**Worksheet - 3**

**Functions / Class / Constructor / Destructor**

Predict the output of following programs. If there are compilation errors, then fix them.

1. #include<iostream>

using namespace std;

  class **Empty** {};

  int main()

{

   cout << sizeof(Empty);

   return 0;

}

Ans) 1 byte because size of class cannot be zero and is usually 1 byte.

1. #include<iostream>

using namespace std;

class Empty { };

int main()

{

Empty a, b;

if (&a == &b)

cout << "impossible " << endl;

else

cout << "Fine " << endl;

return 0;

}

Ans) Fine because address of can never be equal to address of b.

1. class Base {

public:

Base(int a) : member(a)

{

cout << "Base constructor called with " << a << endl;

}

private:

int member;

};

void test(Base obj1)

{

cout << "Base object's member = " << obj1.member;

}

int main()

{

test(333);

}

Ans) error because object members are private and hence here int member is private so obj1.member cant be executed

**Static members**

4.#include<iostream>

class Test {

   static void fun() const {

     return;

   }

};

int main()

{

   getchar();

   return 0;

}

Ans) A static member function cannot be declared as constant. So it is an error.

1. #include <iostream>

using namespace std;

class A

{

public:

    A() { cout << "A's Constructor Called " << endl;  }

};

class B

{

    static A a;

public:

    B() { cout << "B's Constructor Called " << endl; }

};

int main()

{

    B b;

    return 0;

}

Ans) B's constructor called because static function is only declared in class definition.

Inorder to be defined it has to be done explicitly.

1. #include <iostream>

using namespace std;

class A

{

    int x;

public:

    A() { cout << "A's constructor called " << endl;  }

};

class B

{

    static A a;

public:

    B() { cout << "B's constructor called " << endl; }

    static A getA() { return a; }

};

A B::a;

int main()

{

    B b1, b2, b3;

    A a = b1.getA();

    return 0;

}

Ans) A's constructor called

B's constructor called

B's constructor called

B's constructor called

Here, variable a is created in B whose data type is A and before main this gets implemented hence A’s constructor called is executed first and also static data member gets memory before itself.

**“this” pointer**

7.#include<iostream>

using namespace std;

class Test

{

private:

   int x;

public:

   void setX (int x)

   {

       this->x = x;

   }

   void print() { cout << "x = " << x ; }

};

int main()

{

   Test obj;

   int x = 20;

   obj.setX(x);

   obj.print();

   return 0;

}

8.#include<iostream>

using namespace std;

class Test

{

private:

int x;

int y;

public:

Test(int x = 0, int y = 0) { this->x = x; this->y = y; }

Test &setX(int a) { x = a; return \*this; }

Test &setY(int b) { y = b; return \*this; }

void print() { cout << "x = " << x << " y = " << y << endl; }

};

int main()

{

Test obj1(5, 5);

obj1.setX(10).setY(20);

obj1.print();

return 0;

}

Ans)x=10 y=20

|  |
| --- |
| 1. #include<iostream>   using namespace std;    class Test  {  private:    int x;  public:    Test(int x = 0) { this->x = x; }    void change(Test \*t) { this = t; }    void print() { cout << "x = " << x << endl; }  };    int main()  {    Test obj(5);    Test \*ptr = new Test (10);    obj.change(ptr);    obj.print();    return 0;  }  Ans) error l value is required as left operand of  Assignment. |

|  |
| --- |
| 1. #include<iostream>   using namespace std;    class Test  {  private:    int x;    int y;  public:    Test(int x = 0, int y = 0) { this->x = x; this->y = y; }    static void fun1() { cout << "Inside fun1()"; }    static void fun2() { cout << "Inside fun2()"; this->fun1(); }  };    int main()  {    Test obj;    obj.fun2();    return 0;  }  Ans) We cannot use this pointer with static |

|  |
| --- |
| 1. #include<iostream>   using namespace std;    class Test  {  private:    int x;    int y;  public:    Test (int x = 0, int y = 0) { this->x = x; this->y = y; }    Test setX(int a) { x = a; return \*this; }    Test setY(int b) { y = b; return \*this; }    void print() { cout << "x = " << x << " y = " << y ; }  };    int main()  {    Test obj1;    obj1.setX(10).setY(20);    obj1.print();    return 0;  }  Ans)x=10 y=0   1. #include<iostream>   using namespace std;    class A  {  int x;  public:  A() { x = 0;}  void fun() {  delete this;  cout<<x;  }  }; |
| class A  {    public:      void fun()      {          delete this;      }  };    int main()  {    A \*ptr = new A;    ptr->fun();    ptr = NULL;      A a;    a.fun();      getchar();    return 0;  }  Ans) error because class A is redefined |

1. Compare line 1 & 2

MyClass t1, t2;

MyClass t3 = t1;  // ----> (1)

t2 = t1;          // -----> (2)

Ans) error because MyClass does not name a type

1. #include<iostream>

#include<cstring>

using namespace std;

class String

{

private:

char \*s;

int size;

public:

