**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным работам**

**по СП**

Студент ИСПП-21 22.06.25 Казнин В.А.

Преподаватель 22.06.25 Садовский Р.В.

Архангельск 2025

**Лабораторная работа №1  
Изучение процесса разработки линейных алгоритмов на языке ассемблера**

**Цель работы**

* Изучить процесс разработки линейных алгоритмов на языке ассемблера;
* Научиться выполнять вычисление математических выражений на языке ассемблера;
* Закрепить навык отладки приложений в MS Visual Studio
* **Ответы на контрольные вопросы**

* Для чего применяется команда mov?

Ответ: команда MOV используется для копирования данных из одного места в другое. Она переносит значение из регистра, памяти или константы в регистр или ячейку памяти. Эта команда не изменяет источник, только копирует значение.

* Какие арифметические команды применяются в языке ассемблера и какое у них назначение?

Ответ: в языке ассемблера применяются следующие арифметические команды:

ADD – сложение;

SUB – вычитание;

INC – увеличение на 1;

DEC – уменьшение на 1;

MUL/IMUL – умножение (без знака и со знаком);

DIV/IDIV – деление (без знака и со знаком).

Эти команды выполняют базовые математические операции с регистрами и переменными.

* Какие побитовые команды применяются для ускорения умножения и деления и когда они применимы?

Ответ: для ускорения операций умножения и деления применяются побитовые команды SHL (сдвиг влево) и SHR (сдвиг вправо). Сдвиг влево на один бит эквивалентен умножению на 2, а сдвиг вправо – делению на 2. Эти команды эффективны, когда множитель или делитель – степень двойки.

* Каков размер в байтах регистров общего назначения EAX, AX, AH, AL?

Ответ: размеры регистров общего назначения:

EAX – 4 байта (32 бита);

AX– 2 байта (16 бит);

AH и AL – по 1 байту (8 бит), где AH – старший байт AX, а AL – младший.

* какие способы обнуления регистров могут применяться в языке ассемблера?

Ответ: регистры можно обнулить разными способами:

XOR AX, AX – быстрый и часто используемый способ;

MOV AX, 0 – прямое присваивание нуля;

SUB AX, AX – вычитание самого регистра из самого себя.

* Можно ли записать значение переменной типа int в регистр AX и почему?

Ответ: записать значение переменной типа int в регистр AX можно только в том случае, если размер переменной соответствует 16 битам, поскольку AX – 16-битный регистр. Если переменная типа int имеет размер 32 бита, её следует записывать в EAX, иначе произойдёт потеря старших битов.

**Лабораторная работа №2**

**Изучение процесса разработки разветвляющихся алгоритмов на языке ассемблера**

* **Цель работы**
* Изучить процесс разработки разветвляющихся алгоритмов на языке ассемблера;
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Какие логические команды поддерживаются в языке asm?

Ответ: в языке ассемблера поддерживаются различные логические команды, которые выполняют побитовые операции над операндами. К ним относятся команды AND, OR, XOR, NOT и TEST. Команда AND выполняет логическую операцию И, OR — логическую операцию ИЛИ, XOR — исключающее ИЛИ, NOT — побитовую инверсию, а команда TEST проверяет результат побитовой операции И, не изменяя содержимое операндов, но устанавливая соответствующие флаги.

* Какая команда осуществляет сравнение двух операндов?

Ответ: для сравнения двух операндов в языке ассемблера используется команда CMP. Эта команда вычитает второй операнд из первого, не сохраняя результат, но изменяет флаги процессора, которые затем используются для выполнения условных переходов.

* Какие операторы условного перехода имеются в языке asm?

Ответ: условные переходы в языке ассемблера осуществляются при помощи операторов, которые анализируют состояние флагов после выполнения логических или арифметических команд. В зависимости от этих флагов можно выполнить переход по метке. Примерами таких операторов являются JE (или JZ) — переход при равенстве, JNE (или JNZ) — переход при неравенстве, а также JG, JL, JGE, JLE и другие, которые используются для знаковых и беззнаковых сравнений.

* Какой оператор безусловного перехода имеется в языке asm?

Ответ: для выполнения безусловного перехода в языке ассемблера используется оператор JMP. Эта команда позволяет перейти к указанной метке или адресу, независимо от состояния каких-либо флагов, и тем самым изменить порядок выполнения программы.

* **Вывод**
* В ходе выполнения работы была достигнута поставленная цель — изучены принципы разработки разветвляющихся алгоритмов на языке ассемблера.
* Также был закреплён навык отладки приложений в среде MS Visual Studio, что позволило на практике освоить применение условных и безусловных переходов, логических команд и методов анализа флагов процессора.

**Лабораторная работа №4**

**Изучение процесса разработки разветвляющихся алгоритмов на языке ассемблера**

* **Цель работы**
* Изучить процесс разработки разветвляющихся алгоритмов на языке ассемблера;
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Какое минимальное количество раз выполняется цикл с предусловием?

Ответ: цикл с предусловием выполняется только в том случае, если условие истинно уже при первой проверке. Поэтому минимальное количество выполнений такого цикла – ноль раз, если условие изначально ложно.

* Какое минимальное количество раз выполняется цикл с постусловием?

Ответ: в отличие от цикла с предусловием, цикл с постусловием выполняет тело хотя бы один раз, так как проверка условия осуществляется после выполнения тела цикла. Таким образом, минимальное количество выполнений цикла с постусловием — один раз.

* Какова общая форма цикла с предусловием на языке ассемблера?

Ответ: Общая форма цикла с предусловием на языке ассемблера реализуется с помощью команды безусловного перехода и условной проверки перед телом цикла.

* Какова общая форма цикла с постусловием на языке ассемблера?

Ответ: цикл с постусловием в ассемблере организуется так, что сначала выполняется тело цикла, а затем — проверка условия, определяющая необходимость следующей итерации.

* **Вывод**
* В ходе выполнения работы была достигнута поставленная цель – изучены принципы разработки разветвляющихся алгоритмов на языке ассемблера.
* Также был закреплён навык отладки приложений в среде MS Visual Studio, что позволило на практике освоить применение условных и безусловных переходов, логических команд и методов анализа флагов процессора.

