Федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)»

УТВЕРЖДЕНО
Проректор по учебной работе и довузовской подготовке
А.А. Воронов

Рабочая программа дисциплины (модуля)

по дисциплине: Информатика
по направлению: Системный анализ и управление
профиль подготовки: Системный анализ и управление в технических, экономических и социальных системах
Физтех-школа Аэрокосмических Технологий
кафедра информатики и вычислительной математики
курс: 1
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
1 (осенний) - Дифференцированный зачет
2 (весенний) - Дифференцированный зачет

Аудиторных часов: 180 всего, в том числе:
лекции: 60 час.
семинары: 0 час.
лабораторные занятия: 120 час.

Самостоятельная работа: 180 час.

Всего часов: 360, всего зач. ед.: 8

Количество контрольных работ, заданий: 8

Программу составили:

Т.Ф. Хирьянов, старший преподаватель
М.П. Абрамов, ассистент

Программа обсуждена на заседании кафедры информатики и вычислительной математики 06.02.2020

Аннотация

В данном курсе студенты познакомятся с языком программирования Python 3. Кроме этого студенты изучат широкий спектр алгоритмов и структур данных. Среди них: алгоритмы сортировки, алгоритмы динамического программирования, работа со строками, основы теории графов и др.

1. Цели и задачи

Цель дисциплины

Научить студентов программировать на языке Python 3 на уровне, достаточном для использования ИКТ в курсе вычислительной математики, в исследовательской научной и в последующей профессиональной деятельности.

Задачи дисциплины

1. Обеспечить чёткое понимание студентами основ информатики и ИКТ, включая некоторые области математики (системы счисления, логика, дискретная математика, теория графов);
2. Обучить студентов основным алгоритмам обработки числовой и текстовой информации;
3. Сформировать у обучающихся навык использования языка программирования Python 3 для решения конкретных прикладных задач;
4. Научить студентов писать программный код коллективно с использованием промышленного стиля программирования и утилит, необходимых при совместной работе над программным продуктом.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-6 Способен применять математические, системно-аналитические, вычислительные методы и программные средства для решения прикладных задач в области создания систем анализа и автоматического управления и их компонентов	ОПК-6.2 Применяет программные средства для решения прикладных задач в области создания систем анализа и автоматического управления и их компонентов
ПК-1 Способен проводить исследование систем управления и их компонент	ПК-1.2 Имеет глубокое знание и понимание базовых математических дисциплин

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- Основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы дискретной математики;
- основы алгоритмического языка программирования Python;
- общие характеристики интерпретируемых и компилируемых языков программирования;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- приёмы разработки программ;
- принципы программирования структур данных для современных программ, типовые решения, применяемые для создания программ;
- основы работы с пакетами прикладных программ в области математики и физики.

уметь:

• Выбирать оптимальные алгоритмы для современных программ;
• разрабатывать полные законченные программы на одном из языков высокого уровня; программы на одном или нескольких языках программирования, как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
• использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности;
• работать как на уровне языка командного интерпретатора, так и с использованием графического пользовательского интерфейса;
• использовать сигналы и оконные сообщения для взаимодействия процессов между собой и с операционной системой;
• создавать безопасные программы, использовать современные средства для написания и отладки программ;
• работать с пакетами прикладных программ, включая использование развитых графических возможностей этих пакетов.

владеть:

• Языком программирования Python и методами создания программ с использованием стандартных библиотек;
• средствами отладки программ на Python;
• навыками программирования с использованием средств операционной системы для решения исследовательских задач;
• основами работы с прикладными пакетами Python и принципами написания дополнительных модулей;
• навыками освоения современных архитектур ЭВМ.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Знакомство с Python 3	2		4	5
2	Однопроходные алгоритмы	2		4	5
3	Системы счисления	2		4	5
4	Функции	2		4	5
5	Списки и алгоритмы на списках	2		4	5
6	Изменяемость списка list в Python	2		4	5
7	Сортировки	2		4	5
8	Рекурсия	2		4	5
9	Быстрые сортировки	2		4	8
10	Двоичный поиск	2		4	8
11	Динамическое программирование	2		4	8
12	Строки	2		4	8
13	Двумерное динамическое программирование	2		4	8
14	Структуры FIFO и LIFO	2		4	10
15	Конечные и клеточные автоматы	2		4	
16	Сложность задач	2		4	5
17	Хеширование	2		4	5
18	Словари и множества в Python	2		4	5
19	Связные списки	2		4	5
20	Очередь и очередь с приоритетами	2		4	5

