

**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:	Техническая физика
профиль подготовки:	Техническая физика космических летательных аппаратов 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 составил: К.А. Беклемышева, канд. физ.-мат. наук, доцент

Программа обсуждена на заседании кафедры информатики и вычислительной математики 06.02.2020

Аннотация

Годовой курс информатики с C++ в качестве основного языка программирования. Первый семестр концентрируется на базовых конструкциях языка C++, классических алгоритмах и структурах данных. Во втором семестре рассматривается объектно-ориентированное программирование и некоторые продвинутое возможности современного C++.

1. Цели и задачи

Цель дисциплины

- Формирование базовых знаний по информатике для дальнейшего использования в других областях 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
УК-3 Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	УК-3.1 Способен устанавливать разные виды коммуникации (учебную, научную, деловую, неформальную и др.)
	УК-3.2 Взаимодействует с другими членами команды для достижения поставленной задачи
ОПК-6 Способен работать с распределёнными базами данных, с информацией в глобальных компьютерных сетях, применяя современные образовательные и информационные технологии	ОПК-6.1 Знаком с принципами работы с распределёнными базами данных и с информацией в глобальных компьютерных сетях
	ОПК-6.2 Использует современные образовательные и информационные технологии и сервисы сети Интернет при решении задач в области профессиональной деятельности

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;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, хэш-таблицы;
- конструкции языка программирования C++;
- парадигму объектно-ориентированного программирования;
- приемы разработки программ на C++.

уметь:

- Разрабатывать полные законченные программы на языке C++;
- применять объектно-ориентированный подход для написания программ;
- выбирать оптимальные алгоритмы для современных программ;
- использовать современные средства написания и отладки программ;
- использовать знания по информатике для приложения в других областях знания и дисциплинах естественнонаучного содержания.

владеть:

- Языком программирования C++;
- современными средствами написания и отладки программ;
- методами создания программ с использованием библиотек.

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	Базовые конструкции языка C++	6		12	16
2	Представление данных в оперативной памяти	4		8	10
3	Потоки ввода-вывода	2		4	4
4	Введение в теорию алгоритмов	6		12	24
5	Эффективные алгоритмы сортировки	2		4	8
6	Введение в структуры данных	2		4	4
7	Структуры данных: списки	2		4	4
8	Структуры данных: деревья	3		6	10
9	Структуры данных: хэш-таблицы	3		6	10
10	Базовые принципы объектно-ориентированного программирования	4		8	12
11	Наследование, интерфейс и реализация	4		8	12
12	Работа с памятью в современном C++	6		12	18
13	Перегрузка операторов	4		8	12
14	Обработка исключений	2		4	6
15	Шаблоны	2		4	6
16	Библиотека STL	6		12	18
17	Приведение и автоматическое выведение типов	2		4	6
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. Базовые конструкции языка C++

Общая структура простейшей программы. Переменные - имена, служебные слова, стандартные имена. Базовые типы данных. Арифметические операторы. Операторы сравнения. Циклы - for, while, do-while. Условные операции и переходы - if-else, тернарный оператор, switch, break и continue.

Функции. Описание функций. Формальные и фактические параметры. Способы передачи параметров. Рекурсивный вызов функции. Сопоставление итерации и рекурсии.

Составные типы данных. Массивы. Структуры. Битовые поля. Объединения (union). Перечисления (enum). Декларация typedef.

Строки как массив символов со служебным символом конца строки. Работа со строками. Типовые строковые функции.

2. Представление данных в оперативной памяти

Хранение данных в памяти. Модель памяти flat. Понятие адреса и указателя. Связь указателей с массивами. Использование указателей совместно со структурами. Динамическая память. Понятие о стеке и куче. Динамическое выделение и очистка памяти. Понятие об утечках памяти.

3. Потоки ввода-вывода

Представление о потоках ввода-вывода. Работа с файлами как с потоками ввода-вывода. Текстовый и бинарный ввод-вывод.

4. Введение в теорию алгоритмов

Интуитивное понятие алгоритма. Формализация алгоритма для реализации на языке программирования. Понятие вычислительной сложности. Оценка сложности простейших алгоритмов. Локально жадные алгоритмы. Примеры локально жадных алгоритмов. Понятие о динамическом программировании. Примеры задач динамического программирования.