**Лабораторная работа №5  
Изучение принципов работы математического сопроцессора**

* **Цель работы**
* Изучить принципы работы сопроцессора и методы его программирования средствами ассемблера.
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Какая команда используется для инициализации сопроцессора?

Ответ: для инициализации сопроцессора используется команда FINIT (или FNINIT). Она приводит сопроцессор в исходное состояние, очищает регистры и сбрасывает управляющие флаги.

* Какие команды сопроцессора используются для загрузки констант?

Ответ: для загрузки констант в сопроцессор применяются команды FLD1 (загрузка значения 1.0), FLDZ (загрузка нуля), FLDPI (число π), FLDL2T (логарифм 2 по основанию 10), FLDL2E (логарифм 2 по основанию e) и другие. Эти команды помещают заранее определённые константы в стек сопроцессора.

* Что такое ST (0)?

Ответ: ST (0) — это вершина стека сопроцессора. Сопроцессор использует стековую архитектуру из восьми регистров (от ST(0) до ST(7)), где ST(0) указывает на текущий верхний элемент, используемый большинством операций.

* Какие арифметические команды имеются в сопроцессоре?

Ответ: среди арифметических команд сопроцессора — команды для сложения (FADD, FADDP), вычитания (FSUB, FSUBP), умножения (FMUL, FMULP), деления (FDIV, FDIVP), а также команды округления и сравнения (FCHS, FRNDINT, FCOM, FTST и др.). Они выполняются с плавающей точкой и обрабатывают данные, хранящиеся в регистрах стека.

* Какие команды сопроцессора используются для передачи данных?

Ответ: для передачи данных используются команды FLD (загрузка значения в стек), FST (сохранение значения без извлечения), FSTP (сохранение с извлечением из стека), FXCH (обмен значениями между регистрами стека), FILD и FISTP (работа с целыми значениями). Эти команды обеспечивают перемещение данных между памятью и регистрами сопроцессора.

* **Вывод**
* В ходе выполнения работы была достигнута поставленная цель — изучены принципы разработки разветвляющихся алгоритмов на языке ассемблера.
* Также был закреплён навык отладки приложений в среде MS Visual Studio, что позволило на практике освоить применение условных и безусловных переходов, логических команд и методов анализа флагов процессора.

**Лабораторная работа №6**

**Изучение принципов работы цепочных команд**

* **Цель работы**
* Изучить принципы обработки цепочек данных средствами ассемблера.
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Для чего в операциях с цепочками используется флаг DF?

Ответ: флаг направления (DF — Direction Flag) используется в операциях с цепочками данных для определения направления обработки. Если флаг DF сброшен (DF = 0), то обработка данных происходит в направлении от младших адресов к старшим (вперёд по памяти). Если DF установлен (DF = 1), то обработка идёт в обратном направлении — от старших адресов к младшим. Это влияет на то, увеличиваются или уменьшаются указатели SI/EDI и DI/ESI при каждой итерации операций с цепочками.

* Для чего используются префиксы в операциях с цепочками данных?

Ответ: префиксы в операциях с цепочками данных используются для задания повторного выполнения команды до тех пор, пока не будет выполнено определённое условие. Наиболее часто применяются префиксы REP, REPE/REPZ и REPNE/REPNZ. Префикс REP означает, что операция будет повторяться, пока счётчик CX/ECX/RCX не станет равным нулю. Префиксы REPE (повторять при равенстве) и REPNE (повторять при неравенстве) применяются в сравнительных операциях, таких как CMPS, и позволяют прекратить выполнение, если найдено несовпадение.

* Какие команды используются для сравнения цепочек данных?

Ответ: для сравнения цепочек данных в языке ассемблера используются команды CMPSB, CMPSW, CMPSD и CMPSQ, которые сравнивают байты, слова, двойные или четырёхбайтные значения, находящиеся по адресам, заданным регистрами источника (SI/ESI/RSI) и приёмника (DI/EDI/RDI). Часто эти команды применяются с префиксами REPE или REPNE для автоматического повторения сравнения до выполнения условия.

* Операнды каких размеров могут использоваться в операциях с цепочками данных?

Ответ: в операциях с цепочками данных могут использоваться операнды следующих размеров: байты (8 бит), слова (16 бит), двойные слова (32 бит). Соответственно используются команды MOVSB, MOVSW, MOVSD, MOVSQ и аналогичные для других операций с цепочками.

* Какие регистры предназначены для задания источника и приемника в операциях с цепочками данных?

Ответ: для задания источника и приёмника в операциях с цепочками данных используются регистры:

**–** SI/ESI/RSI **–** источник данных (Source Index);

**–** DI/EDI/RDI **–** приёмник данных (Destination Index).

Также используется регистр CX/ECX/RCX как счётчик итераций при использовании префикса REP.

* **Вывод**
* В ходе работы были изучены принципы обработки цепочек данных с использованием команд и префиксов языка ассемблера.
* Также был закреплён навык отладки программ в среде MS Visual Studio, что позволило на практике проследить выполнение операций с цепочками и работу соответствующих регистров.

**Лабораторная работа №7**

**Изучение процесса разработки модулей на языке ассемблера**

* **Цель работы**
* Изучить принципы обработки цепочек данных средствами ассемблера.
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Из каких этапов состоит процесс разработки программ на языке ассемблера?

Ответ: процесс разработки программ на языке ассемблера включает несколько этапов: написание исходного кода, компиляция (ассемблирование), компоновка (линковка), загрузка исполняемого файла в память и выполнение. Также важными этапами являются отладка и тестирование программы.

* Из каких сегментов состоит программа на языке ассемблера и как объявляется каждый из сегментов?

Ответ: программа на языке ассемблера состоит из нескольких сегментов: сегмент данных (.data) — для хранения и инициализации переменных; сегмент кода (.code) — для размещения инструкций программы; и иногда сегмент стека (.stack) — для организации временного хранения данных. Каждый сегмент объявляется соответствующей директивой: .data, .code, .stack.

* Как в программе на языке ассемблера объявить целое число, строку, массив?

