МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ
ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
«КОЛЬСКИЙ НАУЧНЫЙ ЦЕНТР РОССИЙСКОЙ АКАДЕМИИ НАУК»

(ФИЦ КНЦ РАН)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

По дисциплине Б1.О.09 Современные технологии программирования
указывается цикл (раздел) ОП, к которому относится дисциплина, название дисциплины

для направления подготовки (специальности) 09.04.02 Информационные системы и технологии
код и наименование направления подготовки (специальности)

направленность программы (профиль) Информационные системы предприятий и учреждений
наименование профиля /специализаций/образовательной программы

Квалификация выпускника, уровень подготовки
Магистр
(указывается квалификация (степень) выпускника в соответствии с ФГОС ВО)

Апатиты

2020

Лист согласования

1 Разработчик:

доцент	УАиМ		Н.А. Тоичкин
должность		подпись	И.О. Фамилия

2 Методические указания рассмотрены и одобрены на заседании учебно-методической комиссии управления аспирантуры и магистратуры 29 июня 2020 г., протокол № 02.

Председатель УМК УАиМ

29.06.2020		Л.Д. Кириллова
дата	подпись	И.О.Фамилия

Пояснительная записка

1. **Методические указания** составлены в соответствии с требованиями федерального государственного образовательного стандарта по образовательной программе высшего образования – программе магистратуры по направлению подготовки 09.04.02 Информационные системы и технологии, утвержденного приказом Минобрнауки России от 19.09.2017 № 917.

2. **Цель дисциплины (модуля)** формирование у обучающихся целостного представления о современных подходах к созданию программных продуктов; технологиях, методах и инструментальных средств разработки, модификации и сопровождения программных комплексов и систем.

Задачи дисциплины:

- получить навыки использования современных техник программирования разработки прикладного программного обеспечения;
- изучить работу с разнообразными структурами данных;
- научиться готовить тестовые задания для отладки отдельных программных модулей и программного комплекса в целом.

3. Требования к уровню подготовки обучающегося в рамках данной дисциплины.

Процесс изучения дисциплины (модуля) «Современные технологии программирования» направлен на формирование элементов следующих компетенций в соответствии с ФГОС ВО 09.04.02 Информационные системы и технологии (уровень магистратуры), представленных в таблице 1.

Таблица 1 – Компетенции, формируемые в процессе изучения дисциплины «Современные технологии программирования»

№ п/п	Код компетенции	Содержание компетенции
1.	ОПК-2	Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач.

4. Планируемые результаты обучения по дисциплине (модулю) «Современные технологии программирования».

Результаты формирования компетенций и обучения представлены в таблице 2.

Таблица 2 – Планируемые результаты обучения

№ п/п	Код компетенции	Компоненты компетенции, степень их реализации	Результаты обучения
1.	ОПК-2	Компоненты компетенции со-	***Знать:***

		относятся с содержанием дисциплины и компетенция реализуется полностью	– современные концепции программирования: структурный подход, объектно-ориентированный подход, функциональный подходи их применение в языке C#; – технологии обобщенного программирования; – основные структуры данных и их применение при разработке различных алгоритмов; – задачи разработки прикладного программного обеспечения. ***Уметь:*** – разрабатывать иерархические схемы программ; – использовать механизм организации запросов к источнику данных LINQ для написания выразительного декларативного кода; ***Владеть:*** – методами разработки приложений с использованием возможностей современных технологий программирования; – навыками работы в современных средах разработки.

Таблица 3 - Перечень практических работ

№ п/п	Наименование практических работ	Количество часов	Наименование темы по табл. 4
1.	Программирование структур данных и перечисления коллекций.	4	2
2.	Делегаты и функциональное программирование.	6	3
3.	LINQ.	6	

Итого часов	**16**	

Рекомендации к выполнению практических работ

Практика № 1. Программирование структур данных и перечисления коллекций

План:

№ 1.1 «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.