String(const char \*str = NULL); // constructor

~String() { delete [] s; }// destructor

String(const String&); // copy constructor

void print() { cout << s << endl; } // Function to print string

void change(const char \*); // Function to change

};

String::String(const char \*str)

{

size = strlen(str);

s = new char[size+1];

strcpy(s, str);

}

void String::change(const char \*str)

{

delete [] s;

size = strlen(str);

s = new char[size+1];

strcpy(s, str);

}

String::String(const String& old\_str)

{

size = old\_str.size;

s = new char[size+1];

strcpy(s, old\_str.s);

}

int main()

{

String str1("India");

String str2 = str1;

str1.print(); // what is printed ?

str2.print();

str2.change("India is great");

str1.print(); // what is printed now ?

str2.print();

return 0;

}

1. #include <iostream>

using namespace std;

int i;

class A

{

public:

~A()

{

i=10;

}

};

int foo()

{

i=3;

A ob;

return i;

}

int main()

{

cout << "i = " << foo() << endl;

return 0;

}

Ans) i=3

#include <iostream>

using namespace std;

int i;

class A

{

public:

~A()

{

i = 10;

}

};

int& foo()

{

i = 3;

A ob;

return i;

}

int main()

{

cout << "i = " << foo() << endl;

return 0;

}

Ans) i=10

17#include <iostream>

using namespace std;

void func(int a, bool flag =true)

{

if(flag ==true){

cout<<"Flag is true. a = "<< a;

}

else{

cout<<"Flag is false. a = "<< a;

}

}

int main()

{

func(200, false);

return 0;

}

Ans) Flag is false. a = 200

Since value is already issued in actual parameter it is passed into the function ignoring the default argument.

18.#include <iostream>

#include <string>

using namespace std;

string askNumber(string prompt ="Please enter a number: ");

int main()

{

string number =askNumber();

cout<<"Here is your number: "<< number;

return0;

}

string askNumber(string prompt)

{

string number;

cout<< prompt;

cin>> number;

return number;

}

Ans) Please enter a number: 5

Here is your number: 5

First we have a default prompt in prototype and the enter the value which is then returned to main and output is given henceforth.

19.#include <iostream>

using namespace std;

int func(int m = 10, int n)

{

int c;

c = m + n;

return c;

}

int main()

{

cout<<func(5);

return 0;

}

ANS) Will throw an error since default argument should be given from right to left and here default argument is missing for n in the function.

20.#include <iostream>

using namespace std;

double &WeeklyHours()

{

double h =46.50;

double &hours = h;

return hours;

}

int main()

{

double hours =WeeklyHours();

cout<<"Weekly Hours: "<< hours;

return0;

}

Ans) Weekly Hours: 46.5

The trailing 0 s are removed or truncated

21.#include <iostream>

using namespace std;

void square (int \*x)

{

\*x = (\*x + 1) \* (\*x);

}

int main ( )

{

intnum = 10;

square(&num);

cout<<num;

return 0;

}

Ans)110 because here we are passing the paraments and calling and call by address type and hence the value is acquired by the derefenrecing operator.

22.#include <iostream>

using namespace std;

void Sum(int a, int b, int& c)

{

a = b + c;

b = a + c;

c = a + b;

}

int main()

{

int x =2, y =3;

Sum(x, y, y);

cout<< x <<" "<< y;

return0;

}

Ans) 2 15 beacause in the function there is no return value for a but for y in the formal parameter address operator is been used and hence the value changes for y.

23.#include <iostream>

using namespace std;

class CDummy

{

public:

intisitme(CDummy&param);

};

intCDummy::isitme(CDummy&param)

{

if(&param==this)

return true;

else

return false;

}

int main ()

{

CDummy a;

CDummy\*b =&a;

if(b->isitme(a)){

cout<<"execute";

}

else

{

cout<<"not execute";

}

return0;

}

Ans) execute

24.#include <iostream>

using namespace std;

class sample

{

private:

intvar;

public:

void input()

{

cout<<var;

}

void output()

{

cout<<"Variable entered is ";

cout<<var<<"**\n**";

}

};

int main()

{

sample object;

object.input();

object.output();

object.var();

return0;

}

Ans)var is private

25.#include <iostream>

using namespace std;

class A

{

public:

int i;

A()

{

i=10;

}

A(int j=3)

{i = j;}

};

int main()

{

A a();

cout<<a.i;

return 0;

}

Ans) error: request for member ‘i’ in ‘a’, which is of non-class type ‘A()’

26.#include <iostream>

using namespace std;

class A

{

public:

int i, j;

A()

{

i=10;

}

A(int j=3)

{i = j;}

};

int main()

{

A a = 4;

cout<<a.i;

cout<<a.j;

return 0;

}

Ans) 434

1. #include <iostream>

using namespace std;

class A

{

public:

int \*i;

A(int \*j = new int(0))

{

i = j;

}

};

void newval(A &a)

{

int i = \*a.i;

a.i = &(++i);

}

int main()

{

A a;

a.i = 0;

if (a.i)

{

newval(a);

}

cout<< \*a.i;

return 0;

}

Ans) segmentation fault