**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Терво Александр Александрович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи. Вариант 10.

Создать класс Angle для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0–360, сложение и вычитание углов, деление углов, получение значений тригонометрических функций, сравнение углов. Операции сложения, вычитания, деления, сравнения (на равенство, больше и меньше) должны быть выполнены в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа Angle.

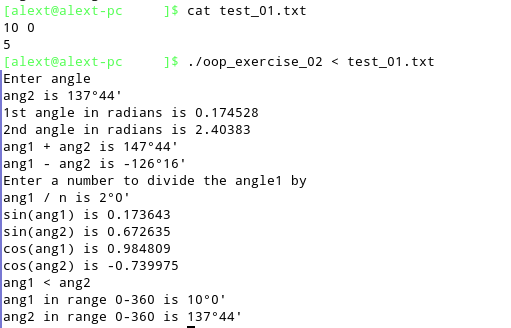
1. Описание программы

Для класса Angle создадим два приватных члена: значения угла в градусах и минутах. Реализуем перегрузку операторов в соответствии с вариантом задания. Создадим методы преобразования в радианы, приведения к диапазону 0-360 и получения значений тригонометрических функций. Реализацию методов класса вынесем в отдельный файл, в основном файле реализуем ввод-вывод данных и взаимодействие с методами класса.

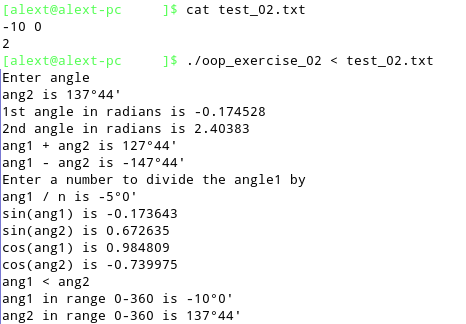
1. Набор тестов и результаты их выполнения

На вход программе подаётся угол в градусах и минутах и число, на которое этот угол необходимо разделить. Программа выводит результат преобразования этого угла в радианы, сумму и разность двух углов (второй угол задан в тексте программы), результат деления заданного угла на число, значения синусов и косинусов от углов, результат их сравнения и приведения к диапазону 0-360°.

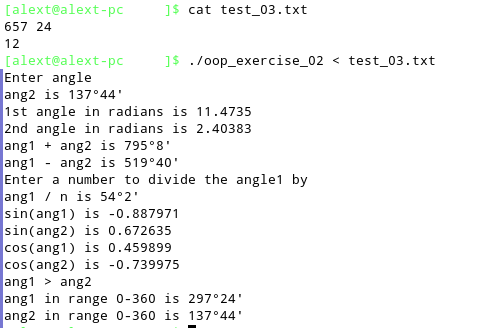
Тест №1



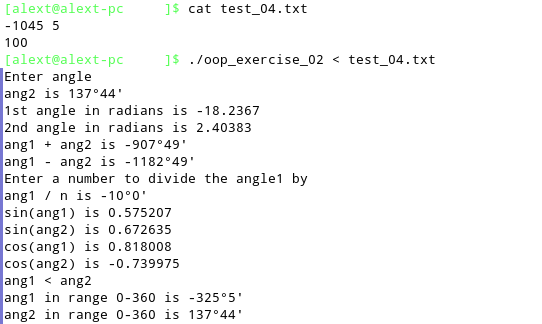
Тест №2



Тест №3



Тест №4



1. Листинг программы

main.cpp:

#include "angle.hpp"

int main() {

int deg;

double min;

std::cout << "Enter angle\n";

std::cin >> deg >> min;

Angle ang1(deg, min);

// std::cout << "Enter 2nd angle\n";

// std::cin >> deg >> min;

Angle ang2 = "137,44"\_angle;

std::cout << "ang2 is " << ang2 << std::endl;

std::cout << "1st angle in radians is " << ang1.radian() << std::endl;

std::cout << "2nd angle in radians is " << ang2.radian() << std::endl;

std::cout << "ang1 + ang2 is " << ang1 + ang2 << std::endl;

std::cout << "ang1 - ang2 is " << ang1 - ang2 << std::endl;

std::cout << "Enter a number to divide the angle1 by\n";

double n;

std::cin >> n;

std::cout << "ang1 / n is " << ang1 / n << std::endl;

std::cout << "sin(ang1) is " << ang1.sin() << "\nsin(ang2) is " << ang2.sin() << std::endl;

std::cout << "cos(ang1) is " << ang1.cos() << "\ncos(ang2) is " << ang2.cos() << std::endl;

if (ang1 > ang2) {

std::cout << "ang1 > ang2\n";

