ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Программирование

Код модуля
1155651(1)

Модуль
Информационно-техническая культура

Екатеринбург

Оценочные материалы составлены автором(ами):

№ п/п	Фамилия, имя, отчество	Ученая степень, ученое звание	Должность	Подразделение
1	Белов Александр Ильич	без ученой степени, без ученого звания	Старший преподаватель	департамент математики, механики и компьютерных наук

Согласовано:

Управление образовательных программ Л.А. Щенникова

Авторы:

- **Белов Александр Ильич, Старший преподаватель, департамент математики, механики и компьютерных наук**

1. СТРУКТУРА И ОБЪЕМ ДИСЦИПЛИНЫ Программирование

1.	**Объем дисциплины в зачетных единицах**	7	
2.	**Виды аудиторных занятий**	Лекции Лабораторные занятия	
3.	**Промежуточная аттестация**	Зачет Экзамен	
4.	**Текущая аттестация**	Контрольная работа	1
		Домашняя работа	1
		Реферат	1
		Отчет по лабораторным работам	1

2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (ИНДИКАТОРЫ) ПО ДИСЦИПЛИНЕ МОДУЛЯ Программирование

Индикатор – это признак / сигнал/ маркер, который показывает, на каком уровне обучающийся должен освоить результаты обучения и их предъявление должно подтвердить факт освоения предметного содержания данной дисциплины, указанного в табл. 1.3 РПМ-РПД.

Таблица 1

Код и наименование компетенции	**Планируемые результаты обучения (индикаторы)**	**Контрольно-оценочные средства для оценивания достижения результата обучения по дисциплине**
1	**2**	**3**
ПК-3 -Способен разрабатывать алгоритмы и компьютерные программы	З-2 - Описывать современные языки программирования, их основные конструкции и возможности П-1 - Выполнять разработку и отладку алгоритмов и компьютерных программ, включая разработку веб-сайтов У-1 - Выбирать инструментальные средства разработки алгоритмов и компьютерных программ	Домашняя работа Зачет Лабораторные занятия Лекции Реферат Экзамен
ПК-4 -Способен осваивать и	З-1 - Привести примеры документации к программным	Домашняя работа Зачет

применять в практической деятельности документацию к программным системам и стандартам в области программирования и информационных систем	системам и стандартам в области программирования и информационных систем П-1 - Иметь практический опыт применения документации к программным системам и стандартам в области программирования	Контрольная работа Лабораторные занятия Лекции Отчет по лабораторным работам Экзамен

3. ПРОЦЕДУРЫ КОНТРОЛЯ И ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ В РАМКАХ ТЕКУЩЕЙ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ МОДУЛЯ В БАЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЕ (ТЕХНОЛОГИЧЕСКАЯ КАРТА БРС)

3.1. Процедуры текущей и промежуточной аттестации по дисциплине

1. Лекции: коэффициент значимости совокупных результатов лекционных занятий – 0.5		
Текущая аттестация на лекциях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
реферат	1,15	*100*
Весовой коэффициент значимости результатов текущей аттестации по лекциям – 0.5		
Промежуточная аттестация по лекциям – зачет **Весовой коэффициент значимости результатов промежуточной аттестации по лекциям – 0.5**		
2. Практические/семинарские занятия: коэффициент значимости совокупных результатов практических/семинарских занятий – не предусмотрено		
Текущая аттестация на практических/семинарских занятиях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
Весовой коэффициент значимости результатов текущей аттестации по практическим/семинарским занятиям– не предусмотрено		
Промежуточная аттестация по практическим/семинарским занятиям–нет **Весовой коэффициент значимости результатов промежуточной аттестации по практическим/семинарским занятиям– не предусмотрено**		
3. Лабораторные занятия: коэффициент значимости совокупных результатов лабораторных занятий –0.5		
Текущая аттестация на лабораторных занятиях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
контрольная работа	1,16	100
Весовой коэффициент значимости результатов текущей аттестации по лабораторным занятиям -1		
Промежуточная аттестация по лабораторным занятиям –нет		

