МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КАРАЧАЕВО-ЧЕРКЕССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ У.Д. АЛИЕВА»

Факультет экономики и управления

УТВЕРЖДАЮ
Декан _____ З.М. Чомаева
«27» апреля 2023 г.
М.П.

Рабочая программа ПМ.01. Разработка модулей программного обеспечения для компьютерных систем
МДК 01.04. «Системное программирование»

Направление подготовки
09.02.07 Информационные системы и программирование
(шифр, название направления)

Среднее профессиональное образование

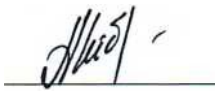
Форма обучения
Очная/очно-заочная

Год начала подготовки - 2023
(по учебному плану)

Карачаевск, 2023

Рабочая программа общеобразовательной учебной дисциплины разработана на основе Федерального государственного образовательного стандарта (далее - ФГОС) СОО в пределах образовательной программы СПО по специальности среднего профессионального образования (далее - СПО) 09.02.07 Информационные системы и программирование.

Одобрено на заседании предметно цикловой комиссии «Информационных, естественно - научных дисциплин» от 23 июня 2023 г., протокол № 6.

Председатель ПЦК
«Информационных,
естественно - научных дисциплин»  Лепшокова А. Н.

СОДЕРЖАНИЕ

Рабочая программа дисциплины
МДК 01.04. «Системное программирование»
09.02.07 Информационные системы и программирование

Цель и задачи изучения дисциплины	Целью изучения данной дисциплины является формирование представлений об общей методологии разработки системно-ориентированных программ с использованием современных алгоритмических языков и систем программирования, а так же подготовка обучающихся в области применения аппаратных и программных средств. Для достижения поставленной цели необходимо решить следующие задачи: - изучить основные подходы к проектированию, разработке и использованию системных программ; - освоить технологии системного программирования с использованием универсальных языков программирования; - изучить использование объектно-ориентированного подхода в программировании системных программ. Рабочая программа учебной дисциплины является частью программы подготовки специалистов среднего звена в соответствии с ФГОС по специальности СПО 09.02.07 Информационные системы и программирование
Место дисциплины в учебном плане	МДК.01.04
Общая трудоемкость дисциплины в часах	162 ч.
Семестр	6
Формируемые компетенции	ПК 5.1. Собирать исходные данные для разработки проектной документации на информационную систему. ПК 5.3. Разрабатывать подсистемы безопасности информационной системы в соответствии с техническим заданием. ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием.

Знания, умения и навыки, получаемые в результате освоения дисциплины	**Знать:** основные этапы разработки программного обеспечения; основные принципы технологии структурного и объектно-ориентированного программирования; основные принципы отладки и тестирования программных продуктов. **Уметь:** осуществлять разработку кода программного модуля на современных языках программирования; создавать программу по разработанному алгоритму как отдельный модуль; выполнять отладку и тестирование программы на уровне модуля; оформлять документацию на программные средства. **Владеть:** разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования; разработки кода программного продукта на основе готовой сертификации на уровне модуля; использование инструментальных средств на этапе отладки программного продукта; проведения тестирования программного модуля по определённому сценарию.
Содержание дисциплины	Введение в системное программирование. Многозадачность в операционных системах. Анализ кода системных программ. Управление памятью. Подсистема ввода-вывода. Архитектура вычислительных машин и систем.
Виды учебной работы	Лекции, практические, тесты, самостоятельная работа.
Перечень основной и дополнительной литературы, необходимой для освоения дисциплины	
а) основная литература 1. Молчанов, А. Ю. Системное программное обеспечение : лабораторный практикум / А. Ю. Молчанов. - Санкт-Петербург : Питер, 2021. - 284 с. - ISBN 978-5-4461-9998-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1857179 – Режим доступа: по подписке. 2. Гуриков, С. Р. Введение в программирование на языке Visual Basic for Applications (VBA) : учебное пособие / С.Р. Гуриков. - Москва : ИНФРА-М, 2021. - 317 с. - (Среднее профессиональное образование). - ISBN 978-5-16-015995-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1074164 – Режим доступа: по подписке.	
б) дополнительная учебная литература	

1. Кузнецов, А.С. Системное программирование : учеб. пособие / А.С. Кузнецов, И.А. Якимов, П.В. Пересунько. - Красноярск : Сиб. федер. ун-т 2018. - 170с. - ISBN 978-5-7638-3885-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1032183 – Режим доступа: по подписке. 2. Побегайло, А. П. Системное программирование в Windows : пособие / А. П. Побегайло. - Санкт-Петербург : БХВ-Петербург, 2006. - 1056 с. - ISBN 5-94157-792-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1842971 – Режим доступа: по подписке.	
в) интернет – ресурсы 1. **Официальный сайт Министерства образования и науки Российской Федерации-http://www.mon.gov.ru** 2. **Федеральный портал "Российское образование"-http://edu.ru** 3. **Информационная система "Единое окно доступа к образовательным ресурсам"-http://window.edu.ru** 4. **Единая коллекция цифровых образовательных ресурсов-http://school-collection.edu.ru** 5. **Федеральный центр информационно-образовательных ресурсов-http://fcior.edu.ru**	
Форма промежуточной аттестации	6 семестр - экзамен.

