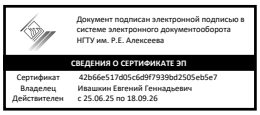
МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Нижегородский государственный технический университет им. Р.Е. Алексеева» (НГТУ)

Учебно-научный институт радиоэлектроники и информационных технологий (ИРИТ)
(Полное и сокращенное название института, реализующего данное направление)



УТВЕРЖДАЮ:
Директор института:
_____ А.В. Мякиньков
подпись ФИО
02 июня 2025г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.Б.13 Основы информатики
(индекс и наименование дисциплины по учебному плану)
для подготовки бакалавров

Направление подготовки: 01.03.02 Прикладная математика и информатика

Направленность: Математическое моделирование и компьютерные технологии

Форма обучения: очная

Год начала подготовки 2023, 2024, 2025

Выпускающая кафедра ПМ

Кафедра-разработчик ПМ
аббревиатура кафедры
Объем дисциплины 72/2
часов/ з. е
Промежуточная аттестация зачёт

Разработчик: Горенкова А. В., ассистент.

Нижний Новгород, 2025

Рабочая программа дисциплины: разработана в соответствии с Федеральным государственным образовательным стандартом высшего образования (ФГОС ВО 3++) по направлению подготовки 01.03.02 Прикладная математика и информатика, утвержденного приказом МИНОБРНАУКИ РОССИИ от 10 января 2018 года № 9, на основании учебного плана принятого УМС НГТУ протокол № 21 от 18.05.2023,
№ 16 от 21.05.2024,
№ 6 от 17.12.2024.

Рабочая программа одобрена на заседании кафедры протокол от 16.05.2025 № 8

Зав. кафедрой д.ф-м.н, профессор А.А. Куркин

Программа рекомендована к утверждению ученым советом ИРИТ.
Протокол от 20.05.2025 № 1

Рабочая программа зарегистрирована в УМУ регистрационный № 01.03.02-п-13
Начальник МО_____ Е.Г. Севрюкова
Заведующая отделом комплектования НТБ _____ Н.И.Кабанина
(подпись)

СОДЕРЖАНИЕ

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ 4

1.1 Цель освоения дисциплины 4
1.2 Задачи освоения дисциплины (модуля) 4

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ 4

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ) 5

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ 6

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам 6
4.2Содержание дисциплины, структурированное по темам 7

5. ТЕКУЩИЙ КОНТРОЛЬ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ 10

5.1 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности 10
5.2 Описание показателей и критериев контроля успеваемости, описание шкал оценивания 10

6. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 13

6.1 Учебная литература, печатные издания библиотечного фонда 13
6.2. Справочно-библиографическая литература. 13

7. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 14

7.1 Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля) 14
7.2 Перечень программных продуктов, используемых при проведении различных видов занятий по дисциплине 14
7.3 Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства необходимого для освоения дисциплины 14

8. ОБРАЗОВАТЕЛЬНЫЕ РЕСУРСЫ ДЛЯ ИНВАЛИДОВ И ЛИЦ С ОВЗ 16

9. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ, НЕОБХОДИМОЕ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ 16

10. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ОБУЧАЮЩИМСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ 17

11. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ 18

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1.1 Цель освоения дисциплины

Целью освоения дисциплины «Основы информатики» является приобретение студентами базовых знаний о принципах функционирования вычислительных систем и операционных систем, навыков работы в командной строке, а также средств автоматизации сборки и контроля версий. Освоение дисциплины направлено на формирование у студентов системного подхода к пониманию архитектуры программного и аппаратного обеспечения, развитие логического и алгоритмического мышления, а также обеспечение формирования компетенций, предусмотренных ФГОС, в части представленных ниже знаний, умений и навыков.

1.2 Задачи освоения дисциплины (модуля)

Дисциплина «Основы информатики» способствует подготовке студентов к решению следующих профессиональных задач:

1. Изучение архитектуры персонального компьютера и принципов его функционирования, включая этапы загрузки и взаимодействие с операционной системой на базовом уровне;
2. Приобретение практических навыков работы в командной строке Unix-подобных систем, использования оболочки bash и написания скриптов;
3. Изучение основ работы с системами контроля версий (например, git), инструментами автоматизации сборки программ (Makefile, CMake).

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Учебная дисциплина «Основы информатики» Б1.Б.13 включена в перечень дисциплин базовой части, определяющий направленность образовательной программы «Прикладная математика и информатика». Дисциплина реализуется в соответствии с требованиями ФГОС, ОП ВО и УП.

Дисциплина «Основы информатики» является основополагающей для изучения дисциплины «Компьютерная графика», «Базы данных», а также для подготовки к сдаче государственного экзамена, выполнения и защиты ВКР.

Рабочая программа дисциплины «Основы информатики» для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается индивидуально с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся.

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)[i]

Таблица 1. Формирование компетенций дисциплинам

Наименование дисциплин, формирующих компетенцию совместно	Семестры, формирования дисциплины Компетенции берутся из Учебного плана по направлению подготовки бакалавра							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
ОПК-4. Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.								
Основы информатики	*							
Компьютерная графика						*		
Базы данных							*	*
Подготовка к сдаче и сдача государственного экзамена								*
Выполнение и защита ВКР								*

ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОПВО

Таблица 2. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения

Код и наименование компетенции	**Код и наименование индикатора достижения компетенции**	**Планируемые результаты обучения по дисциплине**			**Оценочные средства**	
					Текущего контроля	**Промежуточной аттестации**
ОПК-4. Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.	ИОПК-4.1. Использует современные Информационные технологии для решения задач профессиональной деятельности.	**Знать:** основные понятия информатики, основы алгоритмизации и программирования; основы и методы защиты информации и сведений, составляющих государственную тайну, парадигмы программирования, инструментальные средства создания и отладки программного обеспечения	**Уметь:** уверенно работать в качестве пользователя на ПЭВМ с программными средствами общего назначения; выполнять алгоритмизацию; создавать алгоритмы для решения задач ввода-вывода данных, сортировки и поиска, создавать блок-схемы и описания алгоритмов, программную документацию.	**Владеть:** персональным компьютером на уровне уверенного пользователя, навыками работы в любых операционных системах, основами работы в глобальной сети, приемами поиска в сети Интернет, навыками работы с программами подготовки документов, таблиц, презентаций, графических объектов.	Задания для контрольных работ	Вопросы для письменного опроса

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам

Общая трудоёмкость дисциплины составляет 2 зач.ед. 72 часа, распределение часов по видам работ семестрам представлено в таблице 3.

