National University of Computer and Emerging Sciences



Task given in Class

Object Oriented Programming

|  |  |
| --- | --- |
| Name | Muhammad Zain |
| Roll No. | 19F-0228 |
| Course INSTRUCTOR | Dr Danish Shehzad |
| Semester | Spring 2020 |

Example 1

# (Composition )

## Source Code;

#include<cstring>

class Date {

public:

Date( int = 1, int = 1, int = 1900 ); // default constructor

void print() const; // print date in month/day/year format

~Date(); // provided to confirm destruction order

private:

int month; // 1-12

int day; // 1-31 based on month

int year; // any year

// utility function to test proper day for month and year

int checkDay( int );

};

#include <iostream>

using namespace std;

// value for day.

Date::Date( int mn, int dy, int yr )

{

if ( mn > 0 && mn <= 12 ) // validate the month

month = mn;

else {

month = 1;

cout << "Month " << mn << " invalid. Set to month 1.\n";

}

year = yr; // should validate yr

day = checkDay( dy ); // validate the day

cout << "Date object constructor for date ";

print(); // interesting: a print with no arguments

cout << endl;

}

void Date::print() const

{ cout << month << '/' << day << '/' << year; }

// Destructor: provided to confirm destruction order

Date::~Date()

{

cout << "Date object destructor for date ";

print();

cout << endl;

}

// Utility function to confirm proper day value

// based on month and year.

// Is the year 2000 a leap year?

int Date::checkDay( int testDay )

{

static const int daysPerMonth[ 13 ] =

{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if ( testDay > 0 && testDay <= daysPerMonth[ month ] )

return testDay;

if ( month == 2 && // February: Check for leap year

testDay == 29 &&

( year % 400 == 0 ||

( year % 4 == 0 && year % 100 != 0 ) ) )

return testDay;

cout << "Day " << testDay << " invalid. Set to day 1.\n";

return 1; // leave object in consistent state if bad value

}

class Employee {

public:

Employee( char \*, char \*, int, int, int, int, int, int );

void print() const;

~Employee(); // provided to confirm destruction order

private:

char firstName[ 25 ];

char lastName[ 25 ];

const Date birthDate;

const Date hireDate;

};

Employee::Employee( char \*fname, char \*lname,

int bmonth, int bday, int byear,

int hmonth, int hday, int hyear )

: birthDate( bmonth, bday, byear ),

hireDate( hmonth, hday, hyear )

{

// copy fname into firstName and be sure that it fits

int length = strlen( fname );

length = ( length < 25 ? length : 24 );

strncpy( firstName, fname, length );

firstName[ length ] = '\0';

// copy lname into lastName and be sure that it fits

length = strlen( lname );

length = ( length < 25 ? length : 24 );

strncpy( lastName, lname, length );

lastName[ length ] = '\0';

cout << "Employee object constructor: "

<< firstName << ' ' << lastName << endl;

}

void Employee::print() const

{

cout << lastName << ", " << firstName << "\nHired: ";

hireDate.print();

cout << " Birth date: ";

birthDate.print();

cout << endl;

}

// Destructor: provided to confirm destruction order

Employee::~Employee()

{

cout << "Employee object destructor: "

<< lastName << ", " << firstName << endl;

}

int main()

{cout<<"Muhammad Zain 19F-228"<<endl;

Employee e( "Bob", "Jones", 7, 24, 1949, 3, 12, 1988 );

cout << '\n';

e.print();

cout << "\nTest Date constructor with invalid values:\n";

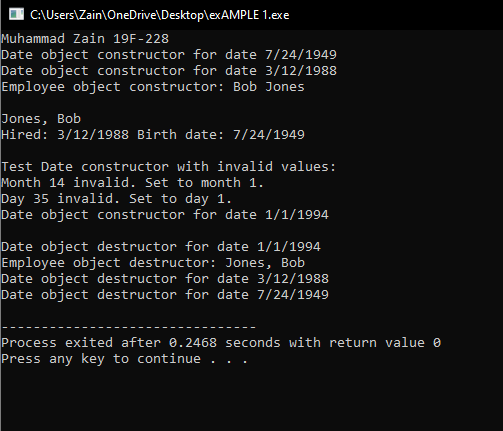
Date d( 14, 35, 1994 ); // invalid Date values

cout << endl;

return 0;

}

# Snip:



Example 2

# (Friend function in one class)

## Source code:

#include<iostream>

using namespace std;

class count{

public:

count() {x=0;}

friend void setX(count&,int);

void print() const {cout<<x<< " Muhammad Zain 19F0228"<<endl;}

private:

int x;

};

void setX(count &c,int val)//ek he class ma check krna tha is llea us class ka address pass krdea

//changing of private data members is allowed

//in friend function

{

c.x=val;

}

int main()

{

count counter ;

cout<<"Counter x after instatiation: ";//constructtor cll hoa value 0 initialize hogai

counter.print();

cout<<"counter x after call to friend function";

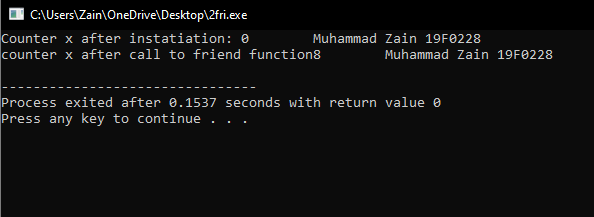
setX(counter,8);//setx bahar class k call hoa to value 8 set hogai

counter.print();

return 0;

}

## Snip:



Example 3

# (Friend function in 2 classess)

# Source Code:

#include<iostream>

using namespace std;

class second;//new concept

//because friend function will be used in bth classess

//so that we declare both class to se there variables

class first{

private:

int numA;

public:

first():numA(50){cout<<"The value of 1st number is "<<numA<<endl;

cout<<"In the constructor of first class before calling of friend function"<<endl;

}

//member initializer list k through initialiation

friend int addition(first,second);

//name of classess from which we can use tthiere variables

//declaration of friend function

};

class second{

private:

int numX;

public:

second():numX(20){cout<<endl<<"The value of 2nd number is "<<numX<<endl;

cout<<"In the constructor of 2nd class before calling of friend function"<<endl;}

//declaration through member initializer list

friend int addition(first,second);

};

int addition(first objectFirst,second objectSecond){

cout<<"The addedd value in the friends function"<<endl;

return (objectFirst.numA+objectSecond.numX);

cout<<endl<<"19F-0228!"<<endl;

}

int main()

{

first obj;

second Obj;

cout<< addition (obj,Obj);

return 0;

}

# Snip:

