МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федерального государственного бюджетного образовательного учреждения высшего образования

«Университет «Дубна»

(филиал «Протвино» государственного университета «Дубна»)

Кафедра «Информационные технологии»

УТВЕРЖДАЮ
Директор

Филиал «Протвино» государственного университета «Дубна»

Евсиков Александр Александрович
я подтверждаю этот документ своей удостоверяющей подписью
2024-10-08 13:47:33

_____А.А. Евсиков

« 28» июня 2024г.

Рабочая программа дисциплины

Структуры и алгоритмы обработки данных

наименование дисциплины (модуля)

Направление подготовки

09.03.01 – «Информатика и вычислительная техника»

код, наименование

Уровень высшего образования

бакалавриат

бакалавриат, магистратура, специалитет

Направленность (профиль) образовательной программы

«Программное обеспечение вычислительной техники и автоматизированных систем»

Форма обучения

очная, заочная

очная, очно-заочная, заочная

Протвино, 2024 г.

Автор программы:
Кульман Т.Н., доцент, к.т.н., кафедра «Информационные технологии»

_____
(подпись)

Программа составлена в соответствии с Федеральным Государственным образовательным стандартом высшего образования и учебным планом по направлению подготовки 09.03.01 «Информатика и вычислительная техника»

Программа рассмотрена на заседании кафедры «Информационные технологии»

Протокол заседания № _11_ от «_20_» ___июня___ 2024 г.

Заведующий кафедрой _____ / Черноверская В.В./
(подпись) (фамилия, имя, отчество)

Рецензент:

Оглавление

1 Цели и задачи освоения дисциплины (модуля)

Целью освоения дисциплины «Структуры и алгоритмы обработки данных» является формирование у студентов профессиональных компетенций в соответствии с требованиями ФГОС ВО по направлению подготовки бакалавров 09.03.01 «Информатика и вычислительная техника» с учетом направленности бакалаврской программы – «Программное обеспечение вычислительной техники и автоматизированных систем».

Дисциплина «Структуры и алгоритмы обработки данных» преподаётся два семестра и посвящена изучению структур данных, применяемых в программировании и современных методов построения и анализа алгоритмов. Весь курс можно разделить на 3 части:

1. изучение базовых структур данных и освоение технологии их программирования;
2. изучение алгоритмов обработки, связанных со структурами данных (сортировка, поиск, копирование и др.); рассмотрение вопросов эффективности алгоритмов;
3. изучение библиотеки шаблонов (STL), – как мощного инструмента, содержащего реализацию многих структур и алгоритмов, повышающего эффективность труда программистов.

Задачами курса являются:

- овладение комплексом знаний основных положений теории структур и алгоритмов обработки данных;
- применение теоретических знаний при создании информационных систем;
- приобретение практических навыков применения структур и алгоритмов, необходимых для программистов;
- обучение использованию библиотеки STL при разработке программ, – при этом повышается надёжность программ, их переносимость, а также уменьшаются расходы на их создание.

Формируемые компетенции: ОПК-2; ОПК-8

Объектами профессиональной деятельности в рамках изучаемой дисциплины (модуля) являются:

- программное обеспечение средств вычислительной техники и автоматизированных систем (программы, программные комплексы и системы);
- автоматизированные системы обработки информации и управления.

2 Место дисциплины (модуля) в структуре ОПОП

Дисциплина «Структуры и алгоритмы обработки данных» относится к Обязательной части Блока 1 дисциплин учебного плана Б1.О.18.03 и преподается два семестра (IV семестр II курса и V семестр III курса).

Приступая к изучению дисциплины «Структуры и алгоритмы обработки данных», студент должен иметь знания и навыки по дисциплинам:

- Программирование на языке высокого уровня,
- Дискретная математика,
- Математическая логика и теория алгоритмов,

Данная дисциплина является предшествующей для следующих дисциплин:

- Базы даны,
- Интерфейсы информационных систем,
- Компьютерная графика,
- Проектирование информационных систем,
- Информационные системы и технологии.

3 Планируемые результаты обучения по дисциплине (модулю)

Раздел заполняется в соответствии с картами компетенций.

Формируемые компетенции: **ОПК-2; ОПК-8**

Формируемые компетенции *(код и наименование)*	**Индикаторы достижения компетенций** *(код и формулировка)*	**Планируемые результаты обучения по дисциплине (модулю)**
ОПК-2: Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	ОПК-2.1 Ориентируется в спектре операционных систем и программных средств отечественного производства на рынке информационных продуктов	Знает общие принципы взаимодействия алгоритмов и данных в объектно-ориентированной технологии
		Знает базовые структуры организации данных и основные алгоритмы обработки данных, применяемые в программировании
	ОПК-2.2 Устанавливает оптимальные критерии работы программных средств при решении задач профессиональной деятельности	Умеет применять к решению прикладных задач базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы
		Умеет работать с библиотекой STL, знать её основные классы и алгоритмы
	ОПК-2.3 Использует современные информационные технологии сбора, обработки, хранения и поиска информации при решении задач профессиональной направленности	Владеет навыками использования сложных структур данных и программной реализации алгоритмов обработки данных для решения задач профессиональной направленности
		Владеет навыками использованию библиотеки шаблонов при разработке программ для повышения надёжности, переносимости и сокращения времени на разработку
ОПК-8: Способен разрабатывать алгоритмы и программы, пригодные для практического применения	ОПК-8.1 Выбирает инструментальные средства, языки программирования и технологии обработки данных на начальном этапе разработки программного продукта	Знает алгоритмические языки программирования, операционные системы и оболочки, современные среды разработки программного обеспечения (C++, Visual Studio)
	ОПК-8.2 Разрабатывает алгоритмы и программные коды программных модулей для практического применения	Умеет составлять алгоритмы, писать и отлаживать коды, тестировать работоспособность программы
		Умеет программировать такие структуры данных, как списки, стеки, очереди, бинарные деревья, множества и др.
	ОПК-8.3 Тестирует работоспособность программ и программных компонентов	Владеет навыками отладки программ и программных компонентов
	ОПК-8.4: Применяет языки программирования и современные программные среды разработки информационных систем для решения прикладных задач различных классов	Умеет писать программы на языках высокого уровня, использовать интегрированную среду разработки приложений для решения прикладных задач
		Владеет навыками программной реализации алгоритмов обработки данных на языке высокого уровня на основе интегрированной среды разработки приложений

Результат обучения сформулирован на основании требований профессиональных стандартов:

– 06.001 «Программист», обобщённая трудовая функция C5 - Интеграция программных модулей и компонент и проверка работоспособности выпусков программного продукта; трудовая функция C/02.5 - Осуществление интеграции программных модулей и компонент и верификации выпусков программного продукта; обобщённая трудовая функция D6 - Разработка требований и проектирование программного обеспечения; трудовая функция D/01.6 - Анализ требований к программному обеспечению;

4 Объем дисциплины (модуля)

Объем дисциплины (модуля) составляет 8 зачетных единицы, всего 288 часа. Дисциплина читается 2 семестра (**IV семестр II курса и V семестр III курса**):

119 часа составляет контактная работа обучающегося с преподавателем:

142 часов составляет самостоятельная работа обучающегося.

27 часов отводится на контроль.

IV семестр:

Зачётных единиц – 3;

Всего часов – 108;

34 часов – лекционные занятия;

34 часа – практические занятия.

40 часов – самостоятельная работа

Вид промежуточного контроля – зачёт.

V семестр:

Зачётных единиц – 5;

Всего часов – 180;

17 часов – лекционные занятия;

34 часа – практические занятия;

102 часа – самостоятельная работа;

27 часов – контроль.

Вид промежуточного контроля – экзамен.

5 Содержание дисциплины (модуля)

Лекции представлены в виде презентаций.

Очная форма обучения

Наименование и краткое содержание разделов и тем дисциплины (модуля) **Форма промежуточной аттестации по дисциплине (модулю)**	**Всего (часы)**	В том числе:							
		Контактная работа (работа во взаимодействии с преподавателем), часы из них					**Самостоятельная работа обучающегося, часы,** из них		
		Лекции	Практические занятия	Лабораторные занятия	Учебные занятия, направленные на проведение текущего контроля (коллоквиумы, контрольные занятия и др.). Входят в практические занятия.	**Всего**	Выполнение домашних заданий, подготовка к к/р	Подготовка творческих заданий.	**Всего**
IV семестр									
Раздел 1. Указатели для формирования структур данных Назначение указателей. Объявление указателей. Операторы работы с указателями. Автоматические и динамические переменные. Использование указателей для формирования динамических структур. Понятие структур данных и их классификация.	8	2	2		2	6	2		2
Раздел 2. Абстрактные типы данных (АТД). Линейные структуры Определение АТД: тип и структура данных, операторы. Классификация АТД. Линейные структуры: стек, очередь, дек. Операции над линейными списками. Реализация операций с использованием массивов и с помощью указателей. Стек и	8	2	2			4	2		2