Таблица 3. Распределение трудоёмкости дисциплины по видам работ по семестрам для студентов очного обучения

Вид учебной работы	**Трудоёмкость в час**	
	Всего час.	**В т.ч. по семестрам**
		1сем
Формат изучения дисциплины	с использованием элементов электронного обучения	
Общая трудоёмкость дисциплины по учебному плану	**72**	**72**
1. Контактная работа:	**38**	**38**
Аудиторная работа, в том числе:	**34**	**34**
занятия лекционного типа (Л)	-	-
занятия семинарского типа (ПЗ-семинары, практ. занятия и др)	-	-
лабораторные работы (ЛР)	34	34
Внеаудиторная, в том числе	**4**	**4**
курсовая работа (проект) (КР/КП) (консультация, защита)	-	-
текущий контроль, консультации по дисциплине	4	**4**
контактная работа на промежуточном контроле (КРА)	-	-
2. Самостоятельная работа (СРС)	**34**	**34**
реферат/эссе (подготовка)	-	-
расчётно-графическая работа (РГР) (подготовка)		
контрольная работа	-	-
курсовая работа/проект (КР/КП) (подготовка)	-	-
самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям, коллоквиум и т.д.)	34	34
Подготовка к экзамену (контроль)	-	-

4.2 Содержание дисциплины, структурированное по темам

Таблица 4. Содержание дисциплины, структурированное по темам для студентов очного обучения

Планируемые (контролируемые) результаты освоения: код УК; ОПК; ПК и индикаторы достижения компетенций	Наименование разделов, тем	Виды учебной работы (час)				Вид СРС	Наименование используемых активных и интерактивных образовательных технологий	Реализация в рамках Практической подготовки (трудоемкость в часах)	Наименование разработанного Электронного курса (трудоемкость в часах)
		Контактная работа			Самостоятельная работа студентов (час)				
		Лекции (час)	Лабораторные работы (час)	Практические занятия (час)					
3 семестр									
Раздел 1. Операционные системы									
ОПК-4 ИОПК-4.1.	**Тема 1.1** Архитектура ЭВМ. Устройство персонального компьютера. Принцип работы процессора.		1		1	Подготовка к лабораторным занятиям [6.1.1., 6.1.2., 6.2.1.]	—	—	—
	Тема 1.2 Архитектура операционной системы. Алгоритм запуска ЭВМ и операционной системы.		0.5		0.5		—	—	—
	Тема 1.3 Виды операционных систем. UNIX-подобные операционные системы.		0.5		0.5		—	—	—
	Итого по 1 разделу		**2**		**2**		—	—	—
Раздел 2. Операционная система Linux									
ОПК-4 ИОПК-4.1.	**Тема 2.1** Выбор дистрибутива Linux. Установка Debian дистрибутивов второй системой на персональный компьютер. Установка Debian дистрибутивов на виртуальную машину.					Подготовка к лабораторным занятиям [6.1.1., 6.1.2., 6.2.1.]	—	—	—
	Тема 2.2 Архитектура файловой системы ext4.						—	—	—
	Тема 2.3 Права и пользователи в Debian дистрибутивах.						—	—	—

Планируемые (контролируемые) результаты освоения: код УК; ОПК; ПК и индикаторы достижения компетенций	Наименование разделов, тем	Виды учебной работы (час)				Вид СРС	Наименование используемых активных и интерактивных образовательных технологий	Реализация в рамках Практической подготовки (трудоемкость в часах)	Наименование разработанного Электронного курса (трудоемкость в часах)
		Контактная работа			Самостоятельная работа студентов (час)				
		Лекции (час)	Лабораторные работы (час)	Практические занятия (час)					
	Тема 2.4 Текстовые и графические оболочки ОС. Работа с текстовой оболочкой Linux. Основные команды оболочки bash.						—	—	—
	Итого по 2 разделу		**8**		**8**		—	—	—
Раздел 3. Bash скрипты									
ОПК-4 ИОПК-4.1.	**Тема 3.1** Структура bash скрипта, шебанг. Запуск и выполнение скриптов.		1		1	Подготовка к лабораторным занятиям [6.1.1., 6.1.2., 6.2.1.]	—	—	—
	Тема 3.2 Виды переменных, работа с переменными.		1		1		—	—	—
	Тема 3.3 Арифметические операции.		2		2		Контрольная работа	—	—
	Тема 3.4 Условные операторы и циклы.		2		2		—	—	—
	Тема 3.5 Работа с массивами в bash скриптах.		2		2		—		
	Тема 3.6 Аргументы командной строки.		2		2		—		
	Тема 3.7 Считывание данных из файла и запись в файл.		4		4		—		
	Тема 3.8 Функции, аргументы функций. Проверка существования аргумента.		2		2		—		
	Тема 3.9 Комплексная работа с bash скриптами		4		4		—		
	Итого по 3 разделу		**20**		**20**		—	—	—
Раздел 4. Системы контроля версий									
ОПК-4 ИОПК-4.1.	**Тема 4.1** Виды систем контроля версий. Различия в хранении diff файлов.		0.5		0.5	Подготовка к лабораторным занятиям [6.1.1., 6.1.2., 6.2.1.]	—	—	—
	Тема 4.2 Система контроля версий		1		1		—	—	—

Планируемые (контролируемые) результаты освоения: код УК; ОПК; ПК и индикаторы достижения компетенций	Наименование разделов, тем	Виды учебной работы (час)				Вид СРС	Наименование используемых активных и интерактивных образовательных технологий	Реализация в рамках Практической подготовки (трудоемкость в часах)	Наименование разработанного Электронного курса (трудоемкость в часах)
		Контактная работа			Самостоятельная работа студентов (час)				
		Лекции (час)	Лабораторные работы (час)	Практические занятия (час)					
	git. Этапы создания коммита.								
	Тема 4.3 Ветвление в git.		1		1		—	—	—
	Тема 4.4 Работа с удалённым репозиторием git.		0.5		0.5		Контрольная работа	—	—
	Тема 4.5 Автоматическая сборка проекта из нескольких файлов с помощью Makefile.		1		1		—		
	Итого по 4 разделу		**4**		**4**		—	—	—
	Итого за 1 семестр		**34**		**34**		—	—	—
	Подготовка к экзамену (контроль)				**4**		—	—	—
	Итого по дисциплине		**34**		**34**		—	—	—

5. ТЕКУЩИЙ КОНТРОЛЬ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Текущий контроль осуществляется по всем видам учебного процесса: беседы, дискуссии по темам лекционных занятий, выполнение практических заданий, расчетно-графической и контрольных работ. Промежуточный контроль проводится в устно-письменной форме.

