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

УТВЕРЖДЕНО
Директор физтех-школы радиотехники и компьютерных технологий
Д.А. Гаврилов

Рабочая программа дисциплины (модуля)

по дисциплине:	Введение в структуры данных и промышленное программирование
по направлению:	Информатика и вычислительная техника
профиль подготовки:	Компьютерные технологии и вычислительная техника Физтех-школа Радиотехники и Компьютерных Технологий кафедра радиоэлектроники и прикладной информатики
курс:	1
квалификация:	бакалавр

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

Аудиторных часов: 120 всего, в том числе:
- лекции: 0 час.
- семинары: 120 час.
- лабораторные занятия: 0 час.

Самостоятельная работа: 60 час.

Всего часов: 180, всего зач. ед.: 4

Количество контрольных работ, заданий: 4

Программу составил: И.Р. Дединский, старший преподаватель

Программа обсуждена на заседании кафедры радиоэлектроники и прикладной информатики 30.01.2024

Аннотация

Курс представляет собой введение в структуры данных и алгоритмы на языке Си. Практическая часть содержит задачи достаточно большого объема. Задачи подобраны по большей части таким образом, что в конце курса каждый студент самостоятельно реализует примитивную модель вычислительной системы (стековой виртуальной машины), инструментальные средства низкоуровневой разработки для него (ассемблер и дизассемблер), а также примитивный высокоуровневый транслятор (проект «нано-GCC»), совместимый с трансляторами других студентов на уровне AST. Это дает возможность использовать кросс-компиляцию программ одного студента для виртуальной машины другого затем выполнение на соответствующей виртуальной машине, а также перевод в исходный текст в формате языка другого студента.

Сложность задач курса легко регулируется их функциональным наполнением.

Для обучения используются следующие принципы:

1. Во главу угла ставится задача, ее решение и, главное, путь от задачи к решению. Во всякой задаче подчеркивается разделение на идею решения и технологию реализации.
2. Самостоятельность решения является ключевым условием.
3. Понимание студентами тех средств, с помощью которых он решил задачу, ставится выше уровня самих средств решения.
4. Аккуратность и надежность решения ставятся выше «программистских трюков», иногда позволяющих в отдельных случаях добиться несколько лучших результатов.

Преподавание курса ведется в предположении, что студенты уже знают язык Питон, Паскаль или аналогичный процедурный язык.

1. Цели и задачи

Цель дисциплины

– научить студентов современным методам программирования и разработки программных систем на языке С, привить навыки надежного, промышленного программирования, работы в команде, подготовить их для участия в проектной работе.

Задачи дисциплины

Задача дисциплины заключается в демонстрации базовых принципов промышленного программирования.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности

решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- основы работы с языком Си;
- основы промышленного программирования;
- основы работы с системами контроля версий;
- различные пути повышения производительности программы.

уметь:

- реализовывать алгоритмы и программы на языке Си.

владеть:

- навыками ведения простейших программных проектов в системах контроля версий.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение в язык С		8		4
2	Трансляция и построение программы		8		4
3	Форматирование кода		8		4
4	Комментирование кода		8		4
5	Тестирование программ		8		4
6	Динамическая память в языке Си		8		4
7	Многомерные массивы		8		4
8	Строки		8		4
9	Структуры		8		4
10	Понятие абстрактных структур данных		8		4
11	Стековый вычислитель. Ассемблер и дизассемблер.		8		4
12	Расширение круга задач для стекового вычислителя		8		4

13	Структура данных «список»		8		4
14	Структура данных «дерево»		8		4
15	Архитектура nGCC		8		4
Итого часов			120		60
Подготовка к экзамену		0 час.			
Общая трудоёмкость		180 час., 4 зач.ед.			

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

Семестр: 1 (Осенний)

1. Введение в язык С

Краткая история и особенности возникновения языка. Ключевая роль Си в обучении программированию. Высокоуровневые и низкоуровневые языки.

