Приложение к ОПОП
«Вычислительные машины, комплексы, системы и сети»

СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

МИНОБРНАУКИ РОССИИ

федеральное государственное автономное образовательное учреждение высшего образования
**«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)**

РАБОЧАЯ ПРОГРАММА

дисциплины

«АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ»

для подготовки бакалавров

по направлению

09.03.01 «Информатика и вычислительная техника»

по профилю

«Вычислительные машины, комплексы, системы и сети»

Санкт-Петербург

2022

ЛИСТ СОГЛАСОВАНИЯ

Разработчики:
старший преподаватель Колинько П.Г.

Рабочая программа рассмотрена и одобрена на заседании кафедры ВТ
19.01.2022, протокол № 1

Рабочая программа рассмотрена и одобрена учебно-методической комиссией ФКТИ з, 24.02.2022, протокол № 2

Согласовано в ИС ИОТ
Начальник ОМОЛА Загороднюк О.В.

1 СТРУКТУРА ДИСЦИПЛИНЫ

Обеспечивающий факультет	ФКТИ
Обеспечивающая кафедра	ВТ
Общая трудоемкость (ЗЕТ)	7
Курс	2, 3
Семестр	5, 4
Виды занятий	
Лекции (академ. часов)	12
Лабораторные занятия (академ. часов)	10
Практические занятия (академ. часов)	8
Иная контактная работа (академ. часов)	6
Все контактные часы (академ. часов)	36
Самостоятельная работа, включая часы на контроль (академ. часов)	216
Всего (академ. часов)	252
Вид промежуточной аттестации	
Дифф. зачет (семестр)	4
Курсовая работа (семестр)	4
Экзамен (семестр)	5

2 АННОТАЦИЯ ДИСЦИПЛИНЫ

«АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ»

Изучаются способы реализации в ЭВМ абстрактных данных и вытекающие из этих способов свойства алгоритмов обработки этих данных. Обсуждаются способы генерации множеств для автоматизации тестирования программ и оборудования. Рассматриваются популярные алгоритмы на ненагруженных и нагруженных графах, жадные алгоритмы, эмпирические алгоритмы для переборных задач. Особое внимание при этом уделяется оптимальной организации данных для этих алгоритмов. Изучаются способы организации данных в реальных задачах, когда одному и тому же набору данных могут применяться одновременно несколько абстрактных моделей. Вводится понятие класса как способа реализации структуры данных в конкретной системе программирования. Даётся способ оценки временной сложности алгоритма в машинном эксперименте.

SUBJECT SUMMARY

«ALGORITHMS AND DATA STRUCTURES»

We study how to implement abstract data into a computer and the resulting properties of these methods of processing these data algorithms. Soba discussed spogenerating sets for the test automation software and equipment. We consider the popular algorithms unloaded and heat-conjugated graphs, greedy, empirical algorithms for search problems. Particular attention is paid to the optimal organization of data for these algorithms. Study ways of organizing data in real per-cottages, where the same data set can be used simultaneously, but more abstract models. The concept of class as a means of impletion of the data structure in a particular programming system. We give a method for evaluating the time complexity of the algorithm in the computer experiment.

3 ОБЩИЕ ПОЛОЖЕНИЯ

3.1 Цели и задачи дисциплины

1. Цели дисциплины:

-изучение абстрактных типов данных и их реализации в объектно-ориентированной программе;

-получение знаний о связи между способами организации данных и вытекающими из них свойствами алгоритма;

-развитие навыков программирования, освоение языка C++ с учётом новейших стандартов;

-получение опыта применения ЭВМ для научных исследований.

2. Задачи дисциплины:

-исследование возможностей популярных структур данных и влияния выбора структуры данных на свойства реализуемого алгоритма;

-изучение популярных алгоритмов и возможных областей их применения;

-приобретение опыта программирования в новейшей версии C++ и постановки экспериментов с полученным программным кодом.

3. Знания:

-особенностей новейших стандартов языка C++, Стандартной библиотеки шаблонов и механизма обработки исключительных ситуаций;

-популярных алгоритмов и возможных областей их применения;

-подходов к решению задач, для которых неизвестны эффективные алгоритмы;

-особенностей кодирования пользовательских структур данных и пользовательских контейнеров.

4. Умения:

-быстро создавать надёжные и эффективные программы в современных системах программирования;

-выбрать оптимальные структуры данных при реализации алгоритма и обосновать свой выбор;

-эффективно реализовать и применить пользовательскую структуру данных и пользовательский контейнер, совместимый с алгоритмами Стандартной библиотеки шаблонов.

5. Навыки проектирования и отладки программ в парадигме объектно-ориентированного программирования в стандарте C++11/14/17/20/23, а также измерения свойств алгоритма в эксперименте на ЭВМ, применения ЭВМ в качестве инструмента для научных исследований.

3.2 Место дисциплины в структуре ОПОП

Дисциплина изучается на основе ранее освоенных дисциплин учебного плана:

1. «Информатика»

2. «Программирование»

и обеспечивает изучение последующих дисциплин:

1. «Математическая логика и теория алгоритмов»

2. «Объектно-ориентированное программирование»

3. «Базы данных»

3.3 Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

В результате освоения образовательной программы обучающийся должен достичь следующие результаты обучения по дисциплине:

Код компетенции/ индикатора компетенции	**Наименование компетенции/индикатора компетенции**
ПК-1	Способен осуществлять проведение научно-исследовательских и опытно-конструкторских разработок по отдельным разделам темы
ПК-1.2	*Осуществляет выполнение экспериментов и оформление результатов исследований и разработок*
ПК-2	Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего масштаба и сложности
ПК-2.1	*Анализирует проблемную ситуацию, планирует разработку системы, осуществляет постановку целей*
ПК-2.3	*Организует оценку соответствия требованиям существующих систем и их аналогов*
ПК-3	Способен разрабатывать требования и проектировать программное обеспечение
ПК-3.1	*Анализирует требования к программному обеспечению*
ПК-3.3	*Проектирует структуры данных*

4 СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1 Содержание разделов дисциплины