Задание:

- Скачайте проект LimitedSizeStack. Реализуйте класс LimitedSizeStack.
- Отладьте его реализацию с помощью тестов в классе LimitedSizeStack_should. Проверьте эффективность операции Push с помощью теста из класса LimitedSizeStack_PerformanceTest.

№ 1.2. Отмена. Продолжайте работу в том же проекте LimitedSizeStack. Если вы запустите проект на исполнение, то увидите окно приложения, в котором можно добавлять новые дела и удалять уже существующие. Однако кнопка "Отмена" пока не работает. Ваша задача — сделать так, чтобы эта кнопка отменяла последнее действие пользователя.

Изучите класс ListModel — в нём реализована логика работы кнопок в приложении.

Задание:

Реализуйте методы Undo и CanUndo. Для этого нужно хранить историю последних limit действий удаления/добавления. Используйте для этого класс LimitedSizeStack из прошлой задачи.

- 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.

№ 1.3. «Экспоненциальное сглаживание»

Задание:

- Скачайте проект Smooth
- В классе ExpSmoothingTask реализуйте функцию экспоненциального сглаживания данных.
- Отладьте реализацию с помощью приложенных модульных тестов. Запустите тестирующее приложение и объясните наблюдаемый результат.

- Для перечисления элементов есть оператор foreach и методы LINQ. В отличие от интерфейса IEnumerator, они просты в использовании и легко читаются. Используйте работу через методы IEnumerator только, если задача не решается с помощью foreach или уже готовых методов LINQ.

№ 1.4. Практика «Скользящее среднее»

Задание:

- Продолжайте работу в том же проекте Smooth
- В классе MovingAverageTask реализуйте функцию скользящего среднего.
- При усреднении с окном размера W, первые W-1 точки результата в действительности должны усредняться по окнам меньшего размера. Так, первая точка должна попасть в результат без изменения. Отладьте реализацию с помощью приложенных модульных тестов.
- Запустите тестирующее приложение и объясните наблюдаемый результат.
- Для этой задачи вам пригодится структура данных Очередь. Не нужно создавать её самостоятельно, воспользуйтесь готовым классом Queue в пространстве имён System.Collections.Generic.

Практическая работа № 2. Делегаты и функциональное программирование

План:

№ 1.1. Проектирование виртуальной машины Brain

Задание:

Скачайте проект brain.

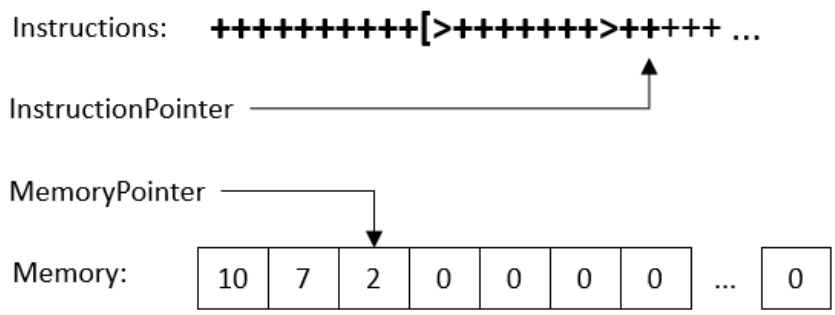
Создавать языки программирования сложно. Но не всегда! Язык программирования Brain — это экстремально простой язык программирования, интерпретатор при желании можно уместить на один экран кода. Программа на Brain состоит только из **символов +-<>.,[]**, поэтому читать такие программы не очень удобно.

В этой серии задач вам предстоит создать этот интерпретатор с возможностью его простого расширения новыми операциями.

В задачах программирования интерпретаторов часто оказываются удобными пройденные в этом блоке делегаты анонимные функции — решите эту задачу с помощью анонимных функций.