функции. Передача параметров в функцию: по значению, по ссылке.									
Раздел 3. Связные списки Определение связного списка. Доступ к элементам списка. Работа со связанными списками: 1. Вывод на экран содержимого списка, 2. Удаление указанного узла, 3. Вставка узла в указанную позицию. Операторы "." и "->". Двусвязные списки.	10	2	4			6	4		4
Раздел 4. Текстовые файлы и потоки ввода/вывода Внутреннее представление файлов. Потоки ввода/вывода. Состояние программы после подключения и отключения потоков. Символьный ввод/вывод. Операторы ввода/вывода ">>" и "<<".	8	2	2			4	4		4
Раздел 5. Деревья Рекурсивное определение структуры дерева. Основные понятия и определения. Бинарные деревья. Обход дерева: прямой, обратный, симметричный. Способы представления бинарного дерева (в виде массива, с помощью указателей). Бинарные деревья поиска, идеально сбалансированные деревья и сбалансированные деревья (АВЛ-деревья). Деревья Фибоначчи.	11	3	2		2	7	4		4
Раздел 6. Восстановление сбалансированных деревьев. Б-деревья Балансировка узлов. Правило поворотов. RR-поворот (LL-поворот), RL-поворот. Включение и исключение в/из АВЛ-деревьев. Б-деревья, определение, свойства, основные операции. Работа с Б-деревьями.	5	3				3	2		2
Раздел 7. Оценка алгоритмов. Сортировки Сложность алгоритма, обозначение О-большое. Определение порядка алгоритма. Сравнение сложности различных алгоритмов. Внутренние	8	2	4			6	2		2

сортировки (выбором, вставками, пузырьком). Рассмотрение алгоритмов и анализ сложности. Быстрая сортировка и анализ сложности.									
Раздел 8. Сортировка слиянием. Сортировка Шелла Разбор алгоритмов сортировки слиянием и сортировки Шелла. Оценки эффективности.	8	2	2		2	6	2		2
Раздел 9. Графы, алгоритмы работы с графами Ориентированные и неориентированные графы. Определение, основные понятия и свойства. Представления графов. Поиск в глубину, поиск в ширину.	6	2	2			4	2		2
Раздел 10. Топологическая сортировка. Алгоритмы поиска кратчайшего пути в графах Отношение частичного порядка. Топологическая сортировка. Кратчайшие пути в графе. 1. Поиск кратчайшего расстояния от одной из вершин графа до всех остальных (алгоритм Дейкстры, алгоритм Беллмана – Форда). 2. Поиск кратчайших путей между всеми парами вершин графа (алгоритм Флойда-Уоршелла). 3. Переборные алгоритмы (поиск в глубину, поиск в ширину).	8	2	2		2	6	2		2
Раздел 11. Алгоритмы поиска. Хеширование Последовательный, бинарный поиск. Понятие хеширования. Хеш-таблицы. Разрешение коллизий. Функции хеширования. Открытое и закрытое хеширование.	4	2	1			3	1		1
Раздел 12. Поиск в тексте. Методы разработки алгоритмов Постановка задачи. Изучение алгоритмов: 1. прямого поиска; 2. Кнута, Морриса и Пратта; 3. Боуера и Мура.	5	2	1			3	2		2

Раздел 13. Рекурсия в алгоритмах и структурах данных Основные понятия, примеры. Рекурсивные алгоритмы. Рекурсия в структурах данных (списки, деревья, графы). Фракталы и их использование.	6	2				2	4		4
Раздел 14. Кодирование и сжатие данных Основные понятия. Обратимые и необратимые методы. Методы сжатия без потери информации (алгоритм Хаффмана, алгоритм Лемпеля-Зива-Велча). Сжатие данных с потерями.	3	2				2	1		1
Раздел 15. Обзор первой части курса	10	4	2			6	4		4
Вид промежуточного контроля – **зачёт** (балльно-рейтинговая система).									
Всего: 108 = 34 + 34 + 40	**108**	**34**	**34= 26+ 8**	←	**8(входят в 34 ч.)**	**68**	40		**40**
V семестр									
Раздел 1. Стандартная библиотека шаблонов в языке программирования С++ Шаблоны функций и классов. Шаблоны и перегрузка функций. Пространства имен. Последовательные контейнеры и итераторы. Свойства последовательных контейнеров.	13	2	4			6	7		7
Раздел 2. Операции с последовательными контейнерами Различные функции (*push_back, pop_back, insert, erase и др.*). Алгоритмы (*sort, find, copy*).	14	2	4			6	8		8
Раздел 3. Операция объединения *merge*. Категории итераторов Алгоритмы (*merge*). Типы, определенные пользователем. Применение *merge* для структур данных. Категории итераторов и операции, применимые к каждой категории. Алгоритмы *replace и reverse.*	16	2	4		2	8	8		8
Раздел 4. Функциональные объекты. Работа с	14	2	4			6	8		8

последовательностями Понятие предиката. Функциональные объекты и их использование. Работа с последовательностями. Алгоритмы *accumulate, count, find_if* и др.									
Раздел 5. Ассоциативные контейнеры Типы ассоциативных контейнеров. Примеры работы со словарями. Алгоритмы работы с ассоциативными контейнерами. Программа «Формирование частотного словаря». Битовые множества.	34	3	6		2	11	3	30	23
Раздел 6. Адаптеры. Работа со словарём Стек, очередь, очередь с приоритетами. Пары и сравнения. Пример работы со словарём. Основные понятия об отношениях между классами.	38	2	4			6	8	24	32
Раздел 7. NP-полные задачи и труднорешаемые задачи. Методы разработки алгоритмов Основные определения. Классификация задач. Методы разработки алгоритмов (метод декомпозиции, поиск с возвратом, динамическое программирование, метод ветвей и границ и др.). Рассмотрение задач: коммивояжера, Ханойская башня, обход конём шахматной доски, о ранце)	11	2	2			4	7		7
Раздел 8. Обзор курса	13	2			2	4	9		9
Промежуточная аттестация – **экзамен** (балльно-рейтинговая система), – 27 часов.									
Всего: 180 = 17 + 34 + 102 + 27 (контроль)	**180= 153+27**	**17**	**34**		**6 (входят в 34ч.)**	**51**	58	44	**102**
Итого за 2 семестра: 288 часов (включая 27 час. контроля)	**288**	**51**	**68**			**119**			**142**

При реализации дисциплины (модуля) организуется практическая подготовка путем проведения практических занятий, предусматривающих участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью (68 часов).

Практическая подготовка также включает в себя отдельные занятия лекционного типа, которые предусматривают передачу учебной информации обучающимся, необходимой для последующего выполнения работ, связанных с будущей профессиональной деятельностью (51 час).

Практическая подготовка при изучении дисциплины реализуется непосредственно в университете (филиале).

6 Перечень учебно-методического обеспечения по дисциплине (модулю)

Для обеспечения реализации программы дисциплины (модуля) разработаны:

- методические материалы к практическим занятиям;
- методические материалы по организации самостоятельной работы обучающихся;
- методические материалы по организации изучения дисциплины (модуля) с применением электронного обучения, дистанционных образовательных технологий;
- методические рекомендации для обучающихся с ограниченными возможностями здоровья и инвалидов по освоению программы дисциплины (модуля);
- методическое обеспечение инновационных форм учебных занятий.

В учебном процессе используются активные и интерактивные формы: обсуждение отдельных разделов дисциплины, опросы на занятиях, совместное и самостоятельное решение студентами практических задач и заданий, разбор конкретных заданий.

Методические материалы по дисциплине (модулю) и образовательной программе в целом представлены на официальном сайте образовательной организации (раздел «Сведения об образовательной организации» – Образование – Образовательные программы).

7 Фонды оценочных средств по дисциплине (модулю)

Для аттестации обучающихся на соответствие их персональных достижений поэтапным требованиям образовательной программы по дисциплине (модулю) разработаны фонды оценочных средств, позволяющие оценить результаты обучения (знания, умения, навыки) и сформированные (формируемые) компетенции.

Эти фонды включают теоретические вопросы, типовые практические задания, контрольные работы, домашние работы, тесты и иные оценочные материалы, используемые при проведении процедур текущего контроля успеваемости и промежуточной аттестации.

Фонды оценочных средств представлены в приложении к рабочей программе.

При необходимости обучающиеся с ограниченными возможностями здоровья и инвалиды обеспечиваются оценочными материалами в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

- в печатной форме увеличенным шрифтом,
- в форме электронного документа,
- в форме аудиофайла.

Для лиц с нарушениями слуха:

- в печатной форме,
- в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

- в печатной форме,
- в форме электронного документа,
- в форме аудиофайла.

8 Ресурсное обеспечение

– Перечень основной и дополнительной учебной литературы

Основная учебная литература

1. Колдаев, В. Д. Структуры и алгоритмы обработки данных: Учебное пособие / Колдаев В.Д. - М.:ИЦ РИОР, НИЦ ИНФРА-М, 2020. - 296 с.: - (Высшее образование: Бакалавриат). - ISBN 978-5-16-101275-8. - Текст : электронный. // ЭБС "Znanium.com". - URL: https://new.znanium.com/catalog/product/1054007 (дата обращения: 13.04.2020) - Режим доступа: ограниченный по логину и паролю
2. Дроздов, С.Н. Структуры и алгоритмы обработки данных : учебное пособие / С.Н. Дроздов ; – Таганрог : Южный федеральный университет, 2016. – 228 с. : схем., ил. – ISBN 978-5-9275-2242-2. - Текст : электронный. // ЭБС Университетская библиотека онлайн. -

URL: http://biblioclub.ru/index.php?page=book&id=493032 (дата обращения: 13.04.2020). Режим доступа: ограниченный по логину и паролю .
3. Парфенов, Д. В. Язык Си: кратко и ясно: Учебное пособие / Д.В. Парфенов. - Москва : Альфа-М: НИЦ ИНФРА-М, 2020. - 320 с. - ISBN 978-5-16-101327-4. - Текст : электронный. // ЭБС "Znanium.com". - URL: https://new.znanium.com/catalog/product/1046077 (дата обращения: 13.04.2020). - Режим доступа: ограниченный по логину и паролю
4. Кульман, Т.Н. Использование стандартной библиотеки шаблонов STL в программировании структур и алгоритмов обработки данных: Учебное пособие / Т.Н. Кульман. – Дубна: Гос. университет «Дубна», 2021. – 83 с.

