

## Рабочая программа дисциплины (модуля) «Программирование», включая оценочные материалы

### 1. Требования к результатам обучения по дисциплине (модулю)

#### 1.1. Перечень компетенций, формируемых дисциплиной (модулем) в процессе освоения образовательной программы

Группа компетенций	Категория компетенций	Коды и содержание компетенций
Универсальные	-	-
Общепрофессиональные	-	ОПК-6. Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий
	-	ОПК-7. Способен осуществлять выбор платформ и инструментальных программно-аппаратных средств для реализации информационных систем
Профессиональные	-	-

#### 1.2. Компетенции и индикаторы их достижения, формируемых дисциплиной (модулем) в процессе освоения образовательной программы

Код компетенции	Код индикатора компетенции	Содержание индикатора компетенции
ОПК-6	ОПК-6.1	Применяет методы проектирования программного обеспечения
ОПК-6	ОПК-6.2	Использует современные языки программирования для разработки алгоритмов и программ
ОПК-6	ОПК-6.3	Применяет методы отладки и тестирования программ
ОПК-7	ОПК-7.1	Анализирует практики использования основных концепций, принципов, теорий и фактов, связанных с информатикой, в профессиональной деятельности
ОПК-7	ОПК-7.2	Способен примечать в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой.
ОПК-7	ОПК-7.3.	Использует в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой.

#### 1.3. Результаты обучения по дисциплине (модулю)

**Цель изучения дисциплины (модуля)** – изучение теоретических основ и выработка навыков применения современных методов программирования

В результате изучения дисциплины (модуля) обучающийся должен

**знать:**

- принципы структурного и объектно-ориентированного программирования; основные синтаксические конструкции современных языков программирования; основные сведения о дискретных структурах, используемых в персональных компьютерах, основные алгоритмы типовых численных методов решения математических задач, один из языков программирования; принципы структурного и объектно-ориентированного программирования; основные синтаксические конструкции современных языков программирования.

**уметь:**

- применять языки программирования высокого уровня для разработки информационных систем в области решения прикладных задач; выполнять анализ алгоритмов, используя методы математического анализа; использовать языки системы программирования для решения профессиональных задач, работать с программными средствами общего назначения; решать типовые задачи по основным разделам курса;

**владеть:**

- методами построения алгоритмической модели профессиональных задач и содержательной интерпретации полученных результатов, навыками разработки программ на одном из универсальных языков программирования высокого уровня в современных средах разработки приложений.

## 2. Объем, структура и содержание дисциплины (модуля)

### 2.1. Объем дисциплины (модуля)

Виды учебной работы	Формы обучения
	Очная
<b>Общая трудоемкость:</b> зачетные единицы/часы	8/288
<b>Контактная работа:</b>	144
Лекции	72
Лабораторные работы	0
Практические занятия, семинары	72
<b>Промежуточная аттестация:</b> экзамен	72
<b>Самостоятельная работа (СР)</b>	72

### 2.2. Темы (разделы) дисциплины (модуля) с указанием отведенного на них количества часов по формам образовательной деятельности

#### Очная форма обучения

№ п/п	Наименование тем (разделов)	Виды учебной работы (в часах)						СР
		Контактная работа						
		Занятия лекционного типа		Занятия семинарского типа				
		Л	Иные	ПЗ	С	ЛР	Иные	
1.	Языки программирования. Введение в программирование на языке C#	2	0	2	0	0	0	4
2.	Инструментальная среда разработки Visual Studio	2	0	2	0	0	0	5
3.	Операции и операторы в языке C#	4	0	4	0	0	0	5
4.	Массивы и строки в C#. Ссылочные типы и типы значения	4	0	4	0	0	0	5
5.	Коллекции, строки, файлы	4	0	4	0	0	0	5
6.	Тестирование	4	0	4	0	0	0	5
7.	Сложность алгоритмов	4	0	4	0	0	0	5
8.	Рекурсивные алгоритмы	4	0	4	0	0	0	5
9.	Алгоритмы поиска и сортировки	4	0	4	0	0	0	5
10.	Основы объектно-ориентированного программирования	2	0	2	0	0	0	4
11.	Наследование	2	0	2	0	0	0	5
12.	Целостность данных	4	0	4	0	0	0	5
13.	Структуры	4	0	4	0	0	0	5
14.	Очереди, стеки, дженерики	4	0	4	0	0	0	5
15.	Yield return	4	0	4	0	0	0	5
16.	Листы и словари	4	0	4	0	0	0	5
17.	Делегаты	4	0	4	0	0	0	5
18.	LINQ	4	0	4	0	0	0	5

**Примечания:**

Л – лекции, ПЗ – практические занятия, С – семинары, ЛР – лабораторные работы, СР – самостоятельная работа.

## 2.3. Содержание дисциплины (модуля), структурированное по темам (разделам) и видам работ

### Содержание лекционного курса

№ п/п	Наименование тем (разделов)	Содержание лекционного курса
1.	Языки программирования. Введение в программирование на языке C#	Эволюция языков программирования. Классификация языков программирования. Базовые концепции платформы .NET: исполняющая среда CLR, общая система типов CTS, CLS, библиотека базовых классов .NET, сборки, CIL код, метаданные типов, манифест сборки, характеристика типов CTS.
2.	Инструментальная среда разработки Visual Studio	Инструментальные средства разработки, на примере Visual Studio: проект, сборка, решение, ссылки на другие сборки, создание проекта, пространства имен.
3.	Операции и операторы в языке C#	Арифметические операции. Операции сравнения и логический тип, сравнение чисел с плавающей точкой, полные и сокращенные операции сравнения.
4.	Массивы и строки в C#. Ссылочные типы и типы значения	Инициализация одномерного массива, обращение к элементам массива, использование foreach, короткая форма записи, тип object (минусы универсальной инициализации). Типы ссылки и типы значения. Место, для хранения контекстов методов. Глобальные переменные.
5.	Коллекции, строки, файлы	Проблемы использования массивов. Списки. Методы: Insert vs Add. Словари, пример, основные методы работы со словарями. Строка vs массив символов. Тип String. Неизменяемость строк, пример на картах памяти.
6.	Тестирование	Понятие тестирования. Автоматическое тестирование. Библиотеки. Простой пример автоматического тестирования: разделение кода, создание библиотеки, создание тестирующего модуля. Модульные тесты. Создание модульных тестов в Visual Studio на простом примере.
7.	Сложность алгоритмов	Важность изучения алгоритмов. Сложность алгоритмов. Базовые понятия: задача, алфавит, алгоритм, программа, временная сложность алгоритма, емкостная сложность алгоритма. Пример расчета сложности простого алгоритма.
8.	Рекурсивные алгоритмы	Простой рекурсивный алгоритм: пример на картах памяти. База рекурсии. Останов рекурсии. Переполнение стека. Дебаг рекурсии. Дерево рекурсии (на примере). Сложность рекурсивного алгоритма. Пример расчета сложности рекурсивного алгоритма.
9.	Алгоритмы поиска и сортировки	Алгоритм линейного поиска: идея, код, анализ. Алгоритм бинарного поиска: идея, код, анализ (проверка корректности работы алгоритма). Алгоритм бинарного поиска: идея, код, анализ (оценка сложности работы алгоритма).
10.	Основы объектно-ориентированного программирования	Классы. Пример инкапсуляции синтаксически не связанных между собой переменных, в отдельную сущность. Преимущества использования классов. Инкапсуляция. Пример карт памяти при использовании классов. Классы vs объекты.
11.	Наследование	Постановка проблемы. Иерархия наследования. Конверсия к типу родителя (upcast). Конверсия к типу наследника (downcast). Класс Array.
12.	Целостность данных	Постановка проблемы. Защита целостности данных, условия целостности, ключевое слово private. Общая проблема безопасности.
13.	Структуры	Объявление структуры. Классы vs структуры. Структуры на картах памяти. Инициализация полей структуры. Передача структур в метод vs передача класса в метод.
14.	Очереди, стеки, дженерики	Стек. Стеки для анализа скобочных выражений. Стеки для вычисления (обратная польская запись). Очередь. Очередь на

		связных списках. Универсальная очередь и даункасты.
15.	Yield return	foreach, IEnumerable и IEnumerator. Сущности интерфейсов IEnumerable и IEnumerator. Реализация интерфейса IEnumerable. Реализация интерфейса IEnumerator. Yield return.
16.	Листы и словари	Листы и индексация. Метод Contains. Листы и индексация. Метод Equals. Листы и индексация. Перегрузка операторов. Хеширующие функции. Примеры.
17.	Делегаты	Делегаты. Постановка проблемы. Делегат как тип данных. Делегат на картах памяти. Дженерик-делегаты. Func и Action.
18.	LINQ	Делегаты для диагностики кода (на примере сортировок). Делегаты в разборе арифметических выражений. Делегаты в вычислении производной. Лямбда-выражения в тестах. LINQ. LINQ to Objects. Метод Where в LINQ: реализация метода Where. Метод Select в LINQ: реализация метода Select. Метод ToList в языке LINQ.

