

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

УТВЕРЖДЕНО
Проректор по учебной работе

А.А. Воронов

	Рабочая программа дисциплины (модуля)
по дисциплине:	Информатика
по направлению:	Прикладные математика и физика
профиль подготовки:	Беспилотные авиационные системы Физтех-школа авиационных и цифровых технологий кафедра информатики и вычислительной математики
курс:	1
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

Аудиторных часов: 120 всего, в том числе:

лекции: 30 час.

семинары: 90 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 60 час.

Всего часов: 180, всего зач. ед.: 4

Программу составили:

В.П. Иванников, д-р физ.-мат. наук, (на удаление) академик, заведующий кафедрой

В.В. Прут, д-р физ.-мат. наук, доцент, профессор

П.Н. Коротин, канд. физ.-мат. наук, доцент

В.К. Хохлов, канд. техн. наук, доцент

Д.С. Северов, канд. физ.-мат. наук, доцент

Программа обсуждена на заседании кафедры информатики и вычислительной математики 12.03.2024

Аннотация

Данный курс предназначен для введения студентов в теорию алгоритмов и информатику. Курс даёт представление о различных структурах данных: массивы, линейные списки, деревья. Рассматриваются классические алгоритмы: сортировка простыми вставками, простым выбором, метод «пузырька», шейкер сортировка. В рамках курса студенты изучают архитектуры системы команд X86, разнообразие систем команд в реальных ЭВМ (CISC, RISC и др.). основные принципы конвейеризации.

1. Цели и задачи

Цель дисциплины

Формирование базовых знаний по информатике для дальнейшего использования в других областях математического знания и дисциплинах естественнонаучного содержания; формирование информационной культуры, исследовательских навыков и способности применять знания на практике.

Задачи дисциплины

- Формирование у обучающихся базовых знаний по информатике;
- формирование информационной культуры: умение логически мыслить, проводить доказательства основных утверждений, устанавливать логические связи между понятиями;
- формирование умений и навыков применять полученные знания для решения информационных задач, самостоятельного анализа полученных результатов.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
ПК-4 Способен критически оценивать применимость используемых методик и методов	ПК-4.1 Знает численные порядки величин, характерных для соответствующей профессиональной области

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- Основы дискретной математики;
- основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы одного или нескольких алгоритмических языков программирования, общие характеристики языков программирования, идеологию объектно-ориентированного подхода;
- приемы разработки программ;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представления информации в ЭВМ и архитектурные принципы повышения их производительности.

уметь:

- Выбирать оптимальные алгоритмы для современных программ;
- разрабатывать полные законченные программы на одном из языков программирования высокого уровня;
- разрабатывать программы на одном или нескольких языках программирования как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- применять объектно-ориентированный подход для написания программ;
- использовать знания по информатике для приложения в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности.

владеть:

- Одним или несколькими современными языками программирования и методами создания программ с использованием библиотек и современных средств их написания и отладки;
- 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	Алгоритмические языки	8	25		15
2	Алгоритмы и структуры данных	10	25		15
3	Введение в алгоритмы	6	20		15
4	Введение в теорию алгоритмов	6	20		15
Итого часов		30	90		60
Подготовка к экзамену		0 час.			
Общая трудоёмкость		180 час., 4 зач.ед.			

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

Семестр: 1 (Осенний)

1. Алгоритмические языки

Характеристика алгоритмических языков и их исполнителей. Понятие трансляции.

Понятие о формальных языках. Способы строгого описания формальных языков, понятие о метаязыках. Алфавит, синтаксис и семантика алгоритмического языка. Описание синтаксиса языка с помощью металингвистических формул и синтаксических диаграмм.

Языки программирования. Общие характеристики языков программирования. Алфавит, имена, служебные слова, стандартные имена, числа, текстовые константы, разделители. Препроцессор и комментарии.

Типы данных, их классификация. Переменные и константы. Скалярные типы данных и операции над ними. Старшинство операций, стандартные функции. Выражения и правила их вычисления. Оператор присваивания.

Файлы. Стандартные функции ввода-вывода.