Ответ: для объявления переменных используются директивы определения данных. Целое число можно объявить, например, так: var1 DW 100 (для 16-битного числа). Строка объявляется как msg DB "Hello", 0, где 0 — завершающий символ. Массив можно задать перечислением элементов: array DB 1, 2, 3, 4.

* Для чего предназначен сегмент кода программы на языке ассемблера?

Ответ: сегмент кода (.code) содержит машинные инструкции, которые процессор выполняет. Это основная часть программы, где размещаются процедуры, основной поток исполнения и вызовы функций.

* Что такое MASM?

Ответ: MASM (Microsoft Macro Assembler) — это макроассемблер от Microsoft, предназначенный для разработки программ на языке ассемблера для архитектуры x86. Он поддерживает макроопределения, удобный синтаксис и интеграцию с Visual Studio.Каков синтаксис вызова процедур в программах на языке ассемблера?

* Каков синтаксис вызова процедур в программах на языке ассемблера?

Ответ: в программах на ассемблере параметры между процедурами передаются несколькими способами: через регистры, через стек, а также с использованием глобальных переменных. Наиболее часто используется стек, так как он сохраняет значения и упрощает передачу данных при вложенных вызовах.

* Какие способы передачи параметров между процедурами используются в программах на языке ассемблера?

Ответ: вызов процедур в ассемблере осуществляется с помощью инструкции CALL имя\_процедуры. Для возврата из процедуры используется команда RET.

* **Вывод**
* В ходе работы были изучены основные принципы обработки цепочек данных с использованием команд и префиксов языка ассемблера.
* Также был закреплён навык отладки программ в среде MS Visual Studio, что позволило лучше понять выполнение операций с памятью и работу регистров при обработке последовательностей данных.

**Лабораторная работа №8**

**Изучение процесса разработки модулей на языке ассемблера**

* **Цель работы**
* Изучить принципы обработки цепочек данных средствами ассемблера.
* Закрепить навык отладки приложений в MS Visual Studio.
* **Ответы на контрольные вопросы**
* Что такое «дизассемблирование»?

Ответ: дизассемблирование – это процесс преобразования исполняемого машинного кода обратно в эквивалентные инструкции ассемблера. Он позволяет увидеть, какие команды выполняются процессором, без исходного кода.

* Для чего выполняется дизассемблирование программного кода?

Ответ: Дизассемблирование выполняется для анализа работы программы на низком уровне, отладки, изучения чужого кода (например, при отсутствии исходников), а также в целях безопасности – для поиска уязвимостей и вредоносных инструкций.

* Что такое «дизассемблер»?

Ответ: дизассемблер – это программа, которая автоматически выполняет процесс дизассемблирования. Она считывает машинный код из исполняемых файлов и отображает его в виде инструкций ассемблера, понятных программисту.

* Какие существуют программы-дизассемблеры?

Ответ: существуют различные программы-дизассемблеры, такие как:

IDA Pro – один из самых мощных интерактивных дизассемблеров;

Ghidra – бесплатный инструмент от NSA с функцией дизассемблирования и декомпиляции;

OllyDbg – отладчик с возможностями дизассемблирования;

Radare2 – кроссплатформенный фреймворк для анализа бинарных файлов;

встроенные средства дизассемблирования в Visual Studio.

* Как открыть окно дизассемблированного программного кода в Visual Studio?

Ответ: в Visual Studio окно дизассемблированного кода можно открыть во время отладки, если установить точку останова и нажать Ctrl+Alt+D, либо выбрать пункт Debug > Windows > Disassembly. Также дизассемблированный код автоматически появляется, если исходный код недоступен в момент отладки.

* для чего выполняется обфускация кода?

Ответ: обфускация кода используется для усложнения анализа программ. Она искажает структуру, имена переменных и логические конструкции таким образом, чтобы программа оставалась работоспособной, но была трудной для чтения и обратной инженерии. Это особенно важно для защиты интеллектуальной собственности и затруднения взлома.

* **Вывод**
* В ходе работы были изучены основные принципы обработки цепочек данных с использованием команд и префиксов языка ассемблера.
* Также был закреплён навык отладки программ в среде MS Visual Studio, что позволило лучше понять выполнение операций с памятью и работу регистров при обработке последовательностей данных.

**Лабораторная работа №9**

**Изучение процесса разработки модулей на языке ассемблера**

* **Цель работы**
* Научиться создавать имитационные модели проектов, работающих под управлением микроконтроллеров для Arduino;
* Получить навыки работы с платой Arduino Uno и макетной платой
* **Ответы на контрольные вопросы**
* Что такое и для чего используется «Arduino Uno»?

Ответ: Arduino Uno – это микроконтроллерная плата, основанная на чипе ATmega328P. Она широко используется для обучения, прототипирования и создания электронных проектов. Arduino Uno предоставляет простую платформу с цифровыми и аналоговыми входами/выходами, а также поддержкой подключения датчиков, дисплеев, моторов и других компонентов.

* Что такое «макетная плата» и для чего она используется?

Ответ: макетная плата, или breadboard, предназначена для сборки электрических схем без пайки. Она используется в учебных и экспериментальных целях, чтобы быстро и удобно соединять электронные компоненты. Внутри платы находятся металлические дорожки, которые соединяют группы отверстий между собой.

* Для чего предназначен каждый из типов рельс на макетной плате?

Ответ: на макетной плате есть разные рельсы: по краям находятся питательные рельсы, через которые подаётся напряжение и земля, а в центральной части — сигнальные рельсы, которые используются для соединения ножек компонентов. Все отверстия в одном ряду внутри группы соединены между собой, что позволяет легко выполнять соединения.

* Как соединить элементы на макетной плате последовательно?

Ответ: чтобы соединить элементы последовательно на макетной плате, выход одного элемента соединяется с входом следующего с помощью проводов. В таком соединении ток проходит через каждый элемент по очереди. При параллельном соединении входы всех элементов подключаются к одной точке, а выходы — к другой, что обеспечивает одинаковое напряжение для всех компонентов.

* Как соединить элементы на макетной плате параллельно?

Ответ: в Visual Studio окно дизассемблированного кода можно открыть во время отладки, если установить точку останова и нажать Ctrl+Alt+D, либо выбрать пункт Debug > Windows > Disassembly. Также дизассемблированный код автоматически появляется, если исходный код недоступен в момент отладки.