Фонд оценочных средств по дисциплине

1. Типовые темы к письменным работам, докладам и выступлениям:

1. Работа на ассемблере с портами ввода-вывода.
2. Визуальный анализ данных.
3. Сравнительные характеристики автоматических дизассемблеров.
4. Создание Windows – приложений на ассемблере.
5. Программирование сопроцессора.
6. Связь ассемблера с другими языками программирования (Pascal – ассемблер и C – ассемблер).
7. Сложные структуры данных в ассемблере (структуры, записи, списки, таблицы).

8. Лексический анализатор.
9. Организация таблиц идентификаторов транслятора по методу бинарного дерева.
10. Организация таблиц идентификаторов транслятора с использованием хеш-функций.
11. Развитие отладчиков в современных системах программирования.
12. Функции BIOS и MS DOS для работы с клавиатурой.
13. Организация ввода-вывода в консольном приложении Windows.
14. Программные средства обслуживания CD и DVD дисков.
15. Алгоритмы работы с памятью на уровне операционной системы (выделение, освобождение).
16. Использование концепции **.net** в системном программировании.
17. Сравнительные характеристики трансляторов.
18. Современные системы программирования.

Критерии оценки доклада, сообщения, реферата:

Отметка «отлично» за письменную работу, реферат, сообщение ставится, если изложенный в докладе материал:

- отличается глубиной и содержательностью, соответствует заявленной теме;
- четко структурирован, с выделением основных моментов;
- доклад сделан кратко, четко, с выделением основных данных;
- на вопросы по теме доклада получены полные исчерпывающие ответы.

Отметка «хорошо» ставится, если изложенный в докладе материал:

- характеризуется достаточным содержательным уровнем, но отличается недостаточной структурированностью;
- доклад длинный, не вполне четкий;
- на вопросы по теме доклада получены полные исчерпывающие ответы только после наводящих вопросов, или не на все вопросы.

Отметка «удовлетворительно» ставится, если изложенный в докладе материал:

- недостаточно раскрыт, носит фрагментарный характер, слабо структурирован;

- докладчик слабо ориентируется в излагаемом материале;

- на вопросы по теме доклада не были получены ответы или они не были правильными.

Отметка «неудовлетворительно» ставится, если:

- доклад не сделан;

- докладчик не ориентируется в излагаемом материале;

- на вопросы по выполненной работе не были получены ответы или они не были правильными.

2. Примерные вопросы к итоговой аттестации

1. Назначение, основные этапы развития операционных систем. Принципы построения ОС.
2. Понятие процесса, потока, ресурса, свойства, классификация. Концепция виртуализации. Концепция прерывания.
3. Состояние процессов. Описание процессов. Взаимодействие процессов. Задача взаимного исключения. Решение задачи взаимного исключения. Задача «производители-потребители» и её решения.
4. Распределение ресурсов, проблема тупиков. Алгоритм банкира. Применение алгоритма банкира.
5. Требования к управлению памятью. Схемы распределения памяти. Страничная организация памяти. Сегментация памяти.
6. Структуризация адресного пространства виртуальной памяти. Задачи управления виртуальной памятью: задача размещения, задача перемещения, задача преобразования адресов, задача замещения.
7. Типы планирования. Алгоритмы планирования. Примеры реализации алгоритмов планирования в современных операционных системах.

8. Организация функций ввода-вывода. Буферизация операций ввода-вывода. Дисковое планирование. Система управление файлами. Организация файлов, доступ к файлам. Управление внешней памятью.
9. Управление памятью в реальном и защищённом режимах. Дескрипторные таблицы и дескрипторы сегментов
10. Понятие процесса, потока, ресурса, свойства, классификация. Концепция виртуализации. Концепция прерывания.
11. Стандарты UNIX. Пользователи системы, атрибуты пользователя. Создание программы, исходный текст, заголовки. Системные вызовы и функции стандартных библиотек. Обработка ошибок. Форматы выполняемых файлов.
12. Типы файлов. Владельцы файлов. Управление правами доступа в файловой системе. Атрибуты файлов. Управление свойствами файлов. Работа с файлами. Структура файловой системы.
13. Процессы в ОС UNIX. Типы процессов. Атрибуты процессов. Создание и управление процессами. Перегрузка процессов. Завершение процессов.
14. Сигналы. Обработка сигналов. Неименованные каналы. Именованные каналы. Дополнительные средства взаимодействия между процессами. Сообщества, семафоры, разделяемая память. Понятие потока ОС UNIX.
15. Архитектура и основные подсистемы ОС Windows. Системный реестр ОС Windows, его назначение и использование.
16. Основные элементы программ с оконным пользовательским интерфейсом. Понятие оконного сообщения. Источники сообщений. Очереди сообщений. Обработка сообщений мыши, клавиатуры.
17. Понятие ресурсов программ. Виды ресурсов.
18. Принципы построения графической подсистемы ОС Windows. Понятие контекста устройства. Вывод графической информации на физическое устройство. Графические инструменты.