4.1.1 Наименование тем и часы на все виды нагрузки

№ п/п	Наименование темы дисциплины	Лек, ач	Пр, ач	Лаб, ач	ИКР, ач	СР, ач
1	Введение	1				1
2	Данные и алгоритмы	1	0	2		4
3	Генерация тестов	1	0	0		10
4	Множество как объект	1	2	2	2	14
5	Введение в Стандартную библиотеку шаблонов (STL)		0			10
6	Графы. Основные понятия, машинное представление и обходы	1		2		20
7	Алгоритмы на графах общего вида	1	2	0	2	14
8	Графы с нагруженными вершинами: сортировка		0	0		8
9	Графы с нагруженными рёбрами: поиск кратчайших путей		0	0		8
10	Графы с нагруженными вершинами и рёбрами: потоки в сетях и смежные задачи		0	0		12
11	Жадные алгоритмы решения информационных задач			0		8
12	Практические способы решения NP-полных задач		0			8
13	Иерархия классов	1	2	2		24
14	Исключительные ситуации	1	2	2		8
15	Способы хранения в памяти изменяющихся множеств	1	0	0		10
16	Деревья двоичного поиска с автобалансировкой	1	0	0		20
17	Поддержка последовательностей в структуре данных для множеств	1	0	0		12
18	Пользовательские дополнения к Стандартной библиотеке шаблонов (STL)		0	0		8
19	Ассоциативные контейнеры			0		10
20	Умные указатели		0			4
21	Измерение временной сложности алгоритма в эксперименте на ЭВМ		0	0	2	2
22	Заключение	1				1
	Итого, ач	12	8	10	6	216
	Из них ач на контроль	0	0	0	0	13
	Общая трудоемкость освоения, ач/зе	252/7				

4.1.2 Содержание

№ п/п	Наименование темы дисциплины	Содержание
1	Введение	Предмет дисциплины, её объём, содержание и связь с другими дисциплинами учебного плана. Роль дисциплины в подготовке инженеров в области разработки средств вычислительной техники, цели и задачи дисциплины. Перечень дисциплин и разделов, усвоение которых необходимо для изучения дисциплины. Обзор литературы по курсу. Свойства алгоритма и их оценка. Временная и ёмкостная сложность. Способы оценки. Обозначения для сравнения скорости роста функций. Оценка сложности алгоритма по его структуре. Последовательные шаги. Выбор. Цикл.
2	Данные и алгоритмы	Понятие данных. Примитивные данные. Стандартные структуры данных. Множества, отображения, последовательности. Очереди и стеки. Память ЭВМ и структуры данных. Связь понятий «структура данных» в теории алгоритмов и «объект» в программировании. Множества. Основные понятия. Способы представления множеств в ЭВМ.
3	Генерация тестов	Автоматическая генерация тестов для программ и оборудования. Получение всех подмножеств заданного множества. Правила работы с датчиком случайных чисел и получение случайного подмножества заданного множества. Получение всех подмножеств в лексикографическом порядке. Понятие о кодовом расстоянии и его применение при создании тестов. Код Грея и его генерация. Множества с повторениями. Генерация подмножеств множества с повторениями. Генерация K-элементных подмножеств. Генерация перестановок.
4	Множество как объект	Объекты и функции-члены: объявление и использование. Функции-члены inline по умолчанию. Создание и уничтожение объектов: конструкторы и де-структоры. Умолчания. Списки инициализации. Константные члены класса и способы их инициализации. Массивы объектов. Перегрузка операций. Общие правила. Одноместные и двуместные операции. Перегрузка операций обычной и дружественной функцией. Перегрузка префиксного и постфиксного инкремента/декремента. Использование одного класса внутри другого. Дружественные классы. Конструктор копирования и перегрузка присваивания. Оптимизация возвращаемого значения. Перемещающее копирование и присваивание.

№ п/п	Наименование темы дисциплины	Содержание
5	Введение в Стандартную библиотеку шаблонов (STL)	Шаблоны. Обобщённые функции. Обобщённые классы. Шаблоны для стека и очереди. Стандартная библиотека шаблонов: обзор. Контейнеры. Итераторы. Функторы. Алгоритмы. Перегрузка операций разыменования и взятия адреса. Итераторы: сущность и классификация. Умные указатели.
6	Графы. Основные понятия, машинное представление и обходы	Абстрактная структура данных «отношение». Бинарное отношение на произвольном множестве и граф как подходящая модель для этого. Машинное представление графов. Деревья: основные понятия и машинное представление. Рекурсивные алгоритмы обхода дерева. Нерекурсивные алгоритмы обхода дерева: в глубину и в ширину. Временная сложность обхода дерева. Обходы графа общего вида. Обход в глубину для орграфа. Обход в ширину. Стягивающие деревья.
7	Алгоритмы на графах общего вида	Фундаментальное множество циклов и его интерпретация. Отыскание фундаментального множества циклов. Компоненты двусвязности. Алгоритм отыскания компонент двусвязности. Сильная связность в орграфе. Эйлеровы пути. Алгоритм нахождения эйлерова цикла. Задачи на полный перебор (NP-полные задачи) и алгоритм с возвратом. Получение стягивающего дерева полным перебором. Алгоритм отыскания Гамильтонова цикла.
8	Графы с нагруженными вершинами: сортировка	Общие понятия о сортировке. Классификация видов сортировки. Оптимальная сортировка. Дерево сортировки и сортировка деревом. Быстрая сортировка. Сортировка слиянием. Сортировка вычерпыванием и поразрядная. Сортировка цепочек.
9	Графы с нагруженными рёбрами: поиск кратчайших путей	Кратчайшие пути в орграфе. Алгоритм отыскания кратчайшего пути. Кратчайший путь от фиксированной вершины: алгоритм Форда-Беллмана. Кратчайший путь в графе с неотрицательными весами: алгоритм Дейкстры. Кратчайшие пути в бесконтурном графе. Кратчайшие пути между всеми парами вершин: алгоритм Флойда. Транзитивное замыкание отношения. Алгоритм Уоршалла. Стягивающее дерево наименьшей стоимости: алгоритмы Прима и Краскала.
10	Графы с нагруженными вершинами и рёбрами: потоки в сетях и смежные задачи	Потоки в сетях: основные понятия. Общая схема отыскания максимального потока в сети. Построение вспомогательного бесконтурного графа. Алгоритм отыскания псевдомаксимального потока во вспомогательном бесконтурном графе. Наибольшее паросочетание в двудольном графе: основные понятия и алгоритм. Трансверсаль и её отыскание. Вершинное покрытие в двудольном графе.

