**Program No:** 01

**Program Topic:** *Overloading Constructor Function.*

**Program Title:**

Write a C++ program to overload a constructor function.

**Objectives:**

To learn how to overload constructor funcions.

**Syntax:**

class ClassName{

public:

ClassName( ){… … … }

ClassName( ){… … …}

…

};

**Source Code:**

#include <iostream>

using namespace std;

class Area{

private:

int length;

int breadth;

public:

Area(): length(5), breadth(2) { }

Area(int l, int b): length(l), breadth(b){ }

void GetLength(){

cout << "Enter length and breadth respectively: ";

cin >> length >> breadth;

}

int AreaCalculation() { return length \* breadth; }

void DisplayArea(int temp){

cout << "Area: " << temp << endl;

}

};

int main(){

Area A1, A2(2, 1);

int temp;

cout << "Default Area when no argument is passed." << endl;

temp = A1.AreaCalculation();

A1.DisplayArea(temp);

cout << "Area when (2,1) is passed as argument." << endl;

temp = A2.AreaCalculation();

A2.DisplayArea(temp);

return 0;

}

**Output:**

Default Area when no argument is passed.

Area: 10

Area when (2,1) is passed as argument.

Area: 2

Process returned 0 (0x0) execution time : 0.084 s

Press any key to continue.

**Explanation:**

Overloaded constructors have the same name (name of the class) but different number of arguments. Constructors are overloaded for three purposes. Those are to gain flexibility, to support arrays, and to create copy constructor.

**Program No:** 02

**Program Topic:** *Copy Constructor.*

**Program Title:**

Write a C++ program using a copy constructor.

**Objectives:**

To learn how to create and use a copy constructor.

**Syntax:**

ClassName (const ClassName &old\_obj){

… … …

};

**Source Code:**

#include<iostream>

using namespace std;

class Point

{

private:

int x, y;

public:

Point(int x1, int y1) { x = x1; y = y1; }

Point(const Point &p2) {x = p2.x; y = p2.y; }

int getX( ){ return x; }

int getY( ){ return y; }

};

int main()

{

Point p1(10, 15);

Point p2 = p1;

cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();

cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();

return 0;

}

**Output:**

p1.x = 10, p1.y = 15

p2.x = 10, p2.y = 15

Process returned 0 (0x0) execution time : 0.102 s

Press any key to continue.

**Explanation:**

A copy constructor is a member function which initializes an object using another object of the same class.

**Program No:** 03

**Program Topic:** *Default Arguments.*

**Program Title:**

Write a C++ program to overload function using default arguments.

**Objectives:**

To learn about default arguments.

**Syntax:**

functionPrototype functionName(dataType a = 0, dataType b = 0);

**Source Code:**

#include<iostream>

using namespace std;

int sum(int x, int y, int z=0, int w=0)

{

return (x + y + z + w);

}

int main()

{

cout << sum(10, 15) << endl;

cout << sum(10, 15, 25) << endl;

cout << sum(10, 15, 25, 30) << endl;

return 0;

}

**Output:**

25

50

80

Process returned 0 (0x0) execution time : 0.225 s

Press any key to continue.

**Explanation:**

A default argument is a value provided in function declaration that is automatically assigned by the compiler if caller of the function doesn’t provide a value for the argument with default value.

These declaration of default arguments are valid:

f(); // a and b default to 0

f (10) ; // a is 10, b defaults to 0

f(10 , 99); // a is 10, b is 99

Note: Note: there is no way to default a and specify b.

**Program No:** 04

**Program Topic:** *Finding address of an overloading function.*

**Program Title:**

Write a C++ program to overload a function finding the address of the overloading function.

**Objectives:**

To learn how to overload a function finding the address of it.

**Syntax:**

int funcName(dataType argument, … …);

int main()

{

int (\*fp)(int a); // pointer to int f(int)

fp = funcName;

… … …

}

**Source Code:**

#include <iostream>

using namespace std;

int myfunc(int a);

int myfunc(int a, int b);

int main()

{

int (\*fp1)(int a);

int (\*fp2)(int a, int b);

fp1 = myfunc;

fp2 = myfunc;

cout << fp1(5) << endl;

cout << fp2(5, 10) << endl;

return 0;

}

int myfunc(int a)

{

return a;

}

int myfunc(int a, int b)

{

return a\*b;

}

**Output:**

5

50

Process returned 0 (0x0) execution time : 0.168 s

Press any key to continue.