21	Основы теории графов	2		4	5
22	Хранение графа в памяти	2		4	5
23	Поиск в глубину	2		4	5
24	Поиск в ширину	2		4	8
25	Поиск кратчайшего пути	2		4	8
26	Остовные деревья	2		4	8
27	Основы теории игр	2		4	8
28	Двоичные деревья поиска	4		8	8
29	Асимптотически сложные задачи на графах	2		4	10
Итого часов		60		120	180
Подготовка к экзамену		0 час.			
Общая трудоёмкость		360 час., 8 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 1 (Осенний)

1. Знакомство с Python 3

Ход исполнения программы. Почему в Python нет goto. Интерактивный режим. Арифметические операции и их приоритеты. Типы данных. Преобразование типа. Ввод-вывод. Именованные параметры print() sep, end. Переменные. Присваивание: =, +=, -=, *=, /=. «Трамвайное присваивание». Множественное присваивание. Обмен переменных значениями. Цикл for и функция range(). Однопроходные алгоритмы: сумма, произведение. Оператор ветвления if. Переменные-счётчики. Среднее арифметическое. Тип bool. Логические операции. Битовые операции &, |, ^.

2. Однопроходные алгоритмы

Обработка потока чисел с терминальным элементом. Поиск числа в потоке. Фильтрация потока чисел. Вложенные ветвления. Каскадные ветвления if-elif-else. Цикл while. Инструкции break и continue. Переменные-флаги. Максимальное число в потоке. Местоположение максимума. Количество равных максимуму. Поиск трёх максимумов за один проход.

3. Системы счисления

Целочисленное деление и взятие остатка, их отличие в C++ и Python. Позиционные системы счисления и литералы целых чисел в Python. Анализ цифр числа в произвольной системе счисления. Переводы из одной системы в другую.

4. Функции

Описание функций с параметрами. Синхронный вызов. Стек вызовов. Локальность переменных. Утиная типизация в Python. Метод грубой силы. Поиск НОД и НОК. Алгоритм Евклида. Тест простоты. Разложение числа на множители.

5. Списки и алгоритмы на списках

Создание списка чисел заданной длины. Функция len(). Индексация элементов от 0 до N-1. Скорость взятия и замены элемента A[i]. Распечатка массива. Задачи на заполнение массива. Заполнение массива числами Фибоначчи. Линейный поиск в массиве. Поэлементное копирование массива. Копирование задом-наперёд. Циклический сдвиг в массиве. Обращение массива.

6. Изменяемость списка list в Python

Ссылочная модель данных. Оператор идентичности is. Добавление и удаление элемента в начале и конце массива. Отличие по скорости A.pop(0) и A.pop(), и почему это так. Списковые включения («генераторы списков»). Решето Эратосфена. Частотный анализ (метод подсчёта).

7. Сортировки

Постановка задачи. Сортировка обезьяны. Сортировка выбором. Сортировка вставками. Ленивые and и or. Проверка упорядоченности массива за O(N). Сортировка дурака. Сортировка методом пузырька (через while с переменной-флагом). Синхронная сортировка нескольких массивов. Устойчивость сортировок. Сортировка подсчётом. Поразрядная сортировка для двоичной СС. Асимптотическая сложность алгоритмов.

8. Рекурсия

Принцип «Разделяй и властвуй». Глубина рекурсии, прямой и обратный ход, рекуррентный и крайний случай. Ханойские башни. Генерация комбинаторных объектов. Перебор с возвратом. Рекурсивная генерация всех чисел длины M. Генерация всех перестановок. Примеры кодирования рекурсии: быстрое возведение в степень, НОД.

