Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа
по курсу «ООП»

Тема:
Операторы, литералы.

Студент:	Косогоров В.В.
Группа:	М8О-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	10
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

angle.hpp

```
#ifndef ANGLE_H
#define ANGLE_H

#include <iostream>
#include <cmath>

class Angle
{
private:
    int deg;
    int min;

public:
    Angle();
    Angle(int d, int m): deg(d), min(m)
    {}

    void To_normal(); //приведение отрицательных углов и минут к диапазону [0..360]deg [0..59]min

    void Read(std::istream &is); //считывание угла
    void Write(std::ostream &os) const; //вывод угла

    double To_decimal() const; //перевод в десятичное представление
    double To_radian() const; //перевод в радианы

    friend Angle operator+ (const Angle& lhs, const Angle& rhs);
    friend Angle operator- (const Angle& lhs, const Angle& rhs);
    friend Angle operator/ (const Angle& lhs, const int divisor);

    double Sin() const;
    double Cos() const;

    friend bool operator> (const Angle& lhs, const Angle& rhs);
    friend bool operator< (const Angle& lhs, const Angle& rhs);
    friend bool operator== (const Angle& lhs, const Angle& rhs);

    friend std::istream& operator>> (std::istream& is, Angle& a);
    friend std::ostream& operator<< (std::ostream& os, Angle& a);
};
```

Angle operator"" _piece(unsigned long long divisor); // возвращает угол 360 градусов, делённый на divisor

#endif // ANGLE_H

angle.cpp:

```
#include "angle.hpp"

Angle::Angle()
{
    deg = 0;
    min = 0;
}

void Angle::To_normal()
{
    if (min < 0) {
        while (min < 0) {
            deg -= 1;
            min += 60;
        }
    } else if (min >= 60) {
        while (min >= 60) {
            deg += 1;
            min -= 60;
        }
    }

    if (deg < 0) {
        deg = 360 - (-deg % 360);
    } else {
        deg %= 360;
    }
}

void Angle::Read(std::istream &is)
{
    is >> deg >> min;
    To_normal();
}
```

```
void Angle::Write(std::ostream &os) const
{
    os << deg << ' ' << min << std::endl;
}

double Angle::To_decimal() const
{
    return deg + (double)min / 60.0;
}

double Angle::To_radian() const
{
    return M_PI * (deg + (double)min / 60.0) / 180.0;
}

Angle operator+ (const Angle& lhs, const Angle& rhs)
{
    Angle res(lhs.deg + rhs.deg, lhs.min + rhs.min);
    res.To_normal();
    return res;
}

Angle operator- (const Angle& lhs, const Angle& rhs)
{
    Angle res(lhs.deg - rhs.deg, lhs.min - rhs.min);
    res.To_normal();
    return res;
}

Angle operator/ (const Angle& lhs, const int divisor)
{
    return Angle(lhs.deg / divisor, lhs.min / divisor);
}

double Angle::Sin() const
{
    if ((deg == 0 || deg == 180) && min == 0) {
        return 0;
    }
    double t = To_radian();
    double sinx = t;

    for (int i = 1; i < 10; ++i)
    {
        double mult = - To_radian() * To_radian() / ((2 * i + 1) * (2 * i));
```

```
        t *= mult;
        sinx += t;
    }
    return sinx;
}

double Angle::Cos() const
{
    if ((deg == 90 || deg == 270) && min == 0) {
        return 0;
    }
    double mul = 1, div = 1,res = 0, x = To_radian();
    for (int i = 1; i < 20; i += 2)
    {
        res += mul / div;
        mul *= -x * x;
        div *= i * (i + 1);
    }
    return res;
}

bool operator> (const Angle& lhs, const Angle& rhs)
{
    return (lhs.To_decimal() > rhs.To_decimal());
}

bool operator< (const Angle& lhs, const Angle& rhs)
{
    return (lhs.To_decimal() < rhs.To_decimal());
}

bool operator== (const Angle& lhs, const Angle& rhs)
{
    return (lhs.To_decimal() == rhs.To_decimal());
}

std::istream& operator>> (std::istream& is, Angle &a)
{
    is >> a.deg >> a.min;
    return is;
}

std::ostream& operator<< (std::ostream& os, Angle &a)
{
    os << a.deg << ' ' << a.min << std::endl;
```

```
    return os;
}

Angle operator"" _piece(unsigned long long divisor)
{
    Angle res(360, 0);
    res = res / divisor;
    return res;
}
```

main.cpp:

```
#include <iostream>

#include "angle.hpp"

int main(void)
{
    Angle a;
    std::cin >> a;

    std::cout << std::endl;

    std::cout << a.To_decimal() << std::endl;
    std::cout << a.To_radian() << std::endl;
    std::cout << std::endl;

    Angle b;
    std::cin >> b;
    std::cout << std::endl;

    Angle result;

    result = a + b;
    std::cout << result;

    result = a - b;
    std::cout << result;

    std::cout << std::endl;

    int divisor;
```

```
    std::cin >> divisor;
    result = a / divisor;
    std::cout << result;
    std::cout << std::endl;

    std::cout << a.Sin() << std::endl;
    std::cout << a.Cos() << std::endl;
    std::cout << std::endl;

    if (a == b) {
        std::cout << "a == b" << std::endl;
    } else if (a > b) {
        std::cout << "a > b" << std::endl;
    } else {
        std::cout << "a < b" << std::endl;
    }
    std::cout << std::endl;

    result = 2_piece;
    std::cout << result;

    result = 6_piece;
    std::cout << result;
    std::cout << std::endl;

    return 0;
}
```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 3.5)