19. Организация многозадачности в ОС Windows. Понятие процесса и потока. Контекст потока. Создание и завершение процессов и потоков. Синхронизация потоков.
20. Понятие динамически подключаемой библиотеки. Структура DLLбиблиотеки. Создание DLL-библиотеки.
21. Отладчики для режима ядра. Режимы отладки. Компоненты отладчика.
22. Прерывания. Уровни прерываний. Подпрограммы обработки прерываний. Отложенные процедуры. Асинхронные процедуры.
23. Пулы памяти. Пул подкачиваемой памяти. Пул неподкачиваемой памяти. Пул сессии, особый пул. Тегирование пулов.
24. Структура драйвера. Точки входа в драйвер. Объект, описывающий драйвер. Объект, описывающий устройство. Объект, описывающий файл. Взаимосвязь объектов.
25. Перехват функций ОС Windows API в пользовательском режиме. Интерфейсный модуль NTDLL.DLL. Функции ОС Windows API в режиме ядра. Технология перехвата функций в ядре за счёт модификации таблиц дескрипторов функций ОС Windows.
26. Понятие о логической и физической организации данных. Массив и связанный список – две основные формы физической организации данных.
27. Абстрактные линейные структуры данных: массив, связанный список.
28. Абстрактные линейные структуры данных: стек, дек, очередь, очередь с приоритетами.

Критерии оценки устного ответа на вопросы по дисциплине «Системное программирование»:

✓ 5 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций лекционного курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение

концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 4 - балла - знание узловых проблем программы и основного содержания лекционного курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 3 балла – фрагментарные, поверхностные знания важнейших разделов программы и содержания лекционного курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 2 балла – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

3. Тестовые задания для проверки знаний студентов

1. Операционная система – это…
+ **Комплекс программ**
- Прикладная программа
+ **Администратор**
- Обеспечение
- Назначение
- Пользование
- Сервисная программа
+ **Менеджер ресурсов компьютера**
2. Операционные системы являются ресурсами … компьютера:
+ **управления**
+ **диспетчеризация**
+ **планирование**
- администратор

- менеджер
- обеспечение
- назначение
- пользование

3. Операционные системы для программирования приложений Win32API:
+ **Windows 98**
- Linux
+ **Windows 2000**
- Unix
+ **Windows XP**
- Windows CE
- Windows NT
- MS DOS

4. Категории объектов ОС Windows приложении:
- Windows 98
- User
+ **Graphics Device Interface**
+ **Unix**
- Windows Shell
+ **Kernel**
- Network Services
- Base Services

5. Ресурсы принадлежащие каждому потоку в ОС Windows:
- потоки интерфейса пользователя
- пользовательские потоки
+ **код исполняемой функции**
- потоки исполняемой функции
+ **набор регистров процессора**
- библиотека общих элементов
- интерфейс пользователя
+ **стек для работы приложения**

6. Действия менеджера потоков, во время переключения процессора на исполнение другого потока:
- обслуживает очередь запускаемого потока
+ **сохранить контекст прерываемого потока**
+ **восстановить контекст запускаемого потока**
+ **передать управление запускаемому потоку**
- управляет доступом для ОС
- выполняет функции ввода и вывода
- обеспечивает функции для вывода графики
- обеспечивает функции для взаймодействия

7. Параметры функции CreateThread:
- TerminateThread
+ **lpThreadAttributes**
- AttachThreadInput

- CreateProcess
- CreateThread
- ExitThread
+ **dwStackSize**
+ **lpStrtAddress**

8. Функции работы с процессами и потоками в Win32 API:
- TerminateThread
- lpThreadAttributes
+ **AttachThreadInput**
+ **CreateProcess**
- ExitThread
+ **CreateThread**
- dwStackSize
- lpStrtAddress

9. Функция Sleep() …
+ **задерживает выполнения потока**
- возвращает нулевое значение
+ **удаляет поток из очереди**
- выполняет выход из потока
- освобождает память
+ **делает короткие паузы**
- завершает выполнение функций
- освобождает ресурсы