Простые и сложные операторы. Пустой, составной, условный операторы. Оператор варианта. Оператор перехода.

Оператор цикла. Программирование рекуррентных соотношений.

Составные типы данных. Массивы.

Описание функций (процедур). Формальные и фактические параметры. Способы передачи параметров. Локализация имен. Побочные эффекты. Итерации и рекурсии.

Ссылочный тип данных. Методы выделения памяти: статический, динамический и автоматический. Структуры. Битовые поля. Объединения. Перечисления. Декларация typedef.

2. Алгоритмы и структуры данных

Абстрактные структуры данных: список, стек, очередь, очередь с приоритетом, ассоциативный массив. Отображение абстрактных структур данных на структуры хранения: массивы, линейные списки, деревья.

Различные реализации ассоциативного массива: двоичные деревья поиска (АВЛ-деревья, красно-чёрные деревья), перемешанные таблицы (с прямой и открытой адресацией, использование техники двойного хэширования при открытой адресации). Оценки алгоритмической сложности операций поиска, добавления и удаления элемента.

Классические алгоритмы: перебор с возвратом, жадные алгоритмы. Примеры алгоритмов работы с графами: поиск минимального остового дерева, поиск кратчайшего пути, задача коммивояжера.

3. Введение в алгоритмы

Понятие внутренней и внешней сортировки. Устойчивая сортировка. Сортировка in-place. Сортировка простыми вставками, простым выбором, метод «пузырька». Шейкер сортировка. Метод Шелла. Быстрая сортировка Хоара. Сортировка слиянием. Пирамидальная сортировка. Оценка трудоемкости.

4. Введение в теорию алгоритмов

Интуитивное понятие алгоритма. Свойства алгоритмов. Понятие об исполнителе алгоритма. Алгоритм как преобразование слов из заданного алфавита. Связь понятия алгоритма с понятием функции. Машина Тьюринга. Нормальные алгоритмы Маркова. Вычислимые функции и их свойства. Невычислимые функции. Различные эквивалентные определения множества вычислимых функций. Алгоритмическая сложность.

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

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).

6. Перечень рекомендуемой литературы

Основная литература

1. Практика и теория программирования [Текст] : в 2 кн. : учеб. пособие для вузов / Н. А. Винокуров, А. В. Ворожцов .— М. : Физматкнига, 2008 .— (Серия "Информатика"). - ISBN 978-5-89155-182-4 (в пер.) .— Кн.2, Ч. 3-4. - 2008. - 288 с.
2. Язык программирования С [Текст] : [учеб. пособие для вузов] / Б. Керниган, Д. Ритчи ; пер. с англ. и ред. В. Л. Бродового .— 2-е изд., перераб. и доп. — М. : Вильямс, 2006, 2007, 2009, 2010, 2012, 2013, 2015 .— 304 с.

Дополнительная литература

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

1. <http://cs.mipt.ru>
2. <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 лекциях используются мультимедийные технологии, включая демонстрацию презентаций. Для контроля и коррекции знаний, обучающиеся могут использовать компьютерное тестирование, в том числе на сайте www.judge.mipt.ru.

В процессе самостоятельной работы обучающихся возможно использование любые среды программирования.

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

Студент, изучающий курс информатики, должен с одной стороны, овладеть общим понятийным аппаратом, а с другой стороны, должен научиться применять теоретические знания на практике. Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

- чтение и конспектирование рекомендованной литературы,
- проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств.

Руководство и контроль за самостоятельной работой студента осуществляется в форме индивидуальных консультаций.

Показателем владения материалом служит умение решать задачи. Для формирования умения применять теоретические знания на практике студенту необходимо решать как можно больше задач. При решении задач каждое действие необходимо аргументировать, ссылаясь на известные теоретические сведения.

При затруднении изучения отдельных тем, вопросов, следует обращаться за консультациями к лектору или преподавателю.

Обязательным требованием является выполнение домашних работ, которые оформляются в специально отведённой для этого тетради и систематически сдаются на проверку.