### Содержание занятий семинарского типа

№ п/п	Наименование тем (разделов)	Тип	Содержание занятий семинарского типа
1.	Языки программирования. Введение в программирование на языке C#	ПЗ	Некоторые особенности языка C#. Базовые понятия языка C#: типы данных, конверсия типов, методы, сигнатура метода, перегрузка методов, переменные, области видимости.
2.	Инструментальная среда разработки Visual Studio	ПЗ	Техника разработки программ. Компиляция проекта. Классификация ошибок в программе. Ошибки на этапе компиляции и на этапе выполнения. Причины run-time ошибок. Блок try-catch. Отладка программ. Стилистические ошибки.
3.	Операции и операторы в языке C#	ПЗ	Операторы if ...else, типичные ошибки ветвлений. Циклы: while, for. Сравнение while и for. Цикл foreach.
4.	Массивы и строки в C#. Ссылочные типы и типы значения	ПЗ	Карты памяти. Хранение массива в памяти. Сборка мусора. Хранение строк. Отличие ссылочных типов, от типов значений. Передача типа значения и ссылочного типа в метод. Многомерные массивы и массивы массивов. Массивы массивов vs двумерные массивы.
5.	Коллекции, строки, файлы	ПЗ	Паттерн Immutable, тип StringBuilder, пример использования StringBuilder. StringvsStringBuilder. Вывод на консоль: специальные символы. Форматированный вывод. Работа с файлами и каталогами. Кодировки.
6.	Тестирование	ПЗ	Метки тестирующих классов и методов. Тело тестирующего метода. Покрытие тестами. Значение тестирования. Внедрение тестов. Функциональное тестирование. TDD. Пример из робототехники. Плюсы и минусы функционального тестирования. Значение тестирования. Внедрение тестов.
7.	Сложность алгоритмов	ПЗ	Масштаб роста функций. О-символика. Пример расчета сложности простого алгоритма, с использованием О-символики (из лекций). О-символика. Пример анализа алгоритма поиска делителей числа n (из лекций). Классы сложности. Сложность задачи.
8.	Рекурсивные алгоритмы	ПЗ	Принцип разделяй и властвуй. Задача перебора всех подмножеств, для заданного множества. Задача разбиения множества чисел на 2 части, таких что $\sum$ чисел в каждой из них одинакова. Задача перестановок. Задача коммивояжера. Задача размещения.
9.	Алгоритмы поиска и сортировки	ПЗ	Сравнение производительности работы алгоритмов линейного и бинарного поиска: обзор кода из лекции. Сортировка пузырьком: идея, код, анализ. Сортировка слиянием: идея, код, анализ. Быстрая сортировка: идея, код, анализ. Сравнение производительности работы алгоритмов сортировки: обзор кода из лекции.
10.	Основы объектно-	ПЗ	Статические и динамические поля класса: карты памяти.

	ориентированного программирования		Статические и динамические методы. Методы расширения. DirectoryInfo, FileInfo. Статические классы vs динамические классы. Паттерн Singleton.
11.	Наследование	ПЗ	Интерфейсы. Примеры работы с интерфейсами: реализация IComparable, реализация IComparer. Полиморфизм. Виртуальные методы.
12.	Целостность данных	ПЗ	Отложенные ошибки и контроль целостности. Свойства в C#, доступ к полям через свойства. Конструкторы, readonly поля.
13.	Структуры	ПЗ	Ключевое слово ref. Boxing/Unboxing. Свойства в структурах. Когда используются структуры.
14.	Очереди, стеки, дженерики	ПЗ	Реализация очереди при помощи дженерик-параметра. Очередь для скользящего среднего. Дженерики и сортировка массивов. Проблема возвращения нескольких значений из метода (7 решений).
15.	Yield return	ПЗ	Ленивые коллекции. Задача генерации бесконечных последовательностей. yield return в рекурсивных методах. Примеры: перебор всех подмножеств, множества, разбиение множества.
16.	Листы и словари	ПЗ	Использование хэшей в алгоритме поиска подстроки в строке. Класс Dictionary. GetHashCode.
17.	Делегаты	ПЗ	Анонимные делегаты. Лямбда-выражение. Примеры лямбд. Замыкание: как работает замыкание, пример на картах памяти, ловушка замыкания (пример)
18.	LINQ	ПЗ	Последовательность вызовов Where и Select. Последовательность вызовов с ToList. Особенности последовательности вызовов ленивых и обычных методов. Фильтрация и преобразование. Методы фильтрации и преобразования. Take, Skip, ToArray, ToList. Method chaining. SelectMany. OrderBy и Distinct. Функции агрегирования. Группировка. ToDictionary и ToLookup.

### Содержание самостоятельной работы

№ п/п	Наименование тем (разделов)	Содержание самостоятельной работы
1.	Языки программирования. Введение в программирование на языке C#	Эволюция языков программирования. Классификация языков программирования.
2.	Инструментальная среда разработки Visual Studio	Отладка программ. Стилистические ошибки.
3.	Операции и операторы в языке C#	Арифметические операции.
4.	Массивы и строки в C#. Ссылочные типы и типы значения	Многомерные массивы и массивы массивов. Массивы массивов vs двумерные массивы.
5.	Коллекции, строки, файлы	Вывод на консоль: специальные символы. Форматированный вывод. Работа с файлами и каталогами. Кодировки.
6.	Тестирование	Плюсы и минусы функционального тестирования. Значение тестирования. Внедрение тестов.
7.	Сложность алгоритмов	Классы сложности. Сложность задачи.
8.	Рекурсивные алгоритмы	Задача перестановок. Задача коммивояжера. Задача размещения.
9.	Алгоритмы поиска и сортировки	Сравнение производительности работы алгоритмов сортировки: обзор кода из лекции.
10.	Основы объектно-ориентированного программирования	Классы. Статические классы vs динамические классы. Паттерн Singleton.
11.	Наследование	Полиморфизм. Виртуальные методы.
12.	Целостность данных	Конструкторы, readonly поля.
13.	Структуры	Свойства в структурах. Когда используются структуры.
14.	Очереди, стеки, дженерики	Проблема возвращения нескольких значений из метода (7

		решений).
15.	Yield return	Примеры: перебор всех подмножеств, множества, разбиение множества.
16.	Листы и словари	Листы и индексация. Листы и индексация.
17.	Делегаты	Делегаты. Постановка проблемы. Делегат как тип данных.
18.	LINQ	Делегаты для диагностики кода (на примере сортировок). Делегаты в разборе арифметических выражений. Делегаты в вычислении производной.

### 3. Оценочные материалы для проведения текущего контроля успеваемости и промежуточной аттестации обучающихся по дисциплине (модулю)

По дисциплине (модулю) предусмотрены следующие виды контроля качества освоения:

- текущий контроль успеваемости;
- промежуточная аттестация обучающихся по дисциплине (модулю).