№ п/п	Наименование темы дисциплины	Содержание
11	Жадные алгоритмы решения информационных задач	Жадные алгоритмы: основные понятия. Понятие о матроидах как основе разработки жадных алгоритмов. Примеры матроидов. Жадный алгоритм для матриц: схема Гаусса. Жадные алгоритмы для графов: алгоритм Дейкстры для отыскания кратчайших путей; схемы Прима и Краскала для стягивающего дерева наименьшей стоимости. Жадный алгоритм для трансверсалей: отыскание частичной трансверсали с наибольшим весом.
12	Практические способы решения NP-полных задач	Некоторые важные NP-полные задачи. Варианты постановки: поиск оптимума и принятие решения. Причина малой практической полезности алгоритма перебора с возвратом. Применение жадных алгоритмов для решения NP-полных задач. Примеры жадных алгоритмов для задачи коммивояжёра, раскладки по ящикам, упаковки рюкзака. Обсуждение результатов. Приложения. Эмпирические алгоритмы решения NP-полных задач. Эмпирический полиномиальный алгоритм решения задачи о раскраске графа. Муравьиный алгоритм решения задачи коммивояжёра.
13	Иерархия классов	Производные классы — что это даёт. Функции-члены. Конструкторы и деструкторы. Пример иерархии классов. Управление доступом к объектам в иерархии: поля типа и их альтернатива — виртуальные функции. Абстрактные классы. Контроль наследования: override и final в новом стандарте. Проектирование иерархии классов: отношения «является» и «содержит». Понятие о паттернах проектирования как типовых схемах построения и организации взаимодействия классов. Закрытое наследование и его альтернатива — включение. Достоинства и недостатки каждого из способов. Контроль доступа к членам базовых классов. Защищённые члены и защищённое включение. Виртуальные базовые классы. Общая идея производных классов и абстрактного базового класса на примере библиотеки фигур. Классы «фигура» и «линия». Необязательность определения ненужных функций. Пример законченной программы с иерархией классов. Множественное наследование. Множественное вхождение базового класса. Разрешение неоднозначности. Виртуальные деструкторы.

№ п/п	Наименование темы дисциплины	Содержание
14	Исключительные ситуации	Использование механизма исключительных ситуаций для проектирования реакции программы на ошибки времени выполнения. Учёт исключительных ситуаций при построении библиотек. Базовые и расширенные гарантии. Функции, не генерирующие исключений. Функции, прозрачные для исключений. Стандартные исключительные ситуации: exception, bad_alloc и др. Неожиданные исключительные ситуации и возможности для их обработки.
15	Способы хранения в памяти изменяющихся множеств	Хеширование как обобщение способа хранения множества в массиве битов. Открытые и закрытые хеш-таблицы: способы разрешения коллизий. Рекомендации по составлению хеш-функций. Двуместные операции с множествами в форме хеш-таблиц. Значение упорядоченности для двуместных операций над множествами в форме последовательностей. Дерево двоичного поиска (ДДП) как расширяемый список с сохранением упорядоченности. Упорядоченный массив как способ хранения дерева двоичного поиска. Двуместные операции над множествами за линейное время. Дерево двоичного поиска: алгоритмы вставки и удаления.
16	Деревья двоичного поиска с автобалансировкой	Вырождение при произвольной вставке последовательности узлов. Деревья с автобалансировкой. АВЛ-дерево: определение и алгоритмы балансировки при вставке и удалении элемента множества. Красно-чёрное дерево: балансировка при вставке и удалении. 2-3-дерево. Алгоритмы вставки и удаления. Рекомендация по кодированию в виде варианта ДДП. Другие варианты деревьев двоичного поиска, устойчивые к вырождению: 1-2-дерево, дерево с хранением в узлах высот или мощностей поддеревьев. Получение дерева с автобалансировкой из упорядоченного массива: АВЛ-и красно-чёрное дерево. Получение 2-3-дерева из упорядоченной последовательности. Двуместные операции с ДДП: алгоритм пошагового обхода двоичного дерева для поддержки схемы слияния. Пошаговый обход 2-3-дерева.

№ п/п	Наименование темы дисциплины	Содержание
17	Поддержка последовательностей в структуре данных для множеств	Хранение последовательностей в структуре данных для множеств. Варианты: обход, номера, списки, справочный массив. Примитивы: определение номера по ключу и ключа по номеру. Двуместные операции над последовательностями: вставка, замена, удаление по интервалу номеров и по образцу, конкатенация и размножение. Множества с повторениями в хеш-таблице, ДДП, АВЛ-дереве, 2-3-дереве для обеспечения поддержки произвольной последовательности. Особенности кодирования ДДП, облегчающие реализацию зеркальных вариантов балансировки. Реализация универсального алгоритма двуместной операции на примере АВЛ-дерева со справочным массивом указателей.
18	Пользовательские дополнения к Стандартной библиотеке шаблонов (STL)	Особенности STL в новых стандартах языка (C++11/14/17/20). Функциональные объекты: указатели на функцию и функторы. Лямбда — функциональный объект, введённый стандартом C++11. Общая характеристика функциональных объектов с точки зрения применимости в алгоритмах. Пользовательский контейнер, пригодный для работы с алгоритмами STL. Пользовательские операторы чтения и вставки для хеш-таблицы и ДДП.
19	Ассоциативные контейнеры	Ассоциативные контейнеры — краткий обзор. Словарь и множество. Мультисловарь и мультимножество. Множество битов. Неупорядоченные ассоциативные контейнеры.
20	Умные указатели	Умные указатели: auto_ptr и его последователи unique_ptr, shared_ptr, week_ptr в новом стандарте. Примеры использования unique_ptr. Указатель unique_ptr для работы с массивами. Пример использования shared_ptr и week_ptr.

№ п/п	Наименование темы дисциплины	Содержание
21	Измерение временной сложности алгоритма в эксперименте на ЭВМ	Использование ЭВМ в научном эксперименте. Математические модели. Особенности постановки на ПЭВМ опытов со случайным результатом. Способы генерации исходных данных. Измерение времени решения задачи. Методика обработки результатов эксперимента: введение в регрессионный анализ. Учебный пример DemoSTL с хеш-таблицей и поддержкой последовательности с использованием вектора итераторов. Объявление класса MyCont и его составляющих, конструкторы и деструктор, перегрузка присваивания. Точные способы измерения времени прохождения теста: — ассемблерная вставка для команды RDTSC; — QueryPerformanceCounter (timepoint) и QueryPerformanceFrequency (frq) из библиотеки Windows.h; — возможности STL: библиотека stl::chrono.
22	Заключение	Порядок итоговой аттестации.

4.2 Перечень лабораторных работ

Наименование лабораторной работы	Количество ауд. часов
1. Данные и алгоритмы	2
2. Множество как объект	2
3. Графы. Основные понятия, машинное представление и обходы	2
4. Иерархия классов	2
5. Исключительные ситуации	2
Итого	10

4.3 Перечень практических занятий

Наименование практических занятий	Количество ауд. часов
1. Множество как объект	2
2. Алгоритмы на графах общего вида	2
3. Иерархия классов	2
4. Исключительные ситуации	2
Итого	8

4.4 Курсовое проектирование

Цель работы (проекта): Реализация в виде программы и исследование алгоритма для работы с графами.