Промежуточный контроль знаний проводится в виде контрольных работ, на которых студенту предлагается решить несколько задач, а также студенту в ходе освоения курса необходимо выполнить две домашние индивидуальные работы с их последующей защитой.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Прикладные математика и физика
профиль подготовки:	Беспилотные авиационные системы Физтех-школа авиационных и цифровых технологий кафедра информатики и вычислительной математики
курс:	<u>1</u>
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

Разработчики:

В.П. Иванников, д-р физ.-мат. наук, (на удаление) академик, заведующий кафедрой
В.В. Прут, д-р физ.-мат. наук, доцент, профессор
П.Н. Коротин, канд. физ.-мат. наук, доцент
В.К. Хохлов, канд. техн. наук, доцент
Д.С. Северов, канд. физ.-мат. наук, доцент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
ПК-4 Способен критически оценивать применимость используемых методик и методов	ПК-4.1 Знает численные порядки величин, характерных для соответствующей профессиональной области

2. Показатели оценивания компетенций

В результате изучения дисциплины «Информатика» обучающийся должен:

знать:

- Основы дискретной математики;
- основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- основы одного или нескольких алгоритмических языков программирования, общие характеристики языков программирования, идеологию объектно-ориентированного подхода;
- приемы разработки программ;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представления информации в ЭВМ и архитектурные принципы повышения их производительности.

уметь:

- Выбирать оптимальные алгоритмы для современных программ;
- разрабатывать полные законченные программы на одном из языков программирования высокого уровня;
- разрабатывать программы на одном или нескольких языках программирования как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- применять объектно-ориентированный подход для написания программ;
- использовать знания по информатике для приложения в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности.

владеть:

- Одним или несколькими современными языками программирования и методами создания программ с использованием библиотек и современных средств их написания и отладки;
- навыками освоения современных архитектур ЭВМ.

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

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Критерии оценивания

Оценка отлично 10 баллов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины, проявляющему интерес к данной предметной области, продемонстрировавшему умение уверенно и творчески применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично 9 баллов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично 8 баллов - выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений, с некоторыми недочетами.

Оценка хорошо 7 баллов - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но недостаточно грамотно обосновывает полученные результаты.

Оценка хорошо 6 баллов - выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности.

Оценка хорошо 5 баллов - выставляется студенту, если он в основном знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач достаточно большое количество неточностей.

Оценка удовлетворительно 4 балла - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он освоил основные разделы учебной программы, необходимые для дальнейшего обучения, и может применять полученные знания по образцу в стандартной ситуации.

Оценка удовлетворительно 3 балла - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, допускающему ошибки в формулировках базовых понятий, нарушения логической последовательности в изложении программного материала, слабо владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и с трудом применяет полученные знания даже в стандартной ситуации.

Оценка неудовлетворительно 2 балла - выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных принципов и не умеет использовать полученные знания при решении типовых задач.

Оценка неудовлетворительно 1 балл - выставляется студенту, который не знает основного содержания учебной программы дисциплины, допускает грубейшие ошибки в формулировках базовых понятий дисциплины и вообще не имеет навыков решения типовых практических задач.

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

Во время проведения дифференцированного зачета обучающиеся могут пользоваться программой дисциплины, а также справочной литературой, вычислительной техникой, конспектами лекций. Дифференцированный зачет может проводиться по итогам текущей успеваемости и сдачи заданий или путем организации специального опроса, проводимого в устной форме.

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

Промежуточная аттестация по дисциплине «Информатика» осуществляется в форме дифференцированного зачета. Дифференцированный зачет выставляется по результатам работы студента в течение семестра на основе оценок за домашние задания, контрольные работы.

1 (осенний) семестр

ЗАДАНИЕ 1

1. Машины Тьюринга

Обозначим как N_0 множество всех неотрицательных целых чисел.

Описать машины Тьюринга, которые реализуют:

1.1. Счетчик четности. Выход машины Тьюринга равен 0 или 1 в зависимости от того, четно или нечетно число единиц в последовательности из 0 и 1, записанной на ленте машины Тьюринга. В конце последовательности стоит символ B . В начальном состоянии головка видит первый левый символ.