5.1 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта в ходе текущего контроля успеваемости представлены в ФОС дисциплины.

Перечень вопросов, выносимых на промежуточную аттестацию в форме экзамена хранятся на кафедре «Прикладная математика» ауд. 1204 по адресу Н. Новгород, ул. Минина, 24 и находятся в свободном доступе.

5.2 Описание показателей и критериев контроля успеваемости, описание шкал оценивания

Для оценки знаний, умений и навыков и формирования компетенций по дисциплине может применяться балльно-рейтинговая/традиционная система контроля и оценивания успеваемости студентов.

Таблица5. Балльно-рейтинговая система оценивания при текущем контроле (контрольные недели) и оценка выполнения контрольных работ

Шкала оценивания	**Зачёт**
$40<R\leq 50$	Зачтено
$30<R\leq 40$	Зачтено
$20<R\leq 30$	Зачтено
$0<R\leq 20$	Не зачтено

Таблица 6. Критерии оценивания результата обучения по дисциплине и шкала оценивания

Код и наименование компетенции	**Код и наименование индикатора достижения компетенции**	**Критерии оценивания результатов обучения**			
		Оценка **«неудовлетворительно»** / «не зачтено» 0-59% от max рейтинговой оценки контроля	Оценка **«удовлетворительно»** / «зачтено» 60-74% от max рейтинговой оценки контроля	Оценка **«хорошо» /** «зачтено» 75-89% от max рейтинговой оценки контроля	Оценка **«отлично» /** «зачтено» 90-100% от max рейтинговой оценки контроля
ОПК-4. Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности.	ИОПК-4.1. Использует современные Информационные технологии для решения задач профессиональной деятельности.	Не способен описать архитектуру персонального компьютера и процессы начальной загрузки; не умеет ориентироваться в структуре операционной системы Linux, не владеет базовыми навыками работы в командной строке; не способен применять инструменты командной оболочки bash, систем контроля версий (git) и средств автоматизации сборки (Makefile, CMake) для решения практических задач; демонстрирует отсутствие понимания процессов, лежащих в основе функционирования программного обеспечения и операционной среды.	Способен воспроизвести базовые сведения об архитектуре персонального компьютера и основных этапах загрузки операционной системы; может выполнять элементарные операции в командной строке Linux и использовать отдельные команды оболочки bash; способен применять git для выполнения базовых операций с репозиторием; умеет использовать готовые Makefile и CMake-файлы или незначительно модифицировать их для сборки несложных программ; способен решать простейшие практические задачи при минимальной помощи.	Способен уверенно описывать архитектуру персонального компьютера, процесс загрузки и начальной инициализации операционной системы; свободно работает в командной строке Linux, применяя основные команды и средства навигации; умеет писать и отлаживать bash-скрипты для автоматизации задач; владеет инструментами git для ведения истории изменений, ветвления и слияния; способен создавать и модифицировать Makefile и CMake-конфигурации для сборки программ средней сложности; может применять полученные знания для решения большинства практических задач без посторонней помощи.	В полном объеме владеет знаниями об архитектуре персонального компьютера и процессе загрузки операционной системы; демонстрирует глубокое понимание принципов функционирования ОС Linux и взаимодействия её компонентов; уверенно использует командную строку для широкого круга задач; разрабатывает и оптимизирует bash-скрипты различной сложности; полноценно использует возможности системы контроля версий git, включая работу с ветками, разрешение конфликтов и ведение совместной разработки; самостоятельно разрабатывает и адаптирует Makefile и CMake-конфигурации для комплексной сборки программных проектов; уверенно применяет приобретённые знания и навыки в практической деятельности и

					реализует нестандартные решения.

Таблица 7. Критерии оценивания

Оценка	Критерии оценивания
Высокий уровень «5» (отлично)	оценку **«отлично»** заслуживает студент, освоивший знания, умения, компетенции и теоретический материал без пробелов; выполнивший все задания, предусмотренные учебным планом на высоком качественном уровне; практические навыки профессионального применения освоенных знаний сформированы.
Средний уровень «4» (хорошо)	оценку **«хорошо»** заслуживает студент, практически полностью освоивший знания, умения, компетенции и теоретический материал, учебные задания не оценены максимальным числом баллов, в основном сформировал практические навыки.
Пороговый уровень «3» (удовлетворительно)	оценку **«удовлетворительно»** заслуживает студент, частично с пробелами освоивший знания, умения, компетенции и теоретический материал, многие учебные задания либо не выполнил, либо они оценены числом баллов близким к минимальному, некоторые практические навыки не сформированы.
Минимальный уровень «2» (неудовлетворительно)	оценку «**неудовлетворительно»** заслуживает студент, не освоивший знания, умения, компетенции и теоретический материал, учебные задания не выполнил, практические навыки не сформированы.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

6.1 Учебная литература, печатные издания библиотечного фонда

Библиотечный фонд укомплектован печатными изданиями из расчета не менее 0,25 экземпляра каждого из изданий, указанных ниже на каждого обучающегося из числа лиц, одновременно осваивающих соответствующую дисциплину (модуль).

6.1.1 Языки программирования и методы трансляции: Учеб.пособие / С. З. Свердлов; СПб. Питер, 2007. - 638 с. - ISBN 978-5-469-00378-6

6.1.2. Информатика. Базовый курс : Учеб.пособие / С. В. Симонович ; СПб. Питер, 2009. - 640 с. - ISBN 5-94723-752-0

6.2. Справочно-библиографическая литература.

6.2.1. https://git-scm.com/book/ru/v2 [Электронный ресурс]: Документация git

7. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Учебный процесс по дисциплине обеспечен необходимым комплектом лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства (состав по дисциплине определен в настоящей РПД и подлежит обновлению при необходимости).

