Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Структурное подразделение «Институт информационных технологий и анализа данных (121)»

УТВЕРЖДЕНА:
на заседании совета института ИТиАД им. Е.И. Попова
Протокол №8 от 16 февраля 2026 г.

Рабочая программа дисциплины

«ПРОГРАММИРОВАНИЕ»

Направление: 09.03.01 Информатика и вычислительная техника

Вычислительные машины, комплексы, системы и сети

Квалификация: Бакалавр

Форма обучения: очная

Документ подписан простой электронной подписью Составитель программы: Столбов Александр Борисович Дата подписания: 16.06.2026	Документ подписан простой электронной подписью Утвердил: Говорков Алексей Сергеевич Дата подписания: 17.06.2026	Документ подписан простой электронной подписью Согласовал: Аношко Алексей Федорович Дата подписания: 17.06.2026

Год набора – 2026

Иркутск, 2026 г.

1 Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы

1.1 Дисциплина «Программирование» обеспечивает формирование следующих компетенций с учётом индикаторов их достижения

Код, наименование компетенции	Код индикатора компетенции
ОПК ОС-2 Способность применять современные информационные технологии при решении задач профессиональной деятельности	ОПК ОС-2.3, ОПК ОС-2.5
ОПК ОС-8 Способность разрабатывать алгоритмы и программы, пригодные для практического применения	ОПК ОС-8.1, ОПК ОС-8.2

1.2 В результате освоения дисциплины у обучающихся должны быть сформированы

Код индикатора	Содержание индикатора	Результат обучения
ОПК ОС-2.3	Применяет различные способы визуализации программных алгоритмов	**Знать** основные понятия, конструкции и структуры алгоритмического языка программирования; принципы и стандарты построения визуализаций алгоритмов, в т.ч. с использованием блок-схем и диаграмм. **Уметь** составлять алгоритмы программ; создавать блок-схемы и диаграммы для представления программных алгоритмов; интерпретировать и анализировать визуализации алгоритмов, выявляя потенциальные проблемы и возможности для оптимизации; использовать современные среды разработки; оформлять полученные рабочие результаты в форме отчёта. **Владеть** компетенцией в применении различных инструментов и программ для визуализации алгоритмов; способностью интегрировать визуализации алгоритмов в документацию и презентации, обеспечивая ясность и доступность информации.
ОПК ОС-2.5	Умеет составлять алгоритмы программ и реализовывать их на языках программирования в среде разработки	**Знать** преимущества и особенности программирования на языке высокого уровня; этапы решения задачи с помощью программы; основные понятия, конструкции и структуры алгоритмического языка

		программирования. **Уметь** составлять алгоритмы, писать и отлаживать коды на языке программирования высокого уровня, тестировать работоспособность программы; использовать современные среды разработки; формлять полученные рабочие результаты в форме отчёта. **Владеть** инструментальными средствами, методами и навыками разработки программного обеспечения с использованием языка программирования высокого уровня.
ОПК ОС-8.1	Способность осваивать и разрабатывать программное обеспечение общего назначения для решения практических задач /способность осваивать одну из сред программирования для решения практических задач на языке программирования (использование развилок, циклов)	**Знать** о различных парадигмах программирования и современном уровне развития языков и технологий программирования; основные понятия, конструкции и структуры алгоритмического языка программирования. **Уметь** применять при решении алгоритмических задач различные виды разветвляющихся и циклических процессов; выбирать и использовать различные типы данных и алгоритмы для решения практических задач; использовать для разработки и отладки программ современные интегрированные среды разработки, в т.ч. настольные (DevC++, Visual Studio); оформить рабочий результат в виде научно-технического отчета или презентации. **Владеть** навыками написания и отладки программ на высокоуровневом языке программирования в интегрированной среде разработки; навыками анализа программных пакетов и библиотеки для разработки программного обеспечения.
ОПК ОС-8.2	Способность осваивать и разрабатывать программное обеспечение для решения практических задач профессиональной	**Знать** о различных парадигмах программирования и современном уровне развития языков и технологий программирования; назначение, устройство и свойства

	деятельности/способность осваивать одну из сред программирования для решения практических задач на языке программирования (использование модульного программирования, создание приложений с графическим интерфейсом), применять программные средства для решения задач вычислительной математики, математической логики и теории алгоритмов	основных абстрактных структур данных: список, очередь, стэк, дерево, хеш-таблица; эффективные алгоритмы поиска и сортировки; принципы создания пользовательских структур данных; принципы и методы функциональной декомпозиции и разработки программ модульной структуры; состав и возможнсоти современных алгоритмических программных библиотек; основы разработки приложений с графическим интерфейсом. **Уметь** применять при решении алгоритмических задач типичные алгоритмы и структуры данных; создавать пользовательские структуры данных; разрабатывать схему иерархии, алгоритмы и программные коды программы модульной структуры; выбирать и использовать различные пакеты и библиотеки языка программирования высокого уровня для решения практических задач, в т.ч. средства для создания приложений с графическим интерфейсом; использовать для разработки и отладки программ современные интегрированные среды разработки, в т.ч. настольные (DevC++, Visual Studio); оформить рабочий результат в виде научно-технического отчета или презентации. **Владеть** навыками написания и отладки программ на высокоуровневом языке программирования в интегрированной среде разработки; навыками анализа программных пакетов и библиотеки для разработки программного обеспечения.