Дополнительная учебная литература

1. Григорьев, А. А. Методы и алгоритмы обработки данных : учеб. пособие / А.А. Григорьев. — Москва : ИНФРА-М, 2018. — 256 с. + Доп. материалы [Электронный ресурс; Режим доступа: https://new.znanium.com]. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-104449-0. – Текст : электронный // ЭБС "Znanium.com". - URL: http://znanium.com/catalog/product/922736 (дата обращения: 13.04.2020). - Режим доступа: ограниченный по логину и паролю
2. Ландовский, В.В. Алгоритмы обработки данных : учебное пособие / В.В. Ландовский – Новосибирск : Новосибирский государственный технический университет, 2018. – 67 с. : ил., табл. – ISBN 978-5-7782-3645-5.. - Текст : электронный. // ЭБС Университетская библиотека онлайн. - URL: http://biblioclub.ru/index.php?page=book&id=574809 (дата обращения: 13.04.2020). Режим доступа: ограниченный по логину и паролю.
3. Культин Н. С/С ++ в задачах и примерах. - СПб.: БХВ-Петербург, 2009. - 288 с.: ил.;

– **Периодические издания**

1. Информационные технологии и вычислительные системы: научный журнал / Учредитель Федеральное государственное учреждение "Федеральный исследовательский центр "Информатика и управление" РАН; гл. ред. Попков Ю.С. - М.: ФГУ Федеральный исследовательский центр "Информатика и управление" РАН. – Журнал выходит 2 раза в полуг. – Основан в 1995 г. - ISSN 2071-8632. – Текст : электронный. Полные электронные версии статей журнала доступны по подписке на сайте научной электронной библиотеки «eLIBRARY.RU»: https://www.elibrary.ru/title_about_new.asp?id=8746
2. Информация и безопасность: научный журнал / Учредители: Воронежский государственный технический университет; гл. ред. Остапенко А.Г. – Воронеж: Воронежский государственный технический университет. – Журнал выходит 2 раза в полуг. - Основан в 1998 году. - ISSN 1682-7813. – Текст : электронный. Полные электронные версии статей журнала доступны на сайте научной электронной библиотеки «eLIBRARY.RU»: http://elibrary.ru/contents.asp?titleid=8748
3. Информатика и системы управления: научное издание / Учредитель: Амурский государственный университет; гл. ред. Е.Л. Еремин. – Благовещенск: Амурский государственный университет. – журнал выходит 2 раза в полуг. - Основан в 2001 г. – ISSN: 1814-2400. - Текст : электронный. Полные электронные версии статей журнала доступны на сайте научной электронной библиотеки «eLIBRARY.RU»: https://www.elibrary.ru/contents.asp?titleid=9793
4. Открытые системы СУБД / Учредитель: ООО «Издательство «Открытые системы»; гл. ред. Д. Волков. – М.: Издательство «Открытые системы». – журнал выходит 2 раза в полуг. - Основан в 1993 году. – ISSN: 1028-7493. – Текст : электронный. – Полные электронные версии статей представлены на сайте журнала: https://www.osp.ru/os/archive
5. Программные продукты и системы: международный научно-практический журнал / Учредитель: Куприянов В.П.; гл. ред. Савин Г.И. - Тверь: Центрпрограммсистем. – журнал выходит 2 раза в полуг. - Основан в 1988 году. – ISSN: 0236-235X. - – Текст : элек-

тронный. – Полные электронные версии статей представлены на сайте журнала: http://swsys.ru/

6. Российские нанотехнологии: научный журнал / Учредитель: НИЦ "Курчатовский институт"; гл. ред. Ковальчук М.В. – М.: Общество с ограниченной ответственностью Парк-медиа – Журнал выходит 6 раз в год. – Основан в 2006 году. - ISSN 1993-4068. – Текст : электронный. – Полные электронные версии статей представлены на сайте журнала: https://nanorf.elpub.ru/jour/issue/viewIssue/16/15#
7. Системный администратор / Учредитель: "Издательский дом "Положевец и партнеры"; гл. ред. Г. Положевец. – М.: Общество с ограниченной ответственностью "Издательский дом "Положевец и партнеры". – Журнал выходит 12 раз в год. - Основан в 2002 году. - ISSN 1813-5579. – Текст : электронный. Полные электронные версии статей журнала доступны по подписке на сайте научной электронной библиотеки «eLIBRARY.RU»: https://elibrary.ru/title_about.asp?id=9973

– **Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

Электронно-библиотечные системы и базы данных

1. ЭБС «Znanium.com»: http://znanium.com/
2. ЭБС «Лань»: https://e.lanbook.com/
3. ЭБС «Юрайт»: https://biblio-online.ru/
4. ЭБС «Университетская библиотека онлайн»: http://biblioclub.ru/
5. Научная электронная библиотека (РУНЭБ) «eLIBRARY.RU»: http://elibrary.ru
6. Национальная электронная библиотека (НЭБ): http://нэб.рф/
7. Базы данных российских журналов компании «East View»: https://dlib.eastview.com/

Научные поисковые системы

1. ArXiv.org - научно-поисковая система, специализируется в областях: компьютерных наук, астрофизики, физики, математики, квантовой биологии. http://arxiv.org/
2. Google Scholar - поисковая система по научной литературе. Включает статьи крупных научных издательств, архивы препринтов, публикации на сайтах университетов, научных обществ и других научных организаций. https://scholar.google.ru/
3. WorldWideScience.org - глобальная научная поисковая система, которая осуществляет поиск информации по национальным и международным научным базам данных и порталам. http://worldwidescience.org/
4. SciGuide - навигатор по зарубежным научным электронным ресурсам открытого доступа. http://www.prometeus.nsc.ru/sciguide/page0601.ssi

Профессиональные ресурсы сети «Интернет»

1. Федеральная информационная система «Единое окно доступа к информационным ресурсам»: http://window.edu.ru/.
2. Проект Инициативного Народного Фронта Образования - ИНФО-проект. Школа программирования Coding Craft http://codingcraft.ru/.
3. Портал Life-prog http://life-prog.ru/.
4. OpenNet www.opennet.ru.
5. Алгоритмы, методы, программы algolist.manual.ru.
6. Сервер министерства высшего образования www.informika.ru.

Необходимое программное обеспечение

Дисциплина обеспечена необходимым лицензионным программным обеспечением, которое находится в свободном доступе (Microsoft Visual Studio 2019).

Все лекции по дисциплине разработаны в форме презентаций.

Необходимое материально-техническое обеспечение

Проведение практических занятий по дисциплине предполагает использование специализированных аудиторий, оснащенных персональными компьютерами, объединенными в локальную сеть и имеющих доступ к ресурсам глобальной сети Интернет.

Студентам предоставляются электронные методические материалы, подготовленные преподавателем.

Для выполнения заданий самостоятельной подготовки обучающиеся обеспечиваются литературой, а также в определённом порядке могут получать доступ к информационным ресурсам Интернета.

Компьютерный класс (15 ПК) (оборудование в собственности).

Проектор для представления презентаций (лекции).

Обучающиеся из числа инвалидов и лиц с ограниченными возможностями здоровья могут использовать специализированное программное и материально-техническое обеспечение:

- обучающиеся с нарушениями опорно-двигательного аппарата при необходимости могут использовать адаптивные технические средства: специально оборудованные джойстики, увеличенные выносные кнопки, клавиатуры с большими клавишами.
- обучающиеся с ограничениями по зрению могут прослушать доступный аудиоматериал или прочитать тексты, увеличив шрифт на экране монитора компьютера. Рекомендуется использовать экранную лупу и другие визуальные вспомогательные средства, чтобы изменить шрифт текста, межстрочный интервал, синхронизацию с речью и т.д., программы экранного доступа (скринридеры для прочтения текстовой информации через синтезированную речь) и/или включить функцию «экранного диктора» на персональном компьютере с операционной системой Windows 7, 8, 10.
- обучающиеся с ограничениями по слуху могут воспользоваться компьютерной аудиогарнитурой при прослушивании необходимой информации и портативной индукционной системой серии «ИСТОК».

При необходимости обучающиеся с ограниченными возможностями здоровья и инвалиды обеспечиваются печатными и (или) электронными образовательными ресурсами (образовательная программа, учебные пособия и др.) в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

- в печатной форме увеличенным шрифтом,
- в форме электронного документа,
- в форме аудиофайла.

Для лиц с нарушениями слуха:

- в печатной форме,
- в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

- в печатной форме,
- в форме электронного документа,
- в форме аудиофайла.

Приложение к рабочей программе дисциплины

Фонды оценочных средств

В результате освоения программы бакалавриата по направлению подготовки 09.03.01 «Информатика и вычислительная техника» с учетом направленности программы – «Программное обеспечение вычислительной техники и автоматизированных систем» выпускник должен обладать следующими профессиональными компетенциями:

ОПК-2: *Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности*
ОПК*-8*: *Способен разрабатывать алгоритмы и программы, пригодные для практического применения*

II курс, IV семестр

В IV семестре видом промежуточного контроля является **зачёт**.