9. Быстрые сортировки

Быстрая сортировка Тони Хоара. Слияние двух упорядоченных массивов. Сортировка слиянием. Неустойчивость сортировок.

10. Двоичный поиск

Бинпоиск. Поиск корня непрерывной функции методом деления пополам. Бинарный поиск по ответу. Бинарный поиск в массиве за O(logN).

11. Динамическое программирование

Вычисление чисел Фибоначчи и проблема перевычислений. Рекурсия с кешированием. Одномерное динамическое программирование. Задачи о Кузнечике. Восстановление пути минимальной стоимости.

12. Строки

Тип str. Неизменяемость строки. Наивный поиск подстроки в строке. Методы строк find, rfind, count, replace. Методы split и join. Разбиение на подстроки, объединение. Срезы строк. Префикс-функция. Алгоритм Кнута-Морриса-Пратта.

13. Двумерное динамическое программирование

Вычисление расстояния Левенштейна. Восстановление последовательности редакционных изменений. Наибольшая общая подпоследовательность. Наибольшая возрастающая подпоследовательность.

14. Структуры FIFO и LIFO

Очереди: FIFO и LIFO. Стек как очередь LIFO. Проверка корректности скобочной последовательности. Обратная польская нотация.

15. Конечные и клеточные автоматы

Машина Тьюринга. Конечный автомат как её упрощение. Конечный автомат для поиска подстроки «abcd». Простейшие клеточные автоматы. Игра «Жизнь» Джона Конвея.

Семестр: 2 (Весенний)

16. Сложность задач

Краткое повторение синтаксиса Python. Сложность задач. Детерминированная и недетерминированная машина Тьюринга. Алгоритмически простые и сложные задачи (классы P и NP). Классы NP-complete и NP-hard.

17. Хеширование

Хеш-функции, хеширование и хеш-таблицы. Что такое хеш-функция. Примеры. Использование хеширования для гарантии целостности файлов и хранения паролей. Полиномиальный хеш. Алгоритм Рабина-Карпа. Открытая и закрытая хеш-таблицы. Проблема удаления из закрытой хеш-таблицы. Перехеширование. Реализация закрытой хеш-таблицы.

18. Словари и множества в Python

Словари и множества в Python. Множество set. Создание и изменение множеств. Работа с элементами. Тип frozenset и зачем он нужен. Операции с множествами, обычные для математики. Словарь dict. Создание и изменение словаря. Пример применения ассоциативного массива. Defaultdict, OrderedDict.

19. Связные списки

Кортежи tuple и контейнер NamedTuple. Списки: односвязный, двусвязный, кольцо (реализация ч/з словари).

20. Очередь и очередь с приоритетами

Очередь и дек (реализация на списках). Контейнер Deque. Куча (повторение). Сортировка кучей. Модуль heapq.

21. Основы теории графов

Введение в теорию графов. Инцидентность, смежность, петля, кратные рёбра, подграф. Эйлеров цикл. Эйлеров путь. Пути в графах. Циклы. Простые пути и циклы. Связность графов. Компоненты связности. Взвешенный граф. Орграфы. Компоненты сильной связности орграфа. Ориентированные ациклические графы. Дерево. Корневое дерево. Остовное дерево графа.

22. Хранение графа в памяти

Список рёбер, матрица смежности и списки смежности. Реализация этих способов и асимптотика их работы. Переходы между различными формами хранения графа. Компактная форма хранения списка смежности для константного графа. Хранение деревьев в памяти.

23. Поиск в глубину

Обход графа в глубину. Выделение компонент связности (обходом в глубину). Выделение компонент сильной связности орграфа. Проверка двудольности графа. Проверка графа на ацикличность и нахождение цикла. Топологическая сортировка. Поиск мостов и точек сочленения.

24. Поиск в ширину

Обход графа в ширину. Очередь при обходе в ширину и её асимптотика. Выделение компонент связности (обходом в ширину). Нахождение кратчайшего цикла в невзвешенном графе.