Содержание работы (проекта): Семестр 4.

Найти в курсе лекций на Образовательном портале ЛЭТИ и рекомендованных литературных источниках описание алгоритма для работы с графами в соответствии с индивидуальным заданием. Сформулировать поставленную задачу в терминах теории множеств. Выбрать структуру данных, оптимальную для представления исходных данных и результата. Реализовать алгоритм в виде программы на ЭВМ. Предложить тесты для проверки правильности работы программы. Оценить временную сложность алгоритма и его реализации на ЭВМ.

Курсовая работа в части таблиц, рисунков и оглавления оформляется в соответствии с ГОСТ 7.32-2017 ”Отчет о научно-исследовательской работе. Структура и правила оформления”.

Курсовая работа состоит из следующих разделов:

-Введение.

-Постановка задачи.

-Формулировка задачи в терминах теории множеств.

-Выбор оптимального способа представления данных и результата.

-Описание алгоритма и оценка его временной сложности.

-Результаты тестирования программы на ЭВМ.

-Заключение.

-Приложение. Текст программы в виде, пригодном для компиляции.

Оформление пояснительной записки на курсовую работу выполняется в соответствии с требованиями к студенческим работам, принятым в СПбГЭТУ. В частности, отчёт по курсовой работе должен содержать:

-титульный лист;

-аннотацию;

-лист содержания;

-задание на работу;

-описание поставленной задачи в терминах теории множеств с обязательной ссылкой на первоисточники (не менее двух);

-обоснование по выбору структур данных для эффективной реализации алгоритма;

-описание алгоритма и пояснения к его реализации;

-результаты тестирования, доказательство корректности реализации алгоритма;

-оценка временной сложности алгоритма;

-выводы по результатам исследования алгоритма;

-перечень ссылочных документов;

-приложение: текст программы (в составе отчёта) или перечень файлов.

Объём пояснительной записки составляет, таким образом, в среднем 8 -10 страниц, без учёта прилагаемых к нему текстов программ. В курсовой работе должно быть использовано не менее 2-х и не более 5-и источников. Курсовая работа оформляется в редакторе документов в формате MS Word или OpenDocument, шрифт для текста (включая текст программы) должен иметь размер 12-14 пунктов. Для основного текста используется шрифт с засечками. для текста программы -моноширинный.

Курсовая работа сдается преподавателю или присылается ему по почте в электронном виде и защищается при личной встрече с ним на занятиях.

Темы:

№ п/п	**Название темы**	**Перевод темы**
1	Получение множества двудольных компонент ориентированного графа.	Preparation of a plurality of components of bipartite undirected graph
2	Проверка ориентированного графа на ацикличность .	Checking on the a cyclic directed graph
3	Поиск изоморфного неориентированного подграфа.	Search undirected subgraph isomorphic
4	Отыскание максимального паросочетания в произвольном неориентированном графе.	Finding a maximum matching in an arbitraryum directed graph

№ п/п	Название темы	Перевод темы
5	Построение полного множества циклов для ориентированного графа.	Construction of the full set of cycles for a directed graph
6	Проверка наличия цикла отрицательной стоимости в ориентированном графе с нагруженными рёбрами.	Check for negative cost cycles in a directed graph with loaded ribs
7	Построение фундаментального множества циклов в неориентированном графе.	Construction of a fundamental set of cycles in an undirected graph

4.5 Реферат

Реферат не предусмотрен.

4.6 Индивидуальное домашнее задание

Индивидуальное домашнее задание - это домашняя контрольная работа, которую выполняют обучающиеся заочной формы обучения в семестра 4 и 5 в соответствие с учебным планом.

Семестр 4.

Тема: Множество в памяти ЭВМ

Исходные данные и требования: вид универсума и одно из 50 возможных выражений для вычисления пятого множества по заданным четырём.

Задание: Напишите программу, реализующую вычисление пятого множества из заданного универсума по четырём имеющимся по заданной формуле, используя представление множества в памяти массивом, списком, массивом битов и машинным словом.

Предложите тесты и выполните тестирование каждого из вариантов реализации.

Оцените их временную сложность.

Семестр 5.

Тема: Иерархия классов.

Исходные данные и требования: одна из 50 возможных фигур для добавки в коллекцию и комбинация из 12 возможных позиций её размещения в итоговой картинке.

Задание: В существующий проект рисования коллекции фигур на экране ЭВМ добавьте ещё одну фигуру, найдя для неё оптимальное место в существующей иерархии классов. Продемонстрируйте результат добавления вашей фигуры в заданные места сводной картинки, создав для этого недостающие функции присоединения.

Оформление работы по общепринятым в университете «Требования к оформлению научно-технических отчетов».

4.7 Доклад

Доклад не предусмотрен.

4.8 Кейс

Кейс не предусмотрен.

4.9 Организация и учебно-методическое обеспечение самостоятельной работы

Изучение дисциплины сопровождается самостоятельной работой студентов с рекомендованными преподавателем литературными источниками и информационными ресурсами сети Интернет.

Планирование времени для изучения дисциплины осуществляется на весь период обучения, предусматривая при этом регулярное повторение пройденного материала. Обучающимся, в рамках внеаудиторной самостоятельной работы, необходимо регулярно дополнять сведениями из литературных источников материал, законспектированный на лекциях. При этом на основе изучения ре-

комендованной литературы целесообразно составить конспект основных положений, терминов и определений, необходимых для освоения разделов учебной дисциплины.

Особое место уделяется консультированию, как одной из форм обучения и контроля самостоятельной работы. Консультирование предполагает особым образом организованное взаимодействие между преподавателем и студентами, при этом предполагается, что консультант либо знает готовое решение, которое он может предписать консультируемому, либо он владеет способами деятельности, которые указывают путь решения проблемы.

