<<endl;
    A a1, a2;
    cout <<"Number of create constructors: "<<A::getCount()
;
    return 0;
}
```

6. Pointer to a Class

What is Pointer to a Class?

A pointer to a C++ class is done exactly the same way as a pointer to a structure and to access members of a pointer to a class you use the member access operator -> operator, just as you do with pointers to structures. Also as with all pointers, you must initialize the pointer before using it.

Code – Demo

```
Constructor created.
Constructor created.
the area of square 1 is: 9
the area of square 2 is: 16
```

```cpp
#include <iostream>
using namespace std;
class Square {
    private:
        int length;     // Length of a square
    public:
      // Constructor non-default
    Square(int l) {
        cout <<"Constructor created." << endl;
        length = l;
    }
    int area() {
        return length * length;
    }
};
int main() {
    Square square1(3);    // Declare Square1
    Square square2(4);    // Declare Square2
    Square *ptr;
    ptr = &square1;
    cout<<"the area of square 1 is: "<<ptr->area()<<endl;
    ptr = &square2;
    cout<<"the area of square 2 is: "<<ptr->area();
    return 0;
}
```