АКТ(ф) СПбГУТ

**ОТЧЕТ**

по практическим работам

по

МДК 01.03 Системное программирование

Студент ИСПП-21 19.06.2025 Э.А. Язиков

Преподаватель 19.06.2025 Р. В. Садовский

Архангельск 2025

**Лабораторная работа №1**

**Изучение процесса разработки линейных алгоритмов на языке**

**Ассемблера**

**1 Цель работы**

1.1 Изучить процесс разработки линейных алгоритмов на языке

ассемблера;

1.2 Научиться выполнять вычисление математических выражений на

языке ассемблера;

1.3 Закрепить навык отладки приложений в MS Visual Studio.

**2 Контрольные вопросы**

2.1 Для копирования данных между регистрами и памятью.

2.2 add, sub, mul, imul, div, idiv — для выполнения арифметических операций.

2.3 Сдвиги (shl, shr) для умножения/деления на степени двойки, применимы при делении/умножении на 2^n.

2.4 EAX — 4 байта, AX — 2 байта, AH — 1 байт, AL — 1 байт.

2.5 mov reg, 0 или xor reg, reg.

2.6 Можно, если значение помещается в 16 бит (регистры AX или его части).

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки линейных алгоритмов на языке

ассемблера;

3.2 В ходе практической работы выполнено вычисление математических выражений на языке ассемблера;

3.3 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №2**

**Изучение процесса разработки разветвляющихся алгоритмов**

**на языке ассемблера**

**1 Цель работы**

1.1 Изучить процесс разработки разветвляющихся алгоритмов на языке ассемблера;

1.2 Закрепить навык отладки приложений в MS Visual Studio

**2 Контрольные вопросы**

2.1 Логические команды: and, or, xor, not.

2.2 Команда cmp.

2.3 Условные переходы: je, jne, jg, jge, jl, jle, и др.

2.4 Безусловный переход: jmp.

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки разветвляющихся алгоритмов на языке ассемблера;

3.2 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №3**

**Изучение процесса разработки циклов со счетчиком на языке ассемблера**

**1 Цель работы**

1.1 Изучить процесс разработки циклов со счетчиком на языке ассемблера;

1.2 Закрепить навык отладки приложений в MS Visual Studio.

**2 Контрольные вопросы**

2.1 Для организации цикла со счетчиком нужны команды:

* MOV — для инициализации счетчика
* CMP — для сравнения счетчика с условием
* Jxx (например, JLE, JL, JGE, JNE) — для перехода по условию
* Метки (label) — для обозначения начала и конца цикла

2.2 Для досрочного выхода из цикла используют команду:

JMP — безусловный переход к выходу или завершению цикла, или команду JZ, JNZ при условии

2.3 Для увеличения значения счетчика:

* INC — увеличить на 1
* ADD — прибавить большее значение (например, ADD reg, value)

2.4 Для уменьшения значения счетчика:

* DEC — уменьшить на 1
* SUB — вычесть большее значение (например, SUB reg, value)

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки циклов со счетчиком на языке ассемблера;

3.2 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №4**

**Изучение процесса разработки разветвляющихся алгоритмов**

**на языке ассемблера**

**1 Цель работы**

1.1 Изучить процесс разработки циклов с предусловием и постусловием на языке ассемблера;

1.2 Закрепить навык отладки приложений в MS Visual Studio.

**2 Контрольные вопросы**

2.1 Минимум 0 раз (если условие изначально ложно).

2.2 Минимум 1 раз (цикл выполняется хотя бы один раз).

2.3 Проверка условия перед выполнением тела:

loop\_start:

; тело цикла

; проверка условия

jnz loop\_start

2.4 Выполнение тела перед проверкой условия:

loop\_body:

; тело цикла

; проверка условия

jnz loop\_body

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки циклов с предусловием и постусловием на языке ассемблера;

3.2 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №5**

**Изучение принципов работы математического сопроцессора**

**1 Цель работы**

1.1 Изучить принципы работы сопроцессора и методы его программирования средствами ассемблера.

1.2 Закрепить навык отладки приложений в MS Visual Studio.

**2 Контрольные вопросы**

2.1 Команда finit (или fninit) для инициализации сопроцессора.