7.1 Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

Перечень программных продуктов, используемых при проведении различных видов занятий по дисциплине (открытый доступ):

1) консультантПлюс [Электронный ресурс]: Справочная правовая система. - Режим доступа: http://www.consultant.ru;
2)научная электронная библиотека E-LIBRARY.ru. – Режим доступа: http://elibrary.ru/defaultx.asp;
3)электронная библиотечная система Поволжского государственного университета сервиса [Электронный ресурс]. – Режим доступа:http://elib.tolgas.ru;
4) электронно-библиотечная система Znanium.com [Электронный ресурс]. - Режим доступа:http://znanium.com;
5) открытое образование [Электронный ресурс]. - Режим доступа: https://openedu.ru;
6) polpred.com. Обзор СМИ. Полнотекстовая, многоотраслевая база данных (БД) [Электронный ресурс]. - Режим доступа:http://polpred.com;
7) базы данных Всероссийского института научной и технической информации (ВИНИТИ РАН) по естественным, точным и техническим наукам [Электронный ресурс]. - Режим доступа: http://www.viniti.ru;
8) университетская информационная система Россия [Электронный ресурс]. - Режим доступа: http://uisrussia.msu.ru.

7.2 Перечень программных продуктов, используемых при проведении различных видов занятий по дисциплине

Таблица 8. Перечень электронных библиотечных систем

№	Наименование ЭБС	Ссылка к ЭБС
1	Консультант студента	http://www.studentlibrary.ru/
2	Лань	https://e.lanbook.com/
3	Юрайт	https://biblio-online.ru/
4	TNT-ebook	https://www.tnt-ebook.ru/

7.3 Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства необходимого для освоения дисциплины

Таблица 9. Перечень программного обеспечения

Программное обеспечение, используемое в университете на договорной основе	Программное обеспечение свободного распространения
Microsoft Windows XP, Prof, S/P3 (подписка DreamSpark Premium, договор №Tr113003 от 25.09.2014)	Calculate Linux (свободное ПО)
Microsoft Windows 7 (подписка MSDN 4689, подписка DreamSparkPremium, договор № Tr113003 от 25.09.2014)	Open Office 4.1.1 (лицензия Apache License 2.0)
Visual Studio 2008 (подписка DreamSpark Premium, договор №Tr113003 от 25.09.2014)	Eclipse (открытое ПО, лицензия Eclipse Public License)

Перечень современных профессиональных баз данных и информационных справочных систем

В таблице 10 указан перечень профессиональных баз данных и информационных справочных систем, к которым обеспечен доступ (удаленный доступ). Данный перечень подлежит обновлению в соответствии с требованиями ФГОС ВО.

Таблица 10

Перечень современных профессиональных баз данных и информационных справочных систем

Наименование профессиональной базы данных, информационно-справочной системы	Доступ к ресурсу (удаленный доступ с указанием ссылки/доступ из локальной сети университета)
База данных стандартов и регламентов РОССТАНДАРТ	*https://www.gost.ru/portal/gost//home/standarts*
Электронная база избранных статей по философии	*http://www.philosophy.ru*
Единый архив экономических и социологических данных	*http://sophist.hse.ru/data_access.shtml*
Базы данных Национального совета по оценочной деятельности	*http://www.ncva.ru*
Информационно-справочная система «Техксперт»	доступ из локальной сети

8. ОБРАЗОВАТЕЛЬНЫЕ РЕСУРСЫ ДЛЯ ИНВАЛИДОВ И ЛИЦ С ОВЗ

В таблице11 указан перечень образовательных ресурсов, имеющих формы, адаптированные к ограничениям их здоровья, а также сведения о наличии специальных технических средств обучения коллективного и индивидуального пользования. При заполнении таблицы может быть использована информация, размещенная в подразделе «Доступная среда» специализированного раздела сайта НГТУ «Сведения об образовательной организации» https://www.nntu.ru/sveden/accenv/

Таблица 11. Образовательные ресурсы для инвалидов и лиц с ОВЗ

№	Перечень образовательных ресурсов, приспособленных для использования инвалидами и лицами с ОВЗ	Сведения о наличии специальных технических средств обучения коллективного и индивидуального пользования
1	ЭБС «Консультант студента»	озвучение книг и увеличение шрифта
2	ЭБС «Лань»	специальное мобильное приложение – синтезатор речи, который воспроизводит тексты книг и меню навигации
3	Образовательная платформа «Юрайт»	версия для слабовидящих

9. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ, НЕОБХОДИМОЕ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Учебные аудитории для проведения занятий по дисциплине, оснащены оборудованием и техническими средствами обучения.

В таблице 12 перечислены: учебные аудитории для проведения учебных занятий, оснащенные оборудованием и техническими средствами обучения; помещения для самостоятельной работы обучающихся, которые должны оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду НГТУ.

Таблица 12. Оснащенность аудиторий для проведения учебных занятий по дисциплине

Номер аудитории	Наименование аудиторий и помещений для самостоятельной работы	Оснащенность аудиторий помещений и помещений для самостоятельной работы	Перечень лицензионного программного обеспечения и реквизиты подтверждающего документа
6421	Мультимедийная аудитория № 6421 учебно-лабораторного корпуса № 6	1. Доска меловая – 1 шт. 3. Экран – 1 шт. 4. Мультимедийный проектор Epson X12 – 1 шт. 5. Компьютер PC MB Asus на чипсете Nvidia/AMDAthlonXII CPU 2.8Ggz/ RAM 4 Ggb/SVGAStandartGraphics +Ge-FORCE Nvidia GT210/HDD 250Ggb,SATAinterface, монитор 19”, с выходом на проектор. 6. Рабочее место студента - 74 7. Рабочее место для преподавателя – 1 шт.	1. Windows 7 32 bit корпоративная; VL 49477S2 2. Adobe Acrobat Reader DC-Russian (беспл.) 3. Microsoft Office Professional Plus 2007 (лицензия № 42470655); 4. Dr.Web (С/н 758S-TDJP-N7HB-ZH2F от 26.05.2025, до 31.05.26)
6543	Помещение для самостоятельной работы студентов (Компьютерный класс № 1) № 6543 учебно-лабораторного	1. Рабочие места студента, оснащенные ПК на базе Intel Core i5 с мониторами – 8 шт. 2. Рабочие места студента, оснащенные ПК на базеCore 2 Duo с	1. Microsoft Windows 7 MSDN реквизиты договора - подписка DreamSpark Premium, договор № 0509/КМР от 15.10.18 2. Бесплатное ПО:

