

Московский Авиационный Институт
(Национальный исследовательский Университет)

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

**Лабораторная работа
по курсу «ООП»**

**Тема:
Простые классы.**

| | |
|----------------|----------------|
| 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>

class Angle
{
private:
    int deg;
    int min;

public:
    Angle();

    void To_normal(); //приведения 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

    Angle Plus(Angle &b) const; //сложение углов
    Angle Minus(Angle &b) const; //вычитание углов
    Angle Division(int number) const; //деление угла на целое число

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

    short Comparison(Angle &b) const; //сравнение углов
};

#endif // ANGLE_H
```

angle.cpp:

```
#include "angle.h"
```

```
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 3.14159265 * (deg + (double)min / 60.0) / 180.0;
}
```

```
Angle Angle::Plus(Angle &b) const
{
    Angle result;
    result.deg = deg + b.deg;
    result.min = min + b.min;

    result.To_normal();

    return result;
}
```

```
Angle Angle::Minus(Angle &b) const
{
    Angle result;

    result.deg = deg - b.deg;
    result.min = min - b.min;

    result.To_normal();

    return result;
}
```

```
Angle Angle::Division(int number) const
{
    Angle result;

    result.deg = deg / number;
    result.min = min / number;

    return result;
}
```

```
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;
}

```

```

short Angle::Comparison(Angle &b) const
{
    double a_dec = To_decimal();
    double b_dec = b.To_decimal();

    if (a_dec == b_dec) {
        return 0;
    } else if (a_dec > b_dec) {
        return 1;
    } else {
        return 2;
    }
}

```

main.cpp:

```
#include <iostream>

#include "angle.h"

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

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

    Angle b;
    b.Read(std::cin);

    a.Plus(b).Write(std::cout);
    a.Minus(b).Write(std::cout);
    std::cout << std::endl;

    int divisor;
    std::cin >> divisor;
    a.Division(divisor).Write(std::cout);
    std::cout << std::endl;

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

    if (a.Comparison(b) == 0) {
        std::cout << "a == b" << std::endl;
    } else if (a.Comparison(b) == 1) {
        std::cout << "a > b" << std::endl;
    } else {
        std::cout << "a < b" << 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_01

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.

```
~/Рабочий стол/what □ ./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
```

```
~/Рабочий стол/what □ ./lab1 < test_01.test
```

59.5333
1.03905

103 2
16 2

29 16

0.861924
0.507037

$a > b$

```
~/Рабочий стол/what .lab1 < test_02.test
```

345.667
6.03302

15 55
315 25

86 10

-0.248012
0.967319

$a > b$

```
~/Рабочий стол/what .lab1 < test_03.test
```

0
0

0 0
0 0

0 0

0
1

$a == b$

```
~/Рабочий стол/what .lab1 < test_04.test
```

180
3.14159

1 30
358 30

25 0

0
-1

$a < b$

~/Рабочий стол/what ./lab1 < test_05.test

180
3.14159

180 0
180 0

1 0

0
-1

$a > b$

~/Рабочий стол/what ./lab1 < test_06.test

170.95
2.98364

102 39
239 15

0 0

0.157296
-0.987551

$a < b$

~/Рабочий стол/what ./lab1 < test_07.test

2
0.0349066

0 0
4 0

1 0

0.0348995
0.999391

$a < b$

```
~/Рабочий стол/what .lab1 < test_08.test
```

354
6.17847

171 0
177 0

5 0

-0.105267
0.99203

$a > b$

```
~/Рабочий стол/what .lab1 < test_09.test
```

195.4
3.41037

158 48
232 0

1 0

-0.265556
-0.964095

$a < b$

```
~/Рабочий стол/what .lab1 < test_10.test
```

11.7667
0.205367

140 12
243 20

0 0

0.203927
0.978986

$a < b$

5. Объяснение результатов работы программы.

- 1) Конструктором Read() переменным deg и min, обозначающие градусы и минуты, присваиваются нули.
- 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 производится методами Plus() и Minus(), после чего результаты приводятся к диапазону [0..360]deg и [0..59]min.
- 7) Вводим целое число, на которое нужно разделить первый угол.
- 8) Делим угол на число из п. 7 с помощью метода Division().
- 9) С помощью формулы Тейлора, использованной в методах Sin() и Cos() считаем приближённые значения синуса и косинуса первого угла.
- 10) С помощью метода Comparison() сравниваем углы 1 и 2, предварительно приведя их к десятичному виду.

6. Вывод.

Выполняя данную лабораторную, я получил опыт работы с классами в C++, системой сборки Stake и распределённой системой управления версиями Git. Реализовал для простого класса методы для приведения членов к другим типам, сложение, вычитание, деление на целое число, сравнение и получение приближённых значений синуса и косинуса с помощью ряда Тейлора.