**Program No:** 05

**Program Topic:** *Operator Overloading.*

**Program Title:**  
Write a C++ program to overload binary operators.

**Objectives:**

To learn how to overload binary operators.

**Syntax:**

return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

#include<iostream>

using namespace std;

class Complex {

private:

int real, imag;

public:

Complex(int r = 0, int i =0) {real = r; imag = i;}

Complex operator + (Complex const &obj) {

Complex res;

res.real = real + obj.real;

res.imag = imag + obj.imag;

return res;

}

Complex operator - (Complex const &obj) {

Complex res;

res.real = real - obj.real;

res.imag = imag - obj.imag;

return res;

}

Complex operator =(Complex const &obj){

Complex res;

res.real = obj.real;

res.imag = obj.imag;

return res;

}

void print() { cout << real << " + i" << imag; }

};

int main()

{

Complex c1(10, 5), c2(2, 4);

Complex c3 = c1 + c2;

Complex c4 = c1 - c2;

Complex c5 = c1;

Complex c6 = c2;

cout << "(";

c5.print( );

cout << ") + (";

c6.print( );

cout << ") = (";

c3.print( );

cout << ")" << endl;

cout << "(";

c5.print( );

cout << ") - (";

c6.print( );

cout << ") = (";

c4.print( );

cout << ")" << endl;

return 0;

}

**Output:**

(10 + i5) + (2 + i4) = (12 + i9)

(10 + i5) - (2 + i4) = (8 + i1)

Process returned 0 (0x0) execution time : 0.247 s

Press any key to continue.

**Explanation:**

Operator functions are same as normal functions. The only differences are, name of an operator function is always operator keyword followed by symbol of operator and operator functions are called when the corresponding operator is used.

There are two important restrictions to remember when you are over-

loading an operator.

1. The precedence of the operator cannot be changed.

2. The number of operands that an operator takes cannot be altered.

The only operators that cannot be overloaded are:

. :: .\* ?

Operator functions cannot have default arguments.

**Program No:** 06

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload relational operators.

**Objectives:**

To learn how to overload relational operators.

**Syntax:**

return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

#include <iostream>

using namespace std;

class Distance {

private:

int feet;

int inches;

public:

Distance() {

feet = 0;

inches = 0;

}

Distance(int f, int i) {

feet = f;

inches = i;

}

void displayDistance() {

cout << "F: " << feet << " I:" << inches <<endl;

}

bool operator <(const Distance& d) {

if(feet < d.feet) {

return true;

}

if(feet == d.feet && inches < d.inches) {

return true;

}

return false;

}

bool operator >(const Distance& d) {

if(feet > d.feet) {

return true;

}

if(feet == d.feet && inches > d.inches) {

return true;

}

return false;

}

bool operator ==(const Distance& d) {

if(feet == d.feet && inches == d.inches) {

return true;

}

return false;

}

};

int main() {

Distance D1(11, 10), D2(5, 11), D3(11, 10), D4(20, 5);

if( D1 > D2 )

cout << "D1 is greater than D2 " << endl;

if(D4 > D2)

cout << "D2 is less than D4 " << endl;

if(D1 == D3)

cout << "D1 is equals to D3 " << endl;

return 0;

}

**Output:**

D1 is greater than D2

D2 is less than D4

D1 is equals to D3

Process returned 0 (0x0) execution time : 0.242 s

Press any key to continue.

**Program No:** 07

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload logical operators.

**Objectives:**

To learn how to overload logical operators.

**Syntax:**

return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

# include <iostream >

using namespace std;

class coord3D{

int x, y, z;

public :

coord3D ( ) { x = 0; y = 0, z = 0; };

coord3D (int i, int j, int k) { x = i; y = j, z = k; }

void get\_xy (int &i, int &j, int &k) { i = x; j = y, k = z; }

int operator &&( coord3D ob2);

};

int coord3D :: operator &&( coord3D ob2){

return (x && ob2.x) && (y && ob2.y) && (z && ob2.z);

}

int main (){

coord3D o1 (10 , 10, 10), o2 (5, 3, 2);

if(o1 && o2)

cout << "o1 && o2 is true \n";

else

cout << "o1 && o2 is false \n";

return 0;

}

**Output:**

o1 && o2 is true

Process returned 0 (0x0) execution time : 0.124 s

Press any key to continue.

**Program No:** 08

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload unary operators.

**Objectives:**

To learn how to overload unary operators.