} else

if (ang1 == ang2) {

std::cout << "ang1 = ang2\n";

} else {

std::cout << "ang1 < ang2\n";

}

ang1.convTo360();

ang2.convTo360();

std::cout << "ang1 in range 0-360 is " << ang1 << std::endl;

std::cout << "ang2 in range 0-360 is " << ang2 << std::endl;

}

angle.hpp:

#ifndef ANGLE\_HPP

#define ANGLE\_HPP

#include <iostream>

#include <cmath>

const double PI = 3.1415;

class Angle {

private:

int deg;

double min;

public:

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

double radian();

int convTo360();

double sin();

double cos();

friend Angle operator + (Angle& lv, Angle& rv);

friend Angle operator - (Angle& lv, Angle& rv);

friend Angle operator / (Angle& lv, double& n);

friend bool operator < (Angle& lv, Angle& rv);

friend bool operator > (Angle& lv, Angle& rv);

friend bool operator == (Angle& lv, Angle& rv);

friend std::ostream& operator << (std::ostream &out, const Angle& ang);

~Angle() {}

};

Angle operator "" \_angle(const char\* str, size\_t size);

#endif

angle.cpp:

#include "angle.hpp"

double Angle::radian() {

return (deg \* PI / 180 + min \* PI / (180 \* 60));

}

int Angle::convTo360() {

deg = deg % 360;

return 0;

}

double Angle::sin() {

return std::sin(radian());

}

double Angle::cos() {

return std::cos(radian());

}

Angle operator + (Angle& lv, Angle& rv) {

return Angle(lv.deg + rv.deg + (lv.min + rv.min) / 60, fmod(lv.min + rv.min, 60));

}

Angle operator - (Angle& lv, Angle& rv) {

int d;

double m;

unsigned int sign = ((lv > rv) || (lv == rv)) ? 1 : -1;

if (((lv.deg < 0) && (rv.deg < 0)) || ((lv.deg >= 0) && (rv.deg >= 0))) {

if (lv.min < rv.min) {

d = sign \* (std::abs(lv.deg - rv.deg) - 1);

m = lv.min + 60.0 - rv.min;

} else {

d = sign \* (std::abs(lv.deg - rv.deg));

m = lv.min - rv.min;

}

} else {

if (lv.min + rv.min >= 60.0) {

d = sign \* (std::abs(lv.deg - rv.deg) + 1);

m = lv.min + rv.min - 60.0;

} else {

d = sign \* (std::abs(lv.deg - rv.deg));

m = lv.min + rv.min;

}

}

return Angle(d, m);

}

Angle operator / (Angle& lv, double& n) {

return Angle(lv.deg / n + (lv.min / n) / 60, fmod(lv.min / n, 60));

}

bool operator < (Angle& lv, Angle& rv) {

bool flag = true;

if (rv.deg < lv.deg) {

flag = false;

} else

if (rv.deg == lv.deg) {

if (rv.min < lv.min) {

flag = false;

}

}

return flag;

}

bool operator > (Angle& lv, Angle& rv) {

bool flag = true;

if (rv.deg > lv.deg) {

flag = false;

} else

if (rv.deg == lv.deg) {

if (rv.min > lv.min) {

flag = false;

}

}

return flag;

}

bool operator==(Angle& lv, Angle& rv) {

bool flag = false;

if ((rv.deg == lv.deg) && (rv.min == lv.min)) {

flag = true;

}

return flag;

}

std::ostream& operator << (std::ostream &out, const Angle& ang) {

out << ang.deg << "°" << ang.min << '\'';

return out;

}

Angle operator "" \_angle(const char\* str, size\_t size) {

//std::cout << size << std::endl;

int deg = 0;

double min = 0;

bool flag = true;

for (int i = 0; i < size; ++i) {

if (flag) {

if (str[i] != ',') {

deg = deg \* 10 + str[i] - '0';

} else {

flag = false;

}

} else {

min = min \* 10 + str[i] - '0';

}

}

return Angle(deg, min);

}

5. Выводы.

В ходе работы я изучил процесс перегрузки операторов, позволяющий программисту определять собственные действия для операторов, а также создание пользовательских литералов, позволяющих проще обрабатывать строки. Также закрепил создание классов и методов и работу с системой контроля версий GIT.

6. Литература.

1.Перегрузка операторов в C++ / Хабр — HabrURL: https://habr.com/ru/post/132014/

2.Пользовательские литералы в С++11 / Хабр — HabrURL: https://habr.com/ru/post/140357/