2 Место дисциплины в структуре ООП

Изучение дисциплины «Программирование» базируется на результатах освоения следующих дисциплин/практик: «Математика», «Программирование», «Дискретная математика»

Дисциплина является предшествующей для дисциплин/практик: «Объектно-ориентированное программирование», «Вычислительная математика», «Инженерная графика»

3 Объем дисциплины

Объем дисциплины составляет – 9 ЗЕТ

Вид учебной работы	**Трудоемкость в академических часах** (Один академический час соответствует 45 минутам астрономического часа)		
	Всего	Семестр № 1	Семестр № 2
Общая трудоемкость дисциплины	324	144	180
Аудиторные занятия, в том числе:	160	80	80
лекции	64	32	32
лабораторные работы	64	32	32
практические/семинарские занятия	32	16	16
Самостоятельная работа (в т.ч. курсовое проектирование)	128	64	64
Трудоемкость промежуточной аттестации	36	0	36
Вид промежуточной аттестации (итогового контроля по дисциплине)	Зачет, Экзамен	Зачет	Экзамен

4 Структура и содержание дисциплины

4.1 Сводные данные по содержанию дисциплины

Семестр № 1

№ п/п	**Наименование раздела и темы дисциплины**	**Виды контактной работы**						СРС		**Форма текущего контроля**
		Лекции		ЛР		ПЗ(СЕМ)				
		№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	
1	2	3	4	5	6	7	8	9	10	11
1	Этапы разработки программ и структурное программирование	1	6	1	6	1	2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	16	Отчет по лабораторной работе
2	Разработка консольного приложения	2	8	1	10	2, 3	4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	17	Отчет по лабораторной работе
3	Составные типы данных	3	10	2	8	4, 5, 6	6	1, 2, 3, 4, 6, 7, 8, 9, 10	16	Отчет по лабораторной работе

4	Модульное программирование	4	8	3	8	7	4	1, 2, 3, 4, 6, 7, 8, 9, 10	15	Творческое задание
	Промежуточная аттестация									Зачет
	Всего		32		32		16		64	

Семестр **№ 2**

№ п/п	Наименование раздела и темы дисциплины	Виды контактной работы						СРС		Форма текущего контроля
		Лекции		ЛР		ПЗ(СЕМ)				
		№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	№	Кол. Час.	
1	2	3	4	5	6	7	8	9	10	11
1	Абстрактные структуры данных	1	10	1	10	1, 2	4	1, 2, 3, 4, 5, 6, 7, 8, 9	15	Отчет по лабораторной работе
2	Введение в библиотеки программирования (на примере STL)	2	6	2	6	3	2	1, 2, 3, 4, 5, 6, 7, 8, 9	15	Отчет по лабораторной работе
3	Классические алгоритмы в программировании	3	8	3	8	4, 5, 6	6	1, 2, 3, 4, 5, 6, 7, 8, 9	18	Отчет по лабораторной работе
4	Разработка библиотеки функций	4	8	4	8	7, 8	4	1, 2, 3, 4, 5, 6, 7, 8	16	Отчет по лабораторной работе
	Промежуточная аттестация								36	Экзамен
	Всего		32		32		16		100	

4.2 **Краткое содержание разделов и тем занятий**

Семестр № 1

№	Тема	Краткое содержание
1	Этапы разработки программ и структурное программирование	Основные определения, архитектуры ЭВМ, приниципы выполнения программ, ЯНУ и ЯВУ, трансляция программ, консольное приложение, инструменты разработчика. Этапы разработки программ. Структурное программирование. Блок-схема алгоритма. Словесное описание алгоритма. Алфавит, лексемы. Переменная: объявление и инициализация. Типы данных. Константы. Структура программы: выражения, вызов функций, операторы и операции.
2	Разработка консольного приложения	Операторы цикла, их особенности и различия. Итерационные циклические процессы. Фундаментальные типы данных и их представления в памяти. Особенности операций со

		знаковыми и беззнаковыми целыми. Стандарт IEEE 754. Кодирование символов. Преобразование типов. Обработка исключений. Отладка программы.Потоки ввода-вывода. Стандартные потоки ввода-вывода. Строковые потоки.
3	Составные типы данных	Виды памяти. Динамическая память. Стек. Указатели. Ссылки. Массивы и указатели. Динамические массивы и память. Алгоритм сортировки пузырьком. Многомерные массивы. Строки и кодировки Unicode. Функции для работы со строками. Пользовательские структуры. Пользовательские типы данных С плюс плюс: структура, перечисление, объединение.
4	Модульное программирование	Функциональная декомпозиция. Функции в С плюс плюс. Принцип передачи параметров. Кадр стека функции. Рекурсия. Область действия и область видимости переменных. Передача параметров функций по указателю и ссылке. Массивы и функции. Директивы препроцессора. Сборка программы на С плюс плюс. Особенности глобальных переменных.

Семестр **№ 2**

№	Тема	Краткое содержание
1	Абстрактные структуры данных	Стек, очередь, списки, бинарные деревья поиска, хэш-таблица. Введение в оценку сложности алгоритмов. Указатели на функцию.Перегрузка функций. Перегрузка операций. Функции в структурах. Разбиение библиотеки на модули. Управление связанными структурами данных.
2	Введение в библиотеки программирования (на примере STL)	Основы обобщенного программирования с использованием шаблонов С плюс плюс. Стандартная библиотека шаблонов STL: контейнеры, итераторы, алгоритмы.
3	Классические алгоритмы в программировании	Обзор типов алгоритмов (жадные, разделяй и влавствуй и т.п.). Алгоритмы сортировки и поиска, алгоритмы на графах, динамическое программирование, алгоритмы с побитовыми операциями и другие алгоритмы.
4	Разработка библиотеки функций	Введение в тестирование программ. Формирование спецификации библиотеки. Машина Тьюринга. Парадигмы программирования: структурное, функциональное, объектно-ориентированное. Идиомы и приёмы программирования. Основные типы языков программирования. Тенденции развития средств и систем для проектирования программ.