#### 3.1. Оценочные материалы для проведения текущей аттестации по дисциплине (модулю)

№ п/п	Контролируемые темы (разделы)	Наименование оценочного средства
1.	Языки программирования. Введение в программирование на языке C#	Устный опрос. Кейсы. Дискуссионные процедуры
2.	Инструментальная среда разработки Visual Studio	Устный опрос. Кейсы. Дискуссионные процедуры
3.	Операции и операторы в языке C#	Устный опрос. Кейсы. Дискуссионные процедуры
4.	Массивы и строки в C#. Ссылочные типы и типы значения	Устный опрос. Кейсы. Дискуссионные процедуры
5.	Коллекции, строки, файлы	Устный опрос. Кейсы. Дискуссионные процедуры
6.	Тестирование	Устный опрос. Кейсы. Дискуссионные процедуры
7.	Сложность алгоритмов	Устный опрос. Кейсы. Дискуссионные процедуры
8.	Рекурсивные алгоритмы	Устный опрос. Кейсы. Дискуссионные процедуры
9.	Алгоритмы поиска и сортировки	Устный опрос. Кейсы. Дискуссионные процедуры
10.	Основы объектно-ориентированного программирования	Устный опрос. Кейсы. Дискуссионные процедуры
11.	Наследование	Устный опрос. Кейсы. Дискуссионные процедуры
12.	Целостность данных	Устный опрос. Кейсы. Дискуссионные процедуры
13.	Структуры	Устный опрос. Кейсы. Дискуссионные процедуры
14.	Очереди, стеки, дженерики	Устный опрос. Кейсы. Дискуссионные процедуры
15.	Yield return	Устный опрос. Кейсы. Дискуссионные процедуры
16.	Листы и словари	Устный опрос. Кейсы. Дискуссионные процедуры
17.	Делегаты	Устный опрос. Кейсы. Дискуссионные процедуры
18.	LINQ	Устный опрос. Кейсы. Дискуссионные процедуры

#### 3.1.1 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности в процессе текущего контроля успеваемости

Устный опрос. Кейсы (ситуации и задачи с заданными условиями). Дискуссионные процедуры (круглый стол, дискуссия, полемика, диспут, дебаты, мини-конференции)

#### Семестр 3

##### Занятие № 1. Ветвления и циклы.

Вопросы для устного опроса:

1. Операции сравнения и логический тип.
2. Сравнение целых чисел.
3. Сравнение чисел с плавающей точкой.
4. Полные и сокращенные операции сравнения.
5. Операторы if ...else, типичные ошибки ветвлений.
6. Циклы: while, for.
7. Сравнение while и for.
8. Цикл foreach.

*Вопросы для дискуссии:*

1. Перечислите операции сравнения в языке программирования C#.
2. Каким образом осуществляется сравнение целых чисел?
3. В чем состоит специфика сравнения чисел с плавающей точкой?
4. Для чего используется конструкция `if ...else`?
5. В чем отличие циклов `while` и `for`? В каких ситуациях лучше использовать каждый из этих циклов?
6. Когда следует использовать `foreach`?

*Кейсы (решение задач)*

1. Протестировать работу приведенных конструкций ветвления и циклов.
2. Напишите метод, склоняющий существительное «рублей» следующее за указанным числительным. Например, для аргумента 10, метод должен вернуть «рублей», для 1 — вернуть «рубль», для 2 — «рубля».
3. Вам даны два прямоугольника на плоскости, со сторонами параллельными осям координат с целочисленными координатами. Реализуйте три метода для работы с прямоугольниками:
  - определение, есть ли у двух прямоугольников хотя бы одна общая точка (и граница и внутренность считаются частью прямоугольника);
  - вычисление площади пересечения;
  - определение, вложен ли один в другой.

Решите задание без использования библиотечных методов, кроме `Min` и `Max`. Обратите внимание, что решение должно корректно работать с вырожденными прямоугольниками: у которых длина или ширина равны 0. В мире компьютерной графики принято, что верхний левый угол экрана имеет координаты (0, 0), а ось Y направлена вниз, а не вверх, как принято в математике. Поэтому в этой задаче нижний край прямоугольника имеет большую координату, чем верхний.

4. Напишите метод вычисления расстояния от отрезка до точки. Расстоянием от отрезка до точки называется расстояние от ближайшей точки отрезка до точки. Это либо расстояние до точки от прямой, содержащей отрезок, либо расстояние до точки от одного из концов отрезка.
5. Подготовиться к ответам на вопросы.

**Занятие № 2. Массивы. Ссылочные типы и типы значения.**

*Вопросы для устного опроса:*

1. Инициализация одномерного массива.
2. Обращение к элементам массива.
3. Использование `foreach`, короткая форма записи.
4. Тип `object`.
5. Типы ссылки и типы значения.
6. Место, для хранения контекстов методов.
7. Глобальные переменные.
8. Карты памяти.
9. Хранение массива в памяти.
10. Сборка мусора.
11. Хранение строк.
12. Отличие ссылочных типов, от типов значений.
13. Передача типа значения и ссылочного типа в метод.
14. Многомерные массивы и массивы массивов.
15. Массивы массивов vs двумерные массивы.

*Вопросы для дискуссии:*

1. Для чего используются массивы?
2. Как массивы хранятся в памяти?
3. Что такое стек и куча?

4. В чем минусы универсальной инициализации?
5. В чем состоит отличие ссылочных типов от типов значений?
6. Для чего применяются карты памяти.
7. Приведите пример инициализации массива на картах памяти.
8. В чем состоит отличие двумерных массивов от массивов массивов.

*Кейсы (решение задач)*

1. Написать функцию для обмена строк двумерного массива с ее помощью отсортировать массив по элементам третьего столбца.
2. Написать процедуру для суммирования матриц. С ее помощью сложить исходную матрицу и транспонированную.
3. Написать функцию для удаления строки из двумерного массива. Оставшиеся строки должны быть расположены плотно, недостающие элементы заменяются 0. С помощью разработанных функций исключить из массива строки с номерами от А до В.
4. Определить является ли матрица ортонормированной, т. е. такой, что скалярное произведение каждой пары различных строк равно 0, а скалярное произведение строки самой на себя равно 1.
5. Подготовиться к ответам на вопросы.

### **Занятие № 3. Коллекции: списки и словари.**

*Вопросы для устного опроса:*

1. Проблемы использования массивов.
2. Устройство списка.
3. Список с дженерик типом.
4. Пример реализации списка.
5. Методы: Insert vs Add.
6. Словари, пример, основные методы работы со словарями.

*Вопросы для дискуссии:*

1. В чем состоит проблема использования массивов?
2. В каком пространстве имен .NET содержится класс для объявления списка?
3. Приведите пример реализации списка через массив.
4. В чем отличие метода Insertи Addдля списков?
5. Какую структуру имеет словарь?
6. Когда применяются словари.
7. Перечислите основные методы для работы со словарями.

*Кейсы (решение задач)*

1. Реализовать парсер предложений, который должен выполнять следующее:
  - разделять текст на предложения, а предложения на слова:
    - Считайте, что слова состоят только из букв (используйте метод `char.IsLetter`) или символа апострофа ' и отделены друг от друга любыми другими символами.
    - Предложения состоят из слов и отделены друг от друга одним из следующих символов. !?;:()
  - приводить символы каждого слова в нижний регистр;
  - пропускать предложения, в которых не оказалось слов;
  - должен возвращать список предложений, где каждое предложение — это список из одного или более слов в нижнем регистре.
2. Реализовать метод, который по списку предложений, составленному в прошлой задаче, составляет словарь самых частотных продолжений биграмм и триграмм. Это словарь, ключами которого являются все возможные начала биграмм и триграмм, а значениями — их самые частотные продолжения. Если есть несколько продолжений с одинаковой частотой, используйте то, которое лексикографически меньше. Для лексикографического сравнения используйте встроенный в .NET



способ сравнения Ordinal, например, с помощью метода `string.CompareOrdinal`. Такой словарь назовём N-граммной моделью текста.

- N-грамма — это N соседних слов в одном предложении. 2-граммы называют биграммами. 3-граммы — триграммами.
- Например, из текста: "She stood up. Then she left." Можно выделить следующие биграммы "she stood", "stood up", "then she" и "she left", none "up then". Идветриграммы "she stood up" и "then she left", none "stood up then".

3. Реализовать алгоритм продолжения текста по N-граммной модели.

Описание алгоритма:

- На вход алгоритму передается словарь `nextWords`, полученный в предыдущей задаче, одно или несколько первых слов фразы `phraseBeginning` и `wordsCount` — количество слов, которые нужно дописать к `phraseBeginning`.
- Словарь `nextWords` в качестве ключей содержит либо отдельные слова, либо пары слов, соединённые через пробел. По ключу `key` содержится слово, которым нужно продолжать фразы, заканчивающиеся на `key`.

Алгоритм должен работать следующим образом:

- Итоговая фраза должна начинаться `sphraseBeginning`. К ней дописывается `wordsCount` слов таким образом:
  - Если фраза содержит как минимум два слова и в словаре есть ключ, состоящий из двух последних слов фразы, то продолжать нужно словом, из словаря по этому ключу.
  - Иначе, если в словаре есть ключ, состоящий из одного последнего слова фразы, то продолжать нужно словом, хранящемся в словаре по этому ключу.
  - Иначе, нужно досрочно закончить генерирование фразы и вернуть сгенерированный на данный момент результат.

4. Подготовиться к ответам на вопросы.