Высокоуровневые языки. Переносимость высокоуровневых программ. Проблемы с производительностью и доступом к вычислительным средствам из языков высокого уровня. Си как язык промежуточного уровня, задуманный и построенный как компромисс между низким и высоким уровнем.

Командная разработка проектов. Системы контроля версий, репозитории и работа с ними. Системы контроля версий. Code review в системе контроля версий. Типичный рабочий процесс при работе с системой контроля версий и репозиторием в Linux и Windows. Рабочий процесс для code review.

2. Трансляция и построение программы

Понятие программного проекта. Модель проекта «один исходный файл – одна программа». Трансляция файла с исходным текстом. Соотнесение исходного текста и исполняемого кода программы. Модель проекта «много исходных файлов – одна программа». Понятие раздельной компиляции. Понятие о включаемых заголовочных файлах, информация, содержащаяся в них, ее использование в процессе трансляции. Понятие библиотеки как объединения объектных файлов.

Структура и синтаксис простейшей программы на языке Си. Раздел включаемых заголовочных файлов, главная программа. Объявление переменных, типы данных (на примере int и double). Ввод и вывод информации, функции printf и scanf. Арифметические выражения, операторы сложения, вычитания, умножения и деления, оператор присваивания. Функция вычисления квадратного корня. Условный оператор.

3. Форматирование кода

Его важность для реализации надежности и командности, его виды, понятие корпоративного стиля форматирования.

Понятие определения функции, ее заголовка и тела. Формальные параметры функции квадратного уравнения (коэффициенты), синтаксис их объявления. Понятие и синтаксис вызова функции. Понятие прототипа функции, синтаксис. Модульный принцип построения программ. Передача данных через оператор return. Анализ возвращаемого значения в main, оператор switch, его синтаксис. Проблема передачи информации о бесконечном количестве корней. Использование «магических чисел» и именованных констант для обозначения бесконечного количества корней. Синтаксис определения именованной константы. Указатели, передача данных через параметр-указатель. Макрос assert, его применение для проверки указательных параметров, информативность для программиста при отладке.

4. Комментирование кода

Примеры: комментарии «о хаках», комментарии-TODO. Блочные комментарии файла и функции

Препроцессор языка С. Соотношение фаз работы компилятора и препроцессора. Директивы препроцессора. Директива include для стандартных заголовочных файлов. Директива define. Использование директивы для задания констант, ее отличия от конструкции с const. Директива define с параметрами. Особенности и побочные эффекты в случае макроопределения с параметрами, ее отличие от функций. Классические примеры построения макроопределения с параметрами с демонстрацией побочных эффектов и защитой параметров скобками. Продолжение макроопределения на следующие строки с помощью символа обратной косой черты. Ошибки применения define: использование аргументов с побочным эффектом (инкремент/декремент переменных, вызовы функций, работающих с потоками и т.п.) Стандартные макроопределения __FILE__, __LINE__, __DATE__, __TIME__. Разбор механизма влияния NDEBUG на assert, условная компиляция. Директивы #ifdef, #ifndef, #if (и синтаксис допустимых в нем конструкций), #else, #elif, #endif

5. Тестирование программ

Тестирование с помощью специальных программ (серверов).

Массивы в языке Си. Использование массивов для хранения серий данных. Объявление и инициализация массива. Ограничения массивов в Си (нумерация, единство типа данных, ограниченный размер). Хранение массивов в оперативной памяти. Адресация к массиву. Имя массива как адрес (указатель) его начального элемента. Типичные ошибки при работе с массивами (выход за границы массива). Проверка допустимости индексации с помощью assert. Решение проблемы волатильности с помощью модификатора const.

6. Динамическая память в языке Си

Понятие «свободной памяти». Функции работы с динамической памятью. Время жизни блока динамической памяти. Динамическая память как ресурс, работа с исчерпанием памяти, реализация стратегий гарантированного освобождения. Пример структуры блока динамической памяти. Последствия выхода за границы блока, двойного освобождения блока, переполнения буфера, находящегося в динамической памяти. Реаллокация блоков динамической памяти, проблема пересчета указателей в случае изменения адреса блока.