1.2. Инверсию заданного слова в алфавите $\{0, 1\}$ (0 заменяет на 1, а 1 – на 0).

1.3. «Переворачивание» заданного слова в алфавите $\{a, b, c\}$.

1.4. Сложение двух чисел из множества N_0 , записанных на ленте в виде последовательности единиц, а именно:

$$0 \rightarrow 0, 1 \rightarrow 01, 2 \rightarrow 011, 3 \rightarrow 0111, 4 \rightarrow 01111, \dots$$

Назовём эту запись *единичной записью* числа. Записи чисел разделены на ленте несколькими пустыми ячейками. Рассмотрите различные варианты начального положения головки машины Тьюринга.

1.5. Сложение двух чисел из N_0 , данных в двоичной системе счисления.

1.6. Распознавание правильных скобочных выражений. Правильное скобочное выражение – это слово в алфавите $A = \{ (,) \}$, которое может получиться, если из арифметического выражения удалить все символы, кроме скобок. Примеры правильных скобочных выражений: пустое слово, $()$, $(())()$, $()()$, $(())()$. Примеры неправильных скобочных выражений: $)()$, $(())()$, $(,))()$, $(())()$. Результат работы: слово «YES», если скобочное выражение правильное, и слово «NO» – иначе.

1.7. Перемножение двух чисел из N_0 , заданных в виде единичных записей. Числа записаны на ленте подряд.

1.8. Вычисление квадрата числа, заданного в виде единичной записи.

2. Алгоритмы Маркова

2.1. Записать нормальные алгоритмы Маркова, которые реализуют:

2.1.1. Приписывание буквы X к входному слову справа.

2.1.2. Задание 1.2.

2.1.3. Задание 1.3.

2.1.4. Задание 1.5.

2.1.5. Задание 1.6.

2.1.6. Удвоение числа, заданного а) в виде единичной записи, б) в двоичной системе счисления.

2.2. В алфавите $A = \{a, b\}$ описать нормальный алгоритм, который выдает в качестве результата пустое слово, если буквы a и b входят во входное слово в равном количестве, и любое непустое слово – иначе. В алгоритме должно быть не более четырех правил подстановки. Докажите правильность придуманного алгоритма.

2.3. Существует ли алгоритм Маркова, применимый только к двоичным записям простых чисел?

2.4. Верно ли, что человек для любого алгоритма Маркова может написать машину Тьюринга, реализующую ту же функцию (то есть множество слов, к которым они применимы, совпадает и для любого элемента из этого множества результаты работы алгоритма Маркова и машины Тьюринга совпадают)? Разрешима ли эта задача алгоритмически?

2.5. Верно ли, что человек для любого алгорифма Маркова и заданного входного слова может определить, применим ли алгорифм Маркова к этому слову или нет? Можно ли поручить эту программу компьютеру (в принципе, не принимая во внимание доступное время и вычислительные мощности)? Ответьте на те же вопросы при условии, что максимальное число правил ограничено числом 10.

3. Решение простых алгоритмических задач

При написании программ в качестве входного и выходного потоков использовать стандартные потоки ввода и вывода.

При решении задач аккуратно форматируйте код согласно правилам. Давайте переменным и функциям значимые имена. Прежде чем сдавать программу преподавателю, проверьте, что она правильно работает на более чем 10 различных входных данных.

В каждой задаче ответьте на вопрос о том, как растет время работы программы и используемая программой память с ростом параметра размера входных данных (например, параметра n).

3.1. «Мах». Написать программу, которая выводит максимальное число из n заданных чисел. В первой строчке входа дано число n , а в следующей строчке указано n целых чисел.

3.2. «Числа Фибоначчи I». Написать программу, которая по данному n находит n -е число Фибоначчи F_n . Числа Фибоначчи определяются соотношениями

$$F_n = F_{n-1} + F_{n-2}, F_1 = F_2 = 1.$$

Подсказка: организуйте цикл, в котором по последним двум вычисленным числам будет вычисляться следующее. Необходимо ли хранить в памяти все вычисленные числа Фибоначчи?