Описание показателей и критериев оценивания компетенций, а также шкал оценивания

ИНДИКАТОР ДОСТИЖЕНИЯ КОМПЕТЕНЦИИ (код и наименование)	**КРИТЕРИИ ОЦЕНИВАНИЯ ШКАЛА оценивания**	
	не зачтено	**зачтено**
ОПК-2: Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности		
ОПК-2.1: Ориентируется в спектре операционных систем и программных средств отечественного производства на рынке информационных продуктов		
Знает общие принципы взаимодействия алгоритмов и данных в объектно-ориентированной технологии	Отсутствие знаний Не знает или слабо знает основные теоретические вопросы. Допускает множественные грубые ошибки.	Удовлетворительно знает основные теоретические вопросы. Допускает отдельные негрубые ошибки, либо не допускает ошибок
Знает базовые структуры данных и основные алгоритмы обработки данных, применяемые в программировании	Отсутствие знаний Не знает или слабо знает базовые структуры данных и основные алгоритмы обработки данных, применяемые в программировании. Допускает множественные грубые ошибки.	Удовлетворительно знает базовые структуры данных и основные алгоритмы обработки данных, применяемые в программировании. Допускает отдельные негрубые ошибки, либо не допускает ошибок
ОПК-2.2: Устанавливает оптимальные критерии работы программных средств при решении задач профессиональной деятельности		
Умеет применять к решению прикладных задач базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы	Отсутствие умений или демонстрирует частичное умение применять базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение применять базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы. Допускает отдельные негрубые ошибки, либо не допускает ошибок

ОПК-2.3: Использует современные информационные технологии сбора, обработки, хранения и поиска информации при решении задач профессиональной направленности		
Владеет навыками использования сложных структур данных и программной реализации алгоритмов обработки данных для решения задач профессиональной направленности	Не владеет или демонстрирует низкий уровень владения навыками использования сложных структур данных и программной реализации алгоритмов обработки данных. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками использования сложных структур данных и программной реализации алгоритмов обработки данных. Допускает отдельные негрубые ошибки, либо не допускает ошибок
Владеет навыками использованию библиотеки шаблонов при разработке программ для повышения надёжности, переносимости и сокращения времени на разработку	Не владеет или демонстрирует низкий уровень владения навыками использованию библиотеки шаблонов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками использованию библиотеки шаблонов. Допускает отдельные негрубые ошибки, либо не допускает ошибок
ОПК-8: Способен разрабатывать алгоритмы и программы, пригодные для практического применения		
ОПК-8.1 Выбирает инструментальные средства, языки программирования и технологии обработки данных на начальном этапе разработки программного продукта		
Знает алгоритмические языки программирования, операционные системы и оболочки, современные среды разработки программного обеспечения (C++, Visual Studio)	Отсутствие знаний. Не знает или слабо знает основные теоретические вопросы. Допускает множественные грубые ошибки.	Удовлетворительно знает основные теоретические вопросы. Допускает отдельные негрубые ошибки, либо не допускает ошибок
ОПК-8.2 Разрабатывает алгоритмы и программные коды программных модулей для практического применения		
Умеет составлять алгоритмы, писать и отлаживать коды, тестировать работоспособность программы	Отсутствие умений или демонстрирует частичное умение составлять алгоритмы, писать и отлаживать коды, тестировать работоспособность программы. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение составлять алгоритмы, писать и отлаживать коды, тестировать работоспособность программы. Допускает отдельные негрубые ошибки, либо не допускает ошибок
Умеет программировать такие структуры данных, как списки, стеки, очереди, бинарные деревья, множества и др.	Отсутствие умений или демонстрирует частичное умение программировать сложные структуры данных. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение программировать сложные структуры данных. Допускает отдельные негрубые ошибки, либо не допускает ошибок
ОПК-8.3 Тестирует работоспособность программ и программных компонентов		
Владеет навыками отладки программ и программных компонентов	Не владеет или демонстрирует низкий уровень владения навыками отладки программ и программных компонентов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками отладки программ и программных компонентов. Допускает отдельные негрубые ошибки, либо не допускает ошибок
ОПК-8.4: Применяет языки программирования и современные программные среды разработки информационных систем для решения прикладных задач различных классов		
Умеет писать программы на языках высокого уровня, использовать интегриро-	Отсутствие умений или демонстрирует частичное умение писать программы на языках высокого уровня, исполь-	Демонстрирует удовлетворительное умение писать программы на языках высокого уровня, использовать интегри-

ванную среду разработки приложений для решения прикладных задач	зовать интегрированную среду разработки приложений. Допускает множественные грубые ошибки.	рованную среду разработки приложений. Допускает отдельные негрубые ошибки, либо не допускает ошибок
Владеет навыками программной реализации алгоритмов обработки данных на языке высокого уровня на основе интегрированной среды разработки приложений	Не владеет или демонстрирует низкий уровень владения навыками программной реализации алгоритмов обработки данных на языке высокого уровня на основе интегрированной среды разработки приложений. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками программной реализации алгоритмов обработки данных на языке высокого уровня на основе интегрированной среды разработки приложений. Допускает отдельные негрубые ошибки, либо не допускает ошибок

Балльно-рейтинговая система
II курс, IV семестр

При балльно-рейтинговой системе все знания, умения и навыки, приобретаемые студентами в результате изучения дисциплины, оцениваются в баллах.

Оценка качества работы в рейтинговой системе является накопительной и используется для оценивания системной работы студентов в течение всего периода обучения.

По итогам работы в семестре студент может получить максимально 100 баллов. Итоговой формой контроля в IV семестре является зачет.

В течение IV семестра студент может заработать баллы за следующие виды работ:

№	Вид работы	Сумма баллов
1	Работа на практических занятиях	34
2	Контрольные работы (ПР-2.1, ПР-2.2, ПР-2.3, ПР-2.4)	49
3	Аудиторные занятия (посещение)	17
	Итого:	100

Если к моменту окончания семестра студент набирает 70 баллов, то он получает оценку «зачтено» автоматически. Если студент не набрал минимального числа баллов (70 баллов), то он в обязательном порядке должен сдавать зачет.

Текущий контроль успеваемости осуществляется в процессе выполнения практических и самостоятельных работ в соответствии с нижеприведенным графиком.

График выполнения самостоятельных работ студентами в IV семестре

Виды работ	Недели учебного процесса																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПР-2.1	ВЗ		ЗЗ														
ПР-2.2				ВЗ			ЗЗ										
ПР-2.3								ВЗ			ЗЗ						
ПР-2.4												ВЗ			ЗЗ		

ПР-2 – контрольная работа (домашняя)
ВЗ – выдача задания
ЗЗ – защита задания

Вопросы к зачёту по дисциплине «Структуры и алгоритмы обработки данных»

Структуры данных

1. Понятие абстрактного типа данных (АТД).
2. Линейные структуры данных. Области применения.

3. Абстрактный список. Определение, свойства, операции.
4. Реализация списков в виде массива и на основе указателей.
5. Работа со связанными списками: вывод на экран, удаление и вставка узла.
6. Абстрактный стек. Определение, свойства, операции.
7. Реализация стека с помощью массива, односвязного списка.
8. Абстрактная очередь. Определение, свойства, операции.
9. Нелинейные структуры данных. Области применения.
10. Деревья. Рекурсивное определение. Основные понятия.
11. Области применения деревьев.
12. Обход дерева: прямой, обратный, симметричный.
13. Операторы работы с деревьями.
14. Абстрактное бинарное дерево. Определение, свойства, операции.
15. Полное, совершенное и сбалансированное бинарные деревья.
16. АВЛ — деревья.
17. Реализация бинарного дерева в виде массива и с помощью указателей.
18. Двоичное дерево поиска.
19. Ориентированный граф. Основные понятия и терминология. Операторы работы с орографом.
20. Реализация ориентированного графа с помощью матрицы смежности и списков смежности.
21. Неориентированный граф. Основные понятия и терминология. Неориентированный граф и операторы работы с ним.
22. Реализация неориентированного графа с помощью матрицы смежности и списков смежности.
23. Графы как абстрактные типы данных.
24. Области применения графов.
25. Кратчайшие пути в графе. Различные алгоритмы.
26. Задача о «Кенигсбергских мостах».
27. B-деревья. Основные понятия, определения и примеры использования.
28. Таблицы с прямым доступом и хэш – таблицы.
29. Понятие хеш-функции, требования к хеш-функции, примеры таких функций. Понятие коллизии.
30. Разрешение коллизий с помощью метода цепочек.
31. Методы расстановки: деление, умножение, универсальное хеширование.
32. Основные понятия сжатия данных Обратимые и необратимые методы.

Алгоритмы

33. Простой алгоритм внутренней сортировки (вставками).
34. Простой алгоритм внутренней сортировки (выбором).
35. Простой алгоритм внутренней сортировки (пузырьковая).
36. Усовершенствованные алгоритмы внутренней сортировки (быстрая сортировка).
37. Алгоритм внешней сортировки (сортировка слиянием).
38. Алгоритм внешней сортировки (сортировка Шелла).
39. Последовательный поиск.
40. Бинарный поиск.
41. Поиск по вторичным ключам. Организация инвертированных индексов.
42. Использование деревьев в задачах поиска.
43. Поиск подстрок в тексте (Прямой поиск).
44. Поиск подстрок в тексте (Алгоритм Боуера, Мура, Хорспула).
45. Поиск подстрок в тексте (Алгоритм Кнута, Мориса, Пратта).
46. Алгоритмы на графах (поиск в глубину).
47. Алгоритмы на графах (поиск в ширину).
48. Алгоритмы на графах (топологическая сортировка).

49. Кратчайшие пути в графе. Взвешенные графы.
50. Алгоритм Дейкстры.
51. Алгоритм Беллмана-Форда.
52. Алгоритм Флойда-Уоршелла.
53. Метод сжатия без потерь (алгоритм Хаффмана).
54. Метод сжатия без потерь (алгоритм Лемпеля-Зива-Велча).
55. Рекурсия. Выполнение рекурсивного вызова, использование стека.
56. Рекурсия с возвратом. Решение задачи об обходе конём шахматной доски.
57. Понятие сложности и эффективность алгоритмов обработки структур данных.
58. Эффективность алгоритмов сортировки и поиска.