4.3 Перечень лабораторных работ

Семестр № 1

№	Наименование лабораторной работы	Кол-во академических часов
1	Разработка консольного приложения для вычисления значений функции	10
1	Введение в этапы разработки программ	6
2	Массивы и пользовательские структуры	8
3	Модульное программирование	8

Семестр **№ 2**

№	Наименование лабораторной работы	Кол-во академических часов
1	Абстрактные структуры данных	10
2	Введение в библиотеки программирования	6
3	Классические алгоритмы	8
4	Семантическое аннотирование библиотек функций	8

4.4 Перечень практических занятий

Семестр № 1

№	Темы практических (семинарских) занятий	Кол-во академических часов
1	Создание программы «Калькулятор» и оформление отчёта	2
2	Потоки ввода-вывода. Последовательность символов. Проверка корректности ввода.	2
3	Вычисление функции, заданной в виде суммы бесконечного ряда	2
4	Работа с указателями и массивами	2
5	Пользовательские структуры данных	2
6	Обработка строковых данных	2
7	Функции и рекурсия	4

Семестр **№ 2**

№	Темы практических (семинарских) занятий	Кол-во академических часов
1	Списки и бинарные деревья поиска	2
2	Использование функций - продвинутый уровень	2
3	Основные элементы библиотеки STL	2
4	Алгоритмы на деревьях	2
5	Алгоритмы сортировки	2
6	Популярные алгоритмы разного типа	2
7	Идиомы программирования	2
8	Приложения с графическим пользовательским интерфейсом	2

4.5 Самостоятельная работа

Семестр № 1

№	Вид СРС	Кол-во академических часов
1	Ведение терминологического словаря	4
2	Выполнение письменных творческих работ (писем, докладов, сообщений, ЭССЕ)	12
3	Выполнение тренировочных и обучающих тестов в дистанционном режиме	4
4	Оформление отчетов по лабораторным и практическим работам	10
5	Подготовка к контрольным работам	2
6	Подготовка к практическим занятиям	4
7	Подготовка к практическим занятиям (лабораторным работам)	8
8	Подготовка к сдаче и защите отчетов	4
9	Проработка разделов теоретического материала	4
10	Прохождение массового открытого онлайн-курса	12

Семестр **№ 2**

№	Вид СРС	Кол-во академических часов
1	Ведение терминологического словаря	4
2	Выполнение письменных творческих работ (писем, докладов, сообщений, ЭССЕ)	12
3	Выполнение тренировочных и обучающих тестов в дистанционном режиме	4
4	Оформление отчетов по лабораторным и практическим работам	12
5	Подготовка к практическим занятиям	4
6	Подготовка к практическим занятиям (лабораторным работам)	8
7	Подготовка к сдаче и защите отчетов	4
8	Проработка разделов теоретического материала	4
9	Прохождение массового открытого онлайн-курса	12

В ходе проведения занятий по дисциплине используются следующие интерактивные методы обучения: В ходе проведения лекций, практических и лабораторных работ используются следующие интерактивные методы обучения: интерактивная демонстрация способа решения типовой проблемы; работа в команде (групповая разработка алгоритмов и их сравнение по различным критериям); взаимная проверка и тестирование алгоритмов; групповое обсуждение эффективности алгоритмов; проектный метод, метод «мозгового штурма» (постановка проблемы, генерация идей, оценка, отбор идеи и реализация алгоритма).

5 Перечень учебно-методического обеспечения дисциплины

5.1 Методические указания для обучающихся по освоению дисциплины

5.1.1 Методические указания для обучающихся по практическим занятиям

Электронный курс по дисциплине «Программирование (1 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=4307 (дата обращения: 20.02.2025).

Электронный курс по дисциплине «Программирование (2 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=6744 (дата обращения: 20.02.2025).

5.1.2 Методические указания для обучающихся по лабораторным работам:

Электронный курс по дисциплине «Программирование (1 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=4307 (дата обращения: 20.02.2025).

Электронный курс по дисциплине «Программирование (2 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=6744 (дата обращения: 20.02.2025).

5.1.3 Методические указания для обучающихся по самостоятельной работе:

Электронный курс по дисциплине «Программирование (1 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=4307 (дата обращения: 20.02.2025).

Электронный курс по дисциплине «Программирование (2 семестр)». Электрон. версия. URL: https://el.istu.edu/course/view.php?id=6744 (дата обращения: 20.02.2025).

6 Фонд оценочных средств для контроля текущей успеваемости и проведения промежуточной аттестации по дисциплине

6.1 Оценочные средства для проведения текущего контроля

6.1.1 семестр 1 | Творческое задание

Описание процедуры.

Дневник программиста является формой творческой работ, в которой обучающийся кратко описывает процесс выполнения самостоятельных занятий и рефлексию по ним.

Тема (раздел)
Дневник программиста заполняется по всем изучаемым темам.

Описание процедуры: Обучающийся отражает в дневнике программиста самостоятельную работу по лабораторной и практике, изучении MOOK, спортивном программировании, контрольной, лекции и дополнительной самостоятельной деятельности. В дневник также можно включить и занятия по другим дисциплинам, связанным с программированием (например, дискретная математика и информатика). На оценку дневника программиста влияют следующие факторы: количество и полнота описания активностей, регулярность записей.
Предлагаются следующие поля дневника для заполнения: общая характеристика записи (дата, продолжительность занятия, тема занятия, тип изучаемого материала), планирование работ (цели и задачи занятия, используемые средства), описание процесса выполнения работ (список допущенных ошибок, способы разрешения ошибок, ресурсы, которые помогли исправить ошибку), анализ результатов занятия, характеристика занятия в свободной форме (рефлексия, выводы, пожелания).