5. Эффективные алгоритмы сортировки

Понятие внутренней и внешней сортировки. Устойчивая сортировка. Сортировка inplace. Сортировка простыми вставками, простым выбором, метод «пузырька». Шейкер сортировка. Метод Шелла. Быстрая сортировка Хоара. Сортировка слиянием. Пирамидальная сортировка. Оценка сложности алгоритмов сортировки для входных данных различных видов.

6. Введение в структуры данных

Абстрактные структуры данных: список, стек, очередь, очередь с приоритетом, ассоциативный массив. Возможность реализации абстрактных структур данных на различных структурах хранения.

7. Структуры данных: списки

Односвязные и двусвязные списки. Варианты реализации. Базовые операции над списками. Алгоритмы на списках. Оценки алгоритмической сложности операций.

8. Структуры данных: деревья

Бинарные деревья поиска. Общая логика и подход к реализации. Необходимость балансировки дерева. Различные алгоритмы балансировки - АВЛ-деревья, красно-чёрные деревья. Оценки алгоритмической сложности операций.

9. Структуры данных: хэш-таблицы

Хэш-таблицы. Общая логика и подход к реализации. Функция хэширования, требования к ней, варианты функций хэширования для различных данных. Возникновение и разрешение коллизий. Хэш-таблицы с прямой и открытой адресацией, использование техники двойного хэширования при открытой адресации. Оценки алгоритмической сложности операций.

Семестр: 2 (Весенний)

10. Базовые принципы объектно-ориентированного программирования

Сравнение процедурного и объектного программирования. Понятие класса и объекта. Базовая структура класса - поля, методы, конструкторы и деструкторы. Статические поля и методы класса.

Базовые принципы ООП: инкапсуляция, наследование, полиморфизм. Публичная и приватная части класса. Полиморфизм функций и методов. Полиморфизм времени компиляции и времени выполнения.

11. Наследование, интерфейс и реализация

Понятие наследования в объектном программировании. Перегрузка при наследовании. Работа конструкторов и деструкторов при наследовании. Модификаторы доступа при наследовании. Дружественные классы. Директивы `override` и `final`. Виртуальные методы. Абстрактные классы. Понятие интерфейса. Интерфейс и реализация. Применение интерфейсов. Таблица вызова виртуальных методов. Множественное наследование.

12. Работа с памятью в современном C++

Ссылки и указатели. Преимущества и недостатки ссылок и указателей, их типовые области применения. Передача параметров и возврат значений по ссылке. Использование указателей для реализации алгоритмов и структур данных.

Директива `const` - логика введения в язык и типовые сценарии применения. Константные методы и константные параметры методов и функций. Сочетание константности с указателями и ссылками. Константная целостность (`const correctness`).

Инициализация в C++. Списки инициализации. Использование инициализации при конструировании класса. Делегирование конструкторов.

Умные указатели. Виды умных указателей. Применение умных указателей для борьбы с утечками памяти.

Семантика перемещений (`move semantics`).

13. Перегрузка операторов

Перегрузка арифметических операторов, операторов ввода и вывода, операторов сравнения. Использование перегрузки операторов для упрощения программы. Использование перегрузки операторов для обеспечения возможностей метапрограммирования.

Перегрузка инкремента и декремента как отдельный случай перегрузки операторов.

Конструктор копирования и оператор присваивания, необходимость их перегрузки в отдельных случаях.

14. Обработка исключений

Механизм исключений. Обработка ошибок и проблем при выполнении программы. Сопоставление механизмов исключений и кодов возврата. Классы исключений. Освобождение ресурсов при обработке исключений.

15. Шаблоны

Понятие метапрограммирования. Параметризованные функции (шаблоны). Параметризованные классы (шаблоны). Сочетание шаблонов с наследованием и с дружественными классами.

16. Библиотека STL

STL - типовая библиотека шаблонов.

Контейнеры в составе STL - последовательные, упорядоченные ассоциативные, неупорядоченные ассоциативные. Внутреннее устройство контейнеров `vector`, `set`, `map`, `unordered_set`, `unordered_map`.

Понятие итератора. Виды итераторов. Внутренняя реализация итераторов.