3.3. «Числа Фибоначчи II». Решить предыдущую задачу, используя идею рекурсии. Оценить число элементарных операций, которое необходимо сделать в рекурсивном и нерекурсивном алгоритмах вычисления числа F_n .

3.4. «Биномиальные коэффициенты». Написать программу, которая для данного натурального числа n находит коэффициенты в разложении

$$(1+x)^n = C_n^0 x^0 + C_n^1 x^1 + \dots + C_n^n x^n.$$

Использовать соотношения

$$C_n^n = C_n^0 = 1, C_n^k = C_{n-1}^k + C_{n-1}^{k-1}.$$

Оценить, как растет время работы вашей программы с ростом n .

3.5. «Простые числа». Написать программу, которая определяет, является ли введенное число n простым.

3.6. Написать программу, вычисляющую площадь односвязной прямоугольной фигуры, заданной перечислением пар целочисленных координат её вершин в произвольном порядке.

3.7. «Задача Иосифа». N человек, имеющие номера от 1 до N и расположившиеся по кругу друг за другом, считаются считалкой, состоящей из M слов. Расчет начинается с номера 1. Человек, на котором считалочка заканчивается – выбывает. А расчет (этой же считалочкой) продолжается с участника, следующего за выбывшим. Написать программу, которая для заданных N и M сообщает (в порядке выбывания) номера трех игроков, выбывших последними.

3.8. «Уравнение». Написать программу, которая в указанном интервале находит нетривиальный корень уравнения $\operatorname{tg} x = x$ с погрешностью 10^{-10} . Сколько итераций необходимо сделать, чтобы достичь указанной точности методами деления пополам, Ньютона, простых итераций?

4. Жадные алгоритмы

4.1. «Атлеты». Написать программу, которая находит «башню» из атлетов максимальной высоты. Атлеты характеризуются двумя параметрами – массой и силой. Сила равна максимальной массе, которую атлет может держать на плечах. Известно, что если атлет тяжелее, то он точно сильнее. Подсказка: упорядочьте атлетов по силе и стройте башню

сверху. Вверх естественно поместить самого слабого. Входом является число атлетов n и n пар (масса, сила).

4.2. «Отрезки». Написать программу, которая для множества заданных отрезков находит минимальное подмножество отрезков, объединение которых покрывает отрезок $S = [0, 10000]$. Число отрезков и координаты их концов заданы на входе. Все координаты целочисленные. Подсказка: покрывайте отрезок S пошагово, двигаясь слева направо. На каждом шаге будет непокрытая часть $[x, 10000]$. Из оставшихся отрезков выбирайте тот, который урежет непокрытую часть до $[y, 10000]$, где y максимальное. Решите эту задачу за время $O(n \log n)$, где n – количество данных отрезков.

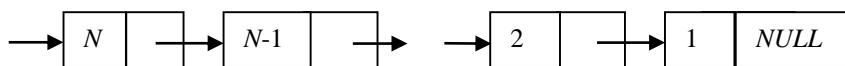
4.3. «Коммивояжер». Написать программу, которая, используя один из возможных жадных алгоритмов, находит на плоскости ломаную линию, идущую из A в B по всем заданным точкам $\{A_1, A_2, A_3, \dots, A_n\}$, как можно меньшей длины. Есть ли такие входные данные, когда написанная программа находит не самый короткий путь? Если есть, приведите пример.

ЗАДАНИЕ 2

1. Структуры данных: списки, стек, очередь, деревья поиска и перемешанные таблицы (хэш-таблицы).

1.1. «Список I». Реализовать односвязный список, элементы которого содержат целые числа. Реализовать при этом функции *list_new()* (создать новый список), *list_delete(l)* (удалить список l и все его элементы), *insert(l, a)* (добавить элемент с заданным целым числом a в начало списка l), *remove(l, a)* (удалить из списка l все элементы, содержащие заданное целое число a), *print(l)* (вывести значения, хранящиеся в элементах списка l). Осуществите массовое и многоплановое тестирование всех реализованных функций.