2.2 Команды fld для загрузки констант и чисел в регистры сопроцессора.

2.3 ST(0) — регистр сопроцессора, основной регистр для операций с плавающей точкой.

2.4 Арифметические команды: fadd, fsub, fmul, fdiv, а также функции для тригонометрии, логарифмов и др.

2.5 Команды передачи данных: fld (загрузка), fst/fstp (выгрузка), fxch (обмен регистров).

**3 Вывод**

3.1 В ходе практической работы изучены принципы работы сопроцессора и методы его программирования средствами ассемблера.

3.2 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №6**

**Изучение принципов работы цепочечных команд**

**1 Цель работы**

1.1 Изучить принципы обработки цепочек данных средствами ассемблера.

1.2 Закрепить навык отладки приложений в MS Visual Studio.

**2 Контрольные вопросы**

2.1 Флаг DF (Direction Flag) определяет направление обработки цепочек: при установке DF операции идут в обратном порядке, при сбросе — вперёд.

2.2 Префиксы (например, rep) используются для автоматического повторения операций с цепочками данных.

2.3 Команды cmpsb, cmpsw, cmpsd (или cmpsq) для сравнения цепочек данных.

2.4 Размер операндов: байты (8 бит), слова (16 бит), double слова (32 бит), quad слова (64 бит).

2.5 Регистры SI/ESI — источник, DI/EDI — приемник.

**3 Вывод**

3.1 В ходе практической работы изучены принципы обработки цепочек данных средствами ассемблера.

3.2 В ходе практической работы закреплен навык отладки приложений в MS Visual Studio.

**Лабораторная работа №7**

**Изучение процесса разработки модулей на языке ассемблера**

**1 Цель работы**

* 1. Изучить процесс разработки модулей с использованием ассемблера MASM.

**2 Контрольные вопросы**

2.1 Этапы разработки: написание кода, ассемблирование, связывание, выполнение.

2.2 Программа состоит из сегментов: .data (данные), .code (код), .stack (стек). Объявляются с помощью директив: segment, ends.

2.3 Целое число: db, dw, dd; строка: db "строка", 0; массив: последовательность данных в сегменте .data.

2.4 Сегмент кода содержит инструкции программы, выполняемые процессором.

2.5 MASM (Microsoft Macro Assembler) — ассемблер для Windows, поддерживающий макросы и высокоуровневые конструкции.

2.6 Вызов процедуры: call имя\_процедуры. Процедура объявляется с меткой и директивой proc.

2.7 Передача параметров: через стек (push/pop) или через регистры (ax, bx, и т.д.).

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки модулей с использованием ассемблера MASM.

**Лабораторная работа №8**

**Дизассемблирование приложений**

**1 Цель работы**

* 1. Научиться применять дизассемблеры для изучения и модификации ПО.

**2 Контрольные вопросы**

2 1 Дизассемблирование — процесс преобразования машинного кода (бинарных инструкций) обратно в ассемблерный код.

2.2 Выполняется для анализа программ, поиска уязвимостей, отладки, реверс-инжиниринга и изучения работы программ.

2.3 Дизассемблер — программа, которая выполняет дизассемблирование, переводя бинарный код в ассемблерные инструкции.

2.4 Примеры: IDA Pro, Ghidra, OllyDbg, x64dbg, Radare2.

2.6 Обфускация кода выполняется для усложнения анализа программы и защиты от реверс-инжиниринга.

**3 Вывод**

3.2 В ходе практической работы применяны дизассемблеры для изучения и модификации ПО.

**Лабораторная работа №9**

**Создание проекта в эмуляторе Arduino**

**1 Цель работы**

1.1 Научиться создавать имитационные модели проектов, работающих под управлением микроконтроллеров для Arduino;

1.2 Получить навыки работы с платой Arduino Uno и макетной платой

**2 Контрольные вопросы**

2. 1 Arduino Uno — это микроконтроллерная плата, предназначенная для разработки и прототипирования электронных устройств и проектов на базе микроконтроллера ATmega328P. Используется для автоматизации, робототехники, обучения электронике и программирования.