#### **Занятие № 4. Рекурсивные алгоритмы. Алгоритм бинарного поиска.**

*Вопросы для устного опроса:*

1. Простой рекурсивный алгоритм: пример на картах памяти.
2. База рекурсии.
3. Останов рекурсии. Переполнение стека. Дебаг рекурсии.
4. Дерево рекурсии (на примере).
5. Сложность рекурсивного алгоритма.
6. Пример расчета сложности рекурсивного алгоритма.
7. Принцип разделяй и властвуй.
8. Задача перебора всех подмножеств, для заданного множества.
9. Задача разбиения множества чисел на 2 части, таких что  $\sum$  чисел в каждой из них одинакова.
10. Задача перестановок.
11. Задача коммивояжера. Задача размещения.
12. Алгоритм бинарного поиска: идея, код, анализ (оценка сложности работы алгоритма).

*Вопросы для дискуссии:*

1. В чем состоит идея рекурсии?
2. Что такое база рекурсии?
3. Каким образом осуществляется расчет сложности рекурсивного алгоритма?
4. Приведите пример построения дерева рекурсии.
5. В чем состоит принцип разделяй и властвуй?
6. В чем состоит принцип решения задачи коммивояжера?
7. Приведите пример работы рекурсивного алгоритма на картах памяти.
8. Чему равна сложность алгоритма бинарного поиска?

*Кейсы (решение задач)*

1. Написать функцию, которая по заданному паролю (некоторое слово) перебирает все возможные пароли, полученные из этого слова заменой регистра, которые должны появляться в лексикографическом порядке, считая, что маленькие буквы меньше больших. Естественно, регистр нужно менять только у букв. Например, для входного слова 'ab42' результат должен быть такой: 'ab42', 'aB42', 'Ab42', 'AB42'. На вход подается слово в нижнем регистре. В результирующем списке не должно быть повторений слов.
2. Робота нужно проехать через указанные точки, посетив каждую хотя бы один раз. Нужно спланировать маршрут так, чтобы суммарный путь был минимален. Функция принимает массив чекпоинтов (точек на плоскости с координатами {x, y}). Робот изначально находится в точке checkpoints[0]. Вернуть нужно порядок посещения чекпоинтов. Например, если функция возвращает массив {0,2,1}, это означает, что робот сначала поедет в чекпоинт с индексом 2, а из него в чекпоинт с индексом 1 и на этом закончит свой путь. Функция должна быть рекурсивной. Реализуйте следующую оптимизацию (отсечение перебора): прекращайте перебор, если текущая длина пути уже больше, чем минимальный путь, найденный ранее.
3. Реализовать алгоритм бинарного поиска с помощью рекурсии.
4. Подготовиться к ответам на вопросы.

**Семестр 4**

**Занятие № 1. Основы объектно-ориентированного программирования.**

*Вопросы для устного опроса:*

1. Классы.
2. Пример инкапсуляции синтаксически не связанных между собой переменных, в отдельную сущность.
3. Преимущества использования классов.
4. Инкапсуляция.
5. Пример карт памяти при использовании классов.
6. Классы vs объекты.
7. Статические и динамические поля класса: карты памяти.
8. Статические и динамические методы.
9. Методы расширения.
10. Статические классы vs динамические классы.
11. Паттерн Singleton.

*Вопросы для дискуссии:*

1. Дайте определение класса.
2. Приведите пример, когда использование классов улучшает программную реализацию. Почему?
3. Что такое инкапсуляция.
4. Приведите пример использования класса на картах памяти.
5. В чем состоит разница между классом и объектом?
6. В чем состоит разница между статическими и динамическими полями/методами?
7. В чем удобство применения методов расширения?
8. Какой параметр обязательно идет первым у метода расширения?
9. Для чего используется паттерн Singleton?

*Кейсы (решение задач)*

1. Практика 1. «Вектор»
  - Создайте новый проект в Visual Studio. Выберите в качестве типа проекта Class Library.
  - В этом проекте создайте два класса, Vector и Geometry, в пространстве имен GeometryTasks.
  - В классе Vector должно быть два публичных поля, X и Y, типа double.

- В классе Geometry должно быть два статических метода: GetLength, который возвращает длину переданного вектора, и Add, который возвращает сумму двух переданных векторов
2. Практика 2. «Отрезок»
- Продолжайте разработку геометрической библиотеки.
  - Создайте класс Segment, представляющий отрезок прямой. Концы его отрезков должны задаваться двумя публичными полями: Begin и End типа Vector.
  - Добавьте метод Geometry.GetLength, вычисляющий длину сегмента, и метод Geometry.IsVectorInSegment(Vector, Segment), проверяющий, что задаваемая вектором точка лежит в отрезке.
3. Практика 3. «Нестатические методы»
- Вы вдруг поняли, что не очень-то удобно писать имя класса Geometry при выполнении любой операции с векторами и сегментами. Однако, отказаться от этого класса вы не можете, потому что за те несколько минут, пока вы сдавали предыдущую задачу, вашу библиотеку скачали и начали использовать в своих проектах тысячи человек.
  - Поэтому вы решили сохранить этот класс, но добавить методы Vector.GetLength(), Segment.GetLength(), Vector.Add(Vector), Vector.Belongs(Segment) и Segment.Contains(Vector) не вместо, а вместе с соответствующими методами класса Geometry.
  - Каждый из этих методов должен вызывать уже существующий метод класса Geometry, чтобы не дублировать код.
4. Практика 4. «256 оттенков серого»
- Некто хочет использовать вашу геометрическую библиотеку для рисования. Для этого ему необходимо, чтобы у вашего класса Segment появился цвет. Однако, вам кажется, что втаскивать цвета в чисто геометрическую сущность - плохая идея.
  - Скачайте проект GeometryPainting, установите в нем reference на вашу библиотеку, и после этого сделайте так, чтобы методы GetColor и SetColor появились в вашем классе Segment.
  - Если цвет не задан, GetColor возвращает Color.Black.

5. Подготовиться к ответам на вопросы.

## **Занятие № 2. Стеки, очереди, дженерики.**

*Вопросы для устного опроса:*

1. Стек.
2. Стеки для анализа скобочных выражений.
3. Стеки для вычисления (обратная польская запись).
4. Очередь.
5. Очередь на связных списках.
6. Универсальная очередь и даункасты.
7. Реализация очереди при помощи дженерик-параметра.
8. Очередь для скользящего среднего.

*Вопросы для дискуссии:*

1. Дайте определение стека.
2. Каким образом можно использовать стек для задачи анализа скобочных выражений?
3. Дайте определение очереди.
4. Приведите пример добавление элемента в очередь на связных списках.
5. Приведите пример удаления элемента из очереди на связных списках.
6. Для чего используется дженерик параметр в объявлении списка/очереди?
7. Приведите пример использования очереди в задаче вычисления скользящего

среднего.

*Кейсы (решение задач)*

1. Практика «Limited Size Stack». В этой задаче вам нужно реализовать стек ограниченного размера. Этот стек работает как обычный стек, однако при превышении максимального размера удаляет самый глубокий элемент в стеке. Таким образом, в стеке всегда будет ограниченное число элементов. Вот пример работы такого стека с ограничением в 2 элемента:

```
// сначала стек пуст
stack.Push(10); // в стеке 10
stack.Push(20); // в стеке 10, 20
stack.Push(30); // в стеке 20, 30
stack.Push(40); // в стеке 30, 40
stack.Pop(); // возвращает 40, в стеке остаётся 30
stack.Pop(); // возвращает 30, стек после этого пуст
```

Операция Push должна иметь сложность  $O(1)$ , то есть никак не зависеть от размера стека.

- У каждой коллекции в C# доступен метод расширения Last(). Однако, работает он за  $O(1)$  только для коллекций, реализующих интерфейс IList (список с доступом к элементам по индексу). Для остальных коллекций он работает за  $O(N)$ , перебирая её элементы до конца. Будьте осторожны.
  - Для эффективной реализации такого стека подойдет двусвязный список. Он уже реализован в классе LinkedList. LinkedList не реализует интерфейс IList, поэтому Last() для него работает медленно. Однако у него есть собственное свойство Last, работающее быстро.
2. Практика «Отмена». Продолжайте работу в том же проекте LimitedSizeStack. Ваша задача — сделать так, чтобы была возможность отменять последнее действие пользователя. Задание:
    - Реализуйте методы Undo и CanUndo. Для этого нужно хранить историю последних limit действий удаления/добавления. Используйте для этого класс LimitedSizeStack из прошлой задачи. Его не нужно включать в отправляемый на проверку файл, считайте, что этот класс уже есть в проекте.
    - Метод Undo отменяет последнее действие из истории.
    - Метод CanUndo возвращает true, если на данный момент история действий не пуста, то есть если вызов Undo будет корректным. Иначе метод должен вернуть false.
    - Если хотите, можете воспользоваться классическим объектно-ориентированным шаблоном Команда. Однако для сдачи данной задачи, точно следовать этому шаблону необязательно.
    - В LimitedSizeStack нужно хранить какие-то данные, описывающие выполненное действие. В них должно быть достаточно информации для отмены действия.