Обобщённые алгоритмы в составе STL и их применение к контейнерам. Особенности синтаксиса. Ограничения, вызванные структурой контейнеров и реализацией итераторов.

Другие компоненты STL: адаптеры, функторы. Стек, очередь, очередь с приоритетами как адаптеры. Реализация адаптеров на типовых контейнерах. Классы-функторы.

Лямбда-функции, их применение совместно с алгоритмами STL.

17. Приведение и автоматическое выведение типов

Динамическое приведение типов и идентификация (RTTI).

Автоматическое выведение типов, ключевое слово auto.

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

Компьютерный класс с доской, проектором или телевизором. Подключение к сети Интернет.

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 C [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.
3. Язык программирования C++ (стандарт C++11) : Краткий курс [Текст] = A Tour of C++, [учеб. пособие для вузов] /Бьерн Страуструп ; пер. с англ. под ред. Н. Н. Мартынова. -М., БИНОМ, 2017
4. C++ : базовый курс [Текст] : [учеб. пособие для вузов] / Г. Шилдт ; пер. с англ. и ред. Н. М. Ручко .— 3-е изд. — М. : Вильямс, 2015, 2016 .— 624 с.

Дополнительная литература

1. Алгоритмы: построение и анализ [Текст] : [учебник для вузов] / Т. Кормен [и др.] ; [пер. с англ. И. В. Красикова и др.] .— 3-е изд. — М. : Вильямс, 2014 .— 1328 с.
2. Алгоритмы и структуры данных [Текст] / Н. Вирт ; пер. с англ. Д. Б. Подшивалова .— 2-е изд., испр. — СПб. : Невский Диалект, 2001,2005 .— 352 с.
3. Алгоритмы [Текст] : [учеб. пособие для вузов] / С. Дасгупта, Х. Пападимитриу, У. Вазиран ; пер. с англ. А. А. Куликова ; под ред. А. Шеня .— М. : МЦНМО, 2014 .— 320 с.

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

Не используются

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 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;
- решение задач, предлагаемых студентам на лекциях и лабораторных занятиях;
- подготовку к лабораторным занятиям, дифференцированному зачёту.

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 каждый шаг решения необходимо аргументировать, ссылаясь на известные теоретические сведения.

При подготовке к лабораторным занятиям необходимо повторять ранее изученные основные понятия. В начале занятия, как правило, проводится короткий (10-15 минут) опрос по материалу прошедших занятий в устной или письменной форме. Обычно придерживаются следующей схемы: изучение материала лекции по конспекту в тот же день, когда была прослушана лекция (10-15 минут); повторение материала накануне следующей лекции (10-15 минут), проработка учебного материала по конспектам лекций, учебной и научной литературе, подготовка ответов на вопросы, предназначенных для самостоятельного изучения (1 час неделю), подготовка к практическому занятию, решение задач (1 час). Важно добиться понимания изучаемого материала, а не механического его запоминания. При затруднении изучения отдельных тем, вопросов, следует обращаться за консультациями к лектору или преподавателю, ведущему лабораторные занятия.

Обязательным требованием является выполнение домашних заданий, которые в отдельных случаях могут быть оформлены как контесты с автоматическим тестированием решений. Проверка текста решения преподавателем является обязательным элементом контроля, вне зависимости от наличия автоматического тестирования.

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

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Техническая физика
профиль подготовки:	Техническая физика космических летательных аппаратов Физтех-школа Аэрокосмических Технологий кафедра информатики и вычислительной математики
курс:	<u>1</u>
квалификация:	бакалавр

Семестры, формы промежуточной аттестации:

- 1 (осенний) - Дифференцированный зачет
- 2 (весенний) - Дифференцированный зачет

Разработчик: К.А. Беклемышева, канд. физ.-мат. наук, доцент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-3 Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	УК-3.1 Способен устанавливать разные виды коммуникации (учебную, научную, деловую, неформальную и др.)
	УК-3.2 Взаимодействует с другими членами команды для достижения поставленной задачи
ОПК-6 Способен работать с распределёнными базами данных, с информацией в глобальных компьютерных сетях, применяя современные образовательные и информационные технологии	ОПК-6.1 Знаком с принципами работы с распределёнными базами данных и с информацией в глобальных компьютерных сетях
	ОПК-6.2 Использует современные образовательные и информационные технологии и сервисы сети Интернет при решении задач в области профессиональной деятельности

2. Показатели оценивания компетенций

В результате изучения дисциплины «Информатика» обучающийся должен:

знать:

- Основы теории алгоритмов;
- свойства алгоритмов, проблемы алгоритмической сложности и алгоритмической неразрешимости;
- общие понятия о структурах данных: стеки, очереди, списки, деревья, хэш-таблицы;
- конструкции языка программирования C++;
- парадигму объектно-ориентированного программирования;
- приемы разработки программ на C++.

