

OBJECT ORIENTED PROGRAMMING

PRACTICAL FILE

Submitted to :--

Dr. SarabjeetSingh

Submitted by :--

Pushaan Sharma(co19349)

INDEX

- 1 Create a new header file and use it in a program.**
- 2 Calculate the range and size of data types in c++.**
- 3 Write a program to show use of class.**
- 4 Write a program to show use of enumerated datatype.**
- 5 Write a program to show use of static member function.**
- 6 Write a program to show use of copy constructor.**
- 7 Write a program to show use of friend function.**
- 8 Write a program to copy the contents of one file to another in reverse order.**

- 9 Write a program to explain use of various keywords used for exception handling in c++.**
- 10 Write a program to overload a function template.**
- 11 Write a program to show use of different access specifiers inc++.**
- 12 Write a program to demonstrate operator overloading.**

Create a new header file and use it in a program.

=)Creating new header file: #
`include<iostream.h> float
sum (float a, float b)`

```

    return ( a + b );
}
{

```

=) Using new header file #
`include<iostream.h> #
include"sum.h"
using namespace std ;
int main()`

```

{
    float x, y, z ;
    cout << "Enter two numbers: "; cin
    >> x >> y ;
    z = sum( x, y ) ;
    cout << " n Sum = " << z ; return 0 ;
}

```

Output:

Enter two numbers: 27 85

Sum = 112

Calculate the range and size of data types in c++.

```
#include<iostream.h>#
#include<math.h>using
namespace std;
int main()
{
    cout<<"Size of char is "<<sizeof(char)<<" bytes "<<endl;cout<<
    "Size of int is "<<sizeof(int)<<" bytes "<<endl;cout<<"Size of float is
    "<<sizeof(float)<<" bytes "<<endl;
    cout<<"Size of double is "<<sizeof(double)<<" bytes "<<endl;cout<<
    "Range of int is from "<<-pow(2,sizeof(int)*8-1<<" to "
    <<pow(2,sizeof(int)*8-1)-1<<endl;
    cout<<"Range of char is from "<<-pow(2,sizeof(char)*8-1<<" to "
    <<pow(2,sizeof(char)*8-1)-1<<endl;return 0
    ;
}
```

Output:

```
Size of char is 1 bytes
Size of int is 4 bytes
Size of float is 4 bytes
Size of double is 8 bytes
Range of int is from -2147483648 to 2147483647
Range of char is from -127 to 127
```

Write a program to show use of class.

```
#include<iostream.h>
using namespace std;
class student
{
    public:
        int roll;
        char name[30];
        int age;
};
int main()
{
    student s1, s2;
    cout << "Enter details of first student: " << endl; cin >>
    s1.roll >> s1.name >> s1.age;
    cout << "n Enter details of second student: " << endl; cin >>
    s2.roll >> s2.name >> s2.age;
    cout << "showing details: ";
    cout << "Details of first student: " << endl;
    cout << "n Name: " << s1.name << "n Roll no: " << s1.roll <<
    "n Age: " << s1.age;
    cout << "Details of second student: " << endl;
    cout << "n Name: " << s2.name << "n Roll no: " << s2.roll <<
    "n Age: " << s2.age; return 0;
}
```

Output:

```
Enter details of first student: 17
Enter details of second student: 55
showing details:
Details of first student:
n Name:
n Roll no:
n Age:
Details of second student:
n Name:
n Roll no:
n Age:
```

Name: Ashish
Age: 17
Roll No.: 25

Age: 18

Write a program to show use of enumerated data type.

```
#include<iostream.h>
using namespace std; enum
year
{JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC;      }
int main()
{
    for( int i = JAN ; i <= DEC ; i++ ) cout << i
        << " ";
    return 0 ;
}
```

Output:

1 2 3 4 5 6 7 8 9 10 11 12

Write a program to show use of static memberfunction.

```
#include<iostream.h>
class demo
{
public :
    static int a ;
    demo()
    {
        a++ ;
    }
    static int fun()
    {
        cout << a ;
        return 0 ;
    }
};
int demo :: a ; int
main()
{
    demo d1, d2, d3 ;
    demo :: fun() ;
    return 0 ;
}
```

Output:

3

Write a program to show use of copy constructor.

```
#include<iostream.h>
using namespace std ; class
point
{
public :
    int x,y;
    point( int x1, int y1 )
    {
        x = x1 ; y
        = y1 ;
    }
    point( const point &p2 )
    {
        x = p2.x ; y
        = p2.y ;
    }
};
int main()
{
    point p1( 10, 15 ) ;
    point p2 = p1 ;
    cout << "p1.x = " << p1.x << " ,p1.y = " << p1.y << endl ; cout << "p2.x
    = " << p2.x << " ,p2.y = " << p2.y << endl ; return 0 ;
}
```