Виртуальная машина хранит следующее:

- Массив памяти, каждая ячейка которого хранит 1 байт. По умолчанию размер памяти — 30000 ячеек.
- Указатель на текущую ячейку памяти. Изначально, указатель указывает на нулевую ячейку.
- Выполняемую программу. Она состоит из инструкций, каждая обозначается одним символом. Программа начинает выполняться с первого символа последовательно.
- Номер выполняемой в данный момент инструкции. После выполнения любой инструкции номер увеличивается на единицу. Как только номер инструкции выходит за пределы программы, выполнение заканчивается.
- Конкретные операции на языке Brain могут читать или менять эти данные.



В этой части вам нужно реализовать виртуальную машину в классе VirtualMachine.cs так, чтобы проходили все тесты из файла VirtualMachineTests.cs.

№ 1.2. Команды виртуальной машины Brain

Задание:

Продолжайте работу в том же проекте brain.

Изучите класс Brain.cs, в частности то, как он использует реализованный ранее класс VirtualMachine.

В классе BrainBasicCommands реализуйте метод, регистрирующий следующие простые команды в виртуальную машину:

Символ	Значение
.	Вывести байт памяти, на который указывает указатель, преобразовав в символ согласно ASCII
+	Увеличить байт памяти, на который указывает указатель
-	Уменьшить байт памяти, на который указывает указатель
,	Ввести символ и сохранить его ASCII-код в байт памяти, на который указывает указатель
>	Сдвинуть указатель памяти вправо на 1 байт
<	Сдвинуть указатель памяти влево на 1 байт
A-Z, a-z, 0-9	сохранить ASCII-код этого символа в байт памяти, на который указывает указатель

Например, программа **++>+++.<.** выводит два символа с ASCII кодами 2 и 3, а память после выполнения команды будет выглядеть так [2, 3, 0, 0, ... 0].

Для ввода и вывода используйте переданные в метод Run функции Func<int> read и Action<char> write.

Тут read по аналогии с Console.Read возвращает либо код введенного символа, либо -1, если ввод закончился. Считайте, что на вход будут подаваться только символы с кодами 0..255 — они точно помещаются в один байт.

Детали реализации инструкций восстановите по тестам в классе BrainfuckBasicCommandsTests. Сделайте так, чтобы все тесты в этом файле проходили.

№ 1.3. Циклы Brain

Задание:

Продолжайте работу в том же проекте brain.

В классе BrainLoopCommands реализуйте метод, регистрирующий следующие команды в виртуальную машину:

Символ	Значение

Символ	Значение
[	(Начало цикла) Перескочить по программе вправо на соответствующий (с учетом вложенности) символ], если текущий байт памяти равен нулю. Продолжать исполнение с этого символа.
]	(Конец цикла) Перескочить по списку инструкций влево на соответствующий (с учетом вложенности) символ [, если текущий байт памяти НЕ равен нулю. Продолжать исполнение с этого символа.

Например, программа **++++++++[>+++++++++<-]>+.** выводит букву A (ASCII-код 65 получается увеличением 8 раз второй ячейки на 8, а потом добавлением ещё единицы).

Детали реализации инструкций восстановите по тестам в классе BrainLoopCommandsTests. Сделайте так, чтобы все тесты в этом файле проходили.

Практическая работа № 3. LINQ

План:

№ 1.1. Median & Bigrams

Задание:

Скачайте проект linq-slideviews.

В файле ExtensionsTask реализуйте два метода расширения: для вычисления медианы и для вычисления списка биграмм.

Эти методы пригодятся в будущем. Вы сможете их использовать на ряду и в перемешку с остальными методами LINQ.

Есть важное замечание по деталям реализации.

Создавая методы, работающие с IEnumerable стоит придерживаться следующих рекомендаций:

1. Если это возможно, не перечисляйте входной IEnumerable до конца. Потому что IEnumerable может теоретически быть бесконечным.
2. Не перечисляйте больше элементов, чем нужно для работы IEnumerable. Возможно, при перечислении лишнего элемента случится ошибка или другой нежелательный побочный эффект.
3. Не полагайтесь на то, что IEnumerable можно будет перечислить дважды. Этого никто не гарантирует. Кстати, некоторые IDE, автоматически находят нарушение этого пункта. Например, подобные предупреждения умеют показывать JetBrains Rider и Visual Studio с установленным Resharper.