Текущая СРС	**Примерная трудоемкость, ач**
Работа с лекционным материалом, с учебной литературой	31
Опережающая самостоятельная работа (изучение нового материала до его изложения на занятиях)	0
Самостоятельное изучение разделов дисциплины	30
Выполнение домашних заданий, домашних контрольных работ	36
Подготовка к лабораторным работам, к практическим и семинарским занятиям	0
Подготовка к контрольным работам, коллоквиумам	36
Выполнение расчетно-графических работ	0
Выполнение курсового проекта или курсовой работы	70
Поиск, изучение и презентация информации по заданной проблеме, анализ научных публикаций по заданной теме	0
Работа над междисциплинарным проектом	0
Анализ данных по заданной теме, выполнение расчетов, составление схем и моделей, на основе собранных данных	0
Подготовка к зачету, дифференцированному зачету, экзамену	13
ИТОГО СРС	216

5 Учебно-методическое обеспечение дисциплины

5.1 Перечень основной и дополнительной литературы, необходимой для освоения дисциплины

№ п/п	Название, библиографическое описание	К-во экз. в библ.
Основная литература		
1	Колинько, Павел Георгиевич. Пользовательские структуры данных [Текст] : учеб.-метод. пособие / П. Г. Колинько, 2020. -63, [1] с.	110
2	Колинько, Павел Георгиевич. Пользовательские контейнеры [Текст] : учеб.-метод. пособие / П. Г. Колинько, 2020. -63 с.	110
Дополнительная литература		
1	Страуструп, Бьярне. Программирование: принципы и практика с использованием C++ [Текст] : пер. с англ. / Б. Страуструп, 2019. -1328 с.	15
2	Поздняков, Сергей Николаевич. Дискретная математика [Текст] : учеб. для вузов по направлениям подгот. ”Информатика и вычисл. техника”, ”Информационные системы”, ”Информационная безопасность” / С.Н. Поздняков, С.В. Рыбин, 2008. -448 с.	491
3	Новиков, Федор Александрович. Дискретная математика для программистов [Текст] : для студентов вузов по направлению подгот. ”Информатика и вычисл. техника” / Ф.А. Новиков, 2008. -383 с.	38
4	Ахо, Альфред В. Построение и анализ вычислительных алгоритмов [Текст] : монография / А.Ахо, Д.Хопкрофт, Дж.Д.Ульман; Пер. с англ. А.О.Слисенко; Под ред. Ю.В.Матиясевича, 1979. -536 с.	11
5	Ахо, Альфред В. Структуры данных и алгоритмы [Текст] : монография / А.В.Ахо, Д.Э.Хопкрофт, Дж.Д.Ульман, 2001. -382 с.	43
6	Липский, Витольд. Комбинаторика для программистов [Текст] / В. Липский ; пер. с польск. В. А. Евстигнеева и О.А. Логиновой ; под ред. А. П. Ершова, 1988. -213 с.	6
7	Макконелл Дж. Основы современных алгоритмов [Текст] : учеб. пособие для вузов по направлению подгот. специалистов ”Информатика и вычислительная техника” : пер. с англ. / Дж. Макконелл ; пер. с англ. под ред. С.К. Ландо ; доп. М.В. Ульянова, 2004. -366 с.	5
8	Кнут Д.Э. Искусство программирования для ЭВМ [Текст] : учеб. пособие : пер. с англ. Т. 3 : Сортировка и поиск / под ред. Ю. М. Баяковского, В. С. Штаркмана, 1978. -844 с.	26
9	Кормен, Томас. Алгоритмы: построение и анализ [Текст] : Учеб. / Т. Кормен; Ч.Лейзерсон, Р.Ривест; Пер. с англ. под ред. А.Шен, 1999. -955 с.	8
10	Седжвик, Роберт. Фундаментальные алгоритмы на C++ [Текст] : [Пер. с англ.]. Ч. 5 : Алгоритмы на графах : монография, 2002. -484 с.	45

5.2 Перечень ресурсов информационно-телекоммуникационной сети «Интернет», используемых при освоении дисциплины

№ п/п	Электронный адрес
1	C++ Reference: Актуальная справка о языке C++11/14/17/20/23, шаблонах, библиотечных фун.http://ru.cppreference.com
2	C++ Reference: Актуальная справка о языке C++11/14/17/20/23, шаблонах, библиотечных фун. (на языке оригинала)http://en.cppreference.com
3	Язык программирования C++ для профессионаловhttps://intuit.ru/studies/courses/98/98/info

5.3 Адрес сайта курса

Адрес сайта курса: https://vec.etu.ru/moodle/course/view.php?id=11726

6 Критерии оценивания и оценочные материалы

6.1 Критерии оценивания

Для дисциплины «Алгоритмы и структуры данных» предусмотрены следующие формы промежуточной аттестации: экзамен, зачет с оценкой.

Экзамен

Оценка	Описание
Неудовлетворительно	Курс не освоен. Студент испытывает серьезные трудности при ответе на ключевые вопросы дисциплины
Удовлетворительно	Студент в целом овладел курсом, но некоторые разделы освоены на уровне определений и формулировок
Хорошо	Студент овладел курсом, но в отдельных вопросах испытывает затруднения. Умеет решать задачи
Отлично	Студент демонстрирует полное овладение курсом, способен применять полученные знания при решении конкретных задач

Зачет с оценкой

Оценка	Описание
Неудовлетворительно	Курс не освоен. Студент испытывает серьезные трудности при ответе на ключевые вопросы дисциплины
Удовлетворительно	Студент в целом овладел курсом, но некоторые разделы освоены на уровне определений и формулировок
Хорошо	Студент овладел курсом, но в отдельных вопросах испытывает затруднения. Умеет решать задачи
Отлично	Студент демонстрирует полное овладение курсом, способен применять полученные знания при решении конкретных задач

Особенности допуска

Семестр 4. К дифференцированному зачёту допускаются студенты, выполнившие индивидуальное домашнее задание (домашнюю контрольную работу), лабораторные работы и подготовившие и защитившие отчёты по ним, а также выполнившие и защитившие курсовую работу.

Дифференцированный зачёт проводится в виде теста из 20 вопросов, на каждый вопрос даётся по одной минуте, правильный ответ оценивается в один балл. Результирующая оценка формируется по результатам итогового теста и результатам защиты отчетов лабораторных работ и курсовой работы.

Семестр 5. К экзамену допускаются студенты, выполнившие индивидуальное домашнее задание (домашняя контрольную работу), лабораторные работы и подготовившие и защитившие отчёты по ним.

Экзамен проводится в виде теста из 20 вопросов, на каждый вопрос даётся по одной минуте, правильный ответ оценивается в один балл.

Студентам, набравшим 19 или 20 баллов, ставится оценка ”отлично”, 17-18 баллов -”хорошо”, 14-16 баллов -”удовлетворительно”, 13 и менее -”неудовлетворительно”.

Результирующая оценка формируется по результатам теста и защит отчетов лабораторных работ.