Весовой коэффициент значимости результатов промежуточной аттестации по лабораторным занятиям – не предусмотрено		
4. Онлайн-занятия: коэффициент значимости совокупных результатов онлайн-занятий –не предусмотрено		
Текущая аттестация на онлайн-занятиях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
Весовой коэффициент значимости результатов текущей аттестации по онлайн-занятиям -не предусмотрено		
Промежуточная аттестация по онлайн-занятиям –нет **Весовой коэффициент значимости результатов промежуточной аттестации по онлайн-занятиям – не предусмотрено**		

3.2. Процедуры текущей и промежуточной аттестации курсовой работы/проекта

Текущая аттестация выполнения курсовой работы/проекта	**Сроки − семестр, учебная неделя**	**Максимальная оценка в баллах**
Весовой коэффициент текущей аттестации выполнения курсовой работы/проекта– не предусмотрено		
Весовой коэффициент промежуточной аттестации выполнения курсовой работы/проекта– защиты – не предусмотрено		

3.1. Процедуры текущей и промежуточной аттестации по дисциплине

2. Лекции: коэффициент значимости совокупных результатов лекционных занятий – 0.5		
Текущая аттестация на лекциях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
домашняя работа	2,12	*100*
Весовой коэффициент значимости результатов текущей аттестации по лекциям – 0.5		
Промежуточная аттестация по лекциям – экзамен **Весовой коэффициент значимости результатов промежуточной аттестации по лекциям – 0.5**		
2. Практические/семинарские занятия: коэффициент значимости совокупных результатов практических/семинарских занятий – не предусмотрено		
Текущая аттестация на практических/семинарских занятиях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
Весовой коэффициент значимости результатов текущей аттестации по практическим/семинарским занятиям– не предусмотрено		
Промежуточная аттестация по практическим/семинарским занятиям–нет **Весовой коэффициент значимости результатов промежуточной аттестации по практическим/семинарским занятиям– не предусмотрено**		
3. Лабораторные занятия: коэффициент значимости совокупных результатов лабораторных занятий –0.5		

Текущая аттестация на лабораторных занятиях	Сроки – семестр, учебная неделя	Максимальная оценка в баллах
отчет по лабораторным работам	2,14	100
Весовой коэффициент значимости результатов текущей аттестации по лабораторным занятиям -1		
Промежуточная аттестация по лабораторным занятиям –нет **Весовой коэффициент значимости результатов промежуточной аттестации по лабораторным занятиям – не предусмотрено**		
4. Онлайн-занятия: коэффициент значимости совокупных результатов онлайн-занятий –не предусмотрено		
Текущая аттестация на онлайн-занятиях	**Сроки – семестр, учебная неделя**	**Максимальная оценка в баллах**
Весовой коэффициент значимости результатов текущей аттестации по онлайн-занятиям -не предусмотрено		
Промежуточная аттестация по онлайн-занятиям –нет **Весовой коэффициент значимости результатов промежуточной аттестации по онлайн-занятиям – не предусмотрено**		

3.2. Процедуры текущей и промежуточной аттестации курсовой работы/проекта

Текущая аттестация выполнения курсовой работы/проекта	Сроки − семестр, учебная неделя	Максимальная оценка в баллах
Весовой коэффициент текущей аттестации выполнения курсовой работы/проекта– не предусмотрено		
Весовой коэффициент промежуточной аттестации выполнения курсовой работы/проекта– защиты – не предусмотрено		

4. КРИТЕРИИ И УРОВНИ ОЦЕНИВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ МОДУЛЯ

4.1. В рамках БРС применяются утвержденные на кафедре/институте критерии (признаки) оценивания достижений студентов по дисциплине модуля (табл. 4) в рамках контрольно-оценочных мероприятий на соответствие указанным в табл.1 результатам обучения (индикаторам).