**Syntax:**

return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

#include<iostream>

using namespace std;

class InDe

{

int a, b;

public:

void accept( ){

cout<<"Enter Two Numbers: \n";

cin >> a;

cin >> b;

}

InDe operator --( ){

a--;

b--;

return \*this;

}

InDe operator++( ){

a++;

b++;

return \*this;

}

void display( ){

cout << "\nA: " << a;

cout<< "\nB: " << b << endl;

}

};

int main()

{

InDe id;

id.accept();

--id;

cout<<"\nAfter Decrementing : ";

id.display( );

++id;

++id;

cout<<"\n\nAfter Incrementing : ";

id.display( );

return 0;

}

**Output:**

Enter Two Numbers:

3

-2

After Decrementing :

A: 2

B: -3

After Incrementing :

A: 4

B: -1

Process returned 0 (0x0) execution time : 5.299 s

Press any key to continue.

**Program No:** 09

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload operator using friend function.

**Objectives:**

To learn how to overload operator using friend function.

**Syntax:**

friend return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

#include <iostream>

using namespace std;

class Point {

int x, y;

public:

Point() {}

Point(int px, int py) {

x = px;

y = py;

}

void show() {

cout << x << " ";

cout << y << "\n";

}

friend Point operator +(Point op1, Point op2);

Point operator =(Point op2);

};

Point operator +(Point op1, Point op2){

Point temp;

temp.x = op1.x + op2.x;

temp.y = op1.y + op2.y;

return temp;

}

Point Point :: operator =(Point op2){

x = op2.x;

y = op2.y;

return \*this;

}

int main()

{

Point ob1(10, 20), ob2( 5, 30);

ob1 = ob1 + ob2;

ob1.show();

return 0;

}

**Output:**

15 50

Process returned 0 (0x0) execution time : 0.136 s

Press any key to continue.

**Explanation:**

A friend function does not have a this pointer. A friend operator function is passed both operands explicitly. For unary operators, the single operand is passed.

A friend function cannot be used to overload the assignment operator.

**Program No:** 10

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload the assignment operator.

**Objectives:**

To learn how to overload the assignment operator.

**Syntax:**

return\_type class\_name :: operator #( arg\_list )

{

… … …

}

**Source Code:**

#include <iostream>

using namespace std;

class Distance {

private:

int feet;

int inches;

public:

Distance() {

feet = 0;

inches = 0;

}

Distance(int f, int i) {

feet = f;

inches = i;

}

void operator = (const Distance &D ) {

feet = D.feet;

inches = D.inches;

}

void displayDistance( ) {

cout << "F: " << feet << " I: " << inches << endl;

}

};

int main( ) {

Distance D1(11, 10), D2(5, 11);

cout << "First Distance : ";

D1.displayDistance( );

cout << "Second Distance :";

D2.displayDistance( );

D1 = D2;

cout << "First Distance :";

D1.displayDistance( );

return 0;

}

**Output:**

First Distance : F: 11 I: 10

Second Distance :F: 5 I: 11

First Distance :F: 5 I: 11

Process returned 0 (0x0) execution time : 0.109 s

Press any key to continue.

**Program No:** 11

**Program Topic:** *Operator Overloading.*

**Program Title:**

Write a C++ program to overload the [ ] Subscript Operator.

**Objectives:**

To learn how to overload the subscript operator.

**Syntax:**

return\_type class\_name :: operator []( int index )

{

// ...

}

**Source Code:**

#include <iostream>

using namespace std;

const int SIZE = 10;

class aray {

private:

int arr[SIZE];

public:

aray( ) {

register int i;

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

arr[i] = i;

}

}

int &operator[ ](int i) {

if( i > SIZE ) {

cout << "Index out of bounds" << endl;

return arr[0];

}

return arr[i];

}

};

int main( ) {

aray A;

cout << "Value of A[2] : " << A[2] <<endl;

cout << "Value of A[5] : " << A[5]<<endl;

cout << "Value of A[12] : " << A[12]<<endl;

return 0;

}

**Output:**

Value of A[2] : 2

Value of A[5] : 5

Index out of bounds

Value of A[12] : 0

Process returned 0 (0x0) execution time : 0.093 s

Press any key to continue.

**Explanation:**

The subscript operator [ ] is normally used to access array elements. This operator can be overloaded to enhance the existing functionality of C++ arrays.

The [ ] can be overloaded only by a member function.