* В каком порядке должно выполняться подключение элементов на схеме?

Ответ: Подключение элементов на схеме должно начинаться с обесточенной платы. Сначала размещаются компоненты, затем они соединяются проводами, после чего выполняется проверка правильности схемы. Только после этого подаётся питание, что предотвращает возможные ошибки и повреждение оборудования.

* **Вывод**
* В ходе работы были освоены принципы создания имитационных моделей для проектов на базе микроконтроллера Arduino.
* Также был получен практический опыт работы с платой Arduino Uno и макетной платой, что способствовало закреплению навыков сборки схем и взаимодействия электронных компонентов.

**Лабораторная работа №10**

**Разработка скетчей для Arduino**

* **Цель работы**
* Научиться создавать скетчи (программное обеспечение на языке Си) для микроконтроллеров Arduino;
* Получить навыки работы с платой Arduino Uno и макетной платой
* **Ответы на контрольные вопросы**
* Для чего предназначены и когда вызываются функции setup() и loop()?

Ответ: Функции setup() и loop() являются основными в программе (скетче) для Arduino. Функция setup() вызывается один раз при запуске или перезагрузке микроконтроллера и предназначена для начальной настройки, такой как задание режимов работы пинов. Функция loop() вызывается бесконечно в цикле после выполнения setup() и содержит основной код, который должен постоянно выполняться.

* Для чего используется функция pinMode(...), какие параметры она принимает?

Ответ: Функция pinMode() используется для установки режима работы пина: вход (INPUT), выход (OUTPUT) или вход с подтягивающим резистором (INPUT\_PULLUP). Она принимает два параметра: номер пина и режим его работы.

* Для чего используется функция digitalWrite(...), какие параметры она принимает?

Ответ: Функция digitalWrite() предназначена для подачи логического сигнала на цифровой пин. Она принимает два параметра: номер пина и значение сигнала – HIGH (высокий уровень) или LOW (низкий уровень).

* Какие функции используются для реализации программной задержки, какие параметры принимают эти функции?

Ответ: Для реализации программной задержки используются функции delay() и delayMicroseconds(). Первая принимает количество миллисекунд, а вторая – количество микросекунд, в течение которых выполнение программы будет приостановлено.

* Что такое «скетч»?

Ответ: Скетч – это программа, написанная для Arduino. Он представляет собой файл с расширением .ino и содержит код, который загружается в микроконтроллер.

* В каком порядке должно выполняться подключение элементов на схеме?

Ответ: Для подключения библиотек к скетчу используется директива #include, после которой указывается имя подключаемой библиотеки в угловых скобках или кавычках. Также можно подключать библиотеки через меню среды Arduino IDE.

* **Вывод**
* В ходе работы были освоены принципы создания имитационных моделей для проектов на базе микроконтроллера Arduino.
* Также был получен практический опыт работы с платой Arduino Uno и макетной платой, что способствовало закреплению навыков сборки схем и взаимодействия электронных компонентов.

**Лабораторная работа №11**

**Разработка приложений для обработки файлов**

* **Цель работы**
* Научиться использовать файловые потоки в приложении на C#;
* Научиться применять классы для работы с файлами в приложениях на C#.
* **Ответы на контрольные вопросы**
* Какое пространство имен содержит себе классы для работы с файловой системой и файловыми потоками?

Ответ: В языке программирования C# пространство имён, содержащее классы для работы с файловой системой и файловыми потоками – это System.IO. Оно предоставляет разнообразные классы и методы для чтения, записи, создания и управления файлами и каталогами.

* Для чего предназначены классы StreamReader и StreamWriter?

Ответ: Классы StreamReader и StreamWriter используются для работы с текстовыми файлами. StreamReader предназначен для чтения текста из файла, а StreamWriter – для записи текста в файл. Они обеспечивают удобный доступ к данным в символьном представлении.

* Для чего предназначены классы BinaryReader и BinaryWriter?

Ответ: Классы BinaryReader и BinaryWriter служат для работы с двоичными файлами. BinaryReader считывает данные в их бинарном виде, а BinaryWriter позволяет записывать данные в файл в бинарном формате, что полезно при необходимости эффективного хранения и передачи информации.

* Какие классы предоставляют информацию о файлах?

Ответ: Информацию о файлах предоставляют классы FileInfo и DirectoryInfo. С их помощью можно получать сведения о размере файла, дате последнего изменения, типе, наличии и других атрибутах, а также выполнять операции над файлами и каталогами.

* **Вывод**
* В ходе работы были изучены и применены на практике основные принципы работы с файловыми потоками в C#. Получены навыки использования классов для чтения, записи и управления файлами, что позволяет эффективно работать с файловой системой в приложениях.Цель работы достигнута.

**Лабораторная работа №12**

**Разработка приложений для поиска файлов**

* **Цель работы**
* Научиться применять классы для работы с файлами и каталогами в приложениях на C#.
* **Ответы на контрольные вопросы**
* В чем отличие между классами Directory и DirectoryInfo?

Ответ: Классы Directory и DirectoryInfo служат для работы с каталогами, но различаются по способу использования. Directory – это статический класс, предоставляющий методы для выполнения операций без создания экземпляра. В отличие от него, DirectoryInfo – это экземплярный класс, позволяющий работать с конкретной директорией через объект, что даёт больше гибкости и возможностей кэширования.

* Для чего предназначены классы StreamReader и StreamWriter?

Ответ: Аналогично, классы File и FileInfo предназначены для работы с файлами. File является статическим и предоставляет быстрый доступ к операциям с файлами, таким как создание, копирование, удаление. FileInfo же требует создания экземпляра, но позволяет получать более подробную информацию и выполнять операции с файлом через объектный подход.

* В чем отличие между классами File и FileInfo?

Ответ: Для получения списка файлов и папок определённого каталога можно использовать методы Directory.GetFiles(), Directory.GetDirectories() или Directory.GetFileSystemEntries() – они возвращают массив строк с путями. В случае с DirectoryInfo, можно использовать методы GetFiles() и GetDirectories().

* Как получить список файлов и папок определенного каталога?

