**Pointers To Class Members:**

**Writer:** Sayak Haldar

**Theory:**

Pointers to members allow you to refer to non-static members of class objects. You cannot use a pointer to member to point to a static class member because the address of a static member is not associated with any particular object. To point to a static class member, you must use a normal pointer.

You can use pointers to member functions in the same manner as pointers to functions. You can compare pointers to member functions, assign values to them, and use them to call member functions. Note that a member function does not have the same type as a nonmember function that has the same number and type of arguments and the same return type.

Pointers to members can be declared and used as shown in the following example:

This is an example:

**#include <iostream>**

**using namespace std;**

**class X {**

**public:**

**int a;**

**void f(int b) {**

**cout << "The value of b is "<< b << endl;**

**}**

**};**

**int main() {**

**// declare pointer to data member**

**int X::\*ptiptr = &X::a;**

**// declare a pointer to member function**

**void (X::\* ptfptr) (int) = &X::f;**

**// create an object of class type X**

**X xobject;**

**// initialize data member**

**xobject.\*ptiptr = 10;**

**cout << "The value of a is " << xobject.\*ptiptr << endl;**

**// call member function**

**(xobject.\*ptfptr) (20);**

**}**

**MCQ Questions:**

**1. Which is referred by pointers to member?**

a) static members of class objects

b) Non-static members of class objects

c) Referring to whole class

d) None of the mentioned

**Answer: b**

Explanation: We cannot use a pointer to member to point to a static class member because the address of a static member is not associated with any particular object.

**2. What should be used to point to a static class member?**

a) Smart pointer

b) Dynamic pointer

c) Normal pointer

d) None of the mentioned

**Answer: c**

**3. Which operator is used in pointer to member function?**

a) .\*

b) ->\*

c) Both .\* & ->\*

d) None of the mentioned

**Answer: c**

Explanation: The pointer to member operators .\* and ->\* are used to bind a pointer to a member of a specific class object.

**#include <iostream>**

**using namespace std;**

**class X**

**{**

**public:**

**int a;**

**void f(int b)**

**{**

**cout<< b << endl;**

**}**

**};**

**int main()**

**{**

**int X :: \*ptiptr = &X :: a;**

**X xobject;**

**xobject.\*ptiptr = 10;**

**cout << xobject.\*ptiptr << endl;**

**return 0;**

**}**

Here, .\* is used.

Now, if we define a pointer to an object of the class, then ->\*would be used.

**4. What is the output of this program?**

#include <iostream>

using namespace std;

class X

{

public:

int a;

void f(int b)

{

cout<< b << endl;

}

};

int main()

{

int X :: \*ptiptr = &X :: a;

void (X :: \* ptfptr) (int) = &X :: f;

X xobject;

xobject.\*ptiptr = 10;

cout << xobject.\*ptiptr << endl;

(xobject.\*ptfptr) (20);

}

a) 10

20

b) 20

10

c) 20

d) 10

Answer: a

#include <iostream>

using namespace std;

class X

{

public:

int a;

void f(int b)

{

cout<< b << endl;

}

};

int main()

{

int X :: \*ptiptr = &X :: a;

**//now, we define a pointer to the member a of the class X.**

void (X :: \* ptfptr) (int) = &X :: f;

**//we define a function pointer to the function f of the class X**

X xobject;

xobject.\*ptiptr = 10;

**//Now, a have the value 10**

cout << xobject.\*ptiptr << endl;

(xobject.\*ptfptr) (20);

**//Now, we call the function**

return 0;

}

Now, one more thing:

**#include<iostream>**

**using namespace std;**

**void f(int b)**

**{**

**cout<<b<<endl;**

**}**

**int main()**

**{**

**void (\*func\_ptr)(int)=&f;**

**func\_ptr(20);**

**return 0;**

**}**

You can have the same code written as:

**#include<iostream>**

**using namespace std;**

**void f(int b)**

**{**

**cout<<b<<endl;**

**}**

**int main()**

**{**

**void (\*func\_ptr)(int)=f;**

**func\_ptr(20);**

**return 0;**

**}**

However, here you cannot replace the given code:

**#include <iostream>**

**using namespace std;**

**class X**

**{**

**public:**

**int a;**

**void f(int b)**

**{**

**cout<< b << endl;**

**}**

**};**

**int main()**

**{**

**int X :: \*ptiptr = &X :: a;**

**void (X :: \* ptfptr) (int) = &X :: f;**

**X xobject;**

**xobject.\*ptiptr = 10;**

**cout << xobject.\*ptiptr << endl;**

**(xobject.\*ptfptr) (20);**

**}**

With

**#include <iostream>**

**using namespace std;**

**class X**

**{**

**public:**

**int a;**

**void f(int b)**

**{**

**cout<< b << endl;**

**}**

**};**

**int main()**

**{**

**int X :: \*ptiptr = &X :: a;**

**void (X :: \* ptfptr) (int) = X :: f;**

**X xobject;**

**xobject.\*ptiptr = 10;**

**cout << xobject.\*ptiptr << endl;**

**(xobject.\*ptfptr) (20);**

**}**

It will generate the following error: classKnowledge7.cpp:15: error: invalid use of non-static member function ‘void X::f(int)’