3. Подготовиться к ответам на вопросы.

### **Занятие № 3. Листы и словари.**

*Вопросы для устного опроса:*

1. Листы и индексация.
2. Метод Contains.
3. Метод Equals.
4. Перегрузка операторов.
5. Хеширующие функции.
6. Примеры.
7. Использование хэшей в алгоритме поиска подстроки в строке.
8. Класс Dictionary. GetHashCode.

*Вопросы для дискуссии:*

1. Приведите на примере внутреннее устройство листов.

2. Как организовать итерацию для листов?
3. Как организовать индексацию для листов?
4. Что такое перегрузка операторов?
5. Приведите пример перегрузки операторов?
6. Для чего используется GetHashCode?

*Кейсы (решение задач)*

1. Практика «Readonly bytes». *Скачайте проект readonly-bytes.*
  - Иногда есть смысл в качестве ключей в Dictionary или HashSet использовать массивы байт. Однако по умолчанию массивы сравниваются по ссылкам, а не по содержимому, а часто нужно именно по содержимому. В таких случаях можно написать класс-обёртку над массивом, который переопределит Equals и GetHashCode так, чтобы сравнение происходило по содержимому. В этой задаче вам нужно создать именно такую обёртку.
  - Задание: В файле ReadonlyBytes.cs создайте класс ReadonlyBytes так, чтобы все тесты из файла ReadonlyBytesTests.cs компилировались и проходили.
2. Практика «Ghosts». *Скачайте проект ghost.*
  - Неаккуратная реализация Equals и GetHashCode может приводить к тому, что добавленный в Dictionary или HashSet ключ внезапно исчезает. Чтобы не попадаться на подобные ошибки в будущем, в этом задании предлагается поизучать всевозможные подобные ошибки.
  - Задание:
  - В проекте вам даны несколько классов с уже реализованными GetHashCode и Equals. Вам нужно придумать, как их использовать, чтобы HashSet стал вести себя некорректно.
  - Изучите тест GhostsTest.cs и в файле GhostsTask.cs создайте класс GhostsTask так, чтобы этот тест проходил
3. Подготовиться к ответам на вопросы.

### **3.1.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности в ходе текущего контроля успеваемости**

#### **Устный ответ**

Оценка знаний предполагает дифференцированный подход к обучающемуся, учет его индивидуальных способностей, степень усвоения и систематизации основных понятий и категорий по дисциплине. Кроме того, оценивается не только глубина знаний поставленных вопросов, но и умение использовать в ответе практический материал. Оценивается культура речи, владение навыками ораторского искусства.

*Критерии оценивания:* последовательность, полнота, логичность изложения, анализ различных точек зрения, самостоятельное обобщение материала, использование профессиональных терминов, культура речи, навыки ораторского искусства. Изложение материала без фактических ошибок.

Оценка «*отлично*» ставится в случае, когда материал излагается исчерпывающе, последовательно, грамотно и логически стройно, при этом раскрываются не только основные понятия, но и анализируются точки зрения различных авторов. Обучающийся не затрудняется с ответом, соблюдает культуру речи.

Оценка «*хорошо*» ставится, если обучающийся твердо знает материал, грамотно и по существу излагает его, знает практическую базу, но при ответе на вопрос допускает несущественные погрешности.

Оценка «*удовлетворительно*» ставится, если обучающийся освоил только основной материал, но не знает отдельных деталей, допускает неточности, недостаточно правильные формулировки, нарушает последовательность в изложении материала, затрудняется с ответами, показывает отсутствие должной связи между анализом, аргументацией и выводами.

Оценка «*неудовлетворительно*» ставится, если обучающийся не отвечает на поставленные вопросы.

#### **Кейсы (ситуации и задачи с заданными условиями)**

Обучающийся должен уметь выделить основные положения из текста задачи, которые требуют анализа и служат условиями решения. Исходя из поставленного вопроса в задаче, попытаться максимально точно определить проблему и соответственно решить ее.

Задачи могут решаться устно и/или письменно. При решении задач также важно правильно сформулировать и записать вопросы, начиная с более общих и, кончая частными.

*Критерии оценивания* – оценка учитывает методы и средства, использованные при решении ситуационной, проблемной задачи.

Оценка «*отлично*» ставится в случае, когда обучающийся выполнил задание (решил задачу), используя в полном объеме теоретические знания и практические навыки, полученные в процессе обучения.

Оценка «*хорошо*» ставится, если обучающийся в целом выполнил все требования, но не совсем четко определяется опора на теоретические положения, изложенные в научной литературе по данному вопросу.

Оценка «*удовлетворительно*» ставится, если обучающийся показал положительные результаты в процессе решения задачи.

Оценка «*неудовлетворительно*» ставится, если обучающийся не выполнил все требования.

#### **Дискуссионные процедуры**

*Круглый стол, дискуссия, полемика, диспут, дебаты, мини-конференции* являются средствами, позволяющими включить обучающихся в процесс обсуждения спорного вопроса, проблемы и оценить их умение аргументировать собственную точку зрения. Задание дается заранее, определяется круг вопросов для обсуждения, группы участников этого обсуждения.

Дискуссионные процедуры могут быть использованы для того, чтобы студенты:

- лучше поняли усвояемый материал на фоне разнообразных позиций и мнений, не обязательно достигая общего мнения;
- смогли постичь смысл изучаемого материала, который иногда чувствуют интуитивно, но не могут высказать вербально, четко и ясно, или конструировать новый смысл, новую позицию;
- смогли согласовать свою позицию или действия относительно обсуждаемой проблемы.

*Критерии оценивания* – оцениваются действия всех участников группы. Понимание проблемы, высказывания и действия полностью соответствуют заданным целям. Соответствие реальной действительности решений, выработанных в ходе игры. Владение терминологией, демонстрация владения учебным материалом по теме игры, владение методами аргументации, умение работать в группе (умение слушать, конструктивно вести беседу, убеждать, управлять временем, бесконфликтно общаться), достижение игровых целей, (соответствие роли – при ролевой игре). Ясность и стиль изложения.

Оценка «*отлично*» ставится в случае, когда все требования выполнены в полном объеме.

Оценка «*хорошо*» ставится, если обучающиеся в целом демонстрируют понимание проблемы, высказывания и действия полностью соответствуют заданным целям. Решения, выработанные в ходе игры, полностью соответствуют реальной действительности. Но некоторые объяснения не совсем аргументированы, нарушены нормы общения, нарушены временные рамки, нарушен стиль изложения.

Оценка «*удовлетворительно*» ставится, если обучающиеся в целом демонстрируют понимание проблемы, высказывания и действия в целом соответствуют заданным целям.

Однако, решения, выработанные в ходе игры, не совсем соответствуют реальной действительности. Некоторые объяснения не совсем аргументированы, нарушены временные рамки, нарушен стиль изложения.

Оценка «неудовлетворительно» ставится, если обучающиеся не понимают проблему, их высказывания не соответствуют заданным целям.