10. Классификация программ в зависимости от количества определяемых ими параллельных потоков управления:
- дуплексный
- многозадачный
+ **многопоточный**
- системный
+ **с параллельным потоком**
- параллельный
+ **однопоточный**
- с прямым потоком

11. Операции над потоком связанные с операционной системой:
- Create
+ **Run**
- Exit
+ **Interrupt**
- Open
+ **Block**
- Read
- Write

12. Ресурсы каждого процесса ОС Windows:
+ **виртуальное адресное пространство**
- код исполняемой функции

+ **маркер доступа**
- набор регистров процессора
- стек для работы приложения
+ **страницы в реальной памяти**
- стек для работы ОС
- консольный ввод
13. Объектам синхронизации первого класса в Windows, которые служат только для решения задач синхронизации параллельных потоков:
+ **мьютекс (mutex)**
+ **событие (event)**
+ **семафор (semaphore)**
- ожидающий таймер
- работа (job)
- процесс (process)
- поток (thread)
- консольный ввод (console input).
14. К третьему классу синхронизации относятся объекты, которые переходят в сигнальное состояние по завершении своей работы:
- мьютекс (mutex)
- событие (event)
+ **работа (job)**
+ **процесс (process)**
+ **поток (thread)**
- семафор (semaphore)
- ожидающий таймер
- консольный ввод (console input)
15. Системные объекты, созданные менеджером объектов в Win32 API:
- класс
- поток
+ **события**
+ **семафор**
- процесс
+ **критический раздел**
- приоритет потока
- контекст
16. Логические комбинации флагов параметра dwDesiredAccess:
+ **EVENT_ALL_ACCESS**
+ **EVENT_MODIFY_STATE**
- ACTIONS_AFTER_EVENT
+ **SYNCHRONIZE**
- SEMAPHORE_ALL_ACCESS
- SEMAPHORE _MODIFY_STATE
- ACTIONS_BEFORE_EVENT
- CREATE_NEW_CONSOLE

17. Логические комбинации флагов параметра dwDesiredAccess, определяющий доступ к семафору:
- EVENT_ALL_ACCESS
- EVENT_MODIFY_STATE
+ **SYNCHRONIZE**
+ **SEMAPHORE_ALL_ACCESS**
- ACTIONS_AFTER_EVENT
+ **SEMAPHORE _MODIFY_STATE**
- ACTIONS_BEFORE_EVENT
- CREATE_NEW_CONSOLE
18. Значение функции WaitForsingieObject, в случае успешного завершения:
- wait_object_p
- wait_forsingie
+ **wait_object_o**
- wait_object
+ **wait_abandoned**
+ **wait_timeout**
- wait_exit
- wait_finally
19. Прототип функции waitForMuitipieObject:
+ **DWORD nCount**
+ **CONST HANDLE *lpHandles**
+ **BOOL bWaitAll**
- DWORD dwReserved
- BOOL bResume
- LPOVERLAPPED lpOverLapped
- HANDLE hThread
- DWORD IDThread
20. Значения функции waitForMuitipieobjects:
+ **ОТ WAIT_OBJECT_0 ДО (WAIT_OBJECT_0 + nCount - 1);**
- ОТ WAIT_OBJECT_0 ДО (WAIT_OBJECT_0);
- ОТ WAIT_OBJECT_0 ДО (WAIT_OBJECT_0 + nCount);
+ **ОТ WAIT_ABANDONED_0 ДО (WAIT_ABANDONED_0 + nCount - 1);**
- WAIT_COMPLETION
+ **WAIT_TIMEOUT**
- ОТ WAIT_ABANDONED_0 ДО (WAIT_ABANDONED_0 + nCount + 1);
- ОТ WAIT_ABANDONED_0 ДО (WAIT_ABANDONED_0);
21. Состояние блока адресов в адресном пространстве:
+ **выделен**
- защищен
+ **зарезвирован**
+ **свободен**
- смещен
- удален

- копирован
- прикреплен
22. Интерфейсы (API) для управления памятью:
- Base Services
- Common Control Library
+ **Virtual Memory**
- Network Services
+ **Memory Mapped File**
+ **Heap Memory**
- Windows Shell
- Windows System Information
23. Куча – это …
- объекты памяти
- страница памяти
+ **динамическая область памяти**
- указатель блока памяти
+ **блок памяти**
- указатель строки
+ **мелкие фрагменты памяти**
- указатель функции
24. Функции для управления памятью кучи:
+ **HeapAlloc()**
- HeapDestroy()
+ **HeapReAlloc()**
- HeapFree()
- HeapSize()
+ **HeapFree()**
- HeapCreate()
- MapViewOfFile()
25. Форматы реального и виртуального адресов:
+ **номер реальной страницы**
- номер фактической страницы
- номер оперативной памяти
+ **номер виртуальной страницы**
- смещение в адресной строке
+ **смещение в реальной и виртуальной странице**
- файлы страницы
- номер динамической страницы
26. Форматы реального и виртуального адресов:
- a
+ **r**
- b
+ **v**
- c
+ **d**