Методические материалы к практическим и самостоятельным занятиям

- Указатели и их применение для решения задач с массивами и со структурами данных (предоставляются готовые проекты).
- Линейные структуры (стеки) с готовыми проектами.
- Различные виды сортировок с решением задач.
- Связные списки с решением задач.
- Программирование очереди и бинарного дерева с решением задач.
- Перегрузка функций с готовым проектом.
- Проекты по созданию приложений в диалоговом режиме.
- Разработанное приложение «Учёт заявок на авиабилеты».
- Разработанное приложение «Обход конём шахматной доски» (перебор с возвратом).
- Вопросы к коллоквиумам по темам:
  - Линейные структуры данных
  - Деревья
  - Графы
  - B-деревья, хеш-функции
  - Алгоритмы сортировки. Эффективность алгоритмов
  - Алгоритмы поиска, включая поиск в тексте
  - Алгоритмы на графах. Рекурсия
  - Кодирование и сжатие данных.

Пример варианта контрольной работы по алгоритмам сортировки

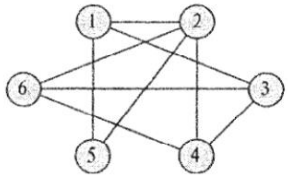
Вариант 3

1. Выполните трассировку алгоритма сортировки слиянием, если нужно упорядочить в порядке возрастания следующий массив:
   5 71 24 35 1 48.
2. Выполните трассировку алгоритма быстрой сортировки, если нужно упорядочить в порядке убывания следующий массив:
   5 71 24 18 35 1 48.
3. Выполните трассировку алгоритма сортировки вставками, если нужно упорядочить в порядке убывания следующий массив: 23 30 20 80 40 15 60.
4. Выполните трассировку алгоритма сортировки Шелла, если нужно упорядочить в порядке возрастания следующий массив:
   5 71 24 18 35 1 48 33.
5. Выполните трассировку алгоритма сортировки методом выбора, если нужно упорядочить в порядке возрастания следующий массив:
   31 25 6 13 78 17 72.
6. Напишите, какой порядок имеет сложность алгоритмов в п.1 – п.5, используйте O(f(n)).

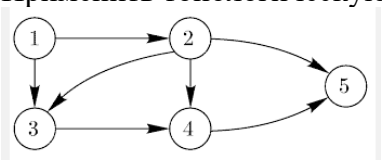
Пример варианта контрольной работы по темам «Деревья и графы»

Вариант 2

1. Деревья. Рекурсивное определение.
2. Прямой обход дерева. Пример.
3. Определение АВЛ — деревьев.
4. Представить алгебраическое выражение в виде бинарного дерева и обойти его, используя обратный обход:
   (A + B) * (C + D) - E
5. Определение ориентированного графа. Пример ОГ.
6. Представить неориентированный граф в виде списка смежности:



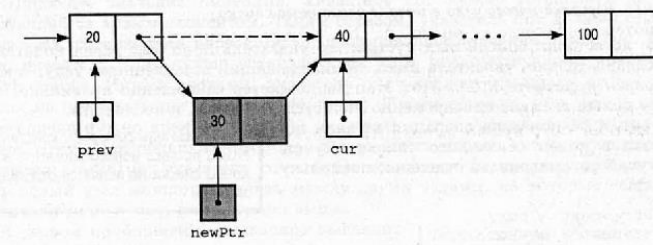
7. Для какой цели в алгоритме поиска в ширину используется очередь?
8. Дать определение пути в графе.
9. Применить топологическую сортировку к графу:



Пример варианта итоговой контрольной работы за IV семестр

Вариант 2

1. Определение линейного списка.
2. Неориентированный граф. Основные понятия и терминология.
3. Сбалансированное дерево, определение, примеры.
4. Представить выражение в виде дерева: (a/b*k-x)*(y+z*c).
5. Обойти полученное в п.4 дерево, методом «слева направо».
6. Отсортировать последовательность чисел 9, 25, 31, 64, 8, 1, 105, 17
в порядке возрастания методом «пузырька».
Показать трассировку процесса сортировки.
7. Очередь и операции над ней.
8. Эффективность алгоритмов сортировки и поиска. Привести 5 оценок различных алгоритмов.
9. Определение дерева поиска.
10. Приведите 3 алгоритма, используемых при построении хеш-функций.
11. В чем преимущества и недостатки списков по сравнению с массивами?
12. Краткое описание алгоритма на графах: поиск в глубину.
13. Опишите с помощью средств C++ узел списка.
14. На рисунке показана схема добавления нового узла. Напишите на C++ операторы вставки этого узла.



Пример задания на реализацию списка с помощью указателей

В программе реализованы следующие функции работы со списком:

- Функция

`void  assign_new_node( node_ptr& ptr )`

Создаёт новый узел списка.

- Функция

`void  assign_list( node_ptr& a_list )`

Присоединяет очередной узел к списку и заполняет его.

- Функция

`void  print_list( node_ptr a_list )`

Выводит элементы списка на экран.

- Функция

`void add_after(node_ptr& list, char a_word[], char word_after[])`

Вставляет в связный список "`list`" после первого встретившегося узла со словом "`a_word`" новый узел со словом "`word_after`". Если в списке "`list`" нет узла со словом "`a_word`", то функция не должна модифицировать список.

- Функция

`void delete_node(node_ptr& a_list, char a_word[])`

Удаляет в связном списке "`a_list`" первый встретившийся узел со словом "`a_word`".

Кроме этого, реализован простой интерфейс работы со списком:

```
Выбор: 1 - Del, 2 - Ins, 3 - List, 0 - Exit:
```

```
#include <iostream.h>
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stddef.h>

const int MAX_WORD_LENGTH = 25;

struct node;
typedef node* node_ptr;

struct node
{
      char word[MAX_WORD_LENGTH];
      node_ptr ptr_to_next_node;
};

char* rus_str( char* str );
void  assign_list( node_ptr& a_list );
void  assign_new_node( node_ptr& ptr );
void  print_list( node_ptr a_list );
void  add_after( node_ptr a_list, char a_word[], char word_after[] );
void  delete_node( node_ptr& a_list, char a_word[] );
```

```
void mainprog()
{
      int choice;
      char a_word[MAX_WORD_LENGTH], word_after[MAX_WORD_LENGTH];
      node_ptr a_list;

      assign_list( a_list );

      cout << rus_str("Выбор: 1 - Del, 2 - Ins, 3 - List, 0 - Exit: ");
      cin >> choice;

while ( choice != 0 )
      {
      switch ( choice )
      {
      case 3 : print_list( a_list );
                  break;
      case 2 : cout << rus_str("После какого слова вставить: ");
                  cin >> a_word;
                  cout << rus_str("Какое слово вставить: ");
                  cin >> word_after;
                  add_after( a_list, a_word, word_after );
                  break;
      case 1 : cout << rus_str("Какое слово хотите удалить: ");
                  cin >> a_word;
                  delete_node( a_list, a_word );
                  break;
      }

      cout << rus_str("\n\nВыбор: 1 - Del, 2 - Ins, 3 - List, 0 - Exit: ");
            cin >> choice;
      }
      cout << "\n\n";
}
```

И т.д.

Задание на тему «Алгоритмы внутренних сортировок и их эффективность»

Сортировка – это процесс упорядочивания набора элементов в возрастающем или убывающем порядке.

Алгоритмы сортировки делятся на:

- Внутренние, когда все данные находятся в оперативной памяти и
- Внешние, - данные могут храниться на вспомогательных устройствах, например, на жёстком диске.

Результат сортировки набора чисел или символов понятен, но можно упорядочивать и объекты, в этом случае выбирается ключ сортировки (ФИО, возраст, адрес).

Анализ алгоритмов – это способы сравнения эффективности разных *методов решения задач*. Сравнение алгоритмов должно быть сосредоточено на их существенных различиях (время выполнения и занимаемая память). Основной способ оценки эффективности алгоритма - подсчёт его операций. Будем рассматривать фактор времени.

Время выполнения алгоритма выражается функцией, зависящей от размера задачи.

Время выполнения алгоритма А прямо пропорционально функции f(n), говорят, что *алгоритм А имеет порядок f(n)* и этот факт обозначается как O(f(n)).

Функция f(n) называется сложностью алгоритма.

Методы сортировки *на том же месте* можно разбить на три категории:

- С помощью прямого выбора,
- С помощью прямого включения,
- С помощью прямого обмена

```
//  П Р О Г Р А М М Ы   С О Р Т И Р О В О К
//  Подготовительная часть
     const int n(10);
     int A[n], B[n+1], C[n], i, j, x, c;
     // Использование датчика случайных чисел
    srand( (unsigned)time( NULL ) );
     for (i=0; i<n; i++)
     {
          A[i] = rand() % 100;
          cout << A[i] << " " ;
          C[i] = B[i+1] = A[i];
     }
     cout << "\n\n";

//*****************************************************
//   С О Р Т И Р О В К А    В Ы Б О Р О М
Этот приём основан на следующих принципах:
```

- Выбирается элемент с наименьшим ключом,
- Он меняется местами с первым элементом,
- Затем этот процесс повторяется с оставшимися n-1 элементами, n-2 элементами и т.д. до последнего.

```
//*****************************************************
cout << "S E L E C T I O N     S O R T \n\n";
     for (i=0; i<n-1; i++)
     {
          int min = i;
          for (j=i+1; j<n; j++)
               if (A[j] < A[min]) min = j;
                  {  // обмен значениями
                          c=A[i];
                          A[i]=A[min];
                          A[min]=c;
     // печать строки после обмена
                          for (int i1=0; i1<n; i1++)
                               {
                                    cout << A[i1] << " " ;
                               }
                          cout << "\n";
                  }
     }
     cout << "\n\n";
```

```
56 19 2 10 77 35 33 31
S E L E C T I O N     S O R T
2 19 56 10 77 35 33 31
2 10 56 19 77 35 33 31
2 10 19 56 77 35 33 31
2 10 19 31 77 35 33 56
2 10 19 31 33 35 77 56
2 10 19 31 33 35 77 56
2 10 19 31 33 35 56 77
```

АНАЛИЗ: Внешний цикл выполняется n-1 раз, внутренний цикл работает от 1 до n-1, => общее количество итераций внешнего цикла = (n-1)+(n-2)+…+1 = n*(n-1)/2.
Поскольку при каждой итерации выполняется одно сравнение,
Их общее количество = n*(n-1)/2. При обмене производится 3 присваивания, => всего присваиваний будет 3*(n-1).

