Приложение к рабочей
программе дисциплины

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Наименование дисциплины	*Программирование на основе C#*
Направление подготовки / Специальность	*09.03.03 Прикладная информатика*
Направленность (профиль) / Специализация	Разработка информационных систем бизнеса
Форма обучения	*очная*

Разработчик Ивашко А.Г. зав. кафедрой программной и системной инженерии

1. Темы дисциплины для самостоятельного освоения обучающимися
Отсутствуют.

2. План самостоятельной работы:

№ п/п	Учебные встречи	Виды самостоятельной работы	Форма отчетности / контроля	Количество баллов	Рекомендуемый бюджет времени на выполнение (ак.ч.)
1	2	3	4	5	6
1	Парадигмы программирования	Выполнение кейсовых заданий для погружения в тематику занятий	Сопоставительный анализ показателей экономического развития стран мира		5
2	Сложности ПО и методы борьбы со слоностями	Прослушивание подкаста для погружения в тематику занятий	Интеллект-карта		5
3	Проектирование сложных систем	Решение практических заданий по материалам лекций	Решение задач по изученным моделям		5
4	UML	Решение практических заданий по материалам лекций	Решение задач по изученным моделям		5
5	Особенности ООП в VS.C#	Решение практических заданий по материалам лекций	Решение задач по изученным моделям		5
6	Очереди, стеки, дженерики	Решение практических заданий по материалам лекций	Решение задач по изученным моделям		5
7	Ветвление, циклы ошибки в C#.	Изучение лекционного материала	Решение практических задач по вопросам лекционного материала		5
8	Массивы, коллекции, строки, файлы	Изучение лекционного материала	Решение практических задач по вопросам лекционного материала		5
9	Тестирование	Решение практических заданий по материалам практических работ	Решение задач по изученным моделям		5
10	Алгоритмы и оценка их сложности.	Изучение лекционного материала	Решение практических задач по вопросам лекционного		5

			материала		
11	Практикум	Изучение лекционного материала	Решение практических задач по вопросам лекционного материала		5
12	Основы ООП в C#	Повторение лекционного материала и решение заданий с практических занятий	Контрольная работа		5
	Итого			0	150

3. Требования и рекомендации по выполнению самостоятельных работ обучающихся, критерии оценивания.

Выполнение кейсовых заданий для погружения в тематику занятий.

Примерное задание.

Система, информационная система и понимание этого термина в различных нормативных документах, состав информационной системы, стандарты, лучшие практики и фреймворки в области информационных систем, CASE (computer-aided software engineering). Классификация ИС и история развития, Информационные системы функциональных областей (Functional Area Information Systems-FAIS), карьерный путь специалиста в области информационных систем и системного аналитика.

Сложность системы, признаки сложной системы, методы управления сложными системами: абстракция, декомпозиция, иерархия, проектирование сложных систем, методы проектирования и их сравнение

Генеалогия языков программирования. История развитие языков программирования. Соотнесение терминов Объектно-ориентированные программирование (OOP), Объектно-ориентированное проектирование (OOD), Объектно-ориентированный анализ (OOA). Основные составляющие объектной модели. Абстрагирование и выделение абстракций. Основные абстракции ООП, примеры абстракций информационной системы. Инкапсуляция, реализация инкапсуляции в C#.

Рекомендации по выполнению:

- изучить материалы лекционных презентаций, конспектов лекций, материалы, размещенные на веб

- освоить основные термины и понятия, макроэкономические показатели и способы их расчета, теоретические положения, формализованное и графическое представление макроэкономических моделей

- самостоятельность (можно пользоваться ИИ, но грамотно интерпретировать результаты анализа)

- визуализация работы за счет построения графиков по показателям

- в выводах необходимо опираться на лекционный материал и изученными в рамках практических занятий модели.

Решение практических заданий по материалам лекций.