2.2 Макетная плата (breadboard) — это пластиковая плата с множеством соединительных контактов, предназначенная для быстрого и удобного монтажа электронных схем без пайки.

2.3 Рельсы (или линии) на макетной плате:

Верхние и нижние рельсы обычно предназначены для подачи питания (например, +5V и GND).

Центральные рельсы — для размещения компонентов и соединений цепи.

2.4 Для последовательного соединения элементов: соедините выводы элементов один за другим по цепи, чтобы ток проходил через все последовательно.

2.5 Для параллельного соединения: соедините все соответствующие выводы элементов между собой, чтобы они имели общий источник питания или общий узел.

2.6 Подключение элементов должно выполняться в логическом порядке: сначала подключите питание и землю, затем компоненты согласно схеме, проверяя правильность соединений перед подачей питания.

**3 Вывод**

3.1 В ходе практической работы создаваны имитационные модели проектов, работающих под управлением микроконтроллеров для Arduino;

3.2 В ходе практической работы получены навыки работы с платой Arduino Uno и макетной платой

**Лабораторная работа №10**

**Разработка скетчей для Arduino**

**1 Цель работы**

1.1Научиться создавать скетчи (программное обеспечение на языке Си)

для микроконтроллеров Arduino;

1.2 Получить навыки работы с платой Arduino Uno и макетной платой

**2 Контрольные вопросы**

2.1 В Arduino функции setup() и loop() предназначены для инициализации и выполнения основной программы.

setup() вызывается один раз при запуске программы и используется для настройки (например, установка режимов пинов).

loop() вызывается постоянно после setup() и содержит основной код, который повторяется бесконечно.

2.2 pinMode(pin, mode) используется для задания режима работы пина:

Параметры: pin — номер пина, mode — режим (INPUT, OUTPUT, или INPUT\_PULLUP).

2.3 digitalWrite(pin, value) управляет состоянием цифрового пина:

Параметры: pin — номер пина, value — уровень сигнала (HIGH или LOW).

2.4 Для задержки используют функции:

delay(ms) — задержка в миллисекундах (принимает параметр ms).

Также есть delayMicroseconds(us) для микросекунд.

2.5 «Скетч» — это программа для Arduino, написанная на языке Arduino (C/C++), которая загружается в плату и выполняется.

2.6 Библиотеки подключаются с помощью директивы #include

**3 Вывод**

* 1. В ходе практической работы созданы скетчи (программное обеспечение на языке Си) для микроконтроллеров Arduino;

1.2 В ходе практической работы получены навыки работы с платой Arduino Uno и макетной платой

**Лабораторная работа №11**

**Разработка приложений для обработки файлов**

**1 Цель работы**

1.1 Научиться использовать файловые потоки в приложении на C#;

1.2 Научиться применять классы для работы с файлами в приложениях на

C#.

**2 Контрольные вопросы**

2.1 Пространство имен, содержащее классы для работы с файловой системой и файловыми потоками, — System.IO.

2.2 Классы StreamReader и StreamWriter предназначены для чтения из текстовых файлов и записи в текстовые файлы соответственно.

2.3 Классы BinaryReader и BinaryWriter используются для чтения и записи данных в бинарном формате, что позволяет работать с двоичными данными (числами, байтами и т.п.).

2.4 Информацию о файлах предоставляют классы: File, FileInfo, Directory, DirectoryInfo.

**3 Вывод**

3.1 В ходе практической использованы файловые потоки в приложении на C#;

3.2 В ходе практической работы применены классы для работы с файлами в приложениях на

C#.

**Лабораторная работа №12**

**Разработка приложений для поиска файлов**

**1 Цель работы**

* 1. Научиться применять классы для работы с файлами и каталогами в приложениях на C#.

**2 Контрольные вопросы**

2 1. Directory — статический класс, предоставляет статические методы для работы с каталогами (например, создание, удаление, получение списков).

DirectoryInfo — класс экземпляра, представляет конкретный каталог и позволяет получать его свойства и выполнять операции через объект.

2.2 File — статический класс с методами для работы с файлами (например, создание, удаление, копирование).

FileInfo — класс экземпляра, содержит свойства и методы для получения информации о файле и его управления через объект.