В сумме выполняется $n*(n-1)/2+3*(n-1)=n^2/2+5*n/2-3$ основных операций. => Сложность алгоритма сортировки методом выбора равна $O(n^2)$.

```
//*************************************************
//   С О Р Т И Р О В К А    В С Т А В К А М И
```

Элементы «делятся» на уже готовую последовательность и исходную последовательность. При каждом шаге, начиная с i=2 и увеличивая i каждый раз на 1, из исходной последовательности извлекается i элемент и перекладывается на нужное место в готовой последовательности.

```
//*************************************************
      cout << "I N S E R T I O N     S O R T \n\n";
      for (i = 1; i <= n; i++)
      {
            x = B[i]; B[0] = x; j = i;
            while (x < B[j-1])
            {
                  // печать массива после обмена
                  for (int i1=0; i1<n; i1++) cout << B[i1+1] << " " ;
                  cout << "\n";
                  B[j] = B[j-1]; j--;
            }
            B[j] = x;
      }
      for (i=0; i<n; i++)   cout << B[i+1] << " " ;
      cout << "\n\n";
```

```
I N S E R T I O N    S O R T
56 19 2 10 77 35 33 31
19 56 2 10 77 35 33 31
2 19 56 10 77 35 33 31
2 10 19 56 77 35 33 31
2 10 19 35 56 77 33 31
2 10 19 33 35 56 77 31
2 10 19 31 33 35 56 77
```

АНАЛИЗ: Внешний цикл выполняется n-1 раз, внутренний цикл работает от 1 до n-1, => общее количество сравнений равно $(n-1)+(n-2)+\ldots+1 = n*(n-1)/2$. Столько же раз в худшем случае внутренний цикл сдвигает элементы. Перемещение элементов на каждой итерации производится дважды, => $2*(n-1)$. Общее количество основных операций равно $n*(n-1)+2*(n-1)=n^2+n-2$. => Сложность алгоритма сортировки методом вставок равна $O(n^2)$.

```
//*************************************************
//   П У З Ы Р Ь К О В А Я    С О Р Т И Р О В К А
```

Этот алгоритм основывается на сравнении и смене мест для пары соседних элементов до тех пор, пока не будут упорядочены все элементы. Если массив рассматривать “вертикально”, то элементы можно интерпретировать как пузырьки в чане с водой. “Вес” каждого пузырька соответствует его ключу.

```
//*************************************************
      cout << "B U B B L E     S O R T \n\n";
for (i=0; i<n; i++)
      {
            cout << C[i] << " " ;
      }
      cout << "\n\n";
```

```
      for (i=1; i<n; i++)
          { for (j = n-1; j>=i; j--)
                  if (C[j-1] > C[j])
                  {          // обмен значениями
                             c=C[j-1];
                             C[j-1]=C[j];
                             C[j]=c;
                             // печать строки после обмена
                  }
           for (int i1=0; i1<n; i1++) cout << C[i1] << " " ;

                             cout << "\n";
            }
      for (i=0; i<n; i++) cout << C[i] << " " ;

      cout << "\n";
}
```

```
B U B B L E    S O R T
56 19 2 10 77 35 33 31
2 56 19 10 31 77 35 33
2 10 56 19 31 33 77 35
2 10 19 56 31 33 35 77
2 10 19 31 56 33 35 77
2 10 19 31 33 56 35 77
2 10 19 31 33 35 56 77
2 10 19 31 33 35 56 77
2 10 19 31 33 35 56 77
```

АНАЛИЗ: Внешний цикл выполняется n-1 раз, при первом проходе выполняется n-1 сравнений и не больше n-1 перестановок, при втором проходе - n-2 сравнений и не больше n-2 перестановок и т.д, => общее количество сравнений и перестановок = (n-1)+(n-2)+…+1 = n*(n-1)/2.
При каждой перестановке производится 3 присваивания, => общее количество основных операций 2*n*(n-1)=2* n^2-2*n.
=> Сложность алгоритма сортировки методом пузырька равна O(n^2).

III курс, V семестр

В V семестре видом промежуточного контроля является **экзамен**.

Описание показателей и критериев оценивания компетенций, а также шкал оценивания

ИНДИКАТОР ДОСТИЖЕНИЯ КОМПЕТЕНЦИИ (код и наименование)	**КРИТЕРИИ ОЦЕНИВАНИЯ РЕЗУЛЬТАТА ОБУЧЕНИЯ по дисциплине (модулю) ШКАЛА оценивания**				
	1	**2**	**3**	**4**	**5**
ОПК-2: Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности					
ОПК-2.1: Ориентируется в спектре операционных систем и программных средств отечественного производства на рынке информационных продуктов					

Знает общие принципы взаимодействия алгоритмов и данных в объектно-ориентированной технологии	Отсутствие знаний	Не знает или слабо знает основные теоретические вопросы Допускает множественные грубые ошибки.	Удовлетворительно знает основные теоретические вопросы Допускает достаточно серьезные ошибки.	Хорошо знает основные основные теоретические вопросы. Допускает отдельные негрубые ошибки	Хорошо знает основные теоретические вопросы. Не допускает ошибок
Знает базовые структуры данных и основные алгоритмы обработки данных, применяемые в программировании	Отсутствие знаний	Удовлетворительно знает базовые структуры данных и основные алгоритмы обработки данных Допускает множественные грубые ошибки.	Хорошо знает базовые структуры данных и основные алгоритмы обработки данных. Допускает достаточно серьезные ошибки	Хорошо знает базовые структуры данных и основные алгоритмы обработки данных. Допускает отдельные негрубые ошибки.	Хорошо знает базовые структуры данных и основные алгоритмы обработки данных. Не допускает ошибок
ОПК-2.2 Устанавливает оптимальные критерии работы программных средств при решении задач профессиональной деятельности					
Умеет применять к решению прикладных задач базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы	Отсутствие умений	Демонстрирует частичное умение применять базовые алгоритмы, выполнять оценку сложности алгоритмов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение применять базовые алгоритмы, выполнять оценку сложности алгоритмов. Допускает достаточно серьезные ошибки.	Демонстрирует достаточно устойчивое умение применять базовые алгоритмы, выполнять оценку сложности алгоритмов.. Допускает отдельные негрубые ошибки.	Демонстрирует устойчивое умение применять базовые алгоритмы, выполнять оценку сложности алгоритмов.. Не допускает ошибок.
Умеет работать с библиотекой STL, знать её основные классы и алгоритмы	Отсутствие умений	Демонстрирует частичное умение работать с библиотекой STL. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение работать с библиотекой STL. Допускает достаточно серьезные ошибки.	Демонстрирует достаточно устойчивое умение работать с библиотекой STL. Допускает отдельные негрубые ошибки.	Демонстрирует устойчивое умение работать с библиотекой STL. Не допускает ошибок.
ОПК-2.3 Использует современные информационные технологии сбора, обработки, хранения и поиска информации при решении задач профессиональной направленности					
Владеет навыками использования сложных структур данных и программной реализации алгоритмов обработки данных	Отсутствие владения	Не владеет или демонстрирует низкий уровень владения навыками	Демонстрирует удовлетворительный уровень владения навыками использо-	Демонстрирует хороший уровень владения навыками использования слож-	Демонстрирует высокий уровень владения навыками использования слож-

для решения задач профессиональной направленности		использования сложных структур данных. Допускает множественные грубые ошибки.	вания сложных структур данных. Допускает достаточно серьезные ошибки.	ных структур данных. Допускает отдельные негрубые ошибки.	ных структур данных. Не допускает ошибок.
Владеет навыками использованию библиотеки шаблонов при разработке программ для повышения надёжности, переносимости и сокращения времени на разработку	Отсутствие владения	Не владеет или демонстрирует низкий уровень владения навыками использованию библиотеки шаблонов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками использованию библиотеки шаблонов. Допускает достаточно серьезные ошибки.	Демонстрирует хороший уровень владения навыками использованию библиотеки шаблонов. Допускает отдельные негрубые ошибки.	Демонстрирует высокий уровень владения навыками использованию библиотеки шаблонов. Не допускает ошибок.
ОПК-8: Способен разрабатывать алгоритмы и программы, пригодные для практического применения					
ОПК-8.1: Выбирает инструментальные средства, языки программирования и технологии обработки данных на начальном этапе разработки программного продукта					
	1	**2**	**3**	**4**	**5**
Знает алгоритмические языки программирования, операционные системы и оболочки, современные среды разработки программного обеспечения (C++, Visual Studio)	Отсутствие знаний	Не знает или слабо знает языки программирования, операционные системы и оболочки. Допускает множественные грубые ошибки.	Удовлетворительно знает языки программирования, операционные системы и оболочки. Допускает достаточно серьезные ошибки.	Хорошо знает языки программирования, операционные системы и оболочки. Допускает отдельные негрубые ошибки.	Демонстрирует свободное и уверенное знание языков программирования, операционных систем и оболочек. Не допускает ошибок.
ОПК-8.2 Разрабатывает алгоритмы и программные коды программных модулей для практического применения					
Умеет составлять алгоритмы, писать и отлаживать коды, тестировать работоспособность программы	Отсутствие умений	Демонстрирует частичное умение составлять алгоритмы, писать и отлаживать коды, тестировать. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение составлять алгоритмы, писать и отлаживать коды, тестировать. Допускает достаточно серьезные ошибки.	Демонстрирует достаточно устойчивое умение составлять алгоритмы, писать и отлаживать коды, тестировать. Допускает отдельные негрубые ошибки.	Демонстрирует устойчивое умение составлять алгоритмы, писать и отлаживать коды, тестировать. Не допускает ошибок.
Умеет программировать такие структуры данных, как списки, стеки, очереди, би-	Отсутствие умений	Демонстрирует частичное умение программи-	Демонстрирует удовлетворительное умение про-	Демонстрирует достаточно устойчивое умение	Демонстрирует устойчивое умение программировать списки,