Ответ: Шаблон поиска задаётся через строку с маской, например "\*.txt" — для всех текстовых файлов. Дополнительно можно указать опции поиска, например SearchOption.AllDirectories для поиска во всех вложенных каталогах, или SearchOption.TopDirectoryOnly – только в текущем.

* Какие свойства класса FileInfo позволяют получить информацию о файле?

Ответ: Свойства класса FileInfo, позволяющие получить информацию о файле, включают:

Length – размер файла в байтах,

CreationTime – дата и время создания файла,

LastAccessTime – дата последнего доступа,

LastWriteTime – дата последнего изменения,

Extension – расширение файла,

FullName – полный путь к файлу.

* **Вывод**
* В ходе работы были освоены основные приёмы работы с файлами и каталогами в языке C#, включая использование классов File, FileInfo, Directory и DirectoryInfo. Полученные навыки позволяют эффективно управлять файловой системой в приложениях, выполняя операции чтения, записи, копирования, удаления и получения информации о файлах и каталогах.

**Лабораторная работа №13**

**Разработка приложений для сортировки файлов**

* **Цель работы**
* Научиться применять классы для работы с файлами и каталогами в приложениях на C#.
* **Ответы на контрольные вопросы**
* Как проверить существование файла по его имени?

Ответ: Чтобы проверить существование файла по его имени, используется метод File.Exists(path). Он возвращает true, если файл существует, и false — если нет.

* Как проверить существование каталога по его имени?

Ответ: Для проверки существования каталога по его имени применяется метод Directory.Exists(path), который аналогично возвращает логическое значение.

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

Ответ: Методы, позволяющие работать с каталогами: Directory.CreateDirectory(path) – создаёт каталог;

* Directory.Delete(path) – удаляет каталог;
  + - * 1. Directory.Move(sourceDirName, destDirName) – перемещает каталог.  
            Для копирования каталога требуется вручную рекурсивно копировать содержимое.
* Какие методы позволяют создавать, удалять, копировать и переносить файлы?

Ответ: Методы для работы с файлами:

* File.Create(path) – создаёт файл;
* File.Delete(path) – удаляет файл;
* File.Copy(sourceFileName, destFileName) – копирует файл;
* File.Move(sourceFileName, destFileName) – перемещает файл.
* **Вывод**
* В результате выполнения работы были изучены и применены на практике классы для работы с файлами и каталогами в языке C#. Полученные знания позволяют создавать, перемещать, копировать и удалять файлы и директории, а также получать информацию о них, что расширяет возможности по управлению данными в приложениях.

**Лабораторная работа №14**

**Разработка утилиты для поиска дубликатов файлов**

* **Цель работы**
* Научиться применять классы для работы с файлами и каталогами в приложениях на C#.
* **Ответы на контрольные вопросы**
* Какое свойство FileInfo возвращает имя файла?

Ответ: Свойство Name класса FileInfo возвращает имя файла без пути.

* Какое свойство FileInfo возвращает расширение файла?

Ответ: Свойство Extension возвращает расширение файла, включая точку (например, ".txt").

* Какое свойство FileInfo возвращает полное имя файла?

Ответ: Свойство FullName возвращает полное имя файла, включая путь.

* Какое свойство FileInfo возвращает дату изменения файла?

Ответ: Свойство LastWriteTime возвращает дату и время последнего изменения файла.

* **Вывод**

В результате выполнения работы были изучены и применены на практике классы для работы с файлами и каталогами в языке C#. Полученные знания позволяют создавать, перемещать, копировать и удалять файлы и директории, а также получать информацию о них, что расширяет возможности по управлению данными в приложениях.

**Лабораторная работа №15**

**Разработка утилиты для просмотра изображений**

* **Цель работы**
* Научиться отображать файлы-изображения в приложениях на C#.
* **Ответы на контрольные вопросы**
* Для чего предназначен класс BitmapImage?

Ответ: Класс BitmapImage предназначен для загрузки и представления растровых изображений (например, PNG или JPEG) в приложениях WPF. Он используется для задания источника изображения, которое можно отобразить в элементах управления, таких как Image.

* Для чего предназначен элемент управления Image?

Ответ: Элемент управления Image в WPF предназначен для отображения изображений в пользовательском интерфейсе. Он поддерживает привязку к источникам изображений, масштабирование, выравнивание и другие визуальные настройки.

* Для чего предназначен элемент управления ScrollView?

Ответ: Элемент управления ScrollViewer предназначен для добавления полос прокрутки к вложенному содержимому, если оно выходит за пределы доступной области. Это позволяет прокручивать большое изображение или другой контент по вертикали и/или горизонтали.

* Как получить высоту и ширину изображения BitmapImage?

Ответ: Чтобы получить высоту и ширину изображения в BitmapImage, используются свойства PixelHeight и PixelWidth соответственно.

* **Вывод**
* В результате выполнения работы была освоена техника отображения файлов-изображений в приложениях на C#, в том числе использование классов BitmapImage и элементов управления Image и ScrollViewer. Полученные навыки позволяют загружать, отображать и управлять изображениями в графическом интерфейсе пользователя.

**Лабораторная работа №16**

**Разработка утилиты Диспетчер задач**

* **Цель работы**
* Научиться получать и отображать информацию о запущенных процессах в приложениях на C#.
* **Ответы на контрольные вопросы**
* Какое пространство имен требуется подключить для работы с процессами?

Ответ: Для работы с процессами требуется подключить пространство имен System.Diagnostics.

* Какой метод запускает процесс?

Ответ: Метод Process.Start() запускает процесс.

* Какой метод завершает процесс?

Ответ: Метод Kill() завершает процесс.

* Как получить список запущенных процессов?

Ответ: Список запущенных процессов получают с помощью метода Process.GetProcesses().

* Как получить процесс по его идентификатору?

Ответ: Процесс по его идентификатору получают с помощью метода Process.GetProcessById(int processId).

* **Вывод**
* В результате выполнения работы была освоена техника получения и отображения информации о запущенных процессах в приложениях на C#, включая использование класса Process и методов для получения списка процессов, их идентификаторов и основных характеристик. Полученные навыки позволяют реализовывать мониторинг и управление процессами в графическом или консольном интерфейсе пользователя.