### 3.2. Оценочные материалы для проведения промежуточной аттестации

#### 3.2.1. Критерии оценки результатов обучения по дисциплине (модулю)

Шкала оценивания	Результаты обучения	Показатели оценивания результатов обучения
ОТЛИЧНО	Знает:	<ul style="list-style-type: none"> <li>- обучающийся глубоко и всесторонне усвоил материал, уверенно, логично, последовательно и грамотно его излагает, опираясь на знания основной и дополнительной литературы,</li> <li>- на основе системных научных знаний делает квалифицированные выводы и обобщения, свободно оперирует категориями и понятиями.</li> </ul>
	Умеет:	- обучающийся умеет самостоятельно и правильно решать учебно-профессиональные задачи или задания, уверенно, логично, последовательно и аргументировано излагать свое решение, используя научные понятия, ссылаясь на нормативную базу.
	Владеет:	<ul style="list-style-type: none"> <li>- обучающийся владеет рациональными методами (с использованием рациональных методик) решения сложных профессиональных задач, представленных деловыми играми, кейсами и т.д.;</li> <li>При решении продемонстрировал навыки</li> <li>- выделения главного,</li> <li>- связкой теоретических положений с требованиями руководящих документов,</li> <li>- изложения мыслей в логической последовательности,</li> <li>- самостоятельного анализа факты, событий, явлений, процессов в их взаимосвязи и диалектическом развитии.</li> </ul>
ХОРОШО	Знает:	<ul style="list-style-type: none"> <li>- обучающийся твердо усвоил материал, достаточно грамотно его излагает, опираясь на знания основной и дополнительной литературы,</li> <li>- затрудняется в формулировании квалифицированных выводов и обобщений, оперирует категориями и понятиями, но не всегда правильно их верифицирует.</li> </ul>
	Умеет:	- обучающийся умеет самостоятельно и в основном правильно решать учебно-профессиональные задачи или задания, уверенно, логично, последовательно и аргументировано излагать свое решение, не в полной мере используя научные понятия и ссылки на нормативную базу.
	Владеет:	<ul style="list-style-type: none"> <li>- обучающийся в целом владеет рациональными методами решения сложных профессиональных задач, представленных деловыми играми, кейсами и т.д.;</li> <li>При решении смог продемонстрировать достаточность, но не глубинность навыков,</li> <li>- выделения главного,</li> <li>- изложения мыслей в логической последовательности,</li> <li>- связки теоретических положений с требованиями руководящих документов,</li> <li>- самостоятельного анализа факты, событий, явлений, процессов в их взаимосвязи и диалектическом развитии.</li> </ul>
УДОВЛЕТВОРИТЕЛЬНО	Знает:	<ul style="list-style-type: none"> <li>- обучающийся ориентируется в материале, однако затрудняется в его изложении;</li> <li>- показывает недостаточность знаний основной и дополнительной литературы;</li> <li>- слабо аргументирует научные положения;</li> <li>- практически не способен сформулировать выводы и обобщения;</li> <li>- частично владеет системой понятий.</li> </ul>
	Умеет:	- обучающийся в основном умеет решить учебно-профессиональную задачу или задание, но допускает ошибки, слабо аргументирует свое решение, недостаточно использует научные понятия и руководящие документы.
	Владеет:	- обучающийся владеет некоторыми рациональными методами

		<p>решения сложных профессиональных задач, представленных деловыми играми, кейсами и т.д.;</p> <p>При решении продемонстрировал недостаточность навыков</p> <ul style="list-style-type: none"> <li>- выделения главного,</li> <li>- изложения мыслей в логической последовательности,</li> <li>- связи теоретических положений с требованиями руководящих документов,</li> <li>- самостоятельного анализа факты, событий, явлений, процессов в их взаимосвязи и диалектическом развитии.</li> </ul>
НЕУДОВЛЕТВОРИТЕЛЬНО	Знает:	<ul style="list-style-type: none"> <li>- обучающийся не усвоил значительной части материала;</li> <li>- не может аргументировать научные положения;</li> <li>- не формулирует квалифицированных выводов и обобщений;</li> <li>- не владеет системой понятий.</li> </ul>
	Умеет:	обучающийся не показал умение решать учебно-профессиональную задачу или задание.
	Владет:	не выполнены требования, предъявляемые к навыкам, оцениваемым «удовлетворительно».

### 3.2.2. Контрольные задания и/или иные материалы для проведения промежуточной аттестации

#### Список вопросов для устных ответов (варианты теста)

##### Варианты теста

1. Какая компонента платформы .NET отвечает за управление памятью?

+: Common Language Runtime

-: Common Type System

-: Common Language Specification

2. Укажите истинные утверждения.

+: В языке программирования C# управление памятью осуществляется автоматически посредством сборки мусора.

-: В языке программирования C# не поддерживается технология аспектно-ориентированного программирования, через атрибуты.

+: В языке программирования C# предлагаются формальные синтаксические конструкции для делегатов.

-: Код, ориентируемый на выполнение в исполняющей среде .NET, называется неуправляемым кодом.

3. Что следует изменить в коде выше? Выберите все верные варианты.

```

1 namespace Slide01
2 {
3     class Program
4     {
5         static void Main()
6         {
7             //требуется вывести на консоль площади трех кругов с радиусами 1, 2 и 3
8             var circleArea1 = 3.14159265 * 1 * 1;
9             var circleArea2 = 3.14159265 * 2 * 2;
10            var circleArea3 = 3.14159265 * 3 * 3;
11            Console.WriteLine(circleArea1);
12            Console.WriteLine(circleArea2);
13            Console.WriteLine(circleArea3);
14        }
15    }
16 }

```

-: Ничего, код великолепен!

+: Использовать константу Math.PI, вместо 3.14

+: Выделить нахождение площади круга в отдельный метод

-: Переименовать Main в PrintNumbers

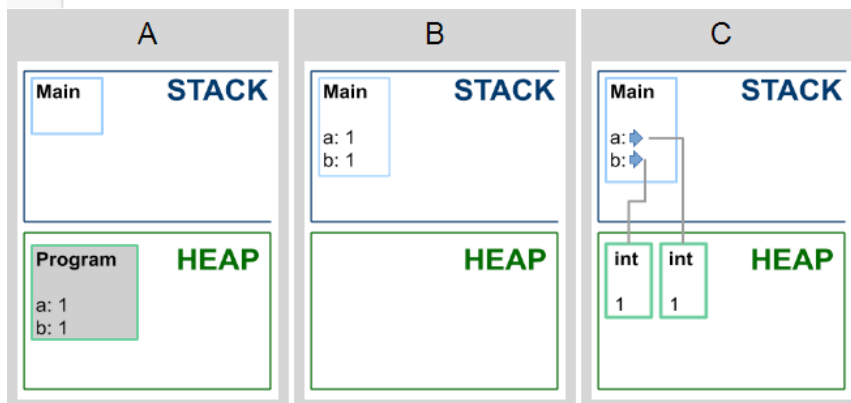
4. Какая из карт памяти соответствует коду к моменту выхода из метода Main?



```

1 public class Program
2 {
3     public static void Main()
4     {
5         int a = 1;
6         int b = 1;
7         // ?
8     }
9 }

```



-: A

+: B

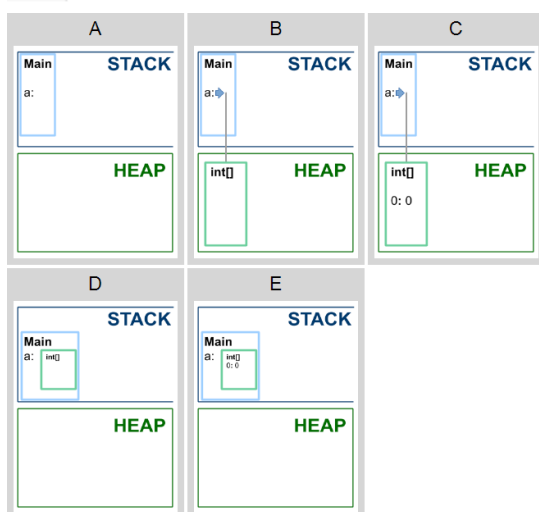
-: C

5.Какая из карт памяти соответствует коду к моменту выхода из метода Main?

```

1 public class Program
2 {
3     static void Main()
4     {
5         int[] a = new int[0];
6         // ?
7     }
8 }

```



-: A

+: B

-: C

-: D

-: E

6.Массивы отлично использовать для...

- + : хранения набора однотипных данных
- : хранения набора разнотипных данных
- : хранения разных свойств одного объекта
- + : поиска значения по его номеру

7.Что напечатает код `Console.WriteLine("12345\n321");`

- + : две строки: 12345 и 321
- : одну строку: 32145
- : одну строку: 12345321
- : одну строку: 12345\n321
- : одну строку: 12345n321

8.Чтобы создать модульный тест с помощью встроенной в Visual Studio системы тестирования, нужно:

- + : Создать новый проект с типом Test Project
- + : Добавить в этом проекте ссылку на сборку с тестируемым кодом
- + : Создать класс с тестами и пометить его атрибутом [TestClass]
- + : Создать в классе нестатический public void-метод без параметров
- + : Пометить этот метод атрибутом [TestMethod]
- + : Написать код подготовки исходных данных, код вызова тестируемого метода и код проверки результата вызова

9.Дополнительные плюсы наличия автоматических тестов

- + : Тратится меньше усилий на выявление и устранение ошибок после старта эксплуатации
- + : Рефакторинг становится менее опасной затеей
- : Повышается скорость разработки
- + : Повышается доверие к коду у других программистов

10.Любой ли алгоритм с конечным входом и выходом можно рассматривать как функцию преобразующую входное слово в выходное слово?