	корпуса № 6	мониторами –2 шт. 3. Рабочее место преподавателя, оснащенное ПК на базе Intel Core i5 с монитором – 1 шт. 4. Проектор Accer, проекционный экран – 1 шт. ПК подключены к сети «Интернет» и обеспечивают доступ в электронную информационно-образовательную среду университета 5. Принтер HP LaserJet 1200 – 1 шт.	Пакет программ Open Office, True Conf, Браузер Google Chrome, Браузер Mozilla Firefox, Браузер Opera, McAfee Security Scan, Adobe Acrobat Reader DC, AutoCAD2013

10. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ОБУЧАЮЩИМСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Дисциплина реализуется посредством проведения контактной работы с обучающимися (включая проведение текущего контроля успеваемости), самостоятельной работы обучающихся и промежуточной аттестации.

Контактная работа: аудиторная, внеаудиторная, а также проводиться в электронной информационно-образовательной среде университета (ЭИОС).

Преподавание дисциплины ведется с применением следующих видов образовательных технологий: балльно-рейтинговая технология оценивания.

При преподавании дисциплины «Основы информатики», используются современные образовательные технологии, позволяющие повысить активность студентов при освоении материала курса и предоставить им возможность эффективно реализовать часы самостоятельной работы.

На лекциях, практических занятиях реализуются интерактивные технологии, приветствуются вопросы и обсуждения, используется личностно-ориентированный подход, технология работы в малых группах, что позволяет студентам проявить себя, получить навыки самостоятельного изучения материала, выровнять уровень знаний в группе.

Все вопросы, возникшие при самостоятельной работе над домашним заданием, подробно разбираются на практических занятиях и лекциях. Проводятся индивидуальные и групповые консультации с использованием, как встреч студентами, так и современных информационных технологий: электронная почта, zoom, google meet.

Инициируется активность студентов, поощряется задание любых вопросов по материалу, практикуется индивидуальный ответ на вопросы студента, рекомендуются методы успешного самостоятельного усвоения материала в зависимости от уровня его базовой подготовки.

Для оценки знаний, умений, навыков и уровня сформированности компетенции применяется балльно-рейтинговая система контроля и оценки успеваемости студентов в процессе текущего контроля.

Промежуточная аттестация проводится в форме экзамена с учётом текущей успеваемости.

Результат обучения считается сформированным на повышенном уровне, если теоретическое содержание курса освоено полностью. При устных собеседованиях студент исчерпывающе, последовательно, четко и логически излагает учебный материал; свободно справляется с задачами, вопросами и другими видами заданий, использует в ответе дополнительный материал. Все предусмотренные рабочей учебной программой задания выполнены в соответствии с установленными требованиями, студент способен анализировать полученные результаты, проявляет самостоятельность при выполнении заданий.

Результат обучения считается сформированным на пороговом уровне, если теоретическое содержание курса освоено полностью. При устных собеседованиях студент последовательно, четко и логически стройно излагает учебный материал; справляется с задачами, вопросами и другими видами заданий, требующих применения знаний; все предусмотренные

рабочей учебной программой задания выполнены в соответствии с установленными требованиями, студент способен анализировать полученные результаты; проявляет самостоятельность при выполнении заданий

Результат обучения считается несформированным, если студент при выполнении заданий не демонстрирует знаний учебного материала, допускает ошибки, неуверенно, с большими затруднениями выполняет задания, не демонстрирует необходимых умений, качество выполненных заданий не соответствует установленным требованиям, качество их выполнения оценено числом баллов ниже трех по оценочной системе, что соответствует допороговому уровню.

10.1. Методические указания для занятий лекционного типа

Лекционные занятия не предусмотрены учебным планом

10.2. Методические указания по освоению дисциплины на лабораторных работах

Подготовку к каждой лабораторной работе студент должен начать с ознакомления с планом занятия, который отражает содержание предложенной темы. Каждая выполненная работа с оформленным отчетом и подлежит защите у преподавателя.

При оценивании лабораторных работ учитывается следующее:

1. качество выполнения экспериментально-практической части работы и степень соответствия результатов работы заданным требованиям;
2. качество оформления отчета по работе;
3. качество устных ответов на контрольные вопросы при защите работы.

10.3. Методические указания по освоению дисциплины на практических занятиях

Практические занятия не предусмотрены учебным планом

10.4. Методические указания по самостоятельной работе обучающихся

Самостоятельная работа обеспечивает подготовку обучающегося к аудиторным занятиям и мероприятиям текущего контроля и промежуточной аттестации по изучаемой дисциплине. Результаты этой подготовки проявляются в активности обучающегося на занятиях и в качестве выполненных практических заданий и других форм текущего контроля.

При выполнении заданий для самостоятельной работы рекомендуется проработка материалов лекций по каждой пройденной теме, а также изучение рекомендуемой литературы, представленной в разделе 6. В процессе самостоятельной работы при изучении дисциплины студенты могут работать на компьютере в специализированных аудиториях для самостоятельной работы (указано в таблице 12). В аудиториях имеется доступ через информационно-телекоммуникационную сеть «Интернет» к электронной информационно-образовательной среде университета (ЭИОС) и электронной библиотечной системе (ЭБС), где в электронном виде располагаются учебные и учебно-методические материалы, которые могут быть использованы для самостоятельной работы при изучении дисциплины.

10.5. Методические указания для выполнения РГР

Расчетно-графические работы не предусмотрены учебным планом

10.6. Методические указания для выполнения курсового проекта/работы

Выполнение курсового проекта/работы не предусмотрено учебным планом.

11. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ КОНТРОЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

11.1. Оценочные средства для проведения текущего контроля успеваемости

Текущий контроль осуществляется по всем видам учебного процесса:

- лабораторные работы,
- контрольные работы.