Указательная арифметика. Операции с указателями в языке См. Формула вычисления адреса для доступа к элементу массива. Использование указательной арифметики для потоковых вычислений на массивах, понятие «текущего элемента».

7. Многомерные массивы

Их объявление, инициализация, адресация в них. Проблема передачи многомерного массива в функцию. Вычисление адреса элемента в многомерном массиве. Необходимость передавать размеры многомерного массива в функцию. Самостоятельное вычисление смещения относительно начала массива и адреса нужного элемента массива, преимущества и недостатки такого подхода. Реализация многомерных массивов в динамической памяти, доступ к таким массивам.

Массивы указателей. Синтаксис объявления и использования массивов указателей. Трактовка операции индексации в случае массивов указателей. Реализация многомерных массивов через массивы указателей, Решение вопроса о хранении массива с «рваным правым краем» (неодинаковым размером строк). Использование разных блоков для хранения разных строк массива, возможность реаллокации для изменения длин строк.

8. Строки

Реализация строк в языке Си, «смысловая» и «свободная» зоны строки. Нулевой символ. Понятие пустой строки. Задачи о копировании и сравнении строк, задача о сжатии пробелов в строке «на месте». Концепция «текущего символа». Проблемы «маляра Шлемиля (Шлемиэля)», их характерные проявления и устранение. Возможности строковой библиотеки языка Си. Массивы строк. Задача о сортировке строк, Обобщение алгоритмов сравнения строк. Указатели на функции. Использование указателей на функции для построения универсальной функции сортировки строк. Библиотечная функция qsort и работа с ней.

Работа с файлами. Функции открытия и закрытия файла. Текстовые файлы, посимвольное и построчное считывание. Состояние «конец файла», константа EOF. Опасность переполнения буфера при чтении. Форматированный текстовый ввод и вывод, опасности, с ним связанные. Символы преобразования данных и форматирования.

9. Структуры

Операции доступа к структурам. Построение структур, размер структуры, выравнивание полей. Передача структур в функцию, способы ее ускорения (через указатель) и защиты доступа (через указатель на константные данные). Реализация «методов класса» средствами языка Си.

10. Понятие абстрактных структур данных

Структура данных «стек». Функции для работы со стеком. Функции конструирования, уничтожения, верификации и технической распечатки (дампа). Пример реализации функции дампа. Реализация верифицируемой функции для работы со структурой данных. Понятие о предусловии, постусловии и инварианте алгоритма. Реализация предусловий с помощью функции верификации и дампа. Тактика двойной верификации с предусловием и постусловием для функций, работающих с неконстантным объектом. Понятие ошибочного объекта.

Использование стека. Задача о вычислении выражений. Вычисление выражений, заданных обратной польской записью. Понятие стекового вычислителя (процессора). Реализация структуры стекового вычислителя и связанных с ней функций. Реализация арифметических команд для стекового вычислителя. Примеры работы стекового вычислителя. Интерактивный режим работы программы вычисления выражений. Задача о построении таблицы значений функции или ее графика. Пример программы в обратной польской записи (Р-программы) для вычисления значения функции в каждой заданной точке. Пример фрагмента программы на языке Си для запуска стековых вычислений функции для каждого заданного значения ее аргумента. Необходимость использования аргумента функции (абсциссы) в обратной польской записи, проблема его хранения в вычислителе. Понятие регистра вычислителя (процессора), введение регистра абсциссы (AX) в стековый вычислитель. Функция на языке Си для загрузки значения абсциссы в вычислитель (mov_ax).

11. Стековый вычислитель. Ассемблер и дизассемблер.