Таблица 4

Критерии оценивания учебных достижений обучающихся

Результаты обучения	Критерии оценивания учебных достижений, обучающихся на соответствие результатам обучения/индикаторам
Знания	Студент демонстрирует знания и понимание в области изучения на уровне указанных индикаторов и необходимые для продолжения обучения и/или выполнения трудовых функций и действий, связанных с профессиональной деятельностью.
Умения	Студент может применять свои знания и понимание в контекстах, представленных в оценочных заданиях, демонстрирует освоение умений на уровне указанных индикаторов и необходимых для

	продолжения обучения и/или выполнения трудовых функций и действий, связанных с профессиональной деятельностью.
Опыт /владение	Студент демонстрирует опыт в области изучения на уровне указанных индикаторов.
Другие результаты	Студент демонстрирует ответственность в освоении результатов обучения на уровне запланированных индикаторов. Студент способен выносить суждения, делать оценки и формулировать выводы в области изучения. Студент может сообщать преподавателю и коллегам своего уровня собственное понимание и умения в области изучения.

4.2 Для оценивания уровня выполнения критериев (уровня достижений обучающихся при проведении контрольно-оценочных мероприятий по дисциплине модуля) используется универсальная шкала (табл. 5).

Таблица 5

Шкала оценивания достижения результатов обучения (индикаторов) по уровням

Характеристика уровней достижения результатов обучения (индикаторов)				
№ п/п	**Содержание уровня выполнения критерия оценивания результатов обучения (выполненное оценочное задание)**	**Шкала оценивания**		
		Традиционная характеристика уровня		**Качественная характеристика уровня**
1.	Результаты обучения (индикаторы) достигнуты в полном объеме, замечаний нет	Отлично (80-100 баллов)	Зачтено	Высокий (В)
2.	Результаты обучения (индикаторы) в целом достигнуты, имеются замечания, которые не требуют обязательного устранения	Хорошо (60-79 баллов)		Средний (С)
3.	Результаты обучения (индикаторы) достигнуты не в полной мере, есть замечания	Удовлетворительно (40-59 баллов)		Пороговый (П)
4.	Освоение результатов обучения не соответствует индикаторам, имеются существенные ошибки и замечания, требуется доработка	Неудовлетворительно (менее 40 баллов)	Не зачтено	Недостаточный (Н)
5.	Результат обучения не достигнут, задание не выполнено	Недостаточно свидетельств для оценивания		Нет результата

5. СОДЕРЖАНИЕ КОНТРОЛЬНО-ОЦЕНОЧНЫХ МЕРОПРИЯТИЙ ПО ДИСЦИПЛИНЕ МОДУЛЯ

5.1. Описание аудиторных контрольно-оценочных мероприятий по дисциплине модуля

5.1.1. Лекции

Самостоятельное изучение теоретического материала по темам/разделам лекций в соответствии с содержанием дисциплины (п. 1.2. РПД)

5.1.2. Лабораторные занятия

Примерный перечень тем

1. Тема 1. Введение в программирование на языке C++.
2. Тема 2. Объектно- ориентированный подход.
3. Тема 3. Рекурсия и динамическая структура данных.
4. Тема 4. Шаблоны и абстрактные типы данных.
5. Тема 5. Анализ правильности программ.
6. Тема 6. Программы, управляемые событиями.

LMS-платформа − не предусмотрена

5.2. Описание внеаудиторных контрольно-оценочных мероприятий и средств текущего контроля по дисциплине модуля

Разноуровневое (дифференцированное) обучение.

Базовый

5.2.1. Контрольная работа

Примерный перечень тем

1. Проверка знаний синтаксиса C++.
2. Бинарные деревья поиска.
3. Структуры

Примерные задания

Вариант задания по теме «Структуры»:

В отдельной библиотеке реализовать структуру с двумя или тремя основными полями / свойствами.

В реализации должны присутствовать:

- конструктор с параметрами для задания значений основных полей;
- дополнительное свойство только для чтения, значение которого вычисляется на основе
- основных полей /свойств;
- переопределение метода ToString() для представления экземпляра структуры в виде строки;
- переопределение метода Equals();
- переопределение метода GetHashCode() так, чтобы для равных с точки зрения метода Equals() экземплярам генерировался одинаковый хеш-код.
- переопределенная(-ые) операция(-и).