уметь:

- Разрабатывать полные законченные программы на языке C++;
- применять объектно-ориентированный подход для написания программ;
- выбирать оптимальные алгоритмы для современных программ;
- использовать современные средства написания и отладки программ;
- использовать знания по информатике для приложения в других областях знания и дисциплинах естественнонаучного содержания.

владеть:

- Языком программирования C++;
- современными средствами написания и отладки программ;
- методами создания программ с использованием библиотек.

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

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 лабораторные работы, домашние задания, контрольные работы.

Вопросы к зачету.

1 (осенний) семестр

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. Вычислимые функции и их свойства. Невычислимые функции. Различные эквивалентные определения множества вычислимых функций. Алгоритмическая сложность.

Алгоритмические языки на примере Си. 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.

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

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 хеширования при открытой адресации). Оценки алгоритмической сложности операций поиска, добавления и удаления элемента.

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

2 (весенний) семестр

Сравнение синтаксиса языка С и С++. Перегрузка имен (полиморфизм) функций. Константы времени компиляции и статическая типизация в объектно-ориентированных языках программирования (по Барбаре Лисков). Структуры в С++. Синтаксис классов

изолированных (невзаимодействующих) объектов в языке C++. Классы, представляющие нормативное знание. Конструкторы и деструкторы. Интерфейс и его реализация. Трансформирование базовых типов языка C++ в классы – основной источник формирования классов в языке C++. Классы-оболочки (Wrappers) с минимальным интерфейсом для базовых типов данных (примитивов). Полиморфизм конструкторов. Методы доступа. Ключевое слово `this`. Постоянные и модифицируемые члены класса. Массивы объектов. Классы-оболочки с полным интерфейсом инкапсулированного базового типа. Перегрузка операторов в C++. Инкапсуляция массивов базовых типов: абстрактные типы данных. Методы доступа. Перегрузка оператора индекса.

Дружественные функции: функции, дружественные классам – функции за пределами классов (нарушение концепции объектно-ориентированного программирования). Инкапсуляция дружественных функций в класс `Visitor`. Объектно-ориентированное программирование без использования дружественных функций: классы данных и классы функций. Потоки ввода-вывода данных в C++. Форматируемый ввод/вывод и манипуляторы ввода/вывода. Перегрузка пользовательских операторов ввода/вывода как дружественных функций. Файловый ввод/вывод. Неформатируемый двоичный ввод/вывод (байтовые потоки). Инкапсуляция объектов и вертикальное делегирование функций.

Адресное пространство приложения. Динамические члены класса. Динамическая память (куча) и статическая память (стек). Операции с динамической памятью (кучей). Конструкторы и деструкторы объектов в динамической памяти. Проблема утечки памяти в C++. Статические переменные – члены класса. Инициализация статических членов класса. Копирование объектов. Присваивание объектов. Передача и возвращение функциями объектов по значению. Конструкторы копий. Клонирование объектов в C++. Передача и возвращение ссылок на объекты.

Инкапсуляция массивов объектов. Классы-оболочки контейнеров с расширенным интерфейсом. Специальные методы создания объектов. Создание объектов посредством статических методов. Класс `Singleton`. Косвенное создание объектов и горизонтальное делегирование функций. Класс-заместитель `Proxy`. Классы-оболочки для указателей базовых типов. Перегрузка операций указывания и разыменования. Интеллектуальные указатели. Повторное использование классов. Открытое наследование классов. Распространение и перегрузка наследуемых методов. Вызов конструкторов базового класса (суперкласса). Наследование функций и операций (Inheriting operations and functions). Перегрузка оператора присвоения. Множественное и виртуальное наследование. Закрытое наследование классов. Классы-адаптеры. Классы – композиты, в объекты которых вложены объекты других классов. Делегирование функций.