**Лабораторная работа №17**

**Разработка утилиты Файловый менеджер**

* **Цель работы**
* Научиться применять элементы управления для отображения файлов и папок в приложениях на C#.
* **Ответы на контрольные вопросы**
* Как получить список дисков?

Ответ: Для получения списка дисков используется класс DriveInfo из пространства имен System.IO. Метод DriveInfo.GetDrives() возвращает массив доступных дисков.

* Как получить список файлов?

Ответ: Для получения списка файлов в папке применяется метод Directory.GetFiles(string path) из пространства имен System.IO, который возвращает массив строк с путями к файлам в указанной директории.

* Для чего предназначен элемент управления TreeView?

Ответ: Элемент управления TreeView предназначен для отображения иерархической структуры данных в виде дерева, позволяя пользователю разворачивать и сворачивать узлы, что удобно для навигации по вложенным объектам (например, каталогам и файлам).

* **Вывод**
* В результате выполнения работы была освоена техника использования элементов управления для отображения файлов и папок в приложениях на C#, в частности работа с классами DriveInfo и Directory, а также применение элемента управления TreeView для представления иерархической структуры файловой системы. Полученные навыки позволяют создавать удобные интерфейсы навигации по дискам и каталогам.

**Лабораторная работа №18**

**Разработка утилиты для анализа дисков**

* **Цель работы**
* Научиться получать и отображать статистическую информацию о дисковом пространстве в приложениях на C#.
* **Ответы на контрольные вопросы**
* Какой класс предоставляет информацию о дисках?

Ответ: Класс DriveInfo предоставляет информацию о дисках.

* Какой класс предоставляет информацию о файлах?

Ответ: Класс FileInfo предоставляет информацию о файлах.

* Какой метод позволяет просуммировать значения в списке?

Ответ: Метод Sum() из пространства имен System.Linq позволяет просуммировать значения в списке.

* Какие методы позволяют отсортировать список по возрастанию и убыванию?

Ответ: Метод Sum() из пространства имен System.Linq позволяет просуммировать значения в списке.

* **Вывод**

В результате выполнения работы была освоена техника получения и отображения статистической информации о дисковом пространстве в приложениях на C#, включая использование классов DriveInfo и методов для получения объема, свободного и занятого места на дисках. Полученные навыки позволяют реализовывать мониторинг состояния дисковых ресурсов в пользовательском интерфейсе.

**Лабораторная работа №19**

**Разработка утилиты Архиватор**

* **Цель работы**
* Научиться применять классы для работы с архивами в приложениях на C#.
* **Ответы на контрольные вопросы**
* Какие пространства имен нужно подключить для работы с архивами?

Ответ: Для работы с архивами нужно подключить пространства имен:

* 1. System.IO;
  2. System.IO.Compression;
  3. System.IO.Compression.FileSystem (для работы с файлами и архивами в файловой системе).
* Какие ссылки нужно подключить для работы с архивами?

Ответ: Для работы с архивами необходимо подключить ссылки на сборки:

* System.IO.Compression.dll
* System.IO.Compression.FileSystem.dll (если используется функционал для работы с файлами в архиве)
* Для чего предназначен класс ZipArchive?

Ответ: Класс ZipArchive предназначен для представления ZIP-архива и работы с его содержимым (чтение, создание и изменение архивных записей).

* Для чего предназначен класс ZipFile?

Ответ: Класс ZipFile предназначен для удобного создания, извлечения и управления ZIP-архивами с помощью статических методов (например, создание архива из каталога или распаковка архива).

* Для чего предназначен класс ZipArchiveEntry?

Ответ: Класс ZipArchiveEntry представляет отдельный файл или папку внутри ZIP-архива, предоставляет доступ к его имени, размеру и содержимому.

* **Вывод**
* В результате выполнения работы была освоена техника использования классов для работы с архивами в приложениях на C#, включая создание, чтение и извлечение ZIP-архивов с помощью классов ZipArchive, ZipFile и ZipArchiveEntry. Полученные навыки позволяют эффективно управлять архивированием файлов и папок в программных решениях.

**Лабораторная работа №20**

**Разработка утилиты скринсейвер**

* **Цель работы**
* Научиться разрабатывать оконные приложения-заставки, используя таймер.
* **Ответы на контрольные вопросы**
* Что такое «утилита»?

Ответ: «Утилита» – это небольшая специализированная программа, предназначенная для выполнения конкретных вспомогательных задач или обслуживания системы.

* Что такое «скринсейвер»?

Ответ: «Скринсейвер» – это программа-заставка, которая автоматически запускается при бездействии компьютера и отображает анимацию или изображение для предотвращения выгорания экрана.

* Какое расширение должно быть у файлов-заставок в Windows?

Ответ: Файлы-заставки в Windows должны иметь расширение.scr.

* **Вывод**
* В результате выполнения работы была освоена технология разработки оконных приложений-заставок на C# с использованием таймера для управления временем отображения и анимацией. Полученные навыки позволяют создавать простые скринсейверы с регулируемым поведением в пользовательском интерфейсе.

**Лабораторная работа №21**

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

* **Цель работы**
* Научиться вычислять хэш-суммы файлов в приложениях на C#, используя встроенные алгоритмы шифрования.
* **Ответы на контрольные вопросы**
* Что такое хэш-сумма?

Ответ: Хэш-сумма – это уникальное фиксированной длины значение, вычисленное по содержимому файла или данных с помощью хэш-функции.

* Для чего применяются хэш-суммы файлов?

Ответ: Хэш-суммы файлов применяются для проверки целостности и подлинности данных, чтобы убедиться, что файл не был изменён или повреждён

* Какие алгоритмы применяются для вычисления хэш-суммы файлов?

Ответ: Для вычисления хэш-суммы файлов применяются алгоритмы, такие как MD5, SHA-1 и SHA-256.

* **Вывод**
* В результате выполнения работы была освоена техника вычисления хэш-сумм файлов в приложениях на C# с использованием встроенных алгоритмов хеширования, таких как MD5 и SHA. Полученные навыки позволяют реализовывать проверку целостности и подлинности файлов в программных решениях.