25. Поиск кратчайшего пути

Алгоритм Дейкстры поиска кратчайшего пути. Алгоритмы Флойда-Уоршелла и Беллмана-Форда.

26. Остовные деревья

Алгоритм Прима. Алгоритм Краскала.

27. Основы теории игр

Игры на ациклических графах. Игра «Ним». Сумма игр. Функция Шпрага-Гранди.

28. Двоичные деревья поиска

Двоичные деревья поиска. Асимптотика основных операций. Баланси-ровка деревьев. АВЛ-дерево и красно-чёрное дерево. Декартово дерево.

29. Асимптотически сложные задачи на графах

Гамильтонов граф. Построение гамильтонова цикла. Задачи о коммивояжере и о китайском почтальоне. Приближенные алгоритмы для NP-полных задач.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Большая лекционная аудитория, подходящая для учебного потока (факультет, оснащённая мультимедиа проектором и экраном для чтения лекций.
Учебные аудитории, учебный сетевой компьютерный класс с установленным необходимым программным обеспечением.

6.Перечень рекомендуемой литературы

Основная литература

1. Алгоритмы [Текст] : [учеб. пособие для вузов] / С. Дасгупта, Х. Пападимитриу, У. Вазирани ; пер. с англ. А. А. Куликова ; под ред. А. Шеня .— М. : МЦНМО, 2014 .— 320 с.

Дополнительная литература

1. Алгоритмы : построение и анализ [Текст] : учебник для вузов / Т. Кормен, Ч. Лейзерсон, Р. Ривест .— М. : МЦНМО, 1999 .— 263 с.

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. https://python.org
2. https://e-maxx.ru/algo/
2. https://github.com
3. http://judge.mipt.ru
4. http://acm.mipt.ru

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

На ПК в компьютерных классах должно быть установлено следующее ПО:

1. Операционная система GNU/Linux;
2. Интерпретатор Python версии не ниже 3.4;
3. Командный интерпретатор Ipython с отладчиком;
4. Среда разработки JetBrains Python Charm community edition;
5. Среда разработки IDLE;
6. Библиотеки ScyPy для Python 3;
7. Библиотеки PyGame;

На лекциях используются мультимедийные технологии, включая демонстрацию презентаций.

Для контроля и коррекции знаний, обучающиеся могут использовать компьютерное тестирование, в том числе на сайте judge.mipt.ru.

В процессе самостоятельной работы обучающихся возможно использование любые среды программирования.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Изложение материала происходит преимущественно на лекциях, сопровождается мультимедиа-презентацией с примерами кода и блок-схемами алгоритмов. На лабораторных занятиях также происходит изложение нового материала: в начале каждой лабораторной работы, по мере необходимости, а также в личных беседах тьютора с рабочими группами. На контекстах изложение нового материала исключено, преподаватель оказывает только консультативное содействие для успешного решения задач.

Учёт, контроль и оценка знаний студентов

В течение семестра успеваемость отслеживается автоматически — по результатам контекстов, а также тьютором по своевременности сдач лабораторных работ. Таким образом достигается раннее выявление отстающих студентов с передачей докладных в деканат.

Посещаемость лекций не отмечается, но каждый контекст завязан на материал прошедшей лекции, что делает посещение лекций насущной необходимостью в течение семестра.

Дифференцированный зачёт принимает лектор в устной форме с учётом оценки по контекстам и оценки по лабораторным работам. Устный ответ практически исключает списывание, показывает владение базовой терминологией предмета, умение говорить на языке информатики, а также позволяет проверить знание сложных алгоритмов, которые долго программируются, но могут быть относительно легко устно объяснены.