№ 1.2. Чтение файла

Задание:

Продолжайте в том же проекте linq-slideviews.

В этой серии задач вам нужно будет проанализировать статистику посещения слайдов этого курса студентами.

Исходные данные содержатся в двух файлах:

slide.txt содержит информацию про каждый из слайдов — идентификатор, тип слайда (теория, задача или тест), и тема соответствующей недели. Пример файла slides.txt:

SlideId;SlideType;UnitTitle
0;theory;Первое знакомство с C#
1;quiz;Первое знакомство с C#
2;theory;Первое знакомство с C#
3;exercise;Первое знакомство с C#

visits.txt содержит по одной записи на первое посещение слайда каждым пользователем. Запись состоит из идентификатора пользователя, идентификатора слайда, даты и времени посещения этим пользователем этого слайда. Пример файла visits.txt:

UserId;SlideId;Date;Time
0;5;2014-09-03;12:20:28
1;6;2014-09-03;12:25:09
1;4;2014-09-03;12:25:24

В этой задаче в классе ParsingTask нужно реализовать методы чтения этих файлов.

Не используйте циклы в решении. Вместо этого используйте LINQ.

Обратите внимание, что в разных методах предлагается реализовать разную реакцию на некорректные строки файлов: в одном случае — игнорировать их, а в другом — выбрасывать исключение на первой же ошибочной строке. Это сделано исключительно в учебных целях — в реальных проектах стоит, конечно, придерживаться какой-то одной выбранной стратегии.

№ 1.3. Статистика

Продолжайте в том же проекте linq-slideviews.

В файле StatisticsTask реализуйте метод GetMedianTimePerSlide. Он должен работать так.

Обозначим T(U, S) время между посещением пользователем U слайда S и ближайшим следующим посещением тем же пользователем U какого-то другого слайда S2 != S.

T(U, S) можно считать примерной оценкой того, сколько времени пользователь U провел на слайде S.

Метод должен для указанного типа слайда, считать медиану значений T(U, S) по всем пользователям и всем слайдам этого типа.

Нужно игнорировать значения меньшие 1 минуты и большие 2 часов при расчете медианы.

Гарантируется, что в тестах и реальных данных отсутствуют записи, когда определенный пользователь заходит на один и тот же слайд более одного раза.

Время нужно возвращать в минутах.

Воспользуйтесь реализованными ранее методами Bigrams и Median.

ПЕРЕЧЕНЬ РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная литература:

1. Павловская Т.А., Программирование на языке высокого уровня C# / Павловская Т.А. - М.: Национальный Открытый Университет "ИНТУИТ", 2016. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/intuit_281.html (дата обращения: 06.12.2020).
2. Чеповский А.М., Common Intermediate Language и системное программирование в Microsoft .NET / Чеповский А.М., Макаров А.В., Скоробогатов С.Ю. - М.: Национальный Открытый Университет "ИНТУИТ", 2016. (Основы информатики и математики) - ISBN 5-94774-410-4. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN5947744104.html (дата обращения: 06.12.2020).
3. Лекции по курсу. Электронное изд. https://moodle.arcticsu.ru/course/view.php?id=159

Дополнительная литература:

4. Седжвик Р. Алгоритмы на C++. М.: Национальный Открытый Университет «ИНТУИТ». Режим доступа: https://biblioclub.ru/index.php?page=book_red&id=429164&sr=1

5. Тоичкин Н.А., Козлова Ю.Г., Богатиков В.Н. Паттерны проектирования: учеб.-метод. пособие по выполнению лаб. работ, (учебное пособие)/ Н.А. Тоичкин, В.Н. Богатиков, Ю.Г. Козлова. Тверь: ТвГТУ, 2015. 48 с. Электронное изд. https://moodle.arcticsu.ru/course/view.php?id=159