Методы, устанавливающие значения свойств или переопределяющие операции при передаче ошибочных значений должны генерировать исключение. Написать модульные тесты для свойств и методов реализованной структуры. Для сравнения чисел с плавающей точкой в методах и тестах установить точность 10 в минус 13 степени.

LMS-платформа − не предусмотрена

5.2.2. Домашняя работа

Примерный перечень тем

1. Конечный автомат и его программная реализация (для нескольких видов автоматов).

2. Разработка своей программы с графическим интерфейсом.

Примерные задания

Пример задания

Даны две параболы. Написать программу, находящую точки их пересечения. Программа должна включать

интерфейсную часть, организующую ввод данных, основной блок вычислений, интерфейс вывода

результатов. Программа должна реагировать на ошибки ввода и ошибки вычислений. Ввод и вывод

производятся консольно.

Критерии оценки: работоспособность программы, выполнение требований задания.

Пример выполнения

```
import math
a1,b1,c1 = map(float, input('введите коэффициенты первой параболы через
пробел: ').split())
a2,b2,c2= map(float, input('введите коэффициенты второй параболы через пробел:
').split())
if (a1==a2):
if(b1==b2):
if(c1==c2):
print('параболы совпадают')
a=a1-a2
b=b1-b2
c=c1-c2
if(a==0):
if(b==0):
print('параболы не пересекаются')
elif(b!=0):
x3=-c/b
y3=a1*x3*x3+b1*x3+c
print('параболы имеют одну точку пересечения (',x3,',',y3,')' )
elif(a!=0):
d=b*b-4*a*c
if(d>0):
x1=(-b+math.sqrt(d))/(2*a)
x2=(-b-math.sqrt(d))/(2*a)
y1=a1*x1*x1+b1*x1+c1
y2=a1*x2*x2+b1*x2+c1
print('параболы имеют 2 точки пересечения (',x1,',',y1,') и
(',x2,',',y2,')')
elif(d==0):
x=-b/(2*a)
```

y=a1*x*x+b1*x+c1
print('параболы имеют одну точку пересечения (',x,',',y,')')
else:
print('параболы не пересекаются')
LMS-платформа − не предусмотрена

5.2.3. Реферат

Примерный перечень тем
1. Структуры.
2. LINQ.
3. Рефлексия типов
Примерные задания
Подготовить реферат по теме LINQ.
Language Integrated Query (LINQ) — проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы.
Выделить структуру реферата: Аббревиатура LINQ, История создания LINQ, LINQ to objects, LINQ to DataSet, LINQ to SQL, LINQ to Entities, Оформление запросов LINQ

LMS-платформа − не предусмотрена

5.2.4. Отчет по лабораторным работам

Примерный перечень тем
1. Объектно- ориентированный подход.
Примерные задания
Структура Angle (угол).
Данная структура предоставляет возможность работу с углами в градусной мере. Имеются в виду не геометрические углы (которые равны, если отличаются на величину, кратную 360°), углы в более общем смысле. Так, поворот ключа в замке на 2 оборота — это поворот на 720°, и это не то же самое, что поворот ключа на 0°.
Свойства:
• Degrees — градусы, целое.
• Minutes — минуты, неотрицательное целое, не более 59.
• Seconds — секунды, неотрицательное целое, не более 59.
• ValueInSeconds — значение угла в секундах, целое, только для чтения

Строковое представление — стандартное, например, 15°24’30” или -30°30’17”.
Определить операции сравнения углов, их сумму, разность и умножение действительного числа на угол.
Выполнение задания
В Visual Studio 2022 создадим новый проект, используя шаблон «Библиотека классов (.NET Framework)». Назовем его AngleStruct. Переименуем модуль Class1.cs в Angle.cs, согласившись с переименованием класса Class1 в Angle. В определении класса Angle заменим ключевое слово class ключевым словом struct. Теперь мы получили определение структуры.
Значение градусов — любое целое число. Сделаем открытое свойство Degrees типа int. Контроль значений для него не нужен, его значение может быть любым целым числом.

Свойства Minutes и Seconds — открытые типа int, но их значения неотрицательные и не могут превосходить 59, поэтому контроль значений осуществим в методе set. Чтобы в коде не было «волшебных чисел», определим максимальное значение 59 как закрытую целочисленную константу.