Самостоятельная домашняя работа предполагается после каждой лабораторной работы.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Системный анализ и управление
профиль подготовки: Системный анализ и управление в технических, экономических и социальных системах
Физтех-школа Аэрокосмических Технологий
кафедра информатики и вычислительной математики
курс: 1
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
1 (осенний) - Дифференцированный зачет
2 (весенний) - Дифференцированный зачет

Разработчики:

Т.Ф. Хирьянов, старший преподаватель
М.П. Абрамов, ассистент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-6 Способен применять математические, системно-аналитические, вычислительные методы и программные средства для решения прикладных задач в области создания систем анализа и автоматического управления и их компонентов	ОПК-6.2 Применяет программные средства для решения прикладных задач в области создания систем анализа и автоматического управления и их компонентов
ПК-1 Способен проводить исследование систем управления и их компонент	ПК-1.2 Имеет глубокое знание и понимание базовых математических дисциплин

2. Показатели оценивания компетенций

В результате изучения дисциплины «Информатика» обучающийся должен:

знать:

- Основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы дискретной математики;
- основы алгоритмического языка программирования Python;
- общие характеристики интерпретируемых и компилируемых языков программирования;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- приёмы разработки программ;
- принципы программирования структур данных для современных программ, типовые решения, применяемые для создания программ;
- основы работы с пакетами прикладных программ в области математики и физики.

уметь:

- Выбирать оптимальные алгоритмы для современных программ;
- разрабатывать полные законченные программы на одном из языков высокого уровня; программы на одном или нескольких языках программирования, как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности;
- работать как на уровне языка командного интерпретатора, так и с использованием графического пользовательского интерфейса;
- использовать сигналы и оконные сообщения для взаимодействия процессов между собой и с операционной системой;
- создавать безопасные программы, использовать современные средства для написания и отладки программ;
- работать с пакетами прикладных программ, включая использование развитых графических возможностей этих пакетов.

владеть:

- Языком программирования Python и методами создания программ с использованием стандартных библиотек;
- средствами отладки программ на Python;
- навыками программирования с использованием средств операционной системы для решения исследовательских задач;
- основами работы с прикладными пакетами Python и принципами написания дополнительных модулей;
- навыками освоения современных архитектур ЭВМ.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

1) Операторы if, elif, else. Цикл while, операторы break, continue, else.
2) Решето Эратосфена. Оценка временной сложности алгоритма.

3) Задача упорядочивания элементов в массиве. Оценка временной сложности задачи в общем случае. Проверка упорядоченности массива за O(N).

4) Рекурсия. Прямой и обратный ход рекурсии. Стек вызовов при рекурсии. Вычисление факториала.

5) Наибольшая возрастающая подпоследовательность.

3. Перечень типовых контрольных заданий, используемых для оценки знаний, умений, навыков

Примерный перечень контрольных вопросов по теории за 1-й семестр:

1. Основы архитектуры компьютера. Принципы фон Неймана.
2. Отличие интерпретируемых и компилируемых языков.
3. Концепция присваивания в Python
4. Обмен двух переменных значениями.
5. Кортежи и их использование.
6. Цикл while. Инструкции управления циклом.
7. Позиционные системы счисления
8. Однопроходные алгоритмы: подсчёт, сумма, произведение.
9. Оператор if. Каскадная условная конструкция elif.
10. Логические операции в Python.
11. Основы алгебры логики
12. Однопроходные алгоритмы: поиск числа в потоке, максимум.
13. Тест простоты числа.
14. Разложение числа на множители.
15. Тип str. Длина строки len(s). Неизменяемость строки.
16. Срезы строк.
17. Методыстрок find, count, replace, startswith, endswith.
18. Наивный поиск подстроки в строке.
19. Ссылочная модель данных в Python. Операторы == и is. Копирование объектов.
20. Алгоритм обращения массива.
21. Алгоритм циклического сдвига в массиве.
22. Срезы списков. Присваивание в срез. Методы списка.
23. Список строк. Методы split и join для строки.
24. Цикл for и его особенности в Python.
25. Listcomprehensions: генерация списков.
26. Двумерные массивы (списки списков). Вложенная генерация.
27. Полиморфизм в Python. Ducktyping.
28. Именованные параметры.
29. Поиск корня функции методом бисекции.
30. Поиск значения в упорядоченном массиве методом бисекции.
31. Сортировка обезьяны.
32. Сортировка вставками.
33. Сортировка выбором.
34. Сортировка методом пузырька.
35. Сортировка дурака
36. Сортировка подсчётом.
37. Поразрядная сортировка.
38. Прагматическая сортировка TimSort.
39. Рекурсия. Прямой и обратный ход рекурсии.
40. Проблема алгоритмической сложности задачи.
41. Ханойские башни.
42. Генерация всех перестановок (рекурсивная)
43. Одномерное динамическое программирование.
44. Двумерное динамическое программирование
45. Рекурсия с кешированием на примере факториала.
46. Рекурсивные сортировки. Быстрая сортировка. Сортировка слиянием.
47. Пирамида (куча). Пирамидальная сортировка.
48. Устойчивость сортировок.
49. Тип set. Множества и работа с ними.