Шаблоны. Полиморфизм функций и параметризованные функции (шаблоны). Параметризованные классы (шаблоны). Статический полиморфизм. Шаблоны интеллектуальных указателей. Шаблоны классов-контейнеров. Вектор (массив). Строка. Связанный список. Обработка исключительных ситуаций.

Внутренние и дружественные классы. Реализация двумерного вектора. Перегрузка оператора двойного индекса. Шаблоны итераторов для классов-контейнеров. Стандартная библиотека шаблонов STL (Standard Template Library). Классы-контейнеры и итераторы. Обобщенные алгоритмы. Функторы.

Классы с виртуальными функциями. Подстановочный критерий Барбары Лисков. Структура объектов и таблицы виртуальных функций. Динамический полиморфизм. Чистые виртуальные функции и абстрактные классы. Полиморфное наследование абстрактных классов в C++. Производящие функции и фабрики объектов. Динамическое приведение типов и идентификация (RTTI) в Visual C++.

Наследование классов с расширением интерфейса. Шаблоны оболочек-адаптеров и внешний полиморфизм.

Одновременное (конкурентное) выполнение потоков вычислений.

1 (осенний) семестр

Задание 1

1. Решение простых алгоритмических задач

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).

1.1. «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.

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

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

1.3. «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 .

1.4. «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

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

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

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

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

1.7. «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.

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

2. Алгоритмы

2.1. «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: упорядочите атлетов по силе и стройте башню сверху. Вверх естественно поместить самого слабого. Входом является число атлетов n и n пар (масса, сила).

2.2. «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.

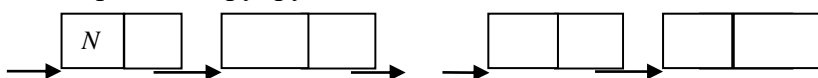
2.3. «Коммивояжер». Написать программу, которая, используя один из возможных жадных алгоритмов, находит на плоскости ломаную линию, идущую из A в B по всем заданным точкам $\{A_1, A_2, A_3, \dots, A_n\}$, как можно меньшей длины. Есть ли такие входные данные, когда написанная программа находит не самый 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 строке 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 AVL-дерево.

1.13. Приведите описание рекурсивной процедуры $check_tree(h, p)$, которая проверяет, является ли дерево AVL-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 $\square = K / N$. 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.)

2 (весенний) семестр

Задание 1

Списки

Реализовать класс List для создания структуры данных двухсвязного списка, каждый элемент которого Node хранит уникальный, случайно сгенерированный ключ.

Класс должен содержать методы:

- добавление элемента в начало и конец списка (*аргументы*: значение ключа для нового узла);
- вывод списка на экран;
- поиск элемента по значению ключа (*аргументы*: значение ключа);
- удаление элемента из начала списка, из конца списка, а также по значению ключа (*аргументы*: значение ключа);
- поиск количества элементов.

Программа должна создать заданный пользователем список и продемонстрировать работоспособность всех методов класса.

Бинарные деревья

Реализовать класс Tree для создания структуры данных бинарного дерева, каждый элемент которого Node хранит уникальный, случайно сгенерированный ключ.

Класс должен содержать методы:

- добавление узла (*аргументы*: значение ключа для нового узла);
- вывод дерева на экран;
- поиск узла дерева по значению ключа (*аргументы*: значение ключа);
- удаление узла по значению ключа (*аргументы*: значение ключа);
- поиск максимального элемента дерева;
- поиск суммы всех элементов дерева;
- поиск количества узлов дерева;
- поиск максимальной глубины дерева.

Программа должна создать дерево заданного размера и продемонстрировать работоспособность всех методов класса.

Векторы и матрицы

Реализовать класс Vector3 для трехмерных векторов. Перегрузить для него основные арифметические операции, в том числе:

- умножение на константу;
- сложение;
- вычитание;
- скалярное и векторное умножение.

Реализовать класс Matrix3 для матриц 3×3 . Перегрузить для него основные арифметические операции:

- сложение;

- вычитание;
- определитель;
- умножение на вектор;
- умножение двух матриц.