- e
- k
27. Описание линейного адреса процесса в ОС Windows:
+ **32 бит**
- 16 бит
+ **от 0x00000000 до 0xFFFFFFFF**
- 2 Гбайт логической памяти
- 4 Гбайт виртуальной памяти
- 2 Гбайт виртуальной памяти
+ **4 Гбайт логической памяти**
- от 0x00000000 до 0x0000FFFF
28. Состояние страницы процесса виртуальной памяти:
- (блокирован);
- (выделен)
+ **free (свободный)**
- (не выделен)
+ **committed (распределены)**
- (готов)
- (не готов)
+ **reserved (зарезвированый)**
29. Функция файловой системы:
- совместно использовать объект файла
+ **открытие доступа к существующему файлу**
- выделят мелкие фрагменты файла
+ **закрытие доступа к существующему файлу**
- закрепление виртуальной памяти
+ **установка указателя файла на нужную запись**
- физическое или форматирование низкого уровня
- разбиение диска на разделы
30. Структура и описание каталога:
+ **древовидная**
- кольцевая
+ **корневой**
- табличная
+ **\ (обратная косая)**
- прямая
- выпуклая
- плоская
31. Имена каталогов и файлов не должны содержат:
- +
+ <
- *
+ **:**
+ /
- ?

- !
- .
32. Параметр dwDesiredAccess задает способ доступа к файлу и принимает значения:
+ **0**
- FILE_SHARE_WRITE
- FILE_SHARE_READ
+ **GENERIC_READ**
+ **GENERIC_WRITE**
- OPEN_EXISTING
- OPEN_ALWAYS
- TRUNCATE_EXISTING
33. Значения параметра dwCreationDisposition при открытии файла:
- 0
- FILE_SHARE_WRITE
- FILE_SHARE_READ
- GENERIC_READ
- GENERIC_WRITE
+ **OPEN_EXISTING**
+ **OPEN_ALWAYS**
+ **TRUNCATE_EXISTING**
34. Правильная запись имени файла:
+ **“C:\\demo_file.dat”**
- “C:\\demo_file.dot”
- “C:\\demo\file.dat”
- “C\\demo_file.dat”
+ **“C:\\new_file.dat”**
- “C:\\ new_filedat”
+ **“C:\\back_file.dat”**
- “\\back_file.dat”
35. Значение функций GetFileType:
- FILE_SHARE_WRITE
- FILE_SHARE_READ
- GENERIC_READ
+ **FILE_TYPE_DISK**
+ **FILE_TYPE_CHAR**
- TRUNCATE_EXISTING
+ **FILE_TYPE_PIPE**
- OPEN_ALWAYS
36. Значения параметра функции CopyFile:
+ **lpExistingFileName**
- lpThreadAttributes
- AttachThreadInput
+ **lpNewFileName**
- dwStackSize

- lpStrtAddress
+ **bFailIfExists**
- TerminateThread
37. Верхние уровни, составляющие основу структуры реестра Windows:
- HKEY_LOCAL
- HKEY_LOCAL_WORD
+ **HKEY_LOCAL_MACHINE**
+ **HKEY_CURRENT_USER**
+ **HKEY_CLASSES_ROOT**
- HKEY_CURRENT_ MACHINE
- HKEY_CLASSES_ MACHINE
- HKEY_CLASSES_USER
38. API функции, применяемые для выполнения операций с реестром:
- RegReserverd
+ **RegCloseKey**
- RegClass
- RegSubKeys
+ **RegCreateKey**
- RegOpen
- RegQery
+ **RegDeleteKey**
39. API функции применяемые в среде Windows NT:
- RegQueryValue()
+ **RegSetKeySecurity()**
- RegCloseKey()
- RegFlushKey()
+ **RegGetKeySecurity()**
- RegLoadKey()
+ **параметры безопасности**
- RegOpenKey()
40. HKey идентифицирует текущий раздел или предопределенные дескрипторы:
- HKEY_LOCAL_MACHINE
- HKEY_LOCAL_WORD
+ **HKEY_CLASSES_ROOT**
- HKEY_CURRENT_ MACHINE
- HKEY_CLASSES_USER
+ **HKEY_CURRENT_USER**
- HKEY_CLASSES_ MACHINE
+ **HKEY_USERS**
41. Параметры функции WriteFile:
- nNumberOfBytes
- lpThreadAttributes
+ **hFile**
+ **lpBuffer**