нарные деревья, множества и др.		ровать списки, стеки, очереди, бинарные деревья, множества и др. Допускает множественные грубые ошибки.	граммировать списки, стеки, очереди, бинарные деревья, множества и др. Допускает достаточно серьезные ошибки.	программировать списки, стеки, очереди, бинарные деревья, множества и др. Допускает отдельные негрубые ошибки.	стеки, очереди, бинарные деревья, множества и др. Не допускает ошибок.
ОПК-8.3: Тестирует работоспособность программ и программных компонентов					
Владеет навыками отладки программ и программных компонентов	Отсутствие владения	Не владеет или демонстрирует низкий уровень владения навыками отладки программ и программных компонентов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками отладки программ и программных компонентов. Допускает достаточно серьезные ошибки.	Демонстрирует хороший уровень владения навыками отладки программ и программных компонентов. Допускает отдельные негрубые ошибки.	Демонстрирует высокий уровень владения навыками отладки программ и программных компонентов. Не допускает ошибок.
ОПК-8.4: Применяет языки программирования и современные программные среды разработки информационных систем для решения прикладных задач различных классов					
Умеет писать программы на языках высокого уровня, использовать интегрированную среду разработки приложений для решения прикладных задач	Отсутствие умений	Демонстрирует частичное умение писать программы, использовать интегрированную среду разработки. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительное умение писать программы, использовать интегрированную среду разработки. Допускает достаточно серьезные ошибки.	Демонстрирует достаточно устойчивое умение писать программы, использовать интегрированную среду разработки. Допускает отдельные негрубые ошибки.	Демонстрирует устойчивое умение писать программы, использовать интегрированную среду разработки. Не допускает ошибок.
Владеет навыками программной реализации алгоритмов обработки данных на языке высокого уровня на основе интегрированной среды разработки приложений	Отсутствие владения	Не владеет или демонстрирует низкий уровень владения навыками программной реализации алгоритмов. Допускает множественные грубые ошибки.	Демонстрирует удовлетворительный уровень владения навыками программной реализации алгоритмов. Допускает достаточно серьезные ошибки.	Демонстрирует хороший уровень владения навыками программной реализации алгоритмов. Допускает отдельные негрубые ошибки.	Демонстрирует высокий уровень владения навыками программной реализации алгоритмов. Не допускает ошибок.

Балльно-рейтинговая система
III курс, V семестр

По итогам работы в семестре студент может получить максимально **70** баллов. Итоговой формой контроля **в V семестре является экзамен**. На экзамене студент может набрать максимально **30** баллов.

В течение V семестра студент может заработать баллы за следующие виды работ:

№	Вид работы	Сумма баллов
1	Работа на практических занятиях	22
2	Контрольные работы (ПР-2.1)	10
3	Контрольные работы (ПР-2.2)	10
4	Творческое домашнее задание (ПР-2.3)	20
5	Аудиторные занятия (посещение)	8
	Итого:	70

Если к моменту окончания семестра студент набирает от **51** до **70** баллов, то он получает допуск к экзамену.

Если студент к моменту окончания семестра набирает от **61** до **70** баллов, то он может получить автоматическую оценку «удовлетворительно». При желании повысить свою оценку, студент имеет право отказаться от автоматической оценки и сдать экзамен.

Если студент не набрал минимального числа баллов (**51** балл), то он не получает допуск к экзамену.

Соответствие рейтинговых баллов и академических оценок

Общая сумма баллов за семестр	Итоговая оценка
86-100	Отлично
71-85	Хорошо
51-70	Допуск к экзамену
в том числе: 61-70	Возможность получения автоматической оценки «удовлетворительно»
51-60	Только допуск к экзамену
0-50 *	Неудовлетворительно (студент не допущен к экзамену)

Текущий контроль успеваемости осуществляется в процессе выполнения практических и самостоятельных работ в соответствии с нижеприведенным графиком.

График выполнения самостоятельных работ студентами в V семестре:

Виды работ	Недели учебного процесса																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ПР-2.1		ВЗ				ЗЗ											
ПР-2.2							ВЗ				ЗЗ						
ПР-2.3										ВЗ					ЗЗ		

ПР-2 – контрольная работа (домашняя)
ВЗ – выдача задания
ЗЗ – защита задания

Методическое обеспечение инновационных форм учебных занятий

Все материалы для практических занятий предоставляются в электронном виде. Готовые творческие работы пересылаются преподавателю через Интернет. Готовые проекты (в Visual Studio) выкладываются на сервер для общего обсуждения и дальнейших самостоятельных доработок.

Инновационные формы проведения учебных занятий

Семестр	Вид учебных занятий	Используемые инновационные формы проведения учебных занятий	Количество академ. часов
IV,V семестры	Практические занятия	1. Разбор конкретных задач и заданий для выработки навыков при выполнении домашних заданий по всем темам, выполнение самостоятельных работ в соответствии с возможностями студентов (индивидуальные задания). 2. Выполнение контрольных и творческих заданий стимулирует поиск и нахождение самостоятельных решений, нацелены на выработку профессиональных умений и навыков.	25
		Всего:	25

Вопросы к экзамену по дисциплине «Структуры и алгоритмы обработки данных»

Структуры данных

1. Понятие абстрактного типа данных (АТД).
2. Линейные структуры данных. Области применения.
3. Реализация списков в виде массива и на основе указателей.
4. Нелинейные структуры данных. Области применения.
5. Деревья. Рекурсивное определение. Основные понятия.
6. Области применения деревьев.
7. Обход дерева: прямой, обратный, симметричный.
8. Полное, совершенное и сбалансированное бинарные деревья.
9. Реализация бинарного дерева в виде массива и с помощью указателей.
10. Двоичное дерево поиска.
11. Ориентированный граф. Основные понятия и терминология. Операторы работы с орографом.
12. Неориентированный граф. Основные понятия и терминология. Неориентированный граф и операторы работы с ним.
13. Области применения графов.
14. B-деревья. Основные понятия, определения и примеры использования.
15. Таблицы с прямым доступом и хэш – таблицы.
16. Понятие хеш-функции, требования к хеш-функции, примеры таких функций. Понятие коллизии.

Алгоритмы

17. Алгоритмы внутренних сортировок (вставками, выбором, пузырьковая).
18. Усовершенствованные алгоритмы внутренней сортировки (быстрая сортировка).
19. Алгоритм внешней сортировки (сортировка слиянием).
20. Бинарный поиск.
21. Поиск подстрок в тексте (известные вам алгоритмы).
22. Алгоритмы на графах (поиск в глубину, поиск в ширину).
23. Кратчайшие пути в графе (известные вам алгоритмы).
24. Понятие сложности и эффективность алгоритмов обработки структур данных. Примеры эффективность алгоритмов сортировки и поиска.

Стандартная библиотека шаблонов STL

25. Общие положения STL.
26. Понятие шаблона (шаблонные функции, шаблонные классы).
27. Пространства имен.
28. Основные составляющие STL (контейнеры, алгоритмы, итераторы).

29. Последовательные контейнеры (векторы, списки, двусторонние очереди).
30. Методы работы с последовательными контейнерами.
31. Алгоритм сортировки в STL.
32. Алгоритм поиска в STL.
33. Алгоритм копирования и итератор вставки.
34. Алгоритм слияния в STL.
35. Категории итераторов.
36. Алгоритмы replace и reverse в STL.
37. Алгоритм accumulate в STL.
38. Алгоритм count в STL.
39. Функциональные объекты, определенные в STL.
40. Ассоциативные контейнеры в STL.
41. Методы и свойства ассоциативных контейнеров.
42. Множества и словари.
43. Множества и словари с дубликатами.
44. Адаптеры контейнеров.
45. Понятие задач класса P.
46. NP-полная задача. Примеры.
47. Методы разработки алгоритмов.