6.2 Оценочные материалы для проведения текущего контроля и промежуточной аттестации обучающихся по дисциплине

Вопросы к экзамену

№ п/п	Описание
1	Какой базовый класс лучше всего использовать для производного класса «треугольник »?
2	Какой тип наследования следует выбрать: private, public или protected?
3	Можно ли вообще не указывать тип наследования?
4	В чём смысл объявления функций в базовом классе как виртуальных?

5	Нужно ли объявлять виртуальной функцию в производном классе, если в базовом она уже объявлена таковой?
6	Можно ли потребовать от компилятора проверить корректность объявления виртуальной функции в производном классе и как это сделать?
7	Можно ли запретить виртуальную функцию в классах-наследниках?
8	Что такое «чисто виртуальная функция»?
9	Обязательно ли переопределять все функции-члены базового класса в производном классе?
10	Как запретить для объекта вызов конструктора по умолчанию?
11	Как запретить вызов конструктора для использования в качестве неявного преобразователя типа?
12	Каким образом следует инициализировать объект базового класса в конструкторе производного класса? Всегда ли это нужно делать?
13	Каким требованиям должны удовлетворять классы, чтобы между ними можно было сформировать отношение «является»?
14	Как установить между двумя классами отношение «содержит»?
15	Что такое исключительная ситуация при выполнении программы?
16	Как можно выявить исключительную ситуацию?
17	Что можно предпринять при выявлении исключительной ситуации?
18	Можно ли передать в обработчик особых ситуаций какуюлибо информацию о произошедшем событии?
19	Можно ли обработать неизвестную особую ситуацию?
20	Можно ли сделать обработчик ситуации пустым?
21	Что можно предпринять, если для корректной обработки ситуации в данном месте программы у обработчика недостаточно данных?
22	Если требуется несколько обработчиков особых ситуаций, в каком порядке следует их размещать в программе?
23	Можно ли перехватывать одним обработчиком несколько различных особых ситуаций?
24	Как действуют обработчики в случае, когда никакой особой ситуации не произошло?
25	Как следует размещать блоки контроля, чтобы получить безопасный программный код?
26	Можно ли продолжить выполнение программы с точки, в которой выявлена ошибка, после внесения исправлений в данные?
27	Какой объём памяти нужно выделять под хеш-таблицу для хранения множеств со средней мощностью 50?
28	Каким требованиям должна удовлетворять «хорошая» хеш-функция?
29	Можно ли построить хеш-таблицу, в которой не будет коллизий?
30	Каков оптимальный алгоритм выполнения двуместной операции над множествами в хеш-таблице? Какова его временная сложность?
31	Можно ли хранить в хеш-таблице множество с повторениями?
32	Какова временная сложность операций вставки и удаления элемента для хеш-таблицы?
33	Почему для операций с хеш-таблицей дают две оценки временной сложности — сложность в худшем случае и сложность в среднем?

34	Что такое вырождение хеш-таблицы и как его избежать?
35	Каким способом следует разметить дерево, чтобы в нём был возможен двоичный поиск? Как его следует нагрузить?
36	Почему для операций с двоичным деревом дают две оценки сложности — «в худшем случае» и «в среднем»? Почему не рассматривается «лучший» случай?
37	Можно ли воспрепятствовать вырождению ДДП?
38	Каков оптимальный алгоритм двуместной операции над множествами, представленными деревьями двоичного поиска?
39	Может ли при двуместной операции над множествами в ДДП получиться вырожденное дерево-результат?
40	Как сделать не вырождающееся ДДП? Зачем оно может понадобиться?
41	Какая структура данных является оптимальной для хранения дерева двоичного поиска?
42	Почему для хранения произвольной последовательности структуру данных для множества (хеш-таблицу или ДДП) приходится дорабатывать?
43	Влияет ли доработка структур данных для множеств для поддержки последовательностей на временную сложность операций над множествами?
44	Какова оптимальная доработка структуры данных и временная сложность для операции исключения части последовательности между указанными позициями?
45	Что такое стандартный контейнер библиотеки STL? Чем он отличается от обычного объекта?
46	Какой стандартный контейнер можно считать наиболее подходящим для работы с множествами?
47	Можно ли использовать стандартные контейнеры для множеств, на которых не определено отношение полного порядка?
48	Существуют ли ограничения на применение стандартных алгоритмов двуместных операций над множествами в контейнерах?
49	С какой целью может понадобиться оформить пользовательскую структуру данных как стандартный контейнер?
50	Каков минимально необходимый набор средств для превращения пользовательской структуры данных в полноценный контейнер?
51	Зачем нужны гарантии устойчивости алгоритмов относительно исключений?
52	Почему базовых гарантий устойчивости алгоритма к исключениям может быть недостаточно?
53	В чём заключаются расширенные гарантии устойчивости алгоритма к исключениям и как их можно обеспечить?

Вариант экзаменационного теста

Вариант 101

1. Какой способ размещения множеств в памяти обеспечивает самый быстрый алгоритм выполнения двуместной операции над ними?

а) массив битов

б) массив элементов

в) машинное слово

2. Можно ли использовать ключевое слово struct в том же смысле, что и class?

а) можно без ограничений

б) нельзя

в) можно, если явно указать все спецификаторы доступа

3. class MyClass {

int A;

public:

int FunA ( );

};

int FunB(MyClass A);

Каким образом функция FunB( ) может использовать переменную A?

а) с уточнением области действия: MyClass :: A

б) через указатель: this->A

в) не может

г) непосредственно

4. class MyClass {

static int A;

public: MyClass ( );

};

Как следует задавать значение статической переменной A?

а) утверждением вида int MyClass :: A = 0; вне всех функций

б) в списке инициализации конструктора

в) в теле конструктора

5. Можно ли объявить класс, который будет играть роль элемента множества списка и множества в целом?

а) можно, но не стоит

б) да, это хороший способ

в) нельзя

6. Каким способом можно перегрузить операцию «=»?

а) задать обычную функцию

б) годятся оба способа

в) задать функцию-член

7. Каким способом объект-множество лучше всего передавать в качестве аргумента в функцию?

а) через указатель

б) по значению

в) константной ссылкой

8. Объявлен массив целых:

int A[100]{0};

Какие элементы массива инициализируются нулём?

а) только нулевой

б) все

в) никакие: ошибка компилятора

9. Сколько может быть рёбер в неориентированном графе из 100 вершин?

а) 101

б) 10000

в) 4500

г) 99

10. Задан граф из 100 вершин и 1000 рёбер. Рёбра нагружены весом, не превышающим 10. Какое значение «бесконечности» достаточно использовать в качестве веса несуществующих рёбер в матрице весов?