- dwStackSize
- lpNewFileName
- bFailIfExists
+ **nNumberOfBytesToWrite**
42. Системный вызов для файла:
+ **chown()**
+ **lchown()**
- chmod
- fchmod
- stat()
- lstat()
+ **fchown()**
- read()
43. Метаданные файла связанные со временем:
- st_blocks
- st_dev
+ **st_ctime**
- st_uid
+ **st_atime**
- st_gid
- st_size
+ **st_mtime**
44. Функции для перемещения файла:
- lpNewFileName
+ **replaceFile**
- readFile
+ **copyFile**
- hFile
+ **moveFile**
- write_File
- delete_File
45. Механизм отображения файлов в память:
+ **отображение содержимого файла**
- установка времени таймера
+ **представление или вид файла**
+ **когерентность данных**
- указывать на функцию завершения
- создать файловое пространство
- открыть файл приложения
- создать файл загрузки
46. Параметры функции CreateFileMapping (создание объекта отображающий файл в память):
- lpBuffer
- numberOfBytes
- completionKey

+ **flProtect**
- readFile
+ **lpAttributes**
- waitCommEvent
+ **hFile**
47. Значения параметра flProtect :
- PAGE_READ
+ **PAGE_READONLY**
- PAGE_WRITEONLY
+ **PAGE_READWRITE**
- PAGE_READCOPY
+ **PAGE_WRITECOPY**
- PAGE_WRITE
- PAGE_COPYONLY
48. Прототип функции MapViewOfFile:
+ **HANDLE hFileMappingObject**
+ **DWORD dwFileOffsetHigh**
+ **DWORD dwFileOffsetLow**
- DWORD dwReserved
- BOOL bResume
- LPOVERLAPPED lpOverLapped
- HANDLE hThread
- DWORD IDThread
49. Прототип функции MapViewOfFileEx, отображающий файл в адресное пространства с некоторого заданного виртуального адреса:
+ **HANDLE hFileMappingObject**
- DWORD dwReserved
+ **DWORD DesireAccess**
+ **LPVOID lpBaseAddress**
- BOOL bResume
- LPOVERLAPPED lpOverLapped
- HANDLE hThread
- DWORD IDThread
50. Механизм отображения файлов в память:
- узнать какой поток ее вызывает
+ **файл отображен несколькими процессами**
- хранить указатели на захваченную память
- запускать приложение
+ **когерентность отображений**
+ **обмен данными между процессами**
- открыть файл приложения
- создать файл загрузки
51. Функции файла stdio.h языка программирования C, создающие стандартную библиотеку ввода-вывода:
- stdout— файл ввода

- stlib —файлы библиотеки
+ **stdin — стандартный файл ввода**
- tanh —математикалық шамаларды шығаратын файл
+ **stdout — стандартный файл вывода**
- strcmp —функция сравнения
+ **stderr —файл вывода сообщения об ошибке**
- main— программаны орындауды бастайтын файл
52. Прототип функции DllMain:
+ **HINSTANCE**
+ **DWORD**
+ **LPVOID**
- HMODULE
- LPCTSTR
- HANDLE
- DWORD
- LPCSTR
53. Параметр fdwReason может иметь одно из следующих значений, которое указывает на причину, по которой операционная система вызывает функцию DllMain:
- dont_resolve_dll_references
- load_library_as_datafile
- load_with_altered_search_path
+ **dll_process_attach**
+ **dll_thread_attach**
+ **dll_process_detach**
- case dll_process_attach
- dwPrewTlsIndex= dwFirstTlsIndex
54. Прототип функции LoadLibraryEx, для загрузки динамически подключаемых библиотек:
+ **LPCTSTR**
+ **HANDLE**
+ **DWORD**
- HINSTANCE
- DWORD_B
- LPVOID
- HMODULE
- LPCSTR
55. Значения параметра dwFiags, задающий флаги управления загрузкой модуля:
+ **dont_resolve_dll_references**
+ **load_library_as_datafile**
+ **load_with_altered_search_path**
- dll_process_attach
- dll_thread_attach
- dll_process_detach

- case dll_process_attach
- dwPrewTlsIndex= dwFirstTlsIndex
56. Действия необходимые для статической загрузки DLL:
- создать файл
+ **поместить библиотеку и файл в каталог**
- хранить указатели на захваченную память
- запускать приложение
- узнать какой поток ее вызывает
+ **ввести имя используемой библиотеки импорта**
- сопоставить каждому потоку свой указатель
+ **описать импортируемые из DLL имена в приложении**
57. Задачи динамической локальной памяти потока:
- создать DLL
- поместить библиотеку в каталог
+ **хранить указатели на захваченную память**
- поместить файл импорта этой библиотеки в каталог
+ **узнать какой поток ее вызывает**
- ввести имя используемой библиотеки импорта
+ **сопоставить каждому потоку свой указатель**
- описать импортируемые из DLL имена в приложении
58. Порядок работы с локальной памятью потока:
+ **распределение указателя**
- сохранить указатель
- использовать указатель
+ **работа с указателем**
+ **освобождение указателя**
- завершение функции
- возвращает значение
- запись значения
59. Динамические подключаемые библиотеки предназначены_____:
+ **для разработки функционально-замкнутых библиотек функций**
+ **для снижения затрат на разработку ПО**
- для открытия файла, который будет отображаться в динамической памяти
- для загрузки в адресное пространство процесса
- для завершения нового потока в процессе
- для сохранения файла и каталога
+ **для уменьшения физической памяти**
- для перехода в адресную строку
60. Функция для работы локальной памятью потока:
+ **TlsAlloc**
- LpBuffer
+ **TlsSetValue**
+ **TlsGetValue**
- CompletionKey
- ReadFile