- : Нет, только алгоритмы работы со строками
- + : Да, так как любая информация (как переданная на вход, так и вычисленная алгоритмом) может быть описана последовательностью байт, а байты можно считать буквами алфавита.

11.Какие оценки сложности алгоритма  $F$  в зависимости от числа  $n$  (а не от размера входа) верны?

```
1 int F(int n){
2     int sum = 0;
3     for(int i=0; i<n; i++)
4         for(int j=0; j<i; j+=2)
5             sum += j;
6     return sum;
7 }
```

- + :  $O(n^2)$
- + :  $O(n^3)$
- :  $O(n)$
- :  $o(n^2)$
- + :  $o(n^3)$
- :  $\Theta(n)$
- + :  $\Theta(n^2)$
- :  $\Theta(n^3)$

**Вопросы для устных ответов**

1. Базовые концепции платформы .NET: исполняющая среда CLR, общая система типов CTS, CLS, библиотека базовых классов .NET, сборки, CIL код, метаданные типов, манифест сборки, характеристика типов CTS.
2. Базовые понятия языка C#: типы данных, конверсия типов, методы, сигнатура метода, перегрузка методов, переменные, области видимости.
3. Инструментальные средства разработки, на примере Visual Studio: проект, сборка, решение, ссылки на другие сборки, создание проекта, пространства имен.
4. Техника разработки программ. Компиляция проекта. Классификация ошибок в программе.
5. Ошибки на этапе компиляции и на этапе выполнения. Причины run-time ошибок. Блок try-catch. Отладка программ. Стилистические ошибки.
6. Арифметические операции. Операции сравнения и логический тип, сравнение чисел с плавающей точкой, полные и сокращенные операции сравнения.
7. Операторы if ...else, типичные ошибки ветвлений. Циклы: while, for. Сравнение while и for. Цикл foreach.
8. Ссылочные типы и типы значения. Инициализация одномерного массива, обращение к элементам массива, использование foreach, короткая форма записи, тип object (минусы универсальной инициализации).
9. Типы ссылки и типы значения. Место, для хранения контекстов методов. Глобальные переменные. Карты памяти.
10. Хранение массива в памяти. Сборка мусора. Хранение строк. Отличие ссылочных типов, от типов значений.
11. Передача типа значения и ссылочного типа в метод.
12. Многомерные массивы и массивы массивов. Массивы массивов vs двумерные массивы.
13. Списки. Методы: Insert vs Add. Словари, пример, основные методы работы со словарями.
14. Строка vs массив символов. Тип String. Неизменяемость строк, пример на картах памяти. Паттерн Immutable, тип StringBuilder, пример использования StringBuilder. StringvsStringBuilder.
15. Вывод на консоль: специальные символы. Форматированный вывод.
16. Работа с файлами и каталогами. Кодировки.
17. Автоматическое тестирование. Библиотеки. Простой пример автоматического тестирования: разделение кода, создание библиотеки, создание тестирующего модуля.
18. Модульные тесты. Создание модульных тестов в Visual Studio на простом примере. Метки тестирующих классов и методов. Тело тестирующего метода.
19. Покрывание тестами. Значение тестирования. Внедрение тестов.
20. Функциональное тестирование. TDD. Пример из робототехники. Плюсы и минусы функционального тестирования.
21. Значение тестирования. Внедрение тестов.
22. Сложность алгоритмов. Базовые понятия: задача, алфавит, алгоритм, программа, временная сложность алгоритма, емкостная сложность алгоритма.
23. Пример расчета сложности простого алгоритма. Масштаб роста функций.
24. O-символика. Пример расчета сложности простого алгоритма, с использованием O-символики (из лекций). O-символика.
25. Пример анализа алгоритма поиска делителей числа n (из лекций). Классы сложности. Сложность задачи.
26. Простой рекурсивный алгоритм: пример на картах памяти. База рекурсии. Останов рекурсии. Переполнение стека. Дебаг рекурсии.
27. Дерево рекурсии (на примере). Сложность рекурсивного алгоритма. Пример расчета сложности рекурсивного алгоритма.

28. Принцип разделяй и властвуй.
29. Задача перебора всех подмножеств, для заданного множества.
30. Задача разбиения множества чисел на 2 части, таких что  $\sum$  чисел в каждой из них одинакова.
31. Задача перестановок.
32. Задача коммивояжера.
33. Задача размещения.
34. Алгоритм линейного поиска: идея, код, анализ.
35. Алгоритм бинарного поиска: идея, код, анализ (оценка сложности работы алгоритма).
36. Сортировка пузырьком: идея, код, анализ.
37. Сортировка слиянием: идея, код, анализ.
38. Быстрая сортировка: идея, код, анализ.
39. Основы объектно-ориентированного программирования. Классы. Пример инкапсуляции синтаксически не связанных между собой переменных, в отдельную сущность. Преимущества использования классов.
40. Инкапсуляция. Пример карт памяти при использовании классов. Классы vs объекты.
41. Статические и динамические поля класса: карты памяти. Статические и динамические методы.
42. Методы расширения. DirectoryInfo, FileInfo. Статические классы vs динамические классы. Паттерн Singleton.
43. Наследование. Постановка проблемы. Иерархия наследования.
44. Иерархия наследования. Конверсия к типу родителя (upcast).
45. Иерархия наследования. Конверсия к типу наследника (downcast).
46. Интерфейсы. Примеры работы с интерфейсами: реализация IComparable, реализация IComparer.
47. Полиморфизм. Виртуальные методы.
48. Целостность данных. Постановка проблемы. Защита целостности данных, условия целостности, ключевое слово private.
49. Общая проблема безопасности. Отложенные ошибки и контроль целостности.
50. Свойства в C#, доступ к полям через свойства. Конструкторы, readonly поля.
51. Структуры. Объявление структуры. Классы vs структуры. Структуры на картах памяти. Инициализация полей структуры.
52. Передача структур в метод vs передача класса в метод. Ключевое слово ref.
53. Boxing/Unboxing. Свойства в структурах. Когда используются структуры.
54. Очереди, стеки, дженерики. Стек. Стеки для анализа скобочных выражений. Стеки для вычисления (обратная польская запись).
55. Очередь. Очередь на связанных списках. Универсальная очередь и даункасты. Реализация очереди при помощи дженерик-параметра.
56. Очередь для скользящего среднего. Дженерики и сортировка массивов.
57. Проблема возвращения нескольких значений из метода (7 решений).
58. Yield return. foreach, IEnumerable и IEnumerator. Сущности интерфейсов IEnumerable и IEnumerator.
59. Реализация интерфейса IEnumerable. Реализация интерфейса IEnumerator. Yield return. Ленивые коллекции.
60. Задача генерации бесконечных последовательностей. yield return в рекурсивных методах. Примеры: перебор всех подмножеств, множества, разбиение множества.
61. Листы и словари. Листы и индексация. Метод Contains. Метод Equals.
62. Листы и индексация. Перегрузка операторов.
63. Хеширующие функции. Примеры. Использование хэшей в алгоритме поиска подстроки в строке.

64. Класс Dictionary. GetHashCode.
65. Делегаты. Делегаты. Постановка проблемы.
66. Делегат как тип данных. Делегат на картах памяти.
67. Дженерик-делегаты. Func и Action. Анонимные делегаты.
68. Лямбда-выражение. Примеры лямбд.
69. Замыкание: как работает замыкание, пример на картах памяти, ловушка замыкания (пример)
70. Делегаты для диагностики кода (на примере сортировок).
71. Делегаты в разборе арифметических выражений.
72. Делегаты в вычислении производной.
73. Лямбда-выражения в тестах. LINQ. LINQ to Objects.
74. Метод Where в LINQ: реализация метода Where. Метод Select в LINQ: реализация метода Select.
75. Метод ToList в языке LINQ. Последовательность вызовов Where и Select. Последовательность вызовов с ToList. Особенности последовательности вызовов ленивых и обычных методов.
76. Фильтрация и преобразование. Методы фильтрации и преобразования. Take, Skip, ToArray, ToList.
77. Method chaining. SelectMany. OrderBy и Distinct.
78. Функции агрегирования. Группировка. ToDictionary и ToLookup.