Свойство ValueInSeconds — открытое типа int, только для чтения, поскольку его значение однозначно определяется значениями свойств Degrees, Minutes и Seconds. Вычислим это значение в методе get. Чтобы в коде не было «волшебных чисел», определим число секунд в минуте и в часе как закрытые константы структуры secondsInMinute и secondsInDegree соответственно.

Напишем конструктор, распределяющий значения градусов, минут и секунд угла по соответствующим свойствам/полям структуры. Чтобы иметь возможность контроля параметров конструктора, мы, прежде чем использовать экземпляр структуры через обращения к его свойствам, задать значения всех полей. Сделаем это через вызов конструктора без парамтров.

Переопределим метод ToString(). В нашем случае строковое представление угла —это просто значения его градусов, минут и секунд, снабженные соответствующими обозначениями градуса (°), минуты (') и секунды (").

Переопределим метод Equals(). Поскольку значение угла можно свести к одному значению в секундах, то углы будут равны, когда совпадут значения полей ValueInSeconds. Следует учесть, что параметр метода имеет тип object. Так как мы будем сравнивать углы только с углами, то нужно проверить, можно ли аргумент привести к типу Angle, и в случае невозможности выбросить исключение.

Переопределим метод GetHashCode(). Опять можно воспользоваться тем, что есть целочисленное значение ValueInSeconds, которое однозначно определяет экземпляр нашей структуры.

Для перегрузки операторов сравнения (== и !=) углов чтобы избежать дублирования кода используем переопределенный метод Equals().

Операции сложения углов и умножения на число проще производить с их значениями в секундах. Но нам нужен результат в градусах, минутах и секундах. Поэтому напишем закрытый статический метод GetAngleByValueInSeconds(), который будет решать эту задачу. Не забываем, что углы могут быть и отрицательными. Для коммутативности операции умножения числа на угол напишем две перегрузки оператора с разным порядком следования операндов.

Теперь модуль Angle.cs содержит код:

```
using System;

namespace AngleStruct
{
public struct Angle
{
const int maxValue = 59;
const int secondsInMinute = 60;
const int secondsInDegree = 3600;

public int Degrees { get; set; }
```

```
    int minutes;
    public int Minutes
    {
    get => minutes;
    set
    {
    if (value < 0 || value > maxValue)
    throw new ArgumentException("Значение должно быть неотрицательным и не более
59");

    minutes = value;
    }
    }

    int seconds;
    public int Seconds
    {
    get => seconds;
    set
    {
    if (value < 0 || value > maxValue)
    throw new ArgumentException("Значение должно быть неотрицательным и не более
59");

    seconds = value;
    }
    }

    public int ValueInSeconds
    {
    get => Math.Sign(Degrees) * (Math.Abs(Degrees) * secondsInDegree +
    Minutes * secondsInMinute + Seconds);
    }

    public Angle(int degrees, int minutes, int seconds) : this()
    {
    Degrees = degrees;
    Minutes = minutes;
    Seconds = seconds;
    }

    public override string ToString() => $"{Degrees}°{Minutes}'{Seconds}\"";

    public override bool Equals(object obj)
    {
    if (obj is Angle)
```

```
return ValueInSeconds == ((Angle)obj).ValueInSeconds;

throw new ArgumentException("Объект для сравнения не является углом");
}

public override int GetHashCode() => ValueInSeconds.GetHashCode();

public static bool operator ==(Angle x, Angle y) => x.Equals(y);
public static bool operator !=(Angle x, Angle y) => !x.Equals(y);

public static Angle operator +(Angle x, Angle y) =>
GetAngleByValueInseconds(x.ValueInSeconds + y.ValueInSeconds);

public static Angle operator *(double k, Angle angle) =>
GetAngleByValueInseconds((int)Math.Round(k * angle.ValueInSeconds));

public static Angle operator *(Angle angle, double k) => k * angle;

private static Angle GetAngleByValueInseconds(int val)
{
int seconds = Math.Abs(val);
int degrees = Math.Sign(val) * (seconds / secondsInDegree);
seconds %= secondsInDegree;
int minutes = seconds / secondsInMinute;
seconds %= secondsInMinute;

return new Angle(degrees, minutes, seconds);
}
}
}
```