- NumberOfBytes
- lpAttributes

61. Параметры функции CreateFileMapping (создание объекта отображающий файл в память):
- lpBuffer
- numberOfBytes
- completionKey
+ **flProtect**
- readFile
+ **lpAttributes**
- waitCommEvent
+ **hFile**

62. Распределение локальной памяти потока в DLL:
+ **case DLL_PROCESS_ATTACH**
- dwPrewTlsIndex= dwFirstTlsIndex
- dwPrewTlsIndex= dwNextTlsIndex
+ **dwlsIndex=TlsAlloc(); If (dwlsIndex ==-1)**
+ **break;**
- return 0;
- dDll=LoadLibrary;
- returnGetLastError();

63. Статическая локальная память:
- оповещает параллельные потоки
+ **использует спецификатор памяти declspec(thread)**
+ **определяет локальные переменные**
- устанавливает соединения между потоками данных
- принимает записи
- содержит очередь пакетов
- обслуживает очередь пакетов
+ **создает отдельный переменной**

64. Значения аргумента dwCreationDisposition для создания файла:
- CREATE_FILE
- CREATE_THREAD
+ **CREATE_NEW**
- OPEN
+ **CREATE_ALWAYS**
- CREATE
+ **OPEN_ALWAYS**
- CREATE_NEWFILE

65. Параметры функции для асинхронной записи данных в файл WriteFileEx:
+ **lpBuffer**
- numberOfBytes
- completionKey
- devceIoControl
- readFile

+ **lpOverLapped**
- waitCommEvent
+ **hFile**
66. Прототип функции UnlockFileEx для асинхронной отмены блокировки области файла:
- LONG lPeriod
+ **HANDLE hFile**
- HANDLE hTimer
+ **DWORD dwReserved**
- BOOL bResume
+ **LPOVERLAPPED lpOverLapped**
- HANDLE hThread
- DWORD IDThread
67. Функции, инициирующие посылку пакетов в порт завершения ввода-вывода:
- CreateIoCompletionPort
- NumberOfBytes
- CompletionKey
+ **DevceIoControl**
+ **ReadFile**
- NumberOfConcurrentThreads
+ **WaitCommEvent**
- hFile
68. Функции, инициирующие посылку пакетов в порт завершения ввода-вывода:
- CreateIoCompletionPort
- NumberOfBytes
- CompletionKey
+ **ConnectNamedPipe**
+ **LockFileEx**
- NumberOfConcurrentThreads
+ **TransactNamedFile**
- hFile
69. Порт завершения ввода-вывода:
+ **объект синхронизации**
- динамическая библиотека
+ **оповещает параллельные потоки**
- устанавливает соединения
- принимает записи
+ **содержит очередь пакетов**
- обслуживает очередь пакетов
- создает параллельные потоки
70. Параметры процедуры ввода-вывода:
- lpBuffer
+ **dwErrorCode**

- lpCompletionRoutine
- dwError
+ **dwNumberOfBytesTrnsferred**
- dwRet
- hFile
+ **lpOverLapped**

71. Механизм структурной обработки исключений:
+ **не допускается использование оператора goto**
- допускается использование оператора goto
+ **допускается использование функций GetExeptionCode**
- допускается использование оператора throw
- не допускается использование функций GetExeptionCode
- не допускается использование оператора throw
+ **допускается использование функций GetExeptionInformation**
- не допускается использование функций GetExeptionInformation

72. Заголовок файла об обработке исключений:
- string.h
+ **windows.h**
- stdio.h
- lm.h
- sddl.h
+ **iostream.h**
+ **float.h**
- math.h

73. Прототип обработки исключений с плавающей точкой:
- unsigned typedef void
+ **unsigned int _controlfp()**
- _se_translator_function
- void se_trans_func
+ **unsigned int new**
- unsigned int
+ **unsigned int mask**
- unsigned code

74. Блок исключений:
+ **_try**
- _search
+ **_except**
- _controlfp
+ **_finally**
- _exception
- _handle
- _function