Output:

p1.x = 10 ,p1.y =15

p2.x = 10 ,p2.y =15

Write a program to show use of friend function.

```
#include<iostream.h>
using namespace std;
class A
{
    int a;
public:
    A()
    {
        a = 0;
    }

    friend class B;
};
class B
{
    int b;
public:
    void showA( A& x )
    {
        cout << "A :: a = " << x.a;
    }
};
int main()
{
    A a;
    B b; b.showA(
    a ); return 0;
}
```

Output:

A :: a = 0

Write a program to copy the contents of one file to another in reverse order.

```
#include<iostream.h>#
#include<fstream.h>#
#include<conio.h>
#include<stdio.h>using
namespace std; void
reverse( char str[] )

{
    char Ch ; ofstream
    ofs ;
    ofs.open("file.txt", ios :: out ) ;
    for( int i = 0 ; str[i] != '\0' ; i++ )
    {
        ofs.put( str[i] ) ;
    }
    int pos = ofs.tellp() ;
    ofs.close() ;
    ifstream ifs ;
    ifs.open( "file.txt", ios :: in ) ;
    ofstream ofs1 ;
    ofs1.open( "file2.txt", ios :: out ) ;
    ifs.seekg( - pos ) ;
    while ( pos >> 0 )
    {
        ifs.get( ch ) ;
        ofs1.put( ch ) ;
        pos-- ; ifs.seekg(
        pos ) ;
    }
    ifs.close() ;
    ofs1.close() ;
    ifstream ifs1 ;
    ifs1.open( "file2.txt", ios :: in ) ;
    while ( ! ifs1.eof() )
    {
        ifs1.get( ch ) ;
        cout << ch ;
    }
    ifs1.close() ;
}
```

```

}
int main()
{
    cout << "Reversing "reverse" .....|....."
    reverse( "reverse" );
    return 0 ;
}

```

Output:

Reversing "reverse"

esrever

Write a program to explain use of various keywords used for exception handling in c++.

```
#include<iostream.h>
using namespace std; double division( int a, int b )
{
    if( b == 0 )
        throw "Division by zero condition!"; return (
            a/b );
}
int main()
{
    int x = 50 ; int
    y = 0 ; double z
    = 0 ; try

        z = division( x,y ) ; cout
        {
            << z << endl ;
        }
        catch ( const char* msg ) cout
        {
            << msg << endl ;
        }
    }return 0 ;
}
```

Output:

Division by zero condition!

Write a program to overload a function template.

```
#include<iostream.h> using
namespace std; template <
class T>
T max( T x, T y )
{
    return ( x > y ) ? x : y; int
}
main()
{
    cout << max<int>( 3, 7 ) << endl;
    cout << max<double>( 3.0, 7.0 ) << endl; cout <<
    max<char>( 'g', 'e' ) << endl; return 0;
}
```

Output:

7
7

g

Write a program to show use of different access specifiers in c++.

```
#include<iostream.h>
using namespace std; class
base
{
private:
    int x;
protected:
    int y;
public:
    int z;
};
class derive1 : private base
{
public :
    void showdata()
    {
        cout << "x is not accessible" << endl; cout << "y
        is a private member" << endl; cout << "z is a
        private member" << endl;
    }
};
class derive2 : protected base
{
public :
    void showdata()
    {
        cout << "x is not accessible" << endl;
        cout << "y is a protected member" << endl; cout
        << "z is a protected member" << endl;
    }
};
class derive3 : public base
{
public :
    void showdata()
    {
        cout << "x is not accessible" << endl;
        cout << "y is a protected member" << endl;
    }
};
```



```

        cout << "z is a public member" << endl ;
    }
};
int main()
{
    derive1 a ;
    derive2 b ;
    cout << "private derived class:" ;
    a.showdata();
    cout << "protected derived class:" ;
    b.showdata();
    cout << "public derived class:" ;
    c.showdata();
    return 0 ;
}

```

Output:

```

private derived class:
a is a private member
protected derived class:
b is a protected member
public derived class:
c is a public member
z is a public member

```

Write a program to demonstrate operator overloading.

```
#include<iostream.h>
using namespace std ; class
complex
{
    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 ;
    }
    void print()
    {
        cout << real << " + i" << imag << endl ;
    }
};
int main()
{
    complex c1( 10, 5 ), c2( 2, 4 ) ;
    complex c3 = c1 + c2 ; c3.print() ;
}
```

Output:

12 + i9