Критерии оценивания.

Отлично.
1. Записи в дневнике программиста ведутся регулярно (не менее двух раз в неделю).
2. В дневнике программиста отражено не менее 80% самостоятельной работы.
3. В дневнике программиста активность описана достоверно и полно.

Хорошо
1. Записи в дневнике программиста ведутся регулярно (не менее одного раза в неделю).
2. В дневнике программиста отражено не менее 50% самостоятельной работы.
3. В дневнике программиста активность описана достоверно, но частично.

Удовлетворительно
1. Записи в дневнике программиста ведутся регулярно (не менее одного раза в неделю).
2. В дневнике программиста отражено не менее 25% самостоятельной работы.
3. В дневнике программиста активность описана достоверно, но кратко.

Неудовлетворительно
1. Дневник программиста не ведётся или ведётся нерегулярно.
2. В дневнике программиста отражено менее 25% самостоятельной работы.
3. В дневнике программиста активность описана не достоверно.

6.1.2 семестр 1 | Отчет по лабораторной работе

Описание процедуры.

Все отчёты сохраняются на электронном ресурсе https://el.istu.edu/ . Обеспечена возможность удаленной обратной связи со студентом (комментарии к работе, чат, видеосвязь).
Все отчёты имеют схожую структуру и должны содержать:
1. Условие задачи.
2. Математическую модель задачи (при наличии).
3. Таблицу внешних спецификаций.
4. Описание основных алгоритмов в словесной форме или в форме блок схем.
5. Таблицу тестов.
6. Проверку правильности алгоритма с помощью таблицы.
7. Код программы на языке С++ или по согласованию с преподавателем на другом языке высокого уровня.

Начиная с 4-ой лабораторной, в отчёт также добавляется:
8. Схема иерархии функций.
9. Таблицы спецификаций функций
10. Таблицы спецификаций пользовательских структур (при наличии)

Тема (раздел)
Каждой теме дисциплины соответствует отдельная лабораторная работа.

Описание процедуры:
В ходе выполнения лабораторной работы студент должен применить теоретические знания по темам, изложенным в курсе «Программирование», а также приобрести практические навыки проектирования, алгоритмизации и кодирования типовых программ.

Для успешной сдачи лабораторной работы студенту необходимо защитить подготовленный отчет. В ходе защиты отчета студенту необходимо дать краткое изложение основных результатов, полученных в ходе выполнения лабораторной работы, устно ответить на теоретические вопросы по теме лабораторной работы, а также продемонстрировать умение ориентироваться в написанном программном коде. Защита отчета является необходимым условием для сдачи лабораторной работы

Контрольные вопросы к лабораторным работам.
Лабораторная работа №1

1. Какие целые типы данных Вы знаете?
2. Какие вещественные типы данных знаете?
3. Для чего служит тип bool?
4. Может ли в условном блоке отсутствовать одна из ветвей?
5. Как объединить несколько операторов в условном блоке?
6. Когда используется оператор выбора?

Лабораторная работа №2

7. Сколько операторов повторения вы знаете? Перечислите их.
8. Оператор For - общий вид, реализуемый алгоритм, примеры.
9. Оператор While - общий вид, реализуемый алгоритм, примеры.
10. Чем цикл с предусловием отличается от цикла с постусловием? Почему таких циклов два вида?
11. Что такое поток ввода/вывода?
12. Особенности стандартных потоков ввода/вывода?
13. Что такое строковый поток?
14. Как представить вещественное число в научном формате, как вывести это формат в поток вывода.

Лабораторная работа №3

1. Что такое массив? Примеры массивов.
2. Основные понятия: массив, элемент массива, индекс элемента, количество элементов.
3. Размерность массива.
4. Описание массивов на Паскале.
5. В каком порядке размещаются элементы массивов в памяти?
6. Каким может быть тип элементов массива?
7. Что такое тип индекса и каким он может быть?
8. Сколько байт занимает литерная переменная?
9. Как определить код ASCII литерной переменной?
10. Какая функция служит для преобразования строчной буквы в прописную?
11. Как получить следующий/предыдущий символ в таблице ASCII?
12. Два способа представления текста в C++.
13. Тип данных string: описание переменных, атрибут длины, ввод, вывод и операция присваивания. Примеры.
14. Сравнение строк.
15. Стандартные процедуры перевода строки в число и числа в строку.
16. Чем отличаются статические и динамические величины?
17. Какая память называется динамически распределяемой?
18. Что такое указатель?
19. Какие виды указателей вам известны?
20. Как определяется адрес переменной?
21. Что такое "разыменование"?

22. Приведите примеры объявления указателей.
23. Как выделить память под динамическую переменную? Как освободить память от динамической переменной?
24. В каком случае возможно присваивание указателей?
25. Какие ситуации приводят к возникновению в динамически распределяемой памяти "мусора"?