**Лабораторная работа №22**

**Разработка утилиты Менеджер паролей**

* **Цель работы**
* Научиться выполнять шифрование и дешифрование данных в приложениях на C#, используя встроенные алгоритмы шифрования.
* **Ответы на контрольные вопросы**
* Что такое «шифрование»?

Ответ: «Шифрование» – это процесс преобразования исходных данных в зашифрованный формат с целью защиты информации от несанкционированного доступа.

* Что такое «дешифрование»?

Ответ: «Дешифрование» – это обратный процесс расшифровки зашифрованных данных для восстановления исходной информации.

* Что такое «AES»?

Ответ: «AES» (Advanced Encryption Standard) – это современный симметричный алгоритм шифрования данных, широко используемый для обеспечения безопасности информации.

* Какими могут быть размеры ключа в алгоритме AES?

Ответ: В алгоритме AES размеры ключа могут быть 128, 192 или 256 бит.

* Какое пространство имен требуется подключить для применения стандартных алгоритмов шифрования?

Ответ: Для применения стандартных алгоритмов шифрования требуется подключить пространство имен System.Security.Cryptography.

* **Вывод**
* В результате выполнения работы была освоена технология шифрования и дешифрования данных в приложениях на C# с использованием встроенных алгоритмов, таких как AES. Полученные навыки позволяют обеспечивать безопасность и конфиденциальность информации в программных продуктах.

**Лабораторная работа №23**

**Изучение процесса разработки DLL**

* **Цель работы**
* Изучить процесс разработки библиотек динамической компоновки на С++.
* Научиться применять библиотеки динамической компоновки написанные на других языках программирования в C#.
* **Ответы на контрольные вопросы**
* Что такое «библиотека динамической компоновки»?

Ответ: «Библиотека динамической компоновки» (Dynamic Link Library, DLL) – это файл, содержащий код и ресурсы, которые могут быть использованы несколькими программами одновременно во время выполнения, позволяя повторное использование и модульность.

* Какое расширение у файлов библиотек?

Ответ: Расширение файлов библиотек –.dll.

* Как подключить библиотеку к проекту?

Ответ: Для подключения библиотеки к проекту в среде разработки (например, Visual Studio) необходимо добавить ссылку на DLL-файл через свойства проекта или через меню «Add Reference» (Добавить ссылку), после чего можно использовать содержащиеся в ней классы и методы.

* **Вывод**
* В результате выполнения работы были изучены основы разработки библиотек динамической компоновки (DLL) на языке C++ и освоены методы их подключения и использования в приложениях на C#. Полученные навыки позволяют создавать и интегрировать модульные компоненты, разработанные на разных языках программирования, обеспечивая расширяемость и повторное использование кода.

**Лабораторная работа №24**

**Использование потоков**

* **Цель работы**
* Научиться разрабатывать многопоточные приложения на C#;
* Научиться создавать и применять фоновые и основные потоки и выполнять обмен данных между ними в программах на C#.
* **Ответы на контрольные вопросы**
* В чем отличие потока от процесса?

Ответ: Отличие потока от процесса: процесс – это самостоятельная выполняющаяся программа с собственной памятью, а поток – это часть процесса, выполняющаяся внутри его контекста и разделяющая память с другими потоками этого процесса.

* Какие преимущества дает многопоточная архитектура?

Ответ: Преимущества многопоточной архитектуры: повышение производительности за счёт параллельного выполнения задач, улучшение отзывчивости приложений, эффективное использование ресурсов процессора.

* Какие существуют основные средства синхронизации потоков?

Ответ: Основные средства синхронизации потоков: мьютексы (Mutex), семафоры (Semaphore), блокировки (lock/Monitor), события (Event), барьеры (Barrier).

* Каким образом на однопроцессорных компьютерах исполняются многопоточные приложения?

Ответ: На однопроцессорных компьютерах многопоточные приложения исполняются с помощью планировщика, который быстро переключает выполнение между потоками, создавая эффект одновременной работы (квазипараллелизм)

* Для чего в С# используется класс Thread?

Ответ: В C# класс Thread используется для создания и управления потоками, позволяя запускать, приостанавливать, возобновлять и завершать отдельные потоки исполнения в приложении.

* **Вывод**
* В результате выполнения работы были освоены принципы разработки многопоточных приложений на C#, включая создание и управление как основными, так и фоновыми потоками, а также организация обмена данными между ними.

**Лабораторная работа №25**

**Разработка сетевого приложения с использованием сокетовЦель работы**

* Цель работы
* Научиться реализовывать и запускать асинхронные операции на C#;
* Научиться выполнять вычисления, используя асинхронные операции на C#.
* **Ответы на контрольные вопросы**
* Что такое «сокет»?

Ответ: Сокет — это программный интерфейс, представляющий собой конечную точку связи в сети, которая используется для организации взаимодействия между компьютерами.

* Каков алгоритм работы сервера, использующего сокеты?

Ответ: алгоритм работы сервера, использующего сокеты, включает создание сокета, его привязку к конкретному адресу и порту, перевод в режим прослушивания, ожидание и принятие входящих соединений, обработку полученных данных и отправку ответов, а также закрытие соединения по завершении работы.

* Каков алгоритм работы клиента, использующего сокеты?

Ответ: алгоритм работы клиента состоит в создании сокета, установлении соединения с сервером, отправке запросов и приёме ответов, после чего соединение закрывается.

* Какие пространства имен требуется подключить для работы с сокетами?

Ответ: для работы с сокетами необходимо подключить пространства имен System.Net и System.Net.Sockets

* Какие параметры требуется указать при создании сокета?

Ответ: при создании сокета указываются параметры адресного семейства (например, IPv4), типа сокета (например, потоковый для TCP) и протокола (например, TCP).

* Как выполнить получение данных с использованием сокетов?

Ответ: Получение данных с использованием сокетов выполняется с помощью метода Receive(), который считывает входящие данные в заданный буфер.

* **Вывод**
* В результате выполнения работы были освоены методы реализации и запуска асинхронных операций на C#, а также выполнение вычислений с их использованием. Полученные навыки позволяют создавать более эффективные и отзывчивые приложения за счёт параллельного выполнения задач без блокировки основного потока.