а) 1000

б) 999

в) 10000

г) 2000

Вопросы к дифф.зачету

№ п/п	Описание
1	Какой объём памяти при работе с множествами в массивах нужно выделить под массив-результат?
2	При вычислении выражения, в которое входит четыре множества, можно ли сэкономить память, если обрабатывать множества не попарно, а все сразу?
3	Какова временная сложность получения результата одновременной обработки нескольких множеств?
4	Какова временная сложность вычисления выражения, в которое входит несколько множеств, в случае представления их массивами битов?
5	Какова временная сложность операций над множествами, представленными машинными словами?
6	Можно ли считать представление множества отображением на универсум универсальным приёмом, применимым для любых задач?
7	Можно ли применить для тестирования алгоритма работы с множествами генератор множества всех подмножеств?
8	Какой из способов генерации случайного множества можно считать самым удобным?
9	В каких случаях тестирования вычисления выражения для множеств целесообразно применить несимметричный генератор тестов?
10	Можно ли применять для генерации подмножества заданной мощности генерацию случайных битов с остановкой по достижении нужного их количества?
11	Как правильно организовать эксперимент для сравнения фактического быстродействия разных способов представления множеств?

12	Сколько раз нужно повторять тест при измерении времени его выполнения функцией clock( )?
13	Какую выгоду можно получить от применения объектов в программе обработки множеств?
14	Как влияет применение объектов на время вычисления множества-результата? Можно ли исключить такое влияние?
15	Все ли созданные в программе множества действительно уничтожаются?
16	В каком порядке происходит уничтожение множеств?
17	Можно ли ожидать в программе использования конструктора копии или присваивания в варианте «с переносом»?
18	Объясняет ли результат трассировки вызовов служебных функций разницу во времени решения задачи в программах без классов и с классами?
19	Чем отличаются алгоритмы для разных способов обхода деревьев?
20	Нужно ли сочетать ввод данных для построения дерева с клавиатуры с его обходом?
21	Можно ли считать применённые вами алгоритмы обхода дерева эффективными?
22	Нужно ли создавать отдельные классы для узла и для дерева в целом или можно ограничиться одним универсальным, рассматривая любой узел как корень некоторого поддерева?

Вариант теста (дифф. зачет)

Вариант 18201

1. Алгоритм какой временной сложности является самым предпочтительным в общем случае?

а) O(n)

б) O(1)

в) O(n^2)

г) O(log n)

2. Алгоритм какой временной сложности является оптимальным для двуместной операции над множествами мощностью n при правильном выборе способа размещения множеств в памяти?

а) O(n^4)

б) O(n)

в) O(1)

г) O(n^2)

3. Какой способ размещения множеств в памяти обеспечивает самый быстрый алгоритм выполнения двуместной операции над ними?

а) массив битов

б) массив элементов

в) машинное слово

4. Каким способом можно перегрузить операцию «[ ]»?

а) задать функцию-член

б) задать функцию-друг

в) годятся оба способа

5. Граф какого вида больше подходит в качестве модели бинарного отношения на произвольном конечном множестве?

а) ни один не подходит

б) оба варианта

в) не ориентированный

г) ориентированный

6. Какое машинное представление графа может быть сделано самым компактным?

а) списки смежности

б) массив пар номеров вершин

в) матрица смежности

г) матрица инциденций

7. Какова максимальная длина элементарного пути между произвольной парой вершин в неориентированном графе из 100 вершин и 1000 рёбер?

а) 1000

б) 901

в) 101

г) 99

8. Какова временная сложность оптимального алгоритма отыскания сильно связных компонент в ориентированном графе из n вершин и m рёбер?

а) O(n^2)

б) O(nm)

в) O(n + m)

г) O(n log n)

9. Сколько фаз увеличения потока может потребоваться в алгоритме Диница для сети из 500 вершин и 2500 рёбер?

а) 499

б) 501

в) 999

г) 2499

10. Какой алгоритм рекомендуется для разреженного графа с нагруженными рёбрами для отыскания в нём дерева наименьшей стоимости?

а) алгоритм Прима

б) алгоритм Гаусса

в) любой из перечисленных

г) алгоритм Краскала

Образцы задач (заданий) для контрольных (проверочных) работ

Тема 1. Множество в памяти ЭВМ

Напишите программу, реализующую вычисление пятого множества латинских букв по четырём заданным по формуле E = A or B and C xor D, используя представление множества в памяти массивом, списком, массивом битов и машинным словом.

Предложите тесты и выполните тестирование каждого из вариантов реализации.

Оцените их временную сложность.

Тема 2. Иерархия классов.

В существующий проект рисования коллекции фигур на экране ЭВМ добавьте ещё одну фигуру - равносторонний треугольник, найдя для неё оптимальное место в существующей иерархии классов. Продемонстрируйте результат добавления вашей фигуры в заданные места сводной картинки, создав для этого недостающие функции присоединения.

Весь комплект контрольно-измерительных материалов для проверки сформированности компетенции (индикатора компетенции) размещен в закрытой части по адресу, указанному в п. 5.3

6.3 График текущего контроля успеваемости

Неделя	Темы занятий	Вид контроля
1	Данные и алгоритмы Генерация тестов	
2		
3		
4		ИДЗ / ИДРГЗ / ИДРЗ
5	Множество как объект	
6		
7		
8		Отчет по лаб. работе
9	Графы. Основные понятия, машинное представление и обходы Алгоритмы на графах общего вида	
10		
11		
12		Отчет по лаб. работе
13	Графы с нагруженными вершинами: сортировка Графы с нагруженными рёбрами: поиск кратчайших путей Графы с нагруженными вершинами и рёбрами: потоки в сетях и смежные задачи Жадные алгоритмы решения информационных задач	
14		
15		
16		Защита КР / КП
18	Иерархия классов	
19		
20		
21		ИДЗ / ИДРГЗ / ИДРЗ
22	Исключительные ситуации	
23		
24		
25		Отчет по лаб. работе
26	Способы хранения в памяти изменяющихся множеств Деревья двоичного поиска с автобалансировкой Поддержка последовательностей в структуре данных для множеств	
27		
28		
29		Отчет по лаб. работе
30	Пользовательские дополнения к Стандартной библиотеке шаблонов (STL) Ассоциативные контейнеры Измерение временной сложности алгоритма в эксперименте на ЭВМ	
31		
32		
33		Отчет по лаб. работе

6.4 Методика текущего контроля

на лекционных занятиях

Текущий контроль включает в себя контроль посещаемости (не менее 80% занятий), по результатам которого студент получает допуск на диффер. зачет /

экзамен.