Лабораторная работа №4

1. Какова структура описания функции?
2. В чем состоит отличие процедуры от функции?
3. Что такое область действия идентификаторов?
4. Каковы основные правила определения области действия для идентификаторов функций?
5. Какие параметры называются формальными и какие – фактическими?
6. По каким признакам различаются параметры?
7. Какие способы передачи параметров реализованы в C++?
8. Каковы правила передачи параметров-значений?
9. Каковы правила передачи параметров-ссылок?
10. В чем особенности бестиповых параметров?
11. Какое определение называется рекурсивным? Приведите собственные примеры рекурсивных определений.
12. Какой вспомогательный алгоритм (подпрограмма) называются рекурсивными?
13. Что такое граничное условие, и каково его назначение в рекурсивной подпрограмме?
14. Что такое рекурсивный спуск?
15. Что такое рекурсивный подъём?
16. Что называется текущим уровнем рекурсии?
17. Что такое глубина рекурсии? Чему равна глубина рекурсии в приведённых выше примерах?
18. На каком этапе выполнения рекурсивной подпрограммы могут выполняться её операторы?
19. В чем преимущества и недостатки рекурсивных процедур по сравнению с итеративными?
20. Что такое стек вызова функций?

Критерии оценивания.

Оценка "Отлично". Работа выполнена полностью.
2. Работа оформлена в соответствии с требованиями.
3.Тестовые наборы являются полными.
4. Программа имеет пользовательский интерфейс с проверкой корректности вводимых данных.
5. Программа работоспособна
6. В случае рекомендации правки программы, она выполнена.
7. Даны ответы на контрольные вопросы по теме работы

Оценка "Хорошо". 1. Работа выполнена полностью.
2. Работа оформлена в соответствии с требованиями.
3. Тестовые наборы не являются полными.
4. Обучающийся иногда путается в ответах на вопросы.

Оценка "Удовлетворительно".
1. Работа выполнена, но рассматривает частный случай.

2. Отчет оформлен с недостатками.
3. Программа имеет небрежно оформленный интерфейс.
4. Программа не проходит тестовые наборы преподавателя.
5. Обучающийся путается в ответах.
Оценка "Неудовлетворительно".
1.Работа не выполнена.
2. Не даны ответы на контрольные вопросы.

6.1.3 семестр 2 | Отчет по лабораторной работе

Описание процедуры.

В ходе выполнения лабораторной работы студент должен применить теоретические знания по темам, изложенным в курсе «Программирование», а также приобрести практические навыки проектирования, алгоритмизации и кодирования типовых программ.

Для успешной сдачи лабораторной работы студенту необходимо защитить подготовленный отчет. В ходе защиты отчета студенту необходимо дать краткое изложение основных результатов, полученных в ходе выполнения лабораторной работы, устно ответить на теоретические вопросы по теме лабораторной работы, а также продемонстрировать умение ориентироваться в написанном программном коде. Защита отчета является необходимым условием для сдачи лабораторной работы

Лабораторная работа №5
1. Что такое абстрактные структуры данных?
2. Что такое динамические структуры данных?
3. Что понимают под "связанным списком"?
4. Как классифицируют связанные списки?
5. Какие основные действия над списками и компонентами списков обычно реализуют?
6. Как описывается список?
7. Что понимают под "двоичным деревом"?
8. Как классифицируют двоичные деревья?
9. Какие основные действия над двоичным деревом поиска обычно реализуют?
10. Как описывается двоичное дерево поиска?
11. В чем особенность сбалансированных двоичных деревьев?

Лабораторная работа №6
1. Что такое очередь?
2. Что такое стек?
3. Что такое хеш-таблица?
4. Что такое множество?
5. В чем заключается идея обобщенного программирования?
6. Что такое контейнер и итератор?
7. Какие абстрактные структуры данных стандартной библиотеки C++ Вы знаете?

Лабораторная работа №7
1. Дайте определение терминам «алгоритм» и «алгоритмический процесс».
2. Дайте характеристику понятия «сложность алгоритма».
3. Поясните смысл оценки сложности алгоритмов в терминах O – нотации.
4. Оцените сложность разработанного алгоритма.

Лабораторная работа №8

1. Какие идиомы переменных и выражений в структурном программировании Вы знаете?
2. Какие идиомы арифметических и логических операций Вы знаете?
3. Какие идиомы для работы со строками Вы знаете?
4. Какие идиомы для работы с массивами фиксированной длины Вы знаете?
5. Какие идиомы для работы с массивами переменной длины Вы знаете?
6. Какие идиомы для работы с картежами и словарями Вы знаете?
7. Какие идиомы для работы с функциями в структурном программировании Вы знаете?

Критерии оценивания.

Оценка "Отлично". Работа выполнена полностью.
2. Работа оформлена в соответствии с требованиями.
3.Тестовые наборы являются полными.
4. Программа имеет пользовательский интерфейс с проверкой корректности вводимых данных.
5. Программа работоспособна
6. В случае рекомендации правки программы, она выполнена.
7. Даны ответы на контрольные вопросы по теме работы

Оценка "Хорошо". 1. Работа выполнена полностью.
2. Работа оформлена в соответствии с требованиями.
3. Тестовые наборы не являются полными.
4. Обучающийся иногда путается в ответах на вопросы.
Оценка "Удовлетворительно".
1. Работа выполнена, но рассматривает частный случай.
2. Отчет оформлен с недостатками.
3. Программа имеет небрежно оформленный интерфейс.
4. Программа не проходит тестовые наборы преподавателя.
5. Обучающийся путается в ответах.
Оценка "Неудовлетворительно".
1.Работа не выполнена.
2. Не даны ответы на контрольные вопросы.