**5. What is the output of this program?**

#include <iostream>

using namespace std;

class Testpm

{

public:

void m\_func1()

{

cout << "func1\n";

}

int m\_num;

};

void (Testpm :: \*pmfn)() = &Testpm :: m\_func1;

int Testpm :: \*pmd = &Testpm :: m\_num;

int main()

{

Testpm ATestpm;

Testpm \*pTestpm = new Testpm;

(ATestpm.\*pmfn)();

(pTestpm ->\* pmfn)();

ATestpm.\*pmd = 1;

pTestpm ->\* pmd = 2;

cout << ATestpm.\*pmd << endl

<< pTestpm ->\* pmd << endl;

}

a) func1

b) func1

func1

c) 1

2

d) func1

func1

1

2

**Answer) d**

Do you need any explanation?

**Check the examples provided in the solution of question of 3.**

**6. What is the output of this program?**

#include <iostream>

using namespace std;

class Car

{

public:

int speed;

};

int main()

{

int Car :: \*pSpeed = &Car :: speed;

Car c1;

c1.speed = 1;

cout << c1.speed << endl;

c1.\*pSpeed = 2;

cout << c1.speed << endl;

return 0;

}

a) 1

b) 2

c) Both 1 & 2

d) None of the mentioned

**Answer: c**

Explanation: In this program, We are printing the value by direct access and another one by using pointer to member.

**7. What is the output of this program?**

#include <iostream>

using namespace std;

class bowl

{

public:

int apples;

int oranges;

};

int count\_fruit(bowl \* begin, bowl \* end, int bowl :: \*fruit)

{

int count = 0;

for (bowl \* iterator = begin; iterator != end; ++ iterator)

count += iterator ->\* fruit;

return count;

}

int main()

{

bowl bowls[2] = {{ 1, 2 },{ 3, 5 }};

cout << "I have " << count\_fruit(bowls, bowls + 2, & bowl :: apples) << " apples\n";

cout << "I have " << count\_fruit(bowls, bowls + 2, & bowl :: oranges) << " oranges\n";

return 0;

}

a) I have 4 apples

I have 7 oranges

b) I have 3 apples

I have 5 oranges

c) I have 1 apples

I have 5 oranges

d) None of the mentioned

View Answer

**Answer: a**

Explanation: In this program, We are passing the value to the class and adding the values and printing it in the main.

What’s more important about this program is that:

You can do:

**#include <iostream>**

**using namespace std;**

**class bowl**

**{**

**public:**

**int apples;**

**int oranges;**

**void display\_bowl\_stats()**

**{**

**cout<<"Apples: "<<apples<<endl;**

**cout<<"Oranges: "<<oranges<<endl;**

**}**

**};**

**int main()**

**{**

**bowl bowls[1] = {{ 1, 2 }};**

**bowls[0].display\_bowl\_stats();**

**return 0;**

**}**

This will print:

Apples: 1

Oranges: 2

But, we cannot do:

**#include <iostream>**

**using namespace std;**

**class bowl**

**{**

**public:**

**int apples;**

**int oranges;**

**void display\_bowl\_stats()**

**{**

**cout<<"Apples: "<<apples<<endl;**

**cout<<"Oranges: "<<oranges<<endl;**

**}**

**};**

**int main()**

**{**

**bowl fruit\_bowl(1,2);;**

**Fruit\_bowl.display\_bowl\_stats();**

**return 0;**

**}**

Why?

**8. What is the output of this program?**

#include <iostream>

using namespace std;

class Foo

{

public:

Foo(int i = 0){ \_i = i;}

void f()

{

cout << "Executed"<<endl;

}

private:

int \_i;

};

int main()

{

Foo \*p = 0;

p -> f();

}

a) Executed

b) Error

c) Runtime error

d) None of the mentioned

**Answer: a**

**9. Which is the best design choice for using pointer to member function?**

a) Interface

b) Class

c) Structure

d) None of the mentioned

**Answer: a**

**10. What is the operation for .\*?**

a) It combines the first operand and the second operand

b) It separates the first operand and the second operand

c) It reduces the data size

d) None of the mentioned

**Answer: a**

Explanation: The binary operator .\* combines its first operand, which must be an object of class type, with its second operand, which must be a pointer-to-member type.