на лабораторных занятиях

Порядок выполнения лабораторных работ, подготовки отчётов и их защиты

В процессе обучения по дисциплине ”Алгоритмы и структуры данных” студент обязан выполнить 5 лабораторных работ. Под выполнением лабораторных работ подразумевается подготовка к работе, разработка и тестирование программы на ЭВМ, подготовка отчета и его защита. Выполнение лабораторных работ осуществляется каждым студентом индивидуально. Отчёт оформляется в электронном виде в соответствии с принятыми в СПбГЭТУ правилами оформления студенческих работ после завершения отладки программы, выполнения исследований с ней и представляется преподавателю на проверку после демонстрации работы программы. Студент отвечает на вопросы по теоретической части, или по тексту программы, или по последующей обработке результатов. В случае если студент демонстрирует достаточное знание вопроса, работа считается защищённой. На защите лабораторной работы студент должен показать: понимание созданного им программного кода, умение объяснять особенности реализованного алгоритма и возможные области его применения и т.д., умение обосновать выбор наиболее подходящих структур данных для алгоритма, доказать полноту его тестирования, зафиксировать навыки и умения, приобретённые при выполнении лабораторной работы.

Результаты лабораторных работ оцениваются по десятибалльной системе:

Оценка от 0 до 5 баллов означает,что работа имеет принципиальные недостатки (не завершена). Количество баллов означает степень незавершённости.

Работа получает оценку ”неудовлетворительно” и должна быть переделана.

Работа может быть принята при оценке не менее 6 баллов, что соответствует ”удовлетворительно” и означает, что студент не провёл качественное тестирование и не смог представить развёрнутых выводов по работе, допустил небрежность в представленном отчёте.

7 баллов (”хорошо”) означает, что тестирование проведено и выводы сделаны, но есть замечания к реализации алгоритма и качеству кодирования.

8 баллов (”отлично”) ставится за работы, выполненные без существенных замечаний как по исполнению, так и по оформлению отчёта.

9 баллов может получить студент, предложивший новые идеи для реализации заданного алгоритма, выполнивший серьёзное его исследование и получивший новые результаты. Работа оценивается на ”отлично” и студент получает бонус в виде освобождения от опроса на диффер. зачёт/экзамене.

Текущий контроль включает в себя выполнение, сдачу в срок отчётов и их защиту по всем контрольным, лабораторным и курсовой работам, по результатам которой студент получает допуск на диффер. зачет / экзамен.

на практических занятиях

Текущий контроль включает в себя контроль посещаемости (не менее 80% занятий), по результатам которого студент получает допуск на диффер. зачет / экзамен.

В ходе проведения практических занятий студенты привлекаются к как можно более активному участию в дискуссиях, решении задач, обсуждению и т. д. При этом активность студентов учитывается преподавателем, как один из способов текущего контроля на практических занятиях.

Самостоятельная работа студентов

Контроль самостоятельной работы студентов осуществляется на лекционных, лабораторных и практических занятиях студентов по методикам, описанным выше.

Индивидуальное домашнее задание (домашняя контрольная работа)

Результаты контрольной работы оформляются в виде отчёта, пересылаемого на электронную почту преподавателя. Предоставление этих результатов - непременное условие для допуска студента к дифференциальному зачёту /экзамену. Результаты контрольных работ оцениваются по методике, указанной выше для лабораторных работ.

Выполнение курсовой работы

Текущий контроль при выполнении курсовой работы осуществляется в соответствии с методическими указаниями по курсовом проектированию и заданием на курсовую работу.

Защита курсового проекта (работы) осуществляется в соответствии с требованиями «Положения о промежуточной аттестации».

Критерии оценивания курсовой работы:

Оценку ”отлично” получает работа, выполненная в полном соответствии с установленными требованиями:

- имеется развёрнутая постановка задачи на языке теории множеств со ссылкой не менее чем на два первоисточника;

- сделан обоснованный выбор структур данных для представления основного и промежуточных результатов;

- приложенная к работе программа компилируется и действительно даёт объявленный результат;

- тестирование проведено достаточно полно и действительно доказывает работоспособность программы;

- сделаны выводы об оптимальности реализации алгоритма и его временной сложности.

При наличии серьёзных замечаний по одному - двум из указанных пунк-

тов ставится оценка ”хорошо”, по трём и более - ”удовлетворительно”.

Оценка ”неудовлетворительно” ставится, если приложенная к работе программа не компилируется, не решает поставленную задачу, не даёт заявленный в отчёте о работе результат.

Отчётные документы по лабораторным и курсовым работам сдаются преподавателю на занятии или присылаются по почте в электронном виде. Тексты программ, включаемые в отчёты или прилагаемые к ним, должны быть пригодны для компиляции.

7 Описание информационных технологий и материально-технической базы

Тип занятий	Тип помещения	Требования к помещению	Требования к программному обеспечению
Лекция	Лекционная аудитория	Количество посадочных мест – в соответствии с контингентом; рабочее место преподавателя, маркерная доска, экран, проектор с разрешение Full HD, ноутбук	1) ОС Linux; 2) Open Office 3.1 или выше; 3) Система программирования Code Bloks 20.3 с поддержкой компилятора C++ стандарта не ниже C++17
Лабораторные работы	Лаборатория	Оснащено компьютерами с подключением к сети Интернет. Количество посадочных мест – в соответствии с контингентом, рабочее место преподавателя.	1) Ос linux; 2) Open Office 3.1 или выше; 3) Система программирования Code Bloks 20 с поддержкой компилятора C++ стандарта не ниже 17
Практические занятия	Аудитория	Оснащено компьютерами с подключением к сети Интернет. Количество посадочных мест – в соответствии с контингентом, рабочее место преподавателя.	1) ОС Linux; 2) Open Office 3.1 или выше; 3) Система программирования Code Bloks 20.3 с поддержкой компилятора C++ стандарта не ниже C++17
Самостоятельная работа	Помещение для самостоятельной работа	Оснащено компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду университета.	1) ОС Linux; 2) Open Office 3.1 или выше; 3) Система программирования Code Bloks 20.3 с поддержкой компилятора C++ стандарта не ниже C++17

8 Адаптация рабочей программы для лиц с ОВЗ

Адаптированная программа разрабатывается при наличии заявления со стороны обучающегося (родителей, законных представителей) и медицинских показаний (рекомендациями психолого-медико-педагогической комиссии). Для инвалидов адаптированная образовательная программа разрабатывается в соответствии с индивидуальной программой реабилитации.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ п/п	Дата	Изменение	Дата и номер протокола заседания УМК	Автор	Начальник ОМОЛА