Проблема скорости работы стекового вычислителя при исполнении стереотипного кода для каждого заданного значения абсциссы. Реализация программирования вычислителя с помощью текстового файла с последовательностью команд в обратной польской записи. Анализ скорости работы такой программы, определение «узких мест» для повышения эффективности. Введение нумерации команд (Р-кода) для повышения эффективности. Решение проблемы низкой мнемоничности Р-кода на стороне языка Си с помощью констант и конструкции enum. Решение аналогичной проблемы при написании программ на Р-коде с помощью реализации транслятора в Р-код (ассемблера). Реализация дизассемблера для целей отладки. Сопряженность программ исполнителя, ассемблера и дизассемблера, ее реализация с помощью единого включаемого файла с именами команд и константами, задающими операции Р-кода. Хранение программы в Р-кодах в массиве команд, преимущества такого подхода перед постоянным чтением их из файла. Отделение фазы загрузки программы из файла с Р-кодом в массив команд от фазы исполнения программы. Реализация эффективного программного комплекса стекового вычислителя для построения графиков или таблиц значений функций и системных утилит для него.

12. Расширение круга задач для стекового вычислителя

Обобщение постановки задачи для построения таблиц значений (графиков) функций с необходимостью единственного исполнения программы и организации перебора значений аргумента в Р-программе. Пример возможной Р-программы для построения таблиц значений (графика) функции. Необходимость команд текстового или графического вывода данных (out), команд условного и безусловного переходов (j*, jmp) для организации цикла в Р-программе. Проблема аргумента в командах переходов. Реализация команд переходов с помощью вручную рассчитанных адресов переходов, недостатки такого подхода. Задача автоматического расчета адресов переходов. Понятие меток как синонимов адресов. Задача сканирования меток и сопоставления им адресов. Методы сопоставления (патчинг кода и многопроходная трансляция). Реализация программы многопроходного (двухпроходного) транслятора в Р-код с использованием нумерованных меток.

Реализация вызова функций в стековом вычислителе. Сравнение работы команд вызова функции и безусловного перехода. Понятие возврата из функции. Необходимость в стеке адресов возвратов для поддержки рекурсии. Реализация команд вызова функции с аргументом в виде метки и возврата из функции с помощью отдельного стека для хранения адресов возврата.

Выполнение задач на стековом вычислителе в виде написания самостоятельно разработанных Р-программ (решение квадратных уравнений с разбором всех частных случаев, выдачей количества и величин их корней, вычисления факториала чисел и чисел Фибоначчи итеративным и рекурсивным способами). Расширение количества регистров (добавление регистров bx, cx, dx), системы команд (добавления команды ввода с клавиатуры in).

13. Структура данных «список»

Использование списков. Односвязные и двусвязные списки. Неэффективность операции индексации для списков (еще один пример «алгоритма маляра Шлемиля»). Проверка валидности списка.

Структура данных «хеш-таблица». Задачи, приводящие к хеш-таблицам. Хеш-функции, их примеры (от простейших и бесполезных к реальным) и свойства, качество хеширования. Характерные размеры хеш-таблиц. Использование хеш-таблиц. Качественное сравнение качества хеширования с помощью гистограммы заполнения хеш-таблицы.

14. Структура данных «дерево»

Примеры различного использование деревьев. Деревья поиска. Перечисление узлов дерева, виды обходов дерева. Верификация деревьев Дамп деревьев. Задачи, использующие деревья.

Структура арифметических выражений. Инфиксная форма записи выражений, ее соответствие порядку действий и дереву вычислений. Задача о грамматическом разборе выражений. Необходимость задания структуры выражений. Понятие языка и грамматики. Способы построения дерева разбора. Алгоритм распознавания языка методом рекурсивного спуска. Итеративное построение грамматики языка программирования и кода распознавателя, использующего рекурсивный спуск. Решение вопроса о приоритете операторов и задачи о подвыражениях в скобках. Достоинства метода рекурсивного спуска, его проблемы и ограничения. Различные обходы деревьев выражений, восстановление линейной инфиксной записи и генерация различных видов польских записей (префиксной, постфиксной). Транслятор инфиксных выражений в ассемблер стекового процессора. Лексический анализ как предварительная фаза перед синтаксическим, его роль в повышении эффективности трансляции и ее упрощении. Понятие лексемы, ее реализация. Рефакторинг транслятора с применением лексического анализа.