1. Задания к лабораторным занятиям
    1. Вычислить результат в двоичном виде. Варианты:
        1.1. $1010_2 + 0101_2$
        1.2. $1111_2 + 0001_2$
        1.3. $1001_2 + 0110_2$
        1.4. $1100_2 + 0011_2$
        1.5. $0111_2 + 0111_2$
        1.6. $0001_2 + 1110_2$
        1.7. $1011_2 + 1001_2$
        1.8. $1101_2 + 1101_2$
        1.9. $10000001_2 + 00000001_2$
        1.10. $11110000_2 + 00001111_2$
    2. Установить свободно распространяемый дистрибутив Ubuntu второй системой или на виртуальную машину.
    3. Выполнить действия с файлом с помощью стандартных команд текстовой оболочки bash. Варианты:
        3.1. Создайте в текущем каталоге подкаталог с именем test_dir и перейдите в него.
        3.2. Внутри test_dir создайте пустой файл с именем file1.txt с помощью команды touch.
        3.3. Запишите строку "Hello, world!" в файл file1.txt с использованием echo и перенаправления.
        3.4. Создайте копию файла file1.txt с именем file2.txt в той же директории.
        3.5. Переименуйте файл file2.txt в file_renamed.txt.
        3.6. Переместите файл file_renamed.txt в родительский каталог.
        3.7. Удалите файл file1.txt.
        3.8. Создайте вложенные директории dir1/dir2/dir3 одной командой.
        3.9. Перейдите в каталог dir3 и выведите полный путь до вашей текущей директории.
        3.10. Выведите список всех файлов и папок текущего каталога, включая скрытые, в длинном формате и с указанием прав доступа.
    4. Реализовать скрипт bash, используя условные операторы и циклы bash. Варианты:
        4.1. Напишите скрипт, который запрашивает у пользователя число и выводит «Число чётное» или «Число нечётное» в зависимости от значения.
        4.2. Напишите скрипт, который сравнивает два введённых пользователем числа и выводит сообщение, какое из них больше (или что они равны).
        4.3. Скрипт запрашивает имя файла и проверяет, существует ли такой файл, и является ли он обычным файлом или каталогом.
        4.4. Используя конструкцию case, напишите скрипт, который принимает на вход день недели (например: "Mon", "Tue", "Sun") и выводит сообщение: будний день или выходной.
        4.5. Напишите скрипт, который выводит на экран все числа от 1 до 10 (используйте цикл for).
        4.6. Напишите скрипт, который выводит квадраты чисел от 1 до 5 с помощью цикла while.
        4.7. Скрипт, который запрашивает у пользователя пароль и сравнивает его с заранее заданной строкой. Если совпадает — вывести "Доступ разрешён", иначе — "Неверный пароль".
        4.8. Напишите скрипт, который считает сумму всех чисел от 1 до 100 и выводит результат.

4.9. Напишите скрипт, который перебирает все файлы в текущем каталоге и выводит их имена по одному в столбик.
4.10. Скрипт, который запрашивает числа у пользователя в бесконечном цикле, пока не будет введено 0. Скрипт должен складывать введенные числа и выводить результат после завершения ввода.

5. Реализуйте bash-скрипт, используя массивы чисел. Варианты:
5.1. Объявите массив из 5 произвольных чисел. Выведите весь массив на экран с помощью @.
5.2. Выведите отдельно первый, последний и третий элементы массива, созданного в первом задании.
5.3. Добавьте в конец массива новый элемент (например, 42) и выведите обновлённый массив.
5.4. Найдите длину массива (количество элементов) и выведите её на экран.
5.5. Напишите цикл for, который проходит по массиву и выводит каждый элемент на новой строке.
5.6. Напишите цикл, который считает сумму всех чисел в массиве.
5.7. Запросите у пользователя 5 строк и сохраните их в массив. Затем выведите этот массив на экран.
5.8. Отсортируйте числовой массив по возрастанию и выведите результат (можно использовать команду sort в сочетании с циклом).
5.9. Напишите скрипт, который проверяет наличие заданного пользователем значения в массиве и выводит сообщение «Найдено» или «Не найдено».
5.10. Создайте два массива и выполните их «слияние» в один третий массив. Выведите содержимое итогового массива.

6. Напишите bash скрипт, выполняющий работу с файлами. Варианты:
6.1. Прочитайте содержимое файла input.txt и выведите его на экран с помощью команды cat.
6.2. Выведите только первую строку файла input.txt (одним из возможных способов: head, read и т.д.).
6.3. Выведите только последнюю строку файла input.txt.
6.4. Считайте содержимое файла input.txt построчно с помощью цикла while и выведите строки на экран с пронумерованными строками (например: "1: строка1", "2: строка2", ...).
6.5. Запишите строку "Привет, мир!" в новый файл output.txt, перезаписав его содержимое.
6.6. Добавьте строку "Это новая строка." в конец файла output.txt, не удаляя уже имеющееся содержимое.
6.7. Считайте строки файла input.txt, которые содержат слово "bash", и запишите их в файл bash_lines.txt.
6.8. Напишите скрипт, который находит количество строк в файле input.txt и записывает это число в файл lines_count.txt.
6.9. Создайте файл numbers.txt, содержащий по одному числу в строке. Напишите скрипт, который читает этот файл и считает сумму всех чисел, записывая результат в файл sum.txt.
6.10. Напишите скрипт, который копирует все строки из одного файла в другой, но при этом удаляет все пустые строки (т.е. в новом файле только непустые строки из исходного).

7. Создайте функции в bash, передайте им аргументы и проверьте их наличие и корректность. Варианты:
7.1. Напишите функцию say_hello, которая выводит на экран "Привет, [имя]!", где имя передаётся как аргумент. Вызовите функцию с разными именами.
7.2. Напишите функцию, которая принимает два аргумента — числа — и выводит

их сумму.

7.3. Модифицируйте функцию из задания 2: если один из аргументов не передан, выводите сообщение об ошибке.

7.4. Напишите функцию, которая проверяет, передан ли хотя бы один аргумент, и выводит количество всех аргументов. Если аргументов нет, вывести "Нет аргументов".

7.5. Напишите функцию, которая принимает путь к файлу в виде аргумента и проверяет, существует ли этот файл. Если да — выводит "Файл существует", иначе — "Файл не найден".

7.6. Создайте функцию, которая принимает строку и число, и выводит эту строку указанное количество раз (цикл for).