2.3 Используя методы:

Directory.GetFiles(path); // список файлов

Directory.GetDirectories(path); // список папок

или комбинированный: Directory.GetFileSystemEntries(path);

2.4 Используя параметры методов:

Directory.GetFiles(path, "\*.txt", SearchOption.AllDirectories);

где "\*.txt" — шаблон поиска, а SearchOption.AllDirectories — опция рекурсивного поиска.

2.5 Некоторые свойства:

Name — имя файла;

FullName — полный путь;

Length — размер файла в байтах;

CreationTime, LastAccessTime, LastWriteTime — даты создания/последнего доступа/изменения;

Extension — расширение файла.

**3 Вывод**

3.1 В ходе практической работы применены классы для работы с файлами и каталогами в приложениях на C#.

**Лабораторная работа №13**

**Разработка приложений для сортировки файлов**

**1 Цель работы**

1.1 Научиться применять классы для работы с файлами и каталогами в приложениях на C#.

**2 Контрольные вопросы**

2.1 Как проверить существование файла по его имени?

Используйте свойство File.Exists(path) — возвращает true, если файл существует.

2.2 Как проверить существование каталога по его имени?

Используйте свойство Directory.Exists(path) — возвращает true, если каталог существует.

2.3 Какие методы позволяют создавать, удалять, копировать и переносить каталоги?

Создавать: Directory.CreateDirectory(path)

Удалять: Directory.Delete(path, recursive) (второй параметр — рекурсивное удаление)

Копировать — напрямую нельзя, нужно вручную создавать новый каталог и копировать содержимое

Переносить (переименовать): Directory.Move(sourceDirName, destDirName)

2.4 Какие методы позволяют создавать, удалять, копировать и переносить файлы?

Создавать: File.Create(path) или File.Copy(source, destination) для копирования

Удалять: File.Delete(path)

Копировать: File.Copy(source, destination)

Переносить (переименовать): File.Move(source, destination)

**3 Вывод**

3.1 В ходе практической работы применены классы для работы с файлами и каталогами в приложениях на C#.

**Лабораторная работа №14**

**Разработка утилиты для поиска дубликатов файлов**

**1 Цель работы**

1.1 Научиться получать и анализировать информацию о файлах в приложениях на C#.

**2 Контрольные вопросы**

2.1 Какое свойство FileInfo возвращает имя файла?

Name — возвращает имя файла без пути.

2.2 Какое свойство FileInfo возвращает расширение файла?

Extension — возвращает расширение файла, включая точку (например, .txt).

2.3 Какое свойство FileInfo возвращает полное имя файла?

FullName — возвращает полный путь к файлу с именем.

2.4 Какое свойство FileInfo возвращает дату изменения файла?

LastWriteTime — дата и время последнего изменения файла.

**3 Вывод**

3.1 В ходе практической получены и анализировать информацию о файлах в приложениях на C#

**Лабораторная работа №15**

**Разработка утилиты для просмотра изображений**

**1 Цель работы**

1.1 Научиться отображать файлы-изображения в приложениях на C#.

**2 Контрольные вопросы**

2.1 Для чего предназначен класс BitmapImage?

BitmapImage — класс, используемый для загрузки, хранения и отображения растровых изображений (например, из файла или URI) в приложениях WPF. Он служит источником изображения для элементов управления, таких как Image.

2.2 Для чего предназначен элемент управления Image?

Image — элемент управления WPF, предназначенный для отображения графических изображений (например, BitmapImage) в интерфейсе пользователя.

2.3 Для чего предназначен элемент управления ScrollView?

ScrollViewer — контейнер, который добавляет прокрутку к содержимому, если оно превышает размеры области отображения. Позволяет прокручивать содержимое по горизонтали и/или вертикали.

2.4 Как получить высоту и ширину изображения BitmapImage?

Используя свойства:

PixelWidth — ширина изображения в пикселях

PixelHeight — высота изображения в пикселях

**3 Вывод**

3.1 В ходе практической работы отображены файлы-изображения в приложениях на C#.

**Лабораторная работа №16**

**Разработка утилиты Диспетчер задач**

**1 Цель работы**

* 1. Научиться получать и отображать информацию о запущенных процессах в приложениях на C#.