Данный вид заданий носит разноплановый характер, нацелены на приобретение обучающимися навыков

В результате освоения дисциплины обучающийся должен:

знать:

– основные технологии программирования;

– основные сведения о дискретных структурах, используемых в персональных компьютерах, основные алгоритмы типовых численных методов решения

математических
задач, один из языков программирования структуру локальных и глобальных компьютерных сетей;
– знать и уметь применять технологии программирования и способы оптимизации передачи данных и способы обеспечения безопасности в компьютерных сетях.
уметь:
– ставить задачу и разрабатывать алгоритм ее решения, использовать прикладные системы программирования;
– работать с современными системами программирования, включая объектноориентированные;
– уметь использовать языки системы программирования для решения профессиональных задач, работать с программными средствами общего назначения;
решать типовые задачи по основным разделам курса, используя методы математического
анализа.
владеть:
– методами и инструментальными средствами разработки программ;
– языками процедурного и объектно-ориентированного программирования,
навыками разработки и отладки программ не менее, чем на одном из алгоритмических
процедурных языков программирования высокого уровня;
– методами построения математической модели профессиональных задач и содержательной интерпретации полученных результатов.

Рекомендации по выполнению:
- изучить материалы лекционных презентаций, конспектов лекций, материалы, размещенные в интернете
- освоить основные термины и понятия, макроэкономические показатели и способы их расчета, теоретические положения, формализованное и графическое представление макроэкономических моделей
- использовать следующие вспомогательные материалы:

1. Хорев, П. Б. Объектно-ориентированное программирование с примерами на C# : учебное пособие / П.Б. Хорев. - Москва : ФОРУМ : ИНФРА-М, 2020. - 200 с. - (Высшее образование: Бакалавриат). - ISBN 978-5-00091-680-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1069921 (дата обращения: 23.11.2022)
2. Гуриков, С. Р. Введение в программирование на языке Visual C# : учеб. пособие / С.Р. Гуриков. - Москва : ФОРУМ : ИНФРА-М, 2019. - 447 с. - (Высшее образование: Бакалавриат). - ISBN 978-5-16-105882-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1017998 (дата обращения: 23.11.2022)
3. Царев, Р.Ю. Информатика и программирование [Электронный ресурс] : учеб. пособие / Р. Ю. Царев, А. Н. Пупков, В. В. Самарин, Е. В. Мыльникова. - Красноярск : Сиб. федер. ун-т, 2014. - 132 с. - ISBN 978-5-7638-3008-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/506203 (дата обращения: 23.11.2022)
4. Бедердинова, О. И. Программирование на языках высокого уровня : учеб. пособие / О.И. Бедердинова, Т.А. Минеева, Ю.А. Водовозова. - Москва : ИНФРА-М, 2019. - 159 с. - Текст : электронный. - URL: https://znanium.com/catalog/product/1044396 (дата обращения: 23.11.2022)

Общие принципы работы преподавателя и студента

1. Именно студент должен нести ответственность за своё образование, а не преподаватель. Научиться программировать — в первую очередь цель

студента. Ваша цель — дать возможности развития, но не развивать насильно — это не работает.

2. Студенты должны действовать осознанно: они должны понимать ради чего они решают задачи и проходят курс, ради чего существуют контрольные мероприятия. Обсуждайте с ними это и объясняйте.
3. Хорошо, если студенты занимают активную позицию, интересуются предметом, организуют консультации, предлагают разумные изменения учебного процесса. Поощряйте такое поведение: поддерживайте их инициативы, благодарите за участие.
4. Поддерживайте у студентов ощущение безопасности. Они должны чувствовать, что преподаватель на их стороне и заинтересован в успехах студентов. В результате студенты будут готовы открыто обсуждать свои проблемы и проблемы курса с преподавателем.
5. Воздержитесь от изменения правил игры по ходу курса — это подрывает доверие. Если правила всё таки приходится изменить, то постарайтесь сделать это в пользу студентов.
6. Вы поможете студентам, если будет сочетать дружелюбие и последовательность. Командно-приказной тон может легко отбить интерес. Но строгость и последовательность соблюдения известных правил должна оставаться — ясные и предсказуемые рамки создают чувство безопасности и комфорт.
7. Давайте свободу действия в этих рамках. Студенты должны учиться пользоваться этой свободой.

Обучение в течение курса

Смешанное обучение (blended learning) — это подход, в котором дома студенты осваивают теоретический материал, а на очных занятиях занимаются практикой: решением задач, созданием проектов, работой в командах. Оно избавляет преподавателя от рутины чтения лекций, давая возможность потратить время на более сложный материал и индивидуальный подход.