**Лабораторная работа №26**

**Сетевое программирование сокетов**

* Цель работы
* Закрепить навыки работы с сетевыми сокетами на C#;
* **Ответы на контрольные вопросы**
* Чем отличаются технологии UDP и TCP?

Ответ: Технологии UDP и TCP отличаются по способу передачи данных. TCP (Transmission Control Protocol) обеспечивает надёжное, ориентированное на соединение соединение с проверкой доставки и восстановлением потерянных пакетов. UDP (User Datagram Protocol) — это более простой, без установления соединения протокол, который не гарантирует доставку, порядок или целостность пакетов, но работает быстрее и эффективнее при передаче небольших сообщений.

* Каков алгоритм работы сервера, использующего сокеты?

Ответ: Алгоритм работы сервера, использующего сокеты, включает создание сокета, привязку его к определённому адресу и порту, перевод сокета в режим прослушивания, ожидание и принятие входящих подключений, обработку полученных данных, отправку ответов клиентам и закрытие соединений по окончании работы.

* **Вывод**
* В результате выполнения работы были закреплены навыки работы с сетевыми сокетами на C#, включая создание серверных и клиентских приложений для обмена данными по сети. Полученные знания позволяют реализовывать сетевое взаимодействие и обеспечивать обмен информацией между удалёнными узлами.

**Лабораторная работа №27**

**Разработка многопоточных приложений**

* Цель работы
* Изучить процесс разработки многопоточных приложений на C#;
* **Ответы на контрольные вопросы**
* Какие методы предоставляет класс Parallel для обработки данных?

Ответ: Класс Parallel предоставляет методы, такие как Parallel.For, Parallel.ForEach и Parallel.Invoke, которые позволяют выполнять циклы и наборы действий параллельно, распределяя работу между доступными потоками для повышения производительности.

* Зачем необходимо синхронизировать потоки и использовать потокобезопасные операции?

Ответ: Синхронизация потоков и использование потокобезопасных операций необходимы для предотвращения ошибок, связанных с одновременным доступом нескольких потоков к общим ресурсам, что может привести к состояниям гонки, повреждению данных и непредсказуемому поведению программы.

* **Вывод**
* В результате выполнения работы был изучен процесс разработки многопоточных приложений на C#, включая использование класса Parallel для эффективного распараллеливания задач и обеспечение безопасности при работе с общими ресурсами. Полученные знания позволяют создавать производительные и корректно работающие многопоточные программы.

**Лабораторная работа №28**

**Обмен данными**

* Цель работы
* Изучить процесс разработки сетевых приложений с использованием SignalR на C#;
* **Ответы на контрольные вопросы**
* Что такое SignalR?

Ответ: SignalR — это библиотека для платформы .NET, которая упрощает добавление функционала реального времени в веб-приложения, обеспечивая двунаправленное асинхронное общение между клиентом и сервером.

* Каким образом осуществляется обмен данными между клиентом и сервером SignalR?

Ответ: Обмен данными между клиентом и сервером в SignalR осуществляется через подключение, которое поддерживает постоянное двунаправленное соединение (например, через WebSockets или альтернативные транспортные протоколы), позволяя серверу и клиенту вызывать методы друг друга напрямую в режиме реального времени..

* **Вывод**
* В результате выполнения работы был изучен процесс разработки сетевых приложений с использованием библиотеки SignalR на C#, включая организацию двунаправленного обмена данными в реальном времени между клиентом и сервером. Полученные навыки позволяют создавать интерактивные и отзывчивые приложения с поддержкой мгновенного обновления информации.

**Лабораторная работа №29**

**Работа с буфером экрана**

* Цель работы
* Изучить процесс разработки консольных приложений с продвинутым пользовательским интерфейсом;
* **Ответы на контрольные вопросы**
* Какие основные элементы управления используются в Terminal.Gui?

Ответ: В Terminal.Gui основными элементами управления являются окна (Window), кнопки (Button), метки (Label), текстовые поля (TextField), списки (ListView), переключатели (CheckBox) и рамки (FrameView), которые позволяют создавать интерактивный консольный интерфейс.

* Как в Terminal.Gui реализуются диалоговые окна?

Ответ: Диалоговые окна в Terminal.Gui реализуются как отдельные окна (Window) или модальные диалоги, которые могут отображать сообщения, запрашивать ввод или подтверждение от пользователя, и управляются с помощью специальных методов для показа и закрытия.

* Как настроить цветовую схему консольного приложения в Terminal.Gui?

Ответ: Цветовая схема консольного приложения в Terminal.Gui настраивается через объекты типа ColorScheme, которые задают цвета для различных элементов интерфейса, таких как фон, текст, выделение и границы, и применяются к контролам для изменения их визуального оформления.

* **Вывод**
* В результате выполнения работы был изучен процесс разработки консольных приложений с использованием библиотеки Terminal.Gui, включая создание и настройку различных элементов управления, диалоговых окон и цветовых схем. Полученные навыки позволяют создавать удобные и функциональные консольные интерфейсы с расширенными возможностями взаимодействия с пользователем.

**Лабораторная работа №30**

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

* Цель работы
* Изучить процесс разработки приложений с использованием технологии gRPC;
* **Ответы на контрольные вопросы**
* В чем заключаются преимущества и недостатки gRPC?

Ответ: Преимущества gRPC заключаются в высокой производительности за счёт использования бинарного протокола Protobuf, поддержке нескольких языков программирования, простоте создания распределённых приложений и встроенной поддержке двунаправленной потоковой передачи данных. Недостатками являются более сложная настройка по сравнению с REST, необходимость использования определённого формата сообщений и ограниченная поддержка браузеров без дополнительных прокси.

* Для чего используются файлы .proto?

Ответ: Файлы .proto используются для описания структуры сообщений и сервисов в формате Protocol Buffers. Они служат основой для генерации кода клиента и сервера, обеспечивая строгое типизированное и эффективное взаимодействие между компонентами системы.

* **Вывод**
* В результате выполнения работы был изучен процесс разработки приложений с использованием технологии gRPC, включая создание и использование файлов .proto для описания сервисов и сообщений.