**2 Контрольные вопросы**

2.1 Какое пространство имен требуется подключить для работы с процессами?

System.Diagnostics

2.2 Какой метод запускает процесс?

Process.Start()

2.3 Какой метод завершает процесс?

Process.Kill()

2.4 Как получить список запущенных процессов?

Process.GetProcesses()

2.5 Как получить процесс по его идентификатору?

Process.GetProcessById(id)

**3 Вывод**

3.1 В ходе практической работы получать и отображать информацию о запущенных процессах в приложениях на C#.

**Лабораторная работа №17**

**Разработка утилиты Файловый менеджер**

**1 Цель работы**

1.1 Научиться применять элементы управления для отображения файлов

и папок в приложениях на C#.

**2 Контрольные вопросы**

2.1 Для получения списка дисков используйте класс DriveInfo из пространства имен System.IO, например: DriveInfo.GetDrives()

2.2 Для получения списка файлов используйте метод Directory.GetFiles(path) из System.IO.

2.3 Элемент управления TreeView предназначен для отображения иерархических данных в виде дерева, например, папок и файлов.

**3 Вывод**

3.1 В ходе практической применены элементы управления для отображения файлов и папок в приложениях на C#..

**Лабораторная работа №18**

**Разработка утилиты для анализа дисков**

**1 Цель работы**

* 1. Научиться получать и отображать статистическую информацию о дисковом пространстве в приложениях на C#.

**2 Контрольные вопросы**

2.1 Класс DriveInfo предоставляет информацию о дисках.

2.2 Класс FileInfo предоставляет информацию о файлах.

2.3 Метод Sum() из LINQ позволяет просуммировать значения в списке.

2.4 Методы OrderBy() и OrderByDescending() позволяют отсортировать список по возрастанию и убыванию соответственно.

**3 Вывод**

3.1 В ходе практической получена и отображена статистическая информацию о дисковом пространстве в приложениях на C#.

**Лабораторная работа №19**

**Разработка утилиты Архиватор**

**1 Цель работы**

1.1 Научиться применять классы для работы с архивами в приложениях

на C#.

**2 Контрольные вопросы**

2.1 System.IO.Compression и System.IO.Compression.FileSystem

2.2 Добавьте ссылку на сборку System.IO.Compression.FileSystem (в проекте Visual Studio через менеджер ссылок).

2.3 ZipArchive — класс, представляющий ZIP-архив, позволяет читать, создавать и изменять содержимое ZIP-файлов.

2.4 ZipFile — статический класс, предоставляющий методы для создания, извлечения и управления ZIP-архивами (например, ExtractToDirectory, CreateFromDirectory).

2.5 ZipArchiveEntry — класс, представляющий отдельный файл или папку внутри ZIP-архива; используется для доступа к содержимому и свойствам конкретной записи в архиве.

**3 Вывод**

3.1 В ходе практической применены классы для работы с архивами в приложениях на C#.

**Лабораторная работа №20**

**Разработка утилиты скринсейвер**

**1 Цель работы**

1.1 Научиться разрабатывать оконные приложения-заставки, используя таймер.

**2 Контрольные вопросы**

2.1 Это небольшая программа или инструмент, предназначенный для выполнения определённых задач или функций в системе, например, для настройки, обслуживания или автоматизации.

2.2 Это программа, которая запускается автоматически после периода бездействия пользователя и отображает анимацию или изображение для защиты экрана и предотвращения выгорания дисплея.

2.3 Файлы-заставки имеют расширение .scr.

**3 Вывод**

3.1 В ходе практической разработаны оконные приложения-заставки, используя таймер.

**Лабораторная работа №21**

**Разработка утилиты для вычисления хэш-суммы файлов**

**1 Цель работы**

1.1 Научиться вычислять хэш-суммы файлов в приложениях на C#,

используя встроенные алгоритмы шифрования.

**2 Контрольные вопросы**

2.1 Хэш-сумма — это уникальное фиксированной длины значение, полученное в результате применения хэш-функции к данным (например, файлу), служащее для проверки целостности и аутентичности.

2.2 Они используются для проверки целостности файла, обнаружения изменений или повреждений, а также для сравнения файлов на идентичность.