1.2. Для структуры данных из задачи 1.1 реализовать функцию *first_integers(N)* от N , которая конструирует список вида



(при $N = 0$ список пустой) и возвращает как свое значение ссылку на этот список.

1.3. «Список II». Реализовать двусвязный список, элементы которого содержат целые числа. Реализовать функции: *list_new()* (создать новый пустой список), *list_delete(l)* (удалить список и все его элементы), *push(l, a)* (добавить новый элемент a в конец списка), *pop(l, x)* (извлечь последний элемент списка), *unshift(l, a)* (добавить новый элемент a в начало) и *shift(l, x)* (извлечь первый элемент списка). Последние пять функций в качестве первого аргумента получают указатель на список, а возвращают 1 или 0 в зависимости от того, успешно ли выполнена операция. Функции *push* и *unshift* во втором аргументе получают добавляемый элемент. Функции *pop* и *shift* во втором аргументе x получают адрес, куда следует поместить извлекаемый элемент. Реализуйте также функцию *reverse*, которая инвертирует список, ссылку на который получает в качестве аргумента.

1.4. «Скобки». Дано слово, состоящее из круглых и фигурных скобок. Написать программу, которая определяет, является ли введенное слово правильным скобочным выражением. Подсказка: постепенно считывайте скобки и используйте стек (можно использовать список из задачи 1.3 с командами *push* и *pop*) для хранения открывающих скобок, для которых пока не считаны парные закрывающие скобки. (Рекурсивное определение правильного скобочного выражения: слово называется правильным скобочным выражением, если все скобки в нем можно разбить на пары, так что в каждой паре скобка, стоящая ближе к левому концу слова, открывающая, а вторая – закрывающая, при этом они имеют один и тот же тип, и, кроме того, слово, стоящее между парными скобками, является правильным скобочным выражением. Например, слова $()$, $\{\}\{\}$, $(\{\}\{\})$ – правильные скобочные выражения, а слова $\{\{, \{\}\}$, $\{\}\{\}$ – неправильные скобочные выражения.)

1.5. «Калькулятор». Написать программу, которая, используя стек (используйте код, полученный при решении задачи 1.3), вычисляет значение арифметического выражения, заданного в постфиксной форме.

1.6. Написать программу, которая получает на вход арифметическое выражение в инфиксной форме и выводит это выражение в постфиксной форме. Программа должна работать за время, ограниченное линейной функцией от размера входного слова.

1.7. «Лабиринт». Написать программу, которая находит кратчайший путь из левого верхнего угла лабиринта в нижний правый угол либо определяет, что такого пути нет. Лабиринт задан в виде прямоугольного клеточного поля, белые клетки в котором считаются проходимыми, а черные – непроходимыми. В первой строчке входа заданы размеры лабиринта N и M по горизонтали и вертикали соответственно. Далее идут N строчек, в каждой из которых дано слово в алфавите $\{\#, .\}$ из M символов. Символ '#' означает черную клетку, а '.' – белую. Выход программы должен содержать последовательность пар координат клеток, по которым нужно идти, либо слово «NO». Использовать очередь (например, список из задачи 1.3 с командами *push* и *shift*) для того, чтобы хранить новые найденные клетки. Последовательно из начала очереди брать (команда *shift*) клетки, чтобы проверить, можно ли из них сделать шаг в новые, не найденные пока клетки, которые помещать (команда *push*) в конец очереди. Как растет время работы алгоритма в зависимости от N и M в худшем случае?

1.8. «Двоичное дерево поиска». Реализовать структуру данных «двоичное дерево поиска» с функциями создания и удаления дерева, добавления пары (ключ, значение) и удаления пары по ключу, где ключ есть целое число, а значение – действительное число. Написать функции $wfs(t)$ и $dfs(t)$ обхода дерева в ширину и в глубину. В первом случае следует использовать очередь, а во втором – рекурсию или стек.