Преимущества освоения теории дома:

- В комфортном темпе:
    - Сильные смотрят в ускоренном темпе и не скучают
    - Слабые смотрят в обычном с паузами и все успевают
    - **Уровень студентов выравнивается**
    - Материал осваивается с желаемым уровнем подробности
- Освоение в удобное время
- Перерывы
- **Можно осваивать, если заболел**
- Онлайн-вопросы
    - **Быстрая обратная связь**
    - Аналитика по домашке позволяет обсуждать затруднения
- Студенты готовы к практике
- Разные форматы материала: видеолекции, аудио, статьи, интерактивные игры

Преимущества практик на очном занятии:

- **Больше времени на ученика**
- **Больше практики**: а мы лучше учимся, когда что-то делаем, а не когда слушаем
- Преподаватель готов помочь
- **Возможна групповая работа**

Видеолекции и упражнения

Видеолекции — один из возможных форматов освоения теоретического материала при смешанном обучении.

Как идругие форматы позволяет выбрать комфортную скорость просмотра, смотреть дома в тапочках или даже с мобилки в транспорте, а также повторить в любой момент.

Небольшие упражнения по видеолекциями позволяют закрепить материал.

Опросы показывают, что студентам видео-формат лекций нравится:

- 50% студентов *за* видеолекции по сравнению с обычными лекциями,
- 5% — *против*,
- 45% — *без разницы*.

*Проверка понимания

Каждый студент воспринимает материал по своему, у каждого возникают свои трудности и вопросы. И преподавателю надо как-то понять, что это за трудности, чтобы объяснить их лучше или дать какое-то задание, в котором проблемный материал будет хорошо освоен.

Проверить понимание можно несколькими способами:

1. Узнать вопросы по пройденному материалу: видеолекциям, текстам и т.д.
    a. Быстро и просто
    b. Студенты могут стесняться или не осознавать своих затруднений
2. Провести опрос
    a. Требует некоторого времени
    b. Ограниченное количество вопросов
    c. Стимулирует диалог и вовлекает в учебный процесс
3. Ведомость модуля
    a. Быстро и просто
    b. Ограниченное количество упражнений
    c. Индивидуальная информация

Взаимопроверка

Прием позволяет поработать студентам в группах. При этом каждому ответить на вопрос.

1. Вы делите участников на 2-3 группы.
2. Озвучиваете задачу: подготовить по пройденной теме вопросы по количеству участников в противоположной группе плюс 2-3 (на случай повторения).
3. Оговаривает время: примерно 5 минут.
4. После окончания времени любой участник 1 группы задает вопрос определенному участнику 2 группы, тот отвечает. Участник 2 группы задает вопрос определенному участнику 1 группы, тот отвечает (если группы 2).
5. Далее по аналогии. Каждый участник получает свой вопрос и отвечает на него.

Преимущества:

- У студентов есть возможность подумать, формулируя вопросы, обменяться своими вариантами.
- Каждый задействован в процессе.
- Прием снижает страх неудачи, т.к. формулируя вопросы, студенты уже повторяют тему.

Подготовка к практическим занятиям.

В ходе подготовки к практическим занятиям рекомендуется изучить презентации с лекций, а также основную и дополнительную литературу, публикации в научных изданиях,

если на них есть отсылки в презентациях, материалы, размещенные на электронных образовательных ресурсах.

Сдача практических заданий заключается в том, что студент демонстрирет работоспособность разработанной программы, обосновывает и поясняет выбранный метод решения, используемый инструмент и применяемые алгоритмы.

Семинарские задачи

Разработка ПО — это не только написание кода. Важно еще уметь придумывать сами решения задач. Студенты в зависимости от темы курса должны научиться применять известные алгоритмы, дорабатывать их для конкретной задачи, научиться проектировать ПО, а также решать другие задачи, не связанные с написанием кода.

Учиться придумывать решения задач более эффективно вместе с другими студентами, а не в одиночку. Потому что в этом случае разные студенты предлагают собственные решения и можно вместе обсудить их плюсы и минусы, обменяться обратной связью, получить обратную связь от преподавателя.

Давайте задачи курса, предполагающие обсуждение группой, называть семинарскими задачами. А занятие, посвященное разбору этих задач — семинаром.

Задачи можно обсуждать по-разному. Студентам самим будет интереснее, если чередовать разные форматы.

Примеры задач

Expr10. Найти сумму всех положительных чисел меньше 1000 кратных 3 или 5.

Expr11. Дано время в часах и минутах.