6.2 Оценочные средства для проведения промежуточной аттестации

6.2.1 Критерии и средства (методы) оценивания индикаторов достижения компетенции в рамках промежуточной аттестации

Индикатор достижения компетенции	**Критерии оценивания**	**Средства (методы) оценивания промежуточной аттестации**
ОПК ОС-2.3	2. Неудовлетворительно (уровень не сформирован). Студентом задание не решено. 3. Удовлетворительно (пороговый уровень). Студентом задание решено с подсказками преподавателя. При этом	Выполнение практических заданий, оформление отчёта по лабораторной

	задание понято правильно, в логическом рассуждении нет существенных ошибок, но задание решено не полностью, или в общем виде, или не верно оформлен отчет. 4. Хорошо (базовый уровень). Студентом задание решено с подсказкой преподавателя. При этом составлен правильный алгоритм решения задания, в логическом рассуждении и решении нет существенных ошибок; но задание решено нерациональным способом или допущено не более двух несущественных ошибок, получен верный ответ. Верно оформлен отчет. 5. Отлично (повышенный уровень). Студентом задание решено самостоятельно. При этом составлен правильный алгоритм решения задания, в логических рассуждениях, в выборе формул и решении нет ошибок, получен верный ответ, задание решено рациональным способом. Верно оформлен отчет. Выполнено дополнительное задание.	работе, устное собеседование по теоретическим вопросам.
ОПК ОС-2.5	2. Неудовлетворительно (уровень не сформирован). Студентом задание не решено. 3. Удовлетворительно (пороговый уровень). Студентом задание решено с подсказками преподавателя. При этом задание понято правильно, в логическом рассуждении нет существенных ошибок, но задание решено не полностью, или в общем виде, или не верно оформлен отчет. 4. Хорошо (базовый уровень). Студентом задание решено с подсказкой преподавателя. При этом составлен правильный алгоритм решения задания, в логическом рассуждении и решении нет существенных ошибок; но задание решено нерациональным способом или допущено не более двух несущественных ошибок, получен верный ответ. Верно оформлен отчет. 5. Отлично (повышенный уровень). Студентом задание решено самостоятельно. При этом составлен	Выполнение практических заданий, оформление отчёта по лабораторной работе, устное собеседование по теоретическим вопросам.

	правильный алгоритм решения задания, в логических рассуждениях, в выборе формул и решении нет ошибок, получен верный ответ, задание решено рациональным способом. Верно оформлен отчет. Выполнено дополнительное задание.	
ОПК ОС-8.1	2. Неудовлетворительно (уровень не сформирован). Студентом задание не решено. 3. Удовлетворительно (пороговый уровень). Студентом задание решено с подсказками преподавателя. При этом задание понято правильно, в логическом рассуждении нет существенных ошибок, но задание решено не полностью, или в общем виде, или не верно оформлен отчет. 4. Хорошо (базовый уровень). Студентом задание решено с подсказкой преподавателя. При этом составлен правильный алгоритм решения задания, в логическом рассуждении и решении нет существенных ошибок; но задание решено нерациональным способом или допущено не более двух несущественных ошибок, получен верный ответ. Верно оформлен отчет. 5. Отлично (повышенный уровень). Студентом задание решено самостоятельно. При этом составлен правильный алгоритм решения задания, в логических рассуждениях, в выборе формул и решении нет ошибок, получен верный ответ, задание решено рациональным способом. Верно оформлен отчет. Выполнено дополнительное задание.	Выполнение практических заданий, оформление отчёта по лабораторной работе, устное собеседование и тестирование по теоретическим вопросам.
ОПК ОС-8.2	2. Неудовлетворительно (уровень не сформирован). Студентом задание не решено. 3. Удовлетворительно (пороговый уровень). Студентом задание решено с подсказками преподавателя. При этом задание понято правильно, в логическом рассуждении нет существенных ошибок, но задание решено не полностью, или в общем виде, или не верно оформлен отчет. 4. Хорошо (базовый уровень).	Выполнение практических заданий, оформление отчёта по лабораторной работе, устное собеседование и тестирование по теоретическим вопросам.

	Студентом задание решено с подсказкой преподавателя. При этом составлен правильный алгоритм решения задания, в логическом рассуждении и решении нет существенных ошибок; но задание решено нерациональным способом или допущено не более двух несущественных ошибок, получен верный ответ. Верно оформлен отчет. 5. Отлично (повышенный уровень). Студентом задание решено самостоятельно. При этом составлен правильный алгоритм решения задания, в логических рассуждениях, в выборе формул и решении нет ошибок, получен верный ответ, задание решено рациональным способом. Верно оформлен отчет. Выполнено дополнительное задание.	

6.2.2 Типовые оценочные средства промежуточной аттестации

6.2.2.1 Семестр 1, Типовые оценочные средства для проведения зачета по дисциплине

6.2.2.1.1 Описание процедуры

Зачёт за первый семестр ставятся по результатам выполнения и защиты лабораторных работ 1-4, посещения лекций, ведения конспекта лекций, прохождения МООК, ведения дневника программиста, знания наизусть терминологического словаря, выполнения тестов с теоретическим вопросами (1-40).
Терминологический словарь - обучающийся воспроизводит наизусть терминологический словарь.
Обучающийся выполняет МООК по всем изучаемым темам.

Пример задания:

Список теоретических вопросов к зачёту.
1. Понятия системы программирования, языка программирования, программы. Архитектуры ЭВМ. Адресное пространство.
2. Алгоритм. Определение и свойства. Способы описания алгоритмов. Блок-схемы: основные элементы и принципы организации.
3. Уровни языков программирования. Трансляция программ. Базовые принципы.
4. Структурное программирование. Виды вычислительных процессов. Утверждение структурного программирования. Идеи Дейкстры.
5. Назначение таблицы внешних спецификаций. Назначение тестирования и отладки программы. Таблица тестов.
6. Структура программы C++. Идентификаторы в C++.
7. Описание переменных в программе. Фундаментальные типы данных C++.
8. Виды констант программы.