7.7. Напишите функцию, которая принимает имя каталога и создаёт его, но только если он не существует. Если каталог уже есть — выводится сообщение.

7.8. Создайте функцию, которая отображает список всех аргументов с указанием их порядковых номеров (нумерация начинается с 1).

7.9. Напишите функцию, которая возвращает (через echo) длину переданной строки.

7.10. Напишите функцию, принимающую список чисел (аргументы) и определяющую максимальное из них.

8. Написать функциональный оболочечный скрипт с обработкой аргументов, вводом/выводом, условиями, циклами и работой с файловой системой. Варианты:

8.1. Написать скрипт, который через аргумент командной строки «-f» получает имя файла, и выводит его содержимое. Пример вызова: ./script arg1 -f filename arg2. Скрипт должен правильно определить положение ключа -f и извлечь имя файла, следующего за ним.

8.2. Написать скрипт, который запрашивает у пользователя имя файла, и создаёт его резервную копию с добавлением к имени текущей даты (например, data_2024-03-12.txt). Если файл не существует или копирование не удалось, вывести сообщение об ошибке.

8.3. Написать скрипт, который запрашивает путь к директории, проверяет её существование и тип, а затем выводит на экран количество обычных файлов и количество поддиректорий внутри неё. В случае ошибки — соответствующее сообщение.

8.4. Написать скрипт, который запрашивает у пользователя 1) имя файла; 2) строку для поиска. Скрипт должен вывести все строки из указанного файла, содержащие заданную строку. Если файл не найден — вывести сообщение об ошибке.

8.5. Написать скрипт, который перебирает все файлы в текущем каталоге и сортирует их по расширению: например, файлы .txt перемещаются в папку text/, .c — в code/, .jpg — в images/. Создаваемые каталоги не должны дублироваться, и команда должна срабатывать повторно корректно.

8.6. Написать скрипт, который принимает на вход пользовательское имя (например, логин) и проверяет, существует ли такой пользователь в системе (используя файл /etc/passwd). Вывести сообщение о результате проверки.

8.7. Написать скрипт, который принимает неограниченное количество чисел в качестве аргументов командной строки, и возвращает минимальное, максимальное и среднее значение. Если аргументы не переданы — вывести ошибку.

8.8. Написать скрипт, который получает имя процесса (например, bash, sshd) и проверяет, запущен ли он в данный момент. Если процесс найден — вывести его PID и количество копий, если нет — соответствующее сообщение.

8.9. Написать скрипт, который архивирует указанную директорию (запрашивается у пользователя) в файл с текущей датой в названии и расширением .tar.gz. Скрипт

должен проверять существование директории перед архивацией и сообщать об успехе или ошибке.

8.10. Написать скрипт, который проверяет все файлы в текущем каталоге на права доступа. Для каждого файла нужно вывести: имя, флаг на чтение, флаг на запись и исполнение для текущего пользователя.

9. Создайте репозиторий и коммит в нём. Варианты:
   9.1. Создайте новый каталог my_project и инициализируйте в нём новый Git-репозиторий.
   9.2. Создайте файл README.md с произвольным содержимым. Добавьте его в индекс с помощью git add и создайте первый коммит с сообщением "Initial commit".
   9.3. Измените содержимое файла README.md, добавьте новую строку текста, сохраните файл и выполните следующий коммит с пояснением изменений.
   9.4. Создайте несколько новых файлов (например, script.sh, notes.txt), добавьте только один из них (git add) и убедитесь с помощью git status, что другой остался неиндексированным.
   9.5. Удалите один из файлов проекта (например, notes.txt), зафиксируйте это изменение в следующем коммите с понятным сообщением.
   9.6. Используя команду git log, отобразите историю коммитов с краткими сообщениями. Затем выведите лог в виде однострочного списка с помощью ключей.
   9.7. Выполните откат изменений в рабочем файле (до последнего коммита), используя git checkout или git restore. Проверьте результат и объясните, что произошло.
   9.8. Создайте новый файл test.txt, запишите в него любой текст, затем добавьте его, выполните коммит с сообщением "Добавлен файл test.txt", а затем удалите последний коммит командой git reset —soft HEAD~1.
   9.9. Создайте новую ветку feature и переключитесь на неё. Внесите в проект любые изменения и сделайте коммит. Переключитесь обратно на ветку main (или master).
   9.10. Создайте конфликт: в ветке main и в ветке feature измените одну и ту же строку одного файла и попытайтесь объединить ветки через git merge. Разрешите конфликт вручную и завершите слияние.
10. Создайте ветку в git. Варианты:
   10.1. Создайте новый репозиторий в каталоге git-branches и выполните первый коммит с файлом README.md. Затем создайте новую ветку dev и переключитесь на неё.
   10.2. В ветке dev создайте файл dev.txt, добавьте в него произвольный текст и зафиксируйте изменения. Вернитесь на ветку main (или master) и убедитесь, что файл dev.txt в ней отсутствует.
   10.3. Создайте ветку feature от текущей основной ветки. Внесите в неё любые изменения и выполните коммит. Затем объедините ветку feature с основной с помощью git merge.
   10.4. В ветке main создайте файл conflict.txt и запишите в него строку A. Переключитесь на ветку dev и в том же файле conflict.txt запишите строку B. Попробуйте слить ветку dev в main и разрешите возникший конфликт вручную.
   10.5. Создайте цепочку коммитов в ветке test: 3 изменения в одном и том же файле. Затем выполните слияние ветки test в main с помощью fast-forward (если возможно). Просмотрите лог истории слияний с использованием git log --graph.
   10.6. Создайте и переключитесь в новую вспомогательную ветку cleanup. Удалите в этой ветке один из файлов проекта. Затем вернитесь в основную ветку и выполните слияние. Убедитесь, что файл был удалён в результате мёрджа.
   10.7. Создайте три ветки: ui, logic и docs — из ветки dev. В каждой ветке создайте по одному уникальному файлу. Поочередно объедините содержимое этих веток в ветку dev, затем в main.

10.8. Переименуйте ветку docs в documentation. Проверьте, что старого имени больше не существует, и вы можете переключиться на новую ветку.

10.9. Удалите локальную ветку test после того, как она была полностью объединена с основной. Предварительно убедитесь, что слияние завершено.