Для создания модульных тестов добавим в наше решение новый проект, используя шаблон «Тестовый проект NUnit». Назовем его AngleStruct.UnitTests. В зависимости проекта добавим ссылку на тестируемую библиотеку (правой кнопкой мыши на «Зависимости» —> выбираем «Добавить ссылку на проект» —> отмечаем проект AngleStruct —> нажимаем OK).

В модуле AngleStruct.UnitTests.cs переименуем класс Tests на AngleTests и пометим его атрибутом [TestFixture]. Также удалим метод SetUp(). Этот метод запускается перед каждым тестированием и служит для настройки тестов, в частности для создания сложных программных сущностей перед тестированием. В нашем случае, когда тестируются простые экземпляры структуры, он не нужен.

Переименуем метод Test1 в ConstructorTest и проверим в нем правильность распределения значений параметров по свойствам экземпляра структуры. Для утверждений (Assert) будем использовать вместо classic model рекомендованную в настоящее время constraint model. Например, вместо

```
Assert.AreEqual(expected, actual);
```

напишем

Assert.That(actual, Is.EqualTo(expected));

Проверим также контроль значений свойств Minutes и Seconds. Соответствующие тесты назовем по шаблону ТестируемыйМетод_УсловияТестирование_ОжидаемыйРезультат. Используем параметризованные тесты.

Также протестируем правильность вычисления значения свойства ValueInSeconds для положительного, нулевого и отрицательного углов. Назовем тест ValueInSecondsTest.

Далее протестируем метод ToString() положительного, нулевого и отрицательного углов параметризованным тестом. Назовем тест ToStringTest.

Для тестирования метода Equals() используем параметризованный тест Equals_TwoAngles_ExpectedResult() для сравнения углов и проверяем исключение (тест Equals_WrongArgument_ArgumentException()), если сравниваем угол и некий объект.

Метод GetHashCode() протестируем через его назначение — этот метод должен возвращать одинаковые значения для одинаковых экземпляров и разные для разных. Напишем тест GetHashCodeTest().

Напишем аналогичный тест для операций сравнения ComparisonTest(), а также параметризованные тесты для операций сложения (AdditionTest) и умножения на число (MultiplicationTest). Параметров у метода получается много, но с этим придется смириться, поскольку мы не можем в атрибуте [TestCase(…)] ссылаться на объекты.

