**M. Ali. Arslan**

**19F-0348**

**Class Activity**

**TASK NO 1:**

**Main.cpp**

#include <iostream>

#include<string>

#include"date1.h"

using namespace std;

int main()

{

Date d1, d2(12, 27, 1992), d3(0, 99, 8045);

cout << "d1 is " << d1

<< "\nd2 is " << d2

<< "\nd3 is " << d3 << "\n\n";

cout << "d2 += 7 is " << (d2 += 7) << "\n\n";

d3.setDate(2, 28, 1992);

cout << " d3 is " << d3;

cout << "\n++d3 is " << ++d3 << "\n\n";

Date d4(3, 18, 1969);

cout << "Testing the preincrement operator:\n"

<< " d4 is " << d4 << '\n';

cout << "++d4 is " << ++d4 << '\n';

cout << " d4 is " << d4 << "\n\n";

cout << "Testing the postincrement operator:\n"

<< " d4 is " << d4 << '\n';

d4++;

cout << "d4++ is " << d4 << '\n';

cout << " d4 is " << d4 << endl;

return 0;

}

**Date1.cpp**

#include <iostream>

#include "date1.h"

int Date::days[] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

Date::Date(int m, int d, int y) { setDate(m, d, y); }

void Date::setDate(int mm, int dd, int yy)

{

month = (mm >= 1 && mm <= 12) ? mm : 1;

year = (yy >= 1900 && yy <= 2100) ? yy : 1900;

// test for a leap year

if (month == 2 && leapYear(year))

day = (dd >= 1 && dd <= 29) ? dd : 1;

else

day = (dd >= 1 && dd <= days[month]) ? dd : 1;

}

Date& Date::operator++()

{

helpIncrement();

return \*this; // reference return to create an lvalue

}

Date Date::operator++(int)

{

Date temp = \*this;

helpIncrement();

return temp; // value return; not a reference return

}

Date& Date::operator+=(int additionalDays)

{

for (int i = 0; i < additionalDays; i++)

helpIncrement();

return \*this; // enables cascading

}

// If the year is a leap year, return true;

// otherwise, return false

bool Date::leapYear(int y)

{

if (y % 400 == 0 || (y % 100 != 0 && y % 4 == 0))

return true; // a leap year

else

return false; // not a leap year

}

// Determine if the day is the end of the month

bool Date::endOfMonth(int d)

{

if (month == 2 && leapYear(year))

return d == 29; // last day of Feb. in leap year

else

return d == days[month];

}

void Date::helpIncrement()

{

if (endOfMonth(day) && month == 12) { // end year

day = 1;

month = 1;

++year;

}

else if (endOfMonth(day)) { // end month

day = 1;

++month;

}

else // not end of month or year; increment day

++day;

}

ostream &operator<<(ostream &output, Date &d)

{

const char\* monthName[13] = { "", "January", "February", "March", "April",

"May", "June", "July", "August", "September", "October", "November", "December" };

output << monthName[d.month] << ' ' << d.day << ", " << d.year;

return output; // enables cascading

}

**Date1.h**

#include<iostream>

#include <ostream>

using namespace std;

class Date

{

friend ostream& operator<<(ostream&, Date&);

public:

Date(int m = 1, int d = 1, int y = 1900); // constructor

void setDate(int, int, int); // set the date

Date& operator++(); // preincrement operator

Date operator++(int); // postincrement operator

Date& operator+=(int); // add days, modify object

bool leapYear(int); // is this a leap year?

bool endOfMonth(int); // is this end of month?

private:

int month;

int day;

int year;

static int days[]; // array of days per month

void helpIncrement(); // utility function

};



**TASK NO 2:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<string>

using namespace std;

class String

{

private:

char\* text;

public:

String(char\* str)

{

text = new char[strlen(str)];

strcpy(text, str);

}

String() {

}

friend ostream& operator<<(ostream&, String

& str);

friend istream& operator>>(istream&, String

& str);

void operator= (char\* str);

String& operator+(String& str);

String& operator+(char\* str);

bool operator==(String& str);

bool operator==(char\* str);

char& operator[] (int Index);

};

bool String::operator == (char\* str)

{

bool val;

val = strcmp(text, str);

if (val == 0)

return true;

else

return false;

}

bool String::operator == (String& par)

{

bool val;

val = strcmp(text, par.text);

if (val == 0)

return true;

else

return false;

}

void String::operator = (char\* str)

{

text = new char[strlen(str)];

strcpy(text, str);

}

String& String::operator + (String& par)

{

String iSt ;

int length = 0;

length = strlen(text);

length += strlen(par.text);

iSt.text = new char[length];

strcpy(iSt.text, text);

strcat(iSt.text, par.text);

return iSt;

}

String& String::operator + (char\* str)

{

String iSt ;

int length = 0;

length = strlen(text);

length += strlen(str);

iSt.text = new char[length];

strcpy(iSt.text, text);

strcat(iSt.text, str);

return iSt;

}

ostream& operator<< (ostream& out, String& str)

{

out << str.text;

return out;

}

istream& operator>> (istream& in, String& str)

{

char temp[200];

in >> temp;

str.text = new char[strlen(temp)];

strcpy(str.text, temp);

return in;

}

char& String::operator[](int Index)

{

return text[Index];

}

int main(void)

{

string string1 = "hello";

string string2 = "";

string1 = "hello world";

cout << "Enter string 2 text" << endl;

cin >> string2;

if (string1 == string2)

cout << "Both strings are equal" << endl;

string2[0] = 'a';

string2[1] = 'b';

cout << "The second string is " << string2 << endl;

cout << " the first character is " << string1[0] << endl;

return 0;

}