15. Архитектура nGCC

Front-end, middle-end и back-end. Достоинства модульного принципа и общего внутреннего формата. Задача о групповой реализации nGCC для n модельных входных языков высокого уровня и m вычислителей с разными системами команд. Разработка общего внутригруппового стандарта промежуточного файла с AST, поддерживающего дополнительные данные (имена переменных и т.п.). Рефакторинг транслятора инфиксных выражений с использованием архитектуры nGCC. Реализация программы для запуска частей транслятора (драйвера). Реализация обратного преобразования (из AST в код модельного высокоуровневого языка).

Работа с переменными в модельных высокоуровневых языках. Модель оперативной памяти высокоуровневого языка. Использование таблиц имен для переменных и других именованных сущностей. Реализация операторов присваивания. Реализация условных операторов, операторов цикла, вызова функции.

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

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

6.Перечень рекомендуемой литературы

Основная литература

1. Язык программирования Си [Текст]/Б. Керниган, Д. Ритчи , пер. с англ. под ред. В. С. Штаркмана, -СПб., Невский Диалект, 2000-2003-2004

Дополнительная литература

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

1. https://www.r-5.org/files/books/computers/languages/c/kr/Brian_Kernighan_Dennis_Ritchie-The_C_Programming_Language-RU.pdf
2. https://git-scm.com/book/ru/v2

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Операционная система Linux или Windows. Компилятор gcc или clang. Среда программирования.

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

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя проработку учебного материала (по учебной литературе), подготовку к практическим занятиям.

Промежуточный контроль знаний проводится в виде сдач проектных заданий.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника
профиль подготовки: Компьютерные технологии и вычислительная техника
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра радиоэлектроники и прикладной информатики
курс: 1
квалификация: бакалавр

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

Разработчик: И.Р. Дединский, старший преподаватель

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

2. Показатели оценивания компетенций

В результате изучения дисциплины «Введение в структуры данных и промышленное программирование» обучающийся должен:

знать:

- основы работы с языком Си;
- основы промышленного программирования;
- основы работы с системами контроля версий;
- различные пути повышения производительности программы.

уметь:

- реализовывать алгоритмы и программы на языке Си.

владеть:

- навыками ведения простейших программных проектов в системах контроля версий.

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

Примеры заданий:

1. Реализовать программу расчета корней квадратного уравнения соответственно принципам промышленного программирования.
2. Реализовать программу сортировки текста.
3. Реализовать структуру данных «Стек» с использованием технологий защиты данных.
4. Реализовать программу, моделирующую поведение простейшего процессора.
5. Реализовать двусвязный список с технологиями графического отображения данных.
6. Реализовать двоичный справочник.
7. Реализовать программу символьного дифференцирования.
8. Реализовать собственный простейший язык программирования высокого уровня.

За каждое задание выставляется оценка в соответствии со следующими критериями оценивания:

Отлично

10) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле.

9) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы.

8) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач.

Хорошо

7) Полностью решены все задачи. Допущены несущественные ошибки

6) Полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

5) Полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

Удовлетворительно.

4) Полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

3) Полностью решено более половины задач.

Неудовлетворительно

2) Решено менее половины задач.

1) Не решено ни одной задачи.

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

1. Операторы языка Си
2. Реализация функций в языке Си
3. Массивы и указатели
4. Указательная арифметика
5. Работа препроцессора языка Си
6. Представление чисел в языке Си
7. Кодировки текстовых файлов. Бинарные файлы
8. Работа динамической памяти
9. Работа с репозиториями
10. Верификация программ
11. Юнит-тестирование
12. Переносимость программ

Критерии оценивания

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

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

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

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

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

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

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

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

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

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

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

Дифференцированный зачет проводится по итогам текущей успеваемости и сдачи заданий, предусмотренных программой дисциплины.

Дифференцированный зачет проводится в устной форме.

При проведении дифференцированного зачета обучающемуся предоставляется 20 минут на подготовку. Опрос обучающегося проводится в течение 30 минут.