Код модуля AngleTests.cs будет выглядеть так:

```
namespace AngleStruct.UnitTests
{
[TestFixture]
public class AngleTests
{
[Test]
public void ConstructorTest()
{
var angle = new Angle(-42, 18, 21);

Assert.That(angle.Degrees, Is.EqualTo(-42));
Assert.That(angle.Minutes, Is.EqualTo(18));
Assert.That(angle.Seconds, Is.EqualTo(21));
}

[TestCase(-30)]
[TestCase(75)]
public void MinutesSet_NegativeOrBigValue_ArgumentException(int val)
{
var angle = new Angle();

Assert.That(() => angle.Minutes = val, Throws.ArgumentException);
}

[TestCase(-30)]
```

```
[TestCase(75)]
public void SecondsSet_NegativeOrBigValue_ArgumentException(int val)
{
var angle = new Angle();

Assert.That(() => angle.Seconds = val, Throws.ArgumentException);
}

[TestCase(15, 42, 18, 56538)]
[TestCase(0, 0, 0, 0)]
[TestCase(-15, 42, 18, -56538)]
public void ValueInSecondsTest(
int degrees, int minutes, int sceconds, int result)
{
var angle = new Angle(degrees, minutes, sceconds);

Assert.That(angle.ValueInSeconds, Is.EqualTo(result));
}

[TestCase(15, 42, 18, "15°42'18\"")]
[TestCase(0, 0, 0, "0°0'0\"")]
[TestCase(-15, 42, 18, "-15°42'18\"")]
public void ToStringTest(int degrees, int minutes, int sceconds, string result)
{
var angle = new Angle(degrees, minutes, sceconds);

Assert.That(angle.ToString(), Is.EqualTo(result));
}

[TestCase(30, 30, true)]
[TestCase(30, 15, false)]
public void Equals_TwoAngles_ExpectedResult(
int degrees1, int degrees2, bool result)
{
var angle1 = new Angle(degrees1, 0, 0);
var angle2 = new Angle(degrees2, 0, 0);

Assert.That(angle1.Equals(angle2), Is.EqualTo(result));
}

[Test]
public void Equals_WrongArgument_ArgumentException()
{
var angle = new Angle();
var smth = new object();
```

```
Assert.That(() => angle.Equals(smth), Throws.ArgumentException);
}

[Test]
public static void GetHashCodeTest()
{
var x = new Angle(45, 18, 31);
var y = new Angle(45, 18, 31);
var z = new Angle(-30, 45, 7);

Assert.That(x.Equals(y), Is.True);
Assert.That(x.Equals(z), Is.False);
}

[Test]
public static void ComparisonTest()
{
var x = new Angle(45, 18, 31);
var y = new Angle(45, 18, 31);
var z = new Angle(-30, 45, 7);

Assert.That(x == y, Is.True);
Assert.That(x != y, Is.False);
Assert.That(x == z, Is.False);
Assert.That(x != z, Is.True);
}

[TestCase(30, 40, 50, 20, 30, 40, 51, 11, 30)]
[TestCase(30, 40, 50, -20, 30, 40, 10, 10, 10)]
[TestCase(-30, 40, 50, 20, 30, 40, -10, 10, 10)]
[TestCase(30, 40, 50, 0, 0, 0, 30, 40, 50)]
public void AdditionTest(
int degrees1, int minutes1, int seconds1,
int degrees2, int minutes2, int seconds2,
int resultDegrees, int resultMinutes, int resultseconds)
{
var angle1 = new Angle(degrees1, minutes1, seconds1);
var angle2 = new Angle(degrees2, minutes2, seconds2);
var result = new Angle(resultDegrees, resultMinutes, resultseconds);

Assert.That (angle1 + angle2, Is.EqualTo(result));
}

[TestCase(2, 31, 41, 51, 63, 23, 42)]
[TestCase(0.5, 31, 41, 51, 15, 50, 56)]
public void MultiplicationTest(double k,
```

```
int degrees, int minutes, int seconds,
int resultDegrees, int resultMinutes, int resultseconds)
{
var angle = new Angle(degrees, minutes, seconds);
var result = new Angle(resultDegrees, resultMinutes, resultseconds);

Assert.That(k * angle, Is.EqualTo(result));
}
}
}
```

LMS-платформа − не предусмотрена

5.3. Описание контрольно-оценочных мероприятий промежуточного контроля по дисциплине модуля

5.3.1. Зачет

Список примерных вопросов

1. Обработка строк. ANSI и Unicode.
2. Проблемы плавающей арифметики.
3. Иерархия классов, представляющих геометрические фигуры.
4. Агрегирование объектов на примере составной фигуры.

LMS-платформа − не предусмотрена

5.3.2. Экзамен

Список примерных вопросов

1. Бинарные деревья поиска.
2. Проблема утечки памяти.
3. Контейнеры и итераторы STL.
4. Ввод-вывод в STL.
5. Отладка и тестирование.
6. Доказательство правильности программ.
7. Введение в Windows GUI.
8. Разработка своей программы с графическим интерфейсом.

LMS-платформа − не предусмотрена

5.4 Содержание контрольно-оценочных мероприятий по направлениям воспитательной деятельности

Направление воспитательной деятельности	Вид воспитательной деятельности	Технология воспитательной деятельности	Компетенция	Результаты обучения	**Контрольно-оценочные мероприятия**
Профессиональное воспитание	учебно-исследовательская, научно-исследовательск	Технология самостоятельной работы	ПК-4	З-1	Лабораторные занятия

	ая				