10.10. Используя git log --oneline --graph --all, отобразите историю всех веток в виде дерева. Проанализируйте структуру ветвлений и слияний, сделанных в рамках предыдущих заданий.

11. Создайте Makefile для сборки файла на языке C. Варианты:

11.1. Создайте простейший Makefile, который компилирует один файл main.c в исполняемый файл main. Используйте переменную для имени компилятора.

11.2. Добавьте ко вчерашнему проекту второй файл utils.c с функцией печати строки. Обновите Makefile так, чтобы проект собирался из двух файлов: main.c и utils.c.

11.3. Разделите проект на три файла: main.c, utils.c, math.c. Каждый должен содержать свою функцию. Обновите Makefile для сборки проекта из нескольких объектных файлов (.o).

11.4. В Makefile реализуйте отдельные правила компиляции для каждого .c-файла (например: utils.o: utils.c…). Используйте автоматические переменные (например: @,@,<).

11.5. Добавьте правило clean, которое удаляет все объектные файлы (.o) и итоговый исполняемый файл. Убедитесь, что команда make clean работает корректно.

11.6. Измените Makefile так, чтобы повторная сборка выполнялась только при изменении исходных файлов, а не при каждом вызове make.

11.7. Реализуйте в Makefile переменные CC (компилятор), CFLAGS (опции компиляции), OBJS (список объектных файлов) и используйте их в соответствующих местах. Попробуйте изменить уровень оптимизации с помощью CFLAGS.

11.8. Дополнительно добавьте в main.c заголовочный файл utils.h. Убедитесь, что при изменении только заголовочного файла make корректно пересобирает нужные части проекта (должна быть настроена зависимость .c-файлов от заголовков).

11.9. Включите условную сборку с помощью переменной DEBUG. Если переменная определена, добавляйте в CFLAGS флаг -g. Позвольте запуск make DEBUG=1 собирать проект в режиме отладки.

11.10. Убедитесь, что при запуске make без изменений в исходных файлах сборка не выполняется (make выводит сообщение, что все цели актуальны). Отредактируйте один из исходников и убедитесь, что только он пересобрался.

2. Задания к контрольным работам

Тема: арифметические операции в bash. Использование переменных

1. Объявите две переменные a и b, присвойте им значения 10 и 5 соответственно. Выведите их сумму.
2. Используя переменные, выполните вычитание значения b из a и выведите результат.
3. Вычислите произведение двух переменных a и b и сохраните результат в переменную result. Выведите переменную result на экран.
4. Найдите остаток от деления значения переменной a на b. Выведите результат в формате: "Остаток: X".
5. Запросите у пользователя два числа и сохраните их в переменные x и y. Посчитайте и выведите их среднее арифметическое, округлённое вниз до целого.
6. Объявите переменную n=7. Посчитайте и выведите квадрат числа n.
7. Объявите три переменные: x=15, y=4, z=2. Вычислите значение выражения (x + y) × z и выведите его.

8. Запросите у пользователя число и сохраните его в переменной num. Напишите условную конструкцию, которая выводит: "Чётное", если num делится на 2, иначе — "Нечётное".
9. Запросите у пользователя три числа. Найдите и выведите максимальное из них.
10. Объявите две переменные с числами и поменяйте их значения местами (не создавая третьей переменной).
11. Запросите у пользователя число и выведите факториал этого числа с использованием цикла while и переменной для накопления результата.
12. Напишите скрипт, который запрашивает радиус окружности, сохраняет его в переменную r и рассчитывает длину окружности по формуле $2\pi r$ (используйте $\pi \approx 3.14$).
13. Создайте переменную с текущим годом и вторую — с вашим годом рождения. Вычислите и выведите возраст.
14. Используя цикл for и переменную-счётчик, выведите таблицу умножения на 5 (от 1 до 10).
15. Напишите скрипт, в котором пользователь вводит секунды, а скрипт с помощью переменных переводит это в часы, минуты и секунды и выводит результат в формате: ЧЧ:ММ:СС.

Тема: работа с удалённым репозиторием Git.

1. Создайте новую папку с именем project-git-control и инициализируйте в ней новый локальный Git-репозиторий.
2. Создайте файл README.md с заголовком проекта и выполните первый коммит с сообщением "Initial commit".
3. Создайте новый файл main.txt, добавьте в него любую информацию (например, краткое описание), выполните git add и сделайте второй коммит с сообщением "Добавлен основной файл".
4. Проверьте статус репозитория (git status) и зафиксируйте его вывод (можно копировать в *.log файл или просто убедиться визуально).
5. Измените файл main.txt, добавив ещё одну строку. Воспользуйтесь командой git diff для просмотра различий. Затем зафиксируйте изменения в третьем коммите.
6. Создайте новую ветку feature-branch и переключитесь в неё. В этой ветке создайте файл feature.txt и выполните коммит.
7. Перейдите в ветку main и объедините в неё изменения из ветки feature-branch с помощью git merge.
8. Проверьте историю коммитов с помощью команды git log --oneline --graph. Убедитесь, что видно объединение веток.
9. Создайте .gitignore и добавьте туда имя временного файла temp.txt. Убедитесь, что git перестал его отслеживать (создайте temp.txt и проверьте git status).
10. Удалите ветку feature-branch после успешного слияния.
11. Отредактируйте README.md, добавив список файлов проекта. Выполните четвёртый коммит с комментарием "Обновлён README.md".
12. Используя git reset --soft HEAD~1, отмените последний коммит, оставив изменения рабочей директории нетронутыми.
13. Повторно закоммитьте отменённые изменения, используя другое сообщение коммита.
14. Создайте тег v1.0 на текущем состоянии проекта с помощью команды git tag. Выведите список всех тегов.
15. Создайте удалённый репозиторий на GitHub (например, с тем же именем project-git-control), подключите его к локальному с помощью git remote add origin https://... и выполните команду git push -u origin main. Убедитесь, что файлы и история появились в удалённом репозитории.

11.2. Оценочные материалы для проведения промежуточной аттестации, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

Форма проведения промежуточной аттестации по дисциплине: зачёт.

Пример билета для подготовки к зачёту:

1. Опишите основные этапы загрузки операционной системы.
2. Создайте bash-скрипт, который получает в качестве аргумента командной строки имя файла и выводит количество строк в этом файле. Проверить, что файл существует.