1.9. Для структуры данных «двоичное дерево поиска» реализовать функцию префиксного обхода дерева $traverse(tree, f)$, которая ко всем значениям, хранящимся в дереве $tree$, применяет функцию $f(item, depth)$. В качестве аргументов функция f получает ссылку на элемент дерева $item$ и глубину $depth$ этого элемента в дереве. Реализовать с помощью функции $traverse$ вывод описания дерева в стандартный поток вывода (префиксное описание дерева).

1.10. «Записная книжка I». Написать программу, которая реализует функциональность телефонной записной книжки. А именно, из стандартного входа программа получает последовательность команд на добавление (*INSERT*) или поиск (*FIND*) записей. Примеры команд: *INSERT Sidorov 1234567*, *INSERT Ivanov 7654321*, *FIND Sidorov*. При выполнении команды *INSERT* программа добавляет пару (фамилия, номер) в своё хранилище и выводит строку «OK», если в хранилище нет записи с такой фамилией, или изменяет соответствующую указанной фамилии запись и выводит строку «*Changed. Old value = X*», если запись с такой фамилией уже есть в хранилище и соответствующий телефонный номер был X . При выполнении команды *FIND* программа выводит телефонный номер для указанной фамилии или выводит «NO», если указанной фамилии нет в справочнике. Следует использовать структуры, состоящие из двух элементов *name* и *number*. Использовать технику динамического выделения памяти для хранения записей. Оценить, как в среднем растёт число элементарных операций при выполнении команд *INSERT* и *FIND* с ростом числа хранимых записей. Записи хранить в массиве или списке. Рассмотреть два случая: а) записи хранятся в отсортированном по фамилиям (в алфавитном порядке) виде; в случае хранения в массиве записей используйте метод деления пополам (см. 3.8); б) записи хранятся в произвольном порядке (например, в порядке добавления).

1.11. «Записная книжка II». Решите задачу 1.10, используя двоичное дерево поиска.

1.12. «Записная книжка III». Решите задачу 1.10, используя АВЛ-дерево.

1.13. Приведите описание рекурсивной процедуры $check_tree(h, p)$, которая проверяет, является ли дерево АВЛ-деревом.

1.14. «Записная книжка IV». Решите задачу 1.10, используя хэш-таблицу с разрешением коллизий методом цепочек либо методом открытой адресации.

1.15. «Поиск путей с заданной суммой». Написать программу, выводящую на экран «YES», если в данном дереве существует путь от корня до листа, сумма элементов в вершинах

которого равна заданному целому числу S , и « NO » – в противном случае. Считается, что в пустом дереве сумма элементов пути равна 0.

1.16. Поставить численный эксперимент, чтобы определить, как зависит суммарное число столкновений и суммарное время работы при добавлении в хэш-таблицу с открытой адресацией и двойным хешированием K случайных ключей при размере таблицы N . Нарисовать графики зависимости суммарного числа столкновений и суммарного времени работы при добавлении K ключей от степени заполнения таблицы $\varepsilon = K / N$. Использовать $N = 10\,007$ и $N = 257$. Сделайте вывод о полезности двойного хеширования, сравнив результаты со случаем, когда $h_2(K) = 1$.

1.17. «Неприкасаемый король». На поле $C3$ шахматной доски неподвижно стоит белый король. Затем на доске устанавливаются имеющие возможность двигаться белый ферзь – на любую свободную клетку и черный король – на любую возможную для него клетку. Доказать, что для любой «возможной и разумной*» начальной позиции мат черному королю может быть объявлен за число ходов меньше 25 (независимо от того, чьим является первый ход).

*Невозможной является, например, позиция, когда два короля оказались на соседних клетках, а неразумной, – когда на соседних клетках оказались черный король и незащищенный белый ферзь.)

Индивидуальные задачи для программирования на языке Ассемблера

Каждому студенту выдается свой вариант задания.

Постановка задачи

Дан текст (последовательность символов), содержащий не более 100 элементов. Признаком конца текста считается символ с кодом 0.

Требуется:

- Ввести текст с клавиатуры и записать его в память ЭВМ.
- Определить, обладает ли этот текст заданным свойством, указанным в Вашем варианте задания.
- Преобразовать текст по правилу 1 Вашего задания, если он обладает заданным свойством, и по правилу 2 – в противном случае.
- Вывести на экран исходный и преобразованный тексты, а также номер и формулировку примененного правила.