project(lab1)

add_executable(lab1
  main.cpp
  angle.cpp
)

set_property(TARGET lab1 PROPERTY CXX_STANDARD 11)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

test.sh:

```
executable=$1

for file in test_??.test
do
  $executable < $file > tmp
  if cmp tmp ${file%%.test}.result
  then
    echo Test "$file": SUCCESS
  else
    echo Test "$file": FAIL
  fi
  rm tmp
done
```

2. Ссылка на репозиторий на GitHub.

https://github.com/vladiq/oop_exercise_02

3. Набор testcases.

test_01.test:

```
59 32
43 30
2
```

test_02.test:

```
-15 40
30 15
4
```

test_03.test:

```
0 0
0 0
15
```

test_04.test:

180 0
182 -30
7

test_05.test:

180 0
0 0
180

test_06.test

-189 -3
-67 -78
189

test_07.test

0 120
0 -120
2

test_08.test

360 -360
180 -180
60

test_09.test

189213 42144
1331 4344
123

test_10.test

-1234423 -314
-31331 -13234
100

4. Результаты выполнения тестов.

…/prog_3_sem/oop_labs/lab_02 ./test.sh ./lab1

Test test_01.test: SUCCESS
Test test_02.test: SUCCESS
Test test_03.test: SUCCESS
Test test_04.test: SUCCESS
Test test_05.test: SUCCESS
Test test_06.test: SUCCESS
Test test_07.test: SUCCESS
Test test_08.test: SUCCESS
Test test_09.test: SUCCESS
Test test_10.test: SUCCESS

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_01.test

59 32
43 30
2

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_01.test

59.5333
1.03905

103 2
16 2

29 16

0.861924
0.507037

a > b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_02.test

-15 40
30 15
4

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_02.test

345.667
6.03302

15 55
315 25

86 10

-0.248012
0.967319

a > b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_03.test

0 0
0 0
15

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_03.test

0
0

0 0
0 0

0 0

0
1

a == b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_04.test

180 0
182 -30

7

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_04.test

180
3.14159

1 30
358 30

25 0

0
-1

a < b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_05.test

180 0
0 0
180

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_05.test

180
3.14159

180 0
180 0

1 0

0
-1

a > b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_06.test

-189 -3
-67 -78
189

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_06.test

170.95
2.98364

102 39
239 15

0 0

0.157296
-0.987551

a < b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_07.test

0 120
0 -120
2

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_07.test

2
0.0349066

0 0
4 0

1 0

0.0348995
0.999391

a < b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_08.test

360 -360
180 -180
60

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_08.test

354
6.17847

171 0
177 0

5 0

-0.105267
0.99203

a > b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_09.test

189213 42144
1331 4344
123

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_09.test

195.4
3.41037

158 48
232 0

1 0

-0.265556
-0.964095

```
a < b

360 0
60 0

…/prog_3_sem/oop_labs/lab_02 cat testcases/test_10.test

-1234423 -314
-31331 -13234
100

…/prog_3_sem/oop_labs/lab_02 ./lab1 < testcases/test_10.test

11.7667
0.205367


140 12
243 20

0 0

0.203927
0.978986

a < b

360 0
60 0
```

5. Объяснение результатов работы программы.

1) Конструктором Read() переменным deg и min, обозначающие градусы и минуты, присваиваются нули, а также вводим конструктор, присваивающий заданные значения членам класса Angle.
2) С помощью метода Read() со стандартного потока ввода мы считываем значения deg и min, где происходит приведение выходящих за диапазон и отрицательных углов и минут к диапазону [0..360] [0..59] соответственно.
3) Вывод значений min и deg на стандартный поток вывода осуществляется методом Write().
4) Первый угол переводится в число типа double и в радианы с помощью методов To_decimal() и To_radian() соответственно.
5) Вводим значения второго угла.
6) Сложение и вычитание углов производится при помощь перегрузки операторов сложения и вычитания.
7) Вводим целое число, на которое нужно разделить первый угол.

8) С помощью перегрузки оператора деления организуется деление угла на целое число.
9) Считаем синус и косинус угла при помощи стандартных функций sin() и cos(), аргументом для которых служит значение угла в радианах.
10) Сравниваем углы при помощи перегрузки операторов сравнения и равенства.
11) Вводим пользовательский литерал _piece, который возвращает угол, градусная мера которого равна 360 / divisor.

6. Вывод.

Выполняя данную лабораторную, я научился перегружать операторы и вводить пользовательские литералы в C++. Реализовал для простого класса методы для приведения членов к другим типам, получения значений синуса и косинуса. Также в ходе работы я реализовал перегрузку бинарных операторов сложения, вычитания, деления, сравнения и равенства, считывание и вывода и реализовал пользовательский литерал, возвращающий значение заданной части угла в 360 градусов.