9. Выражения. Приоритет операций.
10. Арифметические операции, операции отношения, логические операции.
11. Инкремент и декремент. Особенности постфиксной и префиксной записи.
12. Операция запятая, размер, тернарная операция.
13. Операторы следования и перехода (передачи управления).
14. Условный оператор и оператор выбора (switch).
15. Операторы цикла, их особенности и различия
16. Строки. Основные функции для работы со строками
17. Потоки ввода-вывода. Базовые принципы. Текстовые и двоичные потоки. Стандартные потоки ввода-вывода. Примеры C++.
18. Стандартные потоки ввода-вывода. Извлечение из потока и включение в поток. Манипуляторы потоков. Строковые потоки C++.
19. Статические массивы: одномерные и многомерные
20. Метод сортировки пузырьком
21. Виды памяти: статическая, автоматическая, динамическая память. Пример схемы распределения.
22. Автоматическая память. Подробно про жизненный цикл её элементов. Стек.
23. Динамическая память. Подробно про жизненный цикл её элементов. Особенности new и delete. Сборка мусора. Базовые принципы.
24. Указатели. Определение, размер, особенности (нулевые, void*)
25. Указатели. 4 базовые операции. Арифметика указателей.
26. Ссылки.
27. Динамические массивы: одномерные
28. Динамические массивы: многомерные
29. Функциональная декомпозиция программы. Схема иерархии функций программы. Таблицы спецификаций функций.
30. Объявление и определение функции C++. Параметры функции.
31. Вызов функции. Возврат значений из функции. Кадр стека функции.
32. Передача параметров в функцию с использованием указателей и ссылок.
33. Способы передачи массивов в функцию на примере C++. Особенности.
34. Рекурсия.
35. Разбиение программы на модули, примеры на C++.
36. Область действия переменной и область видимости переменной
37. Глобальные переменные: объявление, обращения, использование в нескольких модулях.
38. Препроцессор, примеры директив препроцессора
39. Алгоритм сборки программы, примеры на C++
40. Принцип выполнения программ с примером машины Тьюринга.

Практическое задание (примеры)

1. Пусть сумма элементов обеих диагоналей матрицы C равна S1, а сумма элементов обеих диагоналей матрицы T равна S2. Если S1 <S2, то найти матрицу H=A+S1T, иначе матрицу Z=B+S2(C+T). Здесь A- матрица порядка n, все элементы которой равны S1; B- матрица порядка n, все элементы которой равны S2.
2. Найти k1-количество элементов ниже побочной диагонали матрицы C, меньших a, и k2-количество элементов ниже побочной диагонали матрицы T, меньших b. Если k1<k2, найти F=T+2k1C, иначе найти H=2C-k2T.
3. Найти a – сумму элементов матрицы T , лежащих на и выше главной диагонали, и b- элементов матрицы C, лежащих на и выше главной диагонали. Если a>b, определить H=T-abC; иначе определить F=C+aT.

Примеры терминов: алгоритм, свойства алгоритма; переменная; тип данных; типы памяти;

кадр стека функции; выражение; интерфейс и реализация функции; алгоритм передачи параметров в функцию; препроцессор; директива препроцессора; бинарное дерево поиска; хеш-таблица.

6.2.2.1.2 Критерии оценивания

Зачтено	Не зачтено
Зачтены лабораторные 1-4, дневник программиста, терминологический словарь. Для МООК: 1. Выполнены все теоретические задания МООК по теме. 2. Выполнено не менее 50% практических задач по теме. Для терминологического словаря: все термины воспроизведены наизусть точно. При этом обучающийся демонстрирует навыками работы в среде программирования; решает поставленные задачи, используя язык программирования; грамотно строит алгоритмы и подбирает тестовые наборы; способен доказать корректность программы; четко и ясно аргументирует использование приобретенных знания и умений при решении задач.	Обучающийся не владеет основным материалом курса; демонстрирует навыки работы в среде программирования; строит алгоритмы с ошибками; неверно выбирает типы данных; неверно описывает спецификации; не умеет проектировать тесты; решает поставленные задачи с ошибками. Не зачтён хотя бы один элемент из списка: лабораторные 1-4, дневник программиста, терминологический словарь. Для МООК: 1. Выполнены не все теоретические задания МООК по теме. 2. Выполнено менее 50% практических задач по теме. Для терминологического словаря: не все термины воспроизведены наизусть точно.

6.2.2.2 Семестр 2, Типовые оценочные средства для проведения экзамена по дисциплине

6.2.2.2.1 Описание процедуры

Экзамены за второй семестр ставятся по результатам выполнения лабораторных работ 5-8, посещения лекций, ведения конспекта лекций, прохождения МООК, выполнения тестов, ведения дневника программиста, знания наизусть терминологического словаря, устных ответов на теоретические вопросы из экзаменационного билета и решения практической задачи. В ходе экзамена студент должен устно ответить на вопросы билета, а также быть способен составить программный код для решения практического задания. Допускается письменный ответ на вопросы билета (по решению преподавателя). Теоретические вопросы экзамена для второго семестра (1-41). Практическое задание.

Терминологический словарь - обучающийся воспроизводит наизусть терминологический словарь.