Варианты задания

Свойство исходного текста:

1. Текст оканчивается заглавной латинской буквой, которая больше не встречается в тексте.
2. Текст начинается цифрой и оканчивается цифрой, причем эти цифры различны.
3. Текст начинается латинской буквой и оканчивается латинской буквой.
4. Текст содержит не менее трех латинских букв.
5. Текст содержит равное количество заглавных и строчных латинских букв.
6. Текст не содержит иных символов, кроме цифр и латинских букв.

Правило 1 преобразования текста:

1. Заменить каждую заглавную латинскую букву следующей по алфавиту ($A \rightarrow B, B \rightarrow C, \dots, Z \rightarrow A$).
2. Заменить каждую ненулевую цифру соответствующей ей по порядковому номеру строчной буквой латинского алфавита ($1 \rightarrow a, 2 \rightarrow b$ и т.д.).
3. Заменить каждую заглавную латинскую букву цифрой, числовое значение которой равно остатку от деления порядкового номера буквы в алфавите на десять.
4. Заменить каждую строчную латинскую букву соответствующей заглавной буквой ($a \rightarrow A, b \rightarrow B, \dots, z \rightarrow Z$).
5. Заменить каждую заглавную латинскую букву соответствующей строчной буквой ($A \rightarrow a, B \rightarrow b, \dots, Z \rightarrow z$).

6. Заменить каждую заглавную латинскую букву «симметричной» в алфавите ($A \rightarrow Z, B \rightarrow Y, \dots, Z \rightarrow A$).

Правило 2 преобразования текста:

1. Перенести в начало текста все входящие в него цифры с сохранением порядка их следования.
2. Перевернуть текст, не используя дополнительную память.
3. Повторить каждый символ текста.
4. Удалить из текста все повторные вхождения его первого символа.
5. Оставить в тексте только те символы, которые входят в него ровно один раз.
6. В каждой группе следующих подряд одинаковых символов оставить только один.

Требования к программе

1. Вывод исходного текста должен быть выполнен сразу после ввода, до анализа и преобразования.
2. Вывод преобразованного текста должен быть выполнен только после завершения преобразования.
3. Алгоритмы преобразования по правилам 1 и 2 должны быть оформлены в виде подпрограмм.
4. Программа должна сохранять работоспособность при любых входных данных.

4. Критерии оценивания

За каждый вопрос из контрольного задания студент получает от 0 до максимального балла в зависимости от полноты представленного ответа (решения). Критерии проставления баллов утверждаются на заседании учебно-методической комиссии кафедры. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за теоретические знания по каждому контрольному заданию:

Оценка	Набранные баллы
отлично (10)	более 88%
отлично (9)	от 78% до 88% включительно
отлично (8)	от 68% до 78% включительно
хорошо (7)	от 58% до 68% включительно
хорошо (6)	от 48% до 58% включительно
хорошо (5)	от 38% до 48% включительно
удовлетворительно (4)	от 28% до 38% включительно
удовлетворительно (3)	от 18% до 28% включительно
неудовлетворительно (2)	от 08% до 18% включительно
неудовлетворительно (1)	не более 08%

Каждая лабораторная работа и задача из задания при полном правильном решении оценивается в определенное количество баллов от 5 до 25. Баллы за полностью правильное решение определяются преподавателями и утверждаются на заседании учебно-методической комиссии кафедры. Процент суммарно набранных баллов от максимально возможного количества определяет оценку за практические знания студента по вышеприведенной таблице.

Итоговая оценка за дифференцированный зачет выставляется на основании оценок, полученных за контрольные работы (далее x и y), и оценки за задания, (далее z) по следующей формуле $(x+y)*z/(z+(x+y)/2)$, с округлением результата до целого сверху.

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

Время проведения каждой контрольной работы составляет 2 академических часа.

Во время проведения контрольных работ обучающиеся могут пользоваться программой дисциплины и любыми рукописными материалами.