Вопросы к опросам и коллоквиумам по библиотеке STL

1. Что такое перегрузка функций?
2. Чем понятие «перегрузка функций» отличается от понятия «шаблон» в STL?
3. Какие виды шаблонов вы знаете?
4. Что означают параметры шаблонов и как они записываются?
5. Назовите основные возможности шаблонных функций.
6. Назовите основные возможности шаблонных классов.
7. Что такое инстанцирование шаблона?
8. Какие преимущества даёт использование перегрузки по сравнению с применением шаблонов и наоборот?
9. Что такое «пространство имён» и для каких целей оно используется?
10. Объясните разницу в понятиях определён (declare) и описан (definition) при создании методов класса.
11. Дайте определение последовательному контейнеру.
12. Какие виды последовательных контейнеров вы знаете?
13. Какие заголовочные файлы нужно включить в проект для работы с последовательными контейнерами?
14. Что такое итератор?
15. Какие вы знаете методы, применяемые в контейнерах?
16. Объясните особенности методов begin() и end().
17. Для каких целей в программах используются следующие конструкции: #include <list>, list <int> L; list <int>::iterator iL;?
18. В какие позиции производится вставка и удаление в контейнерах vector, deque и list по умолчанию?
19. Чем отличаются методы шаблонных классов от алгоритмов?
20. Алгоритм sort. Его использование и особенности.
21. Алгоритм find. Его использование и особенности.
22. Как с помощью алгоритма find найти все вхождения искомого элемента?
23. Алгоритм copy. Режимы замещения и вставки.
24. В каких случаях используются итераторы вставки?
25. Каким образом предикат в вызове find_if влияет на результат поиска?
26. Объясните работу алгоритма merge.

27. Для каких контейнеров может применяться алгоритм merge?
28. Для каких целей в заголовочные файлы включается строка #include <iterator>?
29. Каким образом используется алгоритм merge в типах, определённых пользователем?
30. Что означает operator<?
31. Как можно записать operator< для упорядочивания строк / чисел / двух полей (например, год и месяц)?
32. Какие значения возвращает функция strcmp(string1, string2)?
33. Какие вы знаете категории итераторов?
34. Приведите примеры алгоритмов с соответствующими категориями итераторов.
35. Объясните работу потоковых итераторов.
36. Что такое функциональный объект?
37. Для каких целей служит функциональный объект?
38. Какой заголовочный файл нужно подключить, чтобы использовать функциональные объекты, определённые в STL?
39. Что такое предикат?
40. Каким образом в программе задаётся передача параметра по ссылке?
41. Объясните на примере программы калькулятора как можно работать с параметром, который передаётся по ссылке.
42. Какие вы знаете функциональные объекты, определённые в STL? Приведите примеры использования.
43. Чем отличается применение в алгоритмах предикатов и функциональных объектов?
44. Какие вы знаете способы модифицирования алгоритмов в STL?
45. Чем отличаются операторы operator< и operator()?
46. Какой заголовочный файл используется при работе с последовательностями?
47. Какой заголовочный файл применяется при работе с вычислительным алгоритмом?
48. Объясните работу алгоритма replace.
49. Объясните работу алгоритма reverse.
50. Каким образом можно определить длину строки и длину массива?
51. С помощью каких операторов память под массивы выделяется динамически и затем освобождается?
52. Чем отличаются алгоритмы count и count_if?
53. Какие возможности вы знаете при работе вычислительного алгоритма accumulate?
54. Каким целям служит заголовочный файл <iomanip>?
55. Какие манипуляторы применяются для настройки вывода вещественных чисел?
56. Какие различные виды сортировок в STL вы знаете?
57. Что такое частичная сортировка?
58. Какие особенности у стабильной сортировки?
59. В чём особенность сортировки nth_element?
60. Какие манипуляторы заголовочного файла <iomanip> вы узнали в этой практической работе?
61. Что означает перегрузка оператора вывода?
62. Чем объявление функции отличается от определения функции?
63. Что такое прототип функции и зачем он используется?
64. Каким образом можно написать предикат для упорядочения по двум ключам (на примере AgeWeight())?
65. Какие контейнеры называются ассоциативными?
66. Какие структуры лежат в основе физической реализации ассоциативных контейнеров?
67. Как описываются ассоциативные контейнеры?

68. С какой целью в описании указывается способ упорядочивания элементов?
69. Назовите методы и свойства ассоциативных контейнеров.
70. Перечислите особенности множеств и словарей.
71. Перечислите особенности множеств и словарей с дубликатами.
72. Как определяется пара в STL?
73. Каким образом можно получить доступ к отдельным элементам пары?
74. Какие существуют алгоритмы работы с ассоциативными контейнерами как с множествами?
75. Что такое битовое множество?
76. Какие основные операции над битовыми множествами вы знаете?
77. Каким образом можно задать размер битового множества?
78. Каким образом можно использовать битовое множество при хранении и изменении информации о наборах данных?
79. Что такое адаптеры контейнеров?
80. Какие виды адаптеров контейнеров вы знаете?
81. Какие базовые контейнеры могут быть использованя для стека, очереди и очереди с приоритетами?
82. Назовите особенности адаптера «стек» и его основные операции.
83. Назовите особенности адаптера «очередь» и его основные операции.
84. Назовите особенности адаптера «очередь с приоритетами» и его основные операции.
85. Поскольку для стеков, очередей и очередей с приоритетами не применяются итераторы, каким образом можно посмотреть их содержимое?
86. Что означает линейныt списки типа LIFO и FIFO?
87. Каким образом, не применяя операцию push, можно внести данные в адаптеры?
88. Какие различия вы можете назвать между тремя изученными адаптерами контейнеров (стек, очередь и очередь с приоритетами)?

Методический материал по способам модификации алгоритмов

Подведём итог использования алгоритмов в STL с включением предикатов и функциональных объектов. Будем рассматривать алгоритм `sort` для vector<int> v.

1. Применяем стандартный алгоритм `sort`:

`sort (v.begin (), v.end());` – сортировка от меньшего к большему.

2. Затем используем этот алгоритм с предикатом `comparefun()`, для сортировки от большего к меньшему:

```
bool comparefun(int x, int y)
{   return x > y;
}
sort (v.begin (), v.end(), comparefun);
```

3. Используем алгоритм `sort` с функциональным объектом `compare()` – сортировка в убывающем порядке:

```
class compare
{  public:
bool operator() (int x, int y) const
     {   return x > y; }
     };
sort (v.begin (), v.end(), compare());
```

4. Используем алгоритм `sort` с функциональным объектом `greater<int>()`, определённым в STL – сортировка в убывающем порядке:

```
sort (v.begin (), v.end(), greater<int>());
```

В дальнейшем также будем применять указанные способы и для других алгоритмов, тем самым усиливая их возможности.

Пример варианта контрольной работы на последовательные контейнеры

Вариант 6

1. Создать очередь. Ввести переменное количество ненулевых целых (ввод завершается нулем) и вывести их на экран.
2. Используя введённые в п.1 данные, применить функции для вставки и удаления для выполнения следующих действий (после каждого действия следует вывод на экран):
    a. Вставить 5 и 8 в начало.
    b. Перед последним элементом вставить 33.
    c. Удалить последний элемент.
    d. Удалить 4 - элемент.
    e. Отсортировать.
3. Создать вектор, внести в него 8 любых (отсортированных по возрастанию) чисел. Вывести их на экран.
4. Выполнить слияние элементов вектора и очереди в любой контейнер.

Пример варианта контрольной работы на ассоциативные контейнеры

Во всех вариантах меню должно выглядеть следующим образом:

1. Найти заданное значение.
2. Вывести полный список.
3. Внести новое значение.

Вариант 1

1. Создать словарь Country, в котором ключом является название страны, а значением — название столицы этой страны.
2. Заполнить его несколькими значениями (4-5).
3. Создать меню, в котором будут решаться следующие задачи:
    a. Ввести название страны и по ней найти и вывести название столицы.
    b. Вывести на экран полный список стран и их столиц.
    c. Ввести название страны, если такая страна есть, то вывести её столицу, если нет – ввести в словарь название страны и её столицы. После этого вывести на экран пополненный список стран и их столиц.

Методические материалы к практическим и самостоятельным занятиям

- Шаблоны функций и классов с готовыми проектами.
- Изучение последовательных контейнеров (функции, итераторы, алгоритмы) с решением задач и с заданиями для самостоятельного решения.
- Изучение ассоциативных контейнеров (функции, итераторы, алгоритмы) с решением задач и с заданиями для самостоятельного решения.
- Функциональные объекты.
- Разные виды сортировок.
- Готовые проекты «Частотный словарь» и «Библиотека».
- Разработанное приложение «Морской бой».

Для самостоятельной работы студентов (домашние работы, подготовка к контрольным работам и коллоквиумы) выдаются задания различных уровней сложности, решения аналогичных задач, готовые проекты и решения.

Эти материалы позволяют студентам написать по выбранной теме своё приложение в качестве творческой работы.

Автором подготовлено учебное пособие «Использование стандартной библиотеки шаблонов STL в программировании структур и алгоритмов обработки данных» (2021 г.).

Примеры творческих заданий

Вариант 2

Написать программу, отыскивающую проход по лабиринту, с использованием контейнерного класса `stack` из STL.

Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в него возможен со стороны, но не с угла. Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов.

Вариант 8

Написать программу учета заявок на обмен квартир и поиска вариантов обмена.

Каждая заявка содержит сведения о двух квартирах: требуемой (искомой) и имеющейся. Сведения о каждой квартире содержат: количество комнат, площадь, этаж, район.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- ввод заявки на обмен;
- поиск в картотеке подходящего варианта: при совпадении требований и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется из списка, в противном случае поступившая заявка включается в картотеку;
- вывод всей картотеки.

Для хранения данных картотеки использовать контейнерный класс `list`.

Вариант 14

Составить программу моделирования работы автобусного парка.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- начальное формирование данных о всех автобусах в парке в виде списка (ввод с клавиатуры или из файла);
- имитация выезда автобуса из парка: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
- имитация въезда автобуса в парк: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;
- вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте, упорядоченных по номерам автобусов;
- вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте, упорядоченных по номерам маршрутов.

Хранение всех необходимых списков организовать с применением контейнерного класса `map`, в качестве ключа использовать «номер автобуса».