Пример задания:

Список теоретических вопросов к экзамену

1. Фундаментальные типы композиции объектов: агрегирование, конкантенация, делегирование.
2. Пользовательские типы данных C++: перечисление, объединение, псевдонимы.
3. Пользовательские типы данных C++: структура. Объявление, инициализация, создание объектов (к понятию о фабричной функции).
4. Пользовательские типы данных C++: структура и методы. Ключевое слово this.
5. Указатели на функцию и делегирование (на примере C++ или C#)
6. Перегрузка функций (на примере C++ или C#).
7. Перегрузка операций (на примере C++ или C#).
8. Сборка программы из нескольких файлов (cpp, h)
9. Потоки и работа с файлами (на примере C++)
10. Представление в ЭВМ целых чисел без знака: коды, операции, особенности использования.
11. Представление в ЭВМ целых чисел со знаком: коды, операции, особенности использования.
12. Представление в ЭВМ действительных чисел: формы представления, стандарт IEEE 754 (формулировка, перевод числа, особенности операций).
13. Абстрактные структуры данных: определения, примеры.
14. Абстрактная структура данных – список: структура и классификация.
15. Односвязный линейный список: алгоритм основных операций.
16. Стек и очередь: основные операции и реализация на C++
17. Бинарное дерево поиска: определение, операции поиска, вставки, удаления.
18. Бинарное дерево поиска: обход в глубину и ширину.
19. Сбалансированное дерево поиска: определение и примеры.
20. Абстрактная структура данных – хеш-таблица: определение, структура, операции
21. Абстрактные структуры данных: ассоциативный массив (отображение, карта, словарь), множество, мультимножество
22. Основные принципы обобщенного программирования
23. Основные типы контейнеров библиотеки STL, примеры использования
24. Итераторы на примере библиотеки STL
25. Принципы работы машина Тьюринга
26. Основные тенденции развития языков программирования
27. Идиомы переменных и выражений в структурном программировании.
28. Идиомы арифметических и логических операций.
29. Идиомы для работы со строками.
30. Идиомы для работы с массивами фиксированной длины.
31. Идиомы для работы с массивами переменной длины.
32. Идиомы для работы с картежами и словарями.
33. Идиомы для работы с функциями в структурном программировании.
34. Идиомы для работы с потоками и файлами.
35. Идиомы для работы с памятью.
36. Идиомы для работы с сетью.
37. Идиомы модульного тестирования.
38. Идиомы пользовательских типов.
39. Идиомы пространства имён.
40. Идиомы исключений.
41. Идиомы контроля исполнения.

Практические задания к экзамену

1. Реализовать пользовательскую структуру данных для хранения информации, описанной в столбце таблице – «Структура».

2. Создать контейнеры STL vector добавить в него элементы – экземпляры созданной структуры. Написать собственных код сортировки элементов этого вектора любым метод (кроме методы пузырька). Реализовать его явно, а не использовать библиотеки.
3. Создать контейнер STL set и добавить в него элементы – экземпляры созданной структуры. Использовать компараторы.
4. Выполнить действие с контейнером (set или vector), используя соответствующие алгоритмы библиотеки STL.

Примеры терминов: алгоритм, свойства алгоритма; переменная; тип данных; типы памяти; кадр стека функции; выражение; интерфейс и реализация функции; алгоритм передачи параметров в функцию; препроцессор; директива препроцессора; бинарное дерево поиска; хеш-таблица.

6.2.2.2.2 Критерии оценивания

Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Демонстрирует навыками работы в среде программирования. Решает поставленные задачи, используя навыки алгоритмизации. Четко и ясно аргументирует использование самостоятельно приобретённых знаний и умений при решении задач. Способен доказать корректность алгоритма, выполнить тестирование и отладку.	Демонстрирует навыки работы в среде программирования, но при этом в ответах допускает неточности. Владеет теоретическим материалом. Решает поставленные задачи с незначительными неточностями. Способен доказать корректность алгоритма, выполнить тестирование и отладку.	Нечетко отвечает на теоретические вопросы, допускает ошибки при ответах. Демонстрирует навыки работы в среде программирования. Решает поставленные задачи, может допускать ошибки, сомневается в принятом решении. Подбирает неполный набор тестов для проверки корректности решения.	Не может дать правильных ответов на поставленные теоретические вопросы. Допускает серьезные ошибки при решении задач. Неверно выбирает способ решения задачи, технологии программирования. Не зачтён хотя бы один элемент из списка: лабораторные 4-8 (т.е. лабораторные 1-4 второго семестра), дневник программиста, терминологический словарь.

7 Основная учебная литература

1. Немцова Т. И. Программирование на языке высокого уровня : программирование на языке С++ : учебное пособие для студентов среднего специального образования / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной , 2012. - 512.

2. Сосинская С. С. Программирование на языке С++ : учебное пособие / С. С. Сосинская, 2002. - 115.

[Сайт] – URL: http://elib.istu.edu/viewer/view.php?file=/files/er-2281.pdf

3. Егорова Н. Н. Лабораторный практикум по курсу "Программирование на языке высокого уровня" : учебное пособие / Н. Н. Егорова, А. С. Дорофеев, 2003. - 71.

[Сайт] – URL: http://elib.istu.edu/viewer/view.php?file=/files/er-2354.pdf

4. Павловская Татьяна Александровна. С/С++. Программирование на языке высокого уровня : учеб. для вузов по направлению "Информатика и вычисл. техника" / Т. А. Павловская, 2004. - 460.

8 Дополнительная учебная литература и справочная

1. Павловская Т. А. С/С++. Программирование на языке высокого уровня : учеб. для вузов по направлению "Информатика и вычислит. техника" / Т. А. Павловская, 2006. - 460.

9 Ресурсы сети Интернет

1. http://library.istu.edu/
2. https://e.lanbook.com/

10 Профессиональные базы данных

1. http://new.fips.ru/
2. http://www1.fips.ru/

11 Перечень информационных технологий, лицензионных и свободно распространяемых специализированных программных средств, информационных справочных систем

12 Материально-техническое обеспечение дисциплины