Part1: “**this” pointer**

A special pointer called this pointer can be employed in C++ programs. When the object needs to point to itself then this pointer is used. Remember this pointer points (or represents the address of the) object of the class but not the class.

Example

class student:public person

{

private:

......

public:

......

void printAddress(void)

{

cout<<"I am from within the object and my address is"<<this;

}

};

Example:

For any class you created in the previous labsheets, add the following member function:

void printobj()

{

cout << "the addtress of object calling this method" << this<<endl;

}

Use the main function in your code to call printobj() method.

Output:

|  |
| --- |
|  |

Another way to use ‘this’ pointer.

void setd(int f)

{

this->d= f;

}

Part2: **copy constructor**

We already know that no argument constructor can initialize data members to some specific values, and parameterized constructor can initialize data members to values passed as arguments. There is also another way to initialize an object with another object of the same type. It is called the *copy constructor*. It is a one-argument constructor whose argument is an object of the same class. Here a full example:

#include <iostream>

using namespace std;

class mk

{

private:

int d;

public:

mk()

{

d = 0;

cout << "I am the defualt constructor" << endl;

}

mk(int j)

{

d = j;

cout << "I am the paramterise constructor" << endl;

}

void getd()

{

cout << "d=" << d << endl;

}

void setd(int f)

{

d = f;

}

};

int main()

{

mk ob1;

mk ob2(10);

mk ob3 = ob2; //copy the ob2

mk ob4(ob2); //another way to copy an object

ob2.getd();

ob3.getd();

ob4.getd();

ob4.setd(50);

ob2.getd();

ob4.getd();

return 0;

}

**What is the output:**

|  |
| --- |
|  |

**Create an object of mk class (ob5) ising the default constructor. And, copy your object to a new object (ob6).**

**Using setd() method, update the d member variable to 80;**

**Print out the value of d for both ob5 and ob6.**

|  |
| --- |
|  |

Part3: **passing objects as function arguments**

In C++ we can pass class’s objects as arguments and also return them from a function the same way we pass and return other variables. No special keyword or header file is required to do so.

To pass an object as an argument we write the object name as the argument while calling the function the same way we do it for other variables.

function\_name(object\_name);

Example

Proceeding to mk class in the previous part, add the following method to client code, not in the class:

void printobj2(mk r)

{

r.getd();

}

And, inside the main function, add the following code:

mk ob1;

ob1.setd(30);

printobj2(ob1);

output:

|  |
| --- |
|  |

and to return the object as argument, use the following syntax

Class\_name function\_name(parameters)

{

Return object\_name;

}

Example:

Add the following method to your code

mk updateobj()

{

mk op(35);

return op;

}

And to use this type of methods, you need to create an object in your main function and assign the updateobj() to your function, as following

mk hj = updateobj();

output:

|  |
| --- |
|  |

That is the end of labsheet.. Good Luck