**Тексты проблемно-аналитических и (или) практических учебно-профессиональных задач (типовой вариант)**

Реализовать парсер предложений, который должен выполнять следующее:

- разделять текст на предложения, а предложения на слова:
  - o Считайте, что слова состоят только из букв (используйте метод `char.IsLetter`) или символа апострофа и отделены друг от друга любыми другими символами.
  - o Предложения состоят из слов и отделены друг от друга одним из следующих символов: `! ? ; ( )`
- приводить символы каждого слова в нижний регистр;
- пропускать предложения, в которых не оказалось слов;
- должен возвращать список предложений, где каждое предложение — это список из одного или более слов в нижнем регистре.

**Решение:**

```
using System.Collections.Generic;
using System.Text;
namespace TextAnalysis
{
    internal static class SentencesParserTask
    {
        public static readonly char[] SentenceDelimiters = {'!', '!',
                                                            '?', ';', ':', '(', ')'};

        public static List<List<string>> ParseSentences(string text)
        {
            var sentences = text.Split(SentenceDelimiters);
            var sentencesList = new List<List<string>>();
            foreach (var sentence in sentences)
            {
                var words = ParseSentence(sentence);
                if (words.Count > 0) sentencesList.Add(words);
            }
        }
    }
}
```

```

        return sentencesList;
    }
    public static List<string> ParseSentence(string sentence)
    {
        var wordBuilder = new StringBuilder();
        var words = new List<string>();
        foreach (var symbol in sentence + ".")
            if (char.IsLetter(symbol) || symbol == "\")
                wordBuilder.Append(char.ToLower(symbol));
            else if (wordBuilder.Length > 0)
            {
                words.Add(wordBuilder.ToString());
                wordBuilder.Clear();
            }
        return words;
    }
}

```

### 3.2.3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков в ходе промежуточной аттестации

#### Процедура оценивания знаний (тест)

Предлагаемое количество заданий	20
Последовательность выборки	Определена по разделам
Критерии оценки	- правильный ответ на вопрос
«5» если	правильно выполнено 90-100% тестовых заданий
«4» если	правильно выполнено 70-89% тестовых заданий
«3» если	правильно выполнено 50-69% тестовых заданий

#### Процедура оценивания знаний (устный ответ)

Предел длительности	10 минут
Предлагаемое количество заданий	2 вопроса
Последовательность выборки вопросов из каждого раздела	Случайная
Критерии оценки	- требуемый объем и структура - изложение материала без фактических ошибок - логика изложения - использование соответствующей терминологии - стиль речи и культура речи - подбор примеров их научной литературы и практики
«5» если	требования к ответу выполнены в полном объеме
«4» если	в целом выполнены требования к ответу, однако есть небольшие неточности в изложении некоторых вопросов
«3» если	требования выполнены частично – не выдержан объем, есть фактические ошибки, нарушена логика изложения, недостаточно используется соответствующая терминологии

#### Процедура оценивания умений и навыков (решение проблемно-аналитических и практических учебно-профессиональных задач)

Предлагаемое количество заданий	1
Последовательность выборки	Случайная
Критерии оценки:	- выделение и понимание проблемы - умение обобщать, сопоставлять различные точки зрения - полнота использования источников - наличие авторской позиции - соответствие ответа поставленному вопросу - использование социального опыта, материалов СМИ, статистических данных - логичность изложения - умение сделать квалифицированные выводы и обобщения с

	точки зрения решения профессиональных задач - умение привести пример - опора на теоретические положения - владение соответствующей терминологией
«5» если	требования к ответу выполнены в полном объеме
«4» если	в целом выполнены требования к ответу, однако есть небольшие неточности в изложении некоторых вопросов. Затрудняется в формулировании квалифицированных выводов и обобщений
«3» если	требования выполнены частично – пытается обосновать свою точку зрения, однако слабо аргументирует научные положения, практически не способен самостоятельно сформулировать выводы и обобщения, не видит связь с профессиональной деятельностью

#### **4. Учебно-методическое и материально-техническое обеспечение дисциплины (модуля)**

##### **4.1. Электронные учебные издания**

1. Губарь, Ю. В. Введение в математическое программирование : учебное пособие / Ю. В. Губарь. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 225 с. — ISBN 978-5-4497-0872-4. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/101994.html>. — Режим доступа: для авторизир. пользователей
2. Давыдов, А. Н. Линейное программирование: графический и аналитический методы : учебное пособие / А. Н. Давыдов. — Самара : Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2014. — 106 с. — ISBN 978-5-9585-0604-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/43184.html>. — Режим доступа: для авторизир. пользователей
3. Лебедева, Т. Н. Теория и практика объектно-ориентированного программирования : учебное пособие / Т. Н. Лебедева. — 2-е изд. — Челябинск, Саратов : Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 221 с. — ISBN 978-5-4486-0663-2. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/81498.html>. — Режим доступа: для авторизир. пользователей. - DOI: <https://doi.org/10.23682/81498>
4. Объектно-ориентированное программирование на C++ : учебник / И. В. Баранова, С. Н. Баранов, И. В. Баженова [и др.]. — Красноярск : Сибирский федеральный университет, 2019. — 288 с. — ISBN 978-5-7638-4034-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/100067.html>. — Режим доступа: для авторизир. пользователей

##### **4.2. Электронные образовательные ресурсы**

1. Электронная библиотечная система «ЭБС ЮРАЙТ» Biblio-online.ru (ЭБС «Юрайт») [Электронный ресурс]. — URL: <https://urait.ru/>.
2. Электронно-библиотечная система ZNANIUM [Электронный ресурс]. — URL: <https://znanium.com/>.
3. Электронная библиотечная система «Консультант студента» [Электронный ресурс]. — URL: <https://www.studentlibrary.ru/>.
4. e-Library.ru: Научная электронная библиотека [Электронный ресурс]. — URL: <http://elibrary.ru/>.
5. Научная электронная библиотека «КиберЛенинка» [Электронный ресурс]. — URL: <http://cyberleninka.ru/>.

6. Информационная система «Единое окно доступа к образовательным ресурсам» [Электронный ресурс]. – URL: <http://window.edu.ru/>.
7. Федеральный центр информационно-образовательных ресурсов [Электронный ресурс]. – URL: <http://fcior.edu.ru/>.

#### **4.3. Современные профессиональные базы данных и информационные справочные системы**

Обучающимся обеспечен доступ (удаленный доступ) к ниже следующим современным профессиональным базам данных и информационным справочным системам:

1. Словари и энциклопедии на Академике [Электронный ресурс]. – URL: <http://dic.academic.ru>.
2. Система информационно-правового обеспечения «Гарант» [Электронный ресурс]. – URL: <http://ivo.garant.ru/>.

#### **4.4. Комплект лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства**

1. Лицензионное программное обеспечение: операционная система Microsoft Windows, пакет офисных приложений Microsoft Office.
2. Свободно распространяемое программное обеспечение: свободные пакеты офисных приложений Apache Open Office, LibreOffice.
3. Программное обеспечение отечественного производства: справочно-правовая система «Гарант» (Электронный периодический справочник «Система ГАРАНТ»), образовательная платформа ЮРАЙТ (Электронная библиотечная система «ЭБС ЮРАЙТ» Biblio-online.ru (ЭБС «Юрайт»)), электронно-библиотечная система ZNANIUM, электронная библиотечная система «Консультант студента».

#### **4.5. Оборудование и технические средства обучения**

Для реализации дисциплины (модуля) используются учебные аудитории для проведения учебных занятий, которые оснащены оборудованием и техническими средствами обучения, и помещения для самостоятельной работы обучающихся, которые оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечены доступом в электронную информационно-образовательную среду РХТУ им. Д.И. Менделеева. Допускается замена оборудования его виртуальными аналогами.

Наименование учебных аудиторий для проведения учебных занятий и помещений для самостоятельной работы*	Оснащенность учебных аудиторий для проведения учебных занятий и помещений для самостоятельной работы оборудованием и техническими средствами обучения
Учебные аудитории для проведения учебных занятий	Учебная аудитория укомплектована специализированной мебелью, отвечающей всем установленным нормам и требованиям, оборудованием и техническими средствами обучения (мобильное мультимедийное оборудование).
Помещение для самостоятельной работы	Помещение оснащено компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду РХТУ им. Д.И. Менделеева и к ЭБС.

\* Номер конкретной аудитории указан в приказе об аудиторном фонде, расписании учебных занятий и расписании промежуточной аттестации.