Аналитическая геометрия

С использованием класса Vector реализовать класс Line, Plane, Segment и Triangle для прямой, плоскости, отрезка и треугольника соответственно. Реализовать класс CollisionDetector, который умеет находить пересечение любых двух объектов указанных типов (отрезок с отрезком, отрезок с плоскостью, треугольник с плоскостью и т.д.).

Задание 2

Геометрические объекты

Реализовать класс Figure, содержащую общую информацию о геометрическом объекте и основные методы работы с ним:

- положение в пространстве;
- поворот;
- перемещение;
- растяжение вдоль осей.

Реализовать набор классов, наследованных от Figure (и, возможно, друг от друга): Triangle (треугольник), Sphere (сфера), Ellipsoid (эллипсоид), Cube (куб), Parallelepiped (параллелепипед), ...

Реализовать в классе Figure статический метод CollisionDetector, который умеет определять, пересекаются ли два геометрических объекта.

Реализовать в классе Figure счетчик созданных объектов.

«Дождь»

В замкнутом кубическом объеме на верхней границе случайно генерируются различные объекты типа Figure с произвольной начальной скоростью, направленной вниз. После пересечения нижней границы объекты исчезают. Одновременно в объеме должно быть порядка 1000 объектов.

Программа должна работать без утечек памяти, проверить это можно при помощи valgrind.

Дополнительное задание. Визуализация процесса. Максимально увеличить количество объектов, которые могут одновременно находиться в объеме.

Бинарные деревья поиска

На основе класса Tree из первого задания реализовать класс SearchTree для создания структуры данных бинарного дерева поиска.

Универсальный список

На основе класса List из первого задания реализовать аналог list из STL.

Моделирование

Дополнительное задание ко всем задачам данного раздела – визуализация процесса, описанного в задаче.

«Задача трех тел»

В неограниченном трехмерном пространстве находится N шарообразных объектов различной массы и различного радиуса. В начальный момент времени они имеют случайные (либо заданные пользователем) векторы скоростей. Объекты взаимодействуют друг с другом согласно закону всемирного тяготения. Определить, какие из объектов столкнутся, и время столкновения.

Дополнительное задание. Вводится поправка к закону всемирного тяготения, которая обеспечивает диссипацию энергии – например, если спутник находился на стабильной орбите вокруг планеты, после введения поправки он будет на нее постепенно падать. Необходимо разработать модель этой поправки (например, приливные силы или гравитационные волны) и ввести ее в программу.

«Тренировка футболистов»

Команда футболистов отрабатывает индивидуальные действия. Каждый играет за себя. Цель футболиста – завладеть мячом и забить гол. Таким образом, он может выполнять два действия:

1. бороться за мяч.
2. бить по воротам, как только мяч оказался у него.

После удара по воротам вратарь выбивает мяч в произвольное место поля. Футболист забивает гол с определенной вероятностью, зависящей от расстояния до ворот. Команда играет определенное время, потом определяется победитель – по забитым мячам. Вывести на экран счет и последовательность действий каждого футболиста.

«Ветеран Броуновского движения»

В кубическом замкнутом объеме находится N «маленьких» шарообразных частиц массы m и одна «большая» массы M . В начальный момент времени они имеют случайные векторы скоростей. Частицы упруго отталкиваются друг от друга и от стенок. Построить график зависимости скорости «большой» частицы от времени.

«Воробьи»

Бабушка на скамейке периодически (пусть раз в Q минут) бросает воробьям по одной ровно N хлебных крошек. Когда бросают крошку хлеба, первый подлетевший к ней воробей хватается крошку, относит ее в сторону и съедает. За счет этого он пропускает «розыгрыш» следующей крошки. Первоначально у скамейки находится M воробьев. Каждую минуту к стае прилетает еще один воробей. Съев P крошек, воробей наедается и улетает. Чтобы количество воробьев не увеличивалось, должно выполняться соотношение $Q = N/P$.

Итак, воробей может выполнять действия:

2. Прилететь.
3. Ждать раздачи крошек.
4. Драться за крошку.
5. Отлететь с крошкой в сторону и съесть ее.
6. Улететь.

Вывести на экран последовательность действий каждого из воробьев.

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. Процентсуммарно 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. Процентсуммарно набранных баллов от

максимально возможного количества определяет оценку за практические знания студента по вышеприведенной таблице.

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.