2.3 Наиболее распространённые алгоритмы:

MD5

SHA-1

SHA-256

SHA-512

**3 Вывод**

3.1 В ходе практической вычислены хэш-суммы файлов в приложениях на C#, используя встроенные алгоритмы шифрования.

**Лабораторная работа №22**

**Разработка утилиты Менеджер паролей**

**1 Цель работы**

1.1 Научиться выполнять шифрование и дешифрование данных в приложениях на C#, используя встроенные алгоритмы шифрования.

**2 Контрольные вопросы**

2.1 Шифрование — процесс преобразования исходных данных (открытого текста) в зашифрованный вид (шифротекст) с помощью алгоритма и ключа, чтобы обеспечить конфиденциальность.

2.2 Дешифрование — обратный процесс, преобразование зашифрованных данных обратно в исходный открытый текст с помощью алгоритма и ключа.

2.3 AES (Advanced Encryption Standard) — современный симметричный алгоритм шифрования, широко используемый для защиты данных.

2.4 Основные размеры ключей: 128, 192 и 256 бит.

2.5 System.Security.Cryptography

**3 Вывод**

3.1 В ходе практической работы выполнены шифрование и дешифрование данных в приложениях на C#, используя встроенные алгоритмы шифрования.

**Лабораторная работа №23**

**Изучение процесса разработки DLL**

**1 Цель работы**

1.1 Изучить процесс разработки библиотек динамической компоновки на

С++.

1.2 Научиться применять библиотеки динамической компоновки

написанные на других языках программирования в C#.

**2 Контрольные вопросы**

2.1 Это библиотека, которая загружается в память во время выполнения программы и используется для предоставления функций или ресурсов, например, DLL в Windows.

2.2 .dll — Dynamic Link Library (динамическая библиотека).

2.3 Visual Studio: добавьте ссылку на .dll через меню «Добавить ссылку» или настройте проект для использования внешней библиотеки.

Для вызова функций из DLL используйте платформенный вызов (DllImport) в коде C# или соответствующие механизмы в других языках.

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки библиотек динамической компоновки на С++.

3.2 В ходе практической применены библиотеки динамической компоновки

написанные на других языках программирования в C#.

**Лабораторная работа №24**

**Использование потоков**

**1 Цель работы**

1.1 Научиться разрабатывать многопоточные приложения на C#;

1.2 Научиться создавать и применять фоновые и основные потоки и

выполнять обмен данных между ними в программах на C#.

**2 Контрольные вопросы**

2.1 Процессы — независимые единицы с собственными ресурсами; потоки — легкие части процесса, делящие ресурсы.

2.2 Многопоточность повышает скорость, отзывчивость и эффективность приложений.

2.3 Основные средства синхронизации — lock, Mutex, Semaphore, AutoResetEvent.

2.4 На однопроцессорных системах потоки выполняются через быстрое переключение контекста.

2.5 В C# класс Thread используется для создания и управления потоками.

**3 Вывод**

3.1 В ходе практической работы разработаны многопоточные приложения на C#;

3.2 В ходе практической работы созданы и применять фоновые и основные потоки и выполнять обмен данных между ними в программах на C#.

**Лабораторная работа №25**

**Разработка сетевого приложения с использованием сокетов**

**1 Цель работы**

1.1 Научиться реализовывать и запускать асинхронные операции на C#;

1.2 Научиться выполнять вычисления, используя асинхронные операции

на C#.

**2 Контрольные вопросы**

2.1 Сокет — это точка соединения для обмена данными между компьютерами по сети, обеспечивающая взаимодействие приложений.

2.2 Алгоритм работы сервера:

* Создать сокет
* Назначить адрес и порт (bind)
* Перевести в режим прослушивания (listen)
* Принять входящее соединение (accept)
* Обмениваться данными с клиентом

2.3 Алгоритм работы клиента:

* Создать сокет
* Установить соединение с сервером (connect)
* Обмениваться данными

2.4 Пространство имен: System.Net.Sockets

2.5 Параметры при создании сокета:

* Адресный семейство (AddressFamily) — например, InterNetwork для IPv4
* Тип сокета (SocketType) — например, Stream для TCP
* Протокол (ProtocolType) — например, Tcp

2.6 Получение данных:

* Использовать метод Receive для чтения данных из сокета в буфер

**3 Вывод**

3.1 В ходе практической работы реализованы и запущены асинхронные операции на C#;

3.2 В ходе практической работы выполнены вычисления, используя асинхронные операции на C#.