50. Тип dict. Словарь (ассоциативный массив) и операции с ним.
51. Dictcomprehensions: генерация множеств и словарей.
52. Частотный анализ для строк.
53. Генераторы, yield.

Примерный перечень контрольных вопросов по теории за 2-й семестр:

1. Классы в Python. Перегрузка операторов.
2. Исключения в Python. Генерирование и перехват исключений.
3. Списки: односвязный, двусвязный, кольцо.
4. Стек. Дек.
5. Очередь.
6. Очередь с приоритетами. Пирамида (куча).
7. Очередь событий графического приложения.
8. Хеш-функция. Хеширование.
9. Открытая хеш-таблица.
10. Закрытая хеш-таблица.
11. Проблема удаления из закрытой хеш-таблицы. Перехеширование.
12. Взвешенный граф.
13. Расстояние между двумя вершинами.
14. Графы и способы их представления: список рёбер, матрица смежности, списки смежности
15. Определение дерева.
16. Поиск в глубину.
17. Связность неориентированных графов: выделение компонент связности.
18. Поиск в ширину.
19. Алгоритм Дейкстры.
20. Восстановление кратчайшего пути.
21. Постоение гамильтонова цикла.
22. Эйлеров цикл. Эйлеров путь.
23. Минимальное остовное дерево. Алгоритм Прима.
24. Задача о коммивояжере
25. Орграфы.
26. Топологическая сортировка.
27. Двоичное дерево поиска.
28. Декартово дерево («дуча»).
29. Балансировка деревьев. АВЛ-дерево. Красно-чёрное дерево.
30. Проверка равенства строк. Простой и вероятностный алгоритмы.
31. Вычисление расстояния Левенштейна.
32. Поиск подстроки в строке.
33. Алгоритм Рабина-Карпа
34. Конечный автомат для поиска подстроки «abcd», «ababc».

На дифференцированном зачёте предлагается ответить на два-три вопроса по теории и решить одну короткую алгоритмическую задачу на бумаге без использования компьютера.

Пример задания на устном зачёте:

1. Сортировка обезьяны.
2. Рекурсия. Прямой и обратный ход рекурсии.
3. Тип set. Множества и работа с ними.
4. Задача: реализовать слияние двух отсортированных списков.

4. Критерии оценивания

Промежуточная аттестация по дисциплине «Информатика» осуществляется в форме дифференцированного зачета.

Дифференцированный зачёт принимает лектор в устной форме с учётом оценки по контестам и оценки по лабораторному практикуму. Устный ответ практически исключает списывание, показывает владение базовой терминологией предмета, умение говорить на языке информатики, а также позволяет проверить знание сложных алгоритмов, которые долго программируются, но могут быть относительно легко устно объяснены.

Оценка по десятибалльной шкале за работу на лабораторном практикуме выставляется преподавателем практикума исходя из количества и качества выполненных практических работ за семестр. Оценка за выполнение контестов выставляется автоматически исходя из суммарного рейтинга обучающегося в системе Ejudgeи также нормируется к десятибалльной шкале.

Итоговая оценка за зачёт не может отличаться от среднего арифметического оценок по контестам и по практическим лабораторным работам более чем на три балла.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Время проведения зачёта составляет 30 минут на одного обучающегося.
Во время подготовки к ответу обучающиеся не могут пользоваться литературой, печатными материалами, рукописными записями, а также электронными средствами (сотовыми телефонами, планшетами, умными часами и т.п.).