Cond1. Дана начальная и конечная клетки на шахматной доске. Корректный ли это ход на пустой доске для: слона, коня, ладьи, ферзя, короля?Найти угол от часовой к минутной стрелке на обычных часах.

Cond3. (1493. В одном шаге от счастья) Дан номер трамвайного билета, в котором суммы первых трёх цифр и последних трёх цифр отличаются на 1 (но не сказано, в какую сторону). Col1. 1330 Интервалы Дан массив. Сделав его предобработку, научиться быстро находить сумму чисел на отрезке [L, R].

Col2. 1491 Нереальная история Дан массив чисел, заполненный нулями. Поступают запросы вида: увеличить все числа на отрезке [L, R] на X. Нужно быстро обработать их и вывести содержимое массива после всех этих запросов.

Col3. 1296 Гиперпереход В массиве чисел найти подмассив (непрерывный отрезок исходного массива) с максимальной суммой.Правда ли, что предыдущий или следующий билет счастливый?

Col23. Дан массив целых чисел длины N, упорядоченный строго по возрастанию, и целое число X. Найти в массиве любые два различных элемента с суммой X, совершая при этом менее квадратичного числа операций (то есть быстрее, чем перебор всех пар элементов). Запрещается использовать встроенные коллекции (Dictionary, HashSet и т.п.)

Rec2. Вычислить i-ое число Фибоначчи через быстрое возведение в степень специально подобранной матрицы 2 на 2. Θ(logi). Рекурсивный и нерекурсивный варианты.

OOP1. Домино. Спроектировать классы для моделирования игры в домино. Кости кладутся в одну линию без изгибов. Выделить основные сущности, решить какие из этих сущностей будут моделироваться классами, какие в этих классах будут данные и методы, как классы будут взаимодействовать друг с другом. Считайте, что пользовательский интерфейс и искусственный интеллект находятся за рамками этой задачи и именно они будут вызывать ваши классы, а не наоборот.

Подготовка к зачету.

Промежуточный контроль освоения и усвоения материала дисциплины осуществляется в формет письменного зачета по вопросам выбранного билета Оценка

студента в рамках традиционной системы оценок выставляется на основе ответа студента на теоретические вопросы, а также выполнения заданий, примерный уровень которых соответствует уровню заданий, выполняемых в семестре при проведении лабораторных работ. Если студент не выполнил задания в течение семестра, то в качестве дополнительного вопроса преподаватель может предложить выполнить невыполненные лабораторные работы

Рекомендации для подготовки:

Зачет проводится по билетам. В билете 3 вопроса, выбранных из списка привденного ниже.

Вопросы к зачету:

1. Информация и ее свойства. Кодирование информации. Двоичная система счисления.
2. Алгоритмы. Виды алгоритмов.
3. Язык программирования C#. Пространство имен. Основные конструкции языка C#.
4. Типы данных. Встроенные типы данных. Преобразования типов. Константы и
5. переменные. Перечисления. Консоль: организация ввода-вывода.
6. Операции в языке C#. Приоритет. Выражения.
7. Операторы. Операторы следования, ветвления. Операторы перехода.
8. Операторы цикла.
9. Язык программирования C#. Методы: основные понятия. Параметры-значения,
10. параметры-ссылки, параметры-массивы и выходные значения.
11. Язык программирования C#. Рекурсивные методы.
12. Язык программирования C#. Обработка исключений.
13. Язык программирования C#. Работа с массивами. Списки List.
14. Язык программирования C#. Обработка текстовой информации в C#. Обработка символьной информации Char: методы и свойства. Методы и свойства неизменяемых сток String. Изменяемые строки StringBuilder.
15. Язык программирования C#. Регулярные выражения.
16. Потоки в C#. Байтовый поток. Символьный поток. Двоичный поток.
17. Приложения для обработки текстовой информации. Основные приемы форматирования
18. текста. Стили и шаблоны.
19. Обработка исключений.
20. Работа с массивами. Списки List.
21. Обработка текстовой информации в C#. Обработка символьной информации Char: методы и свойства. Методы и свойства неизменяемых сток String. Изменяемые строки StringBuilder.
22. Потоки в C#. Байтовый поток. Символьный поток. Двоичный поток. Класс File, методы
23. класса.
24. Классы: основные понятия. Данные: переменные и константы. Методы. Свойства.
25. Конструкторы. Деструкторы. События.
26. Наследование
27. Оценка сложности алгоритма
28. Методы Тестирования