**Лабораторная работа №26**

**Сетевое программирование сокетов**

**1 Цель работы**

1.1 Закрепить навыки работы с сетевыми сокетами на C#;

**2 Контрольные вопросы**

2.1 TCP — протокол с установлением соединения, обеспечивает надежную доставку данных, порядок и контроль ошибок.

UDP — протокол без установления соединения, быстрее, но не гарантирует доставку или порядок данных.

2.2 Алгоритм работы сервера:

* Создать сокет
* Назначить адрес и порт (bind)
* Перевести в режим прослушивания (listen)
* Принять входящее соединение (accept) (для TCP) или ждать сообщений (для UDP)
* Обмениваться данными с клиентом

**3 Вывод**

3.1 В ходе практической работы закреплены навыки работы с сетевыми сокетами на C#.

**Лабораторная работа №27**

**Разработка многопоточных приложений**

**1 Цель работы**

1.1 Изучить процесс разработки многопоточных приложений на C#;

**2 Контрольные вопросы**

2.1 Класс Parallel предоставляет методы For, ForEach, Invoke для параллельной обработки данных.

2.2 Синхронизация потоков и потокобезопасные операции нужны для предотвращения конфликтов, ошибок и некорректных данных при одновременном доступе нескольких потоков к одним ресурсам.

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки многопоточных приложений на C#.

**Лабораторная работа №28**

**Обмен данными**

**1 Цель работы**

1.1 Изучить процесс разработки сетевых приложений с использованием

SignalR на C#;

**2 Контрольные вопросы**

2.1 SignalR — это библиотека для ASP.NET, которая обеспечивает реальное время обмена данными между клиентами и сервером, автоматически управляя соединениями и обновлениями.

2.2 Обмен данными осуществляется через постоянное соединение (например, WebSocket), позволяющее серверу и клиентам отправлять сообщения друг другу в реальном времени.

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки сетевых приложений с использованием.

**Лабораторная работа №29**

**Работа с буфером экрана**

**1 Цель работы**

1.1 Изучить процесс разработки консольных приложений с продвинутым

пользовательским интерфейсом;

**2 Контрольные вопросы**

2.1 В Terminal.Gui используются основные элементы управления: Label, Button, TextField, CheckBox, RadioGroup, ListView, FrameView и другие.

2.2 Диалоговые окна реализуются с помощью создания новых окон (Window) или диалогов (Dialog), которые отображаются поверх основного интерфейса и могут содержать элементы управления для взаимодействия.

2.3 Цветовую схему можно настроить через свойства ColorScheme или установить цвета для элементов управления, задавая цвета фона и текста, а также применяя темы или собственные стили.

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки консольных приложений с продвинутым пользовательским интерфейсом.

**Лабораторная работа №30**

**Разработка приложения для загрузки и отправки данных по**

**сети**

**1 Цель работы**

1.1 Изучить процесс разработки приложений с использованием технологии

gRPC;

**2 Контрольные вопросы**

2.1 Преимущества gRPC:

* Быстрая и эффективная передача данных (использует Protocol Buffers)
* Поддержка нескольких языков программирования
* Простота определения интерфейсов и автоматическая генерация кода
* Поддержка потоковой передачи и асинхронных вызовов

Недостатки gRPC:

* Требует поддержки Protocol Buffers, что усложняет настройку для новичков
* Не так широко распространен, как REST, и может быть сложнее в реализации для некоторых задач
* Ограничен в браузерах (не работает напрямую без прокси или WebSocket)

2.2 Файлы .proto используются для определения интерфейсов сервисов, сообщений и структур данных в gRPC. Они служат контрактом между клиентом и сервером, на основе которых автоматически генерируется код для взаимодействия

**3 Вывод**

3.1 В ходе практической работы изучен процесс разработки приложений с использованием технологии

gRPC.