75. Блок исключений:
+ **_try**
- _search

+ **_except**
- _controlfp
+ **_leave**
- _exception
- _handle
- _function
76. Параметры функции RaiseException:
+ **lpNewFileName**
- dwExceptionFlag
- setHandleInformation
+ **nNumberOfArguments**
+ **lpArguments**
- hReadPipe
- write_File
- delete_File
77. Значения параметра new для управления исключениями:
+ **_EM_DENORMAL**
- _EM_FUNCTION
- _EM_TRANSLATION
+ **_EM_OVERFLOW**
- _EM_BYZERO
- _EM_FLOAT
- _EM_WORD
+ **_EM_ZERODIVIDE**
78. Функция-транслятор_____:
+ **преобразовывает исключения**
- реализует исключения
+ **использует инструкцию throw языка C++**
- возвращает начальные значения
- завершает выполнение блока
- начинает раскрутку стека
+ **описывается в заголовочном файле eh.h**
- использует функцию _leave
79. Элементы безопасности объектов Win32:
- создает маркер доступа
+ **поддержка защищенных каналов**
- разрешает доступ к каналу
- создает аудит доступа к объекту
+ **поддержка интеллектуальных карточек**
- изменяет информацию безопасности
+ **встроенная поддержка функций API**
- следит за изменениями потока
80. Функции интеллектуальных карточек:
- поддержка защищенных каналов
+ **аутентификация пользователей**

+ **проведение финансовых операции**
- разрешает доступ к каналу
- создает аудит доступа к объекту
- изменяет информацию безопасности
- встроенная поддержка функций API
+ **хранение информации о человеке**
81. Режимы доступа к объектам:
+ **R, W;**
- RA
+ **WA;**
- CA
- WR
+ **WC;**
- RW
- CW
82. Модели управления в дискреционной модели безопасности:
+ **иерархическое управление**
- дистанционное управление
- сетевое управление
+ **либеральное управление**
- доступное управление
+ **централизованное управление**
- серверное управление
- древовидное управление
83. Основные функции списка управления доступами:
+ **создается владельцем объекта**
- создается сервером
- запрещает доступ к объекту
- создается операционной системой
+ **открывает требуемый доступ к объекту**
+ **хранится в виде списка**
- ограничивает код доступа
- используется сервером
84. Идентификатор безопасности (Security Identifier):
+ **создается операционной системой**
- создается сервером
- хранится в оперативной памяти
+ **хранится в базе данных SAM**
- хранится в логической памяти
- представляет учетную запись
+ **бинарное представление учетной записи**
- создает учетную запись
85. Символы обозначающие идентификатор безопасности:
- A
+ **S**

+ **R**
+ **I**
- T
- F
- L
- M

86. Дескрипторы безопасности, известные на платформах Windows:
- R -1-5-1
+ **S-1-5-1**
- R -1-5-5
- S -1-5-17
+ **S -1-5-10**
- I-1-5-16
+ **S -1-5-18**
- S-1-5-20

87. Главное отличие привилегий от прав доступа …
- права доступа ограничивает доступ субъекта к объектам
- никаких отличии привилегий от прав доступа
+ **правами доступа к объекту управляет владелец этого объекта**
+ **привилегии касаются субъектов, а не охраняемых объектов системы**
- привилегии назначаются по умолчанию в операционных системах Windows
- привилегии касаются охраняемых объектов системы
+ **привилегии назначаются субъектам администратором системы**
- права доступа выполнить некоторое действие по отношению объектам системы

88. Информация сохраненная в маркере доступа:
- SID учетной записи пользователя
+ **идентификатор безопасности текущей сессии (logon session)**
- каждое сообщение содержит информацию о субъекте
- статистическая информация о маркере доступа
+ **содержит информацию о субъекте, который выполнил действие**
- хранятся учетные записи пользователей и групп
+ **данные, определяющие политику безопасности на локальной машине**
- базу данных учетных записей

89. Уровни безопасности потока-сервера для обработки запросов потока-клиента:
+ **Security Anonymous level**
+ **Security Identification level**
+ **Security Impersonation level**
- Security Delegation
- Security Reference Monitor
- Security Account Manager
- Security Accounts Database
- Security_attributes

90. Уровни безопасности потока-сервера для обработки запросов потока-клиента:
+ **Object Open**
- ObjectType
+ **Object Deleted**
- Object_inherit
- pObjectName
+ **Object Open for Delete**
- se_file_object
- ObjectSecurity
91. Виды связей между процессами-отправителями и адресатами:
- между собой связаны три процесса
+ **между собой связаны только два процесса**
- между собой связаны четыре процесса
- один процесс связан с N-1 процессами
- один процесс связан с N+1 процессами
+ **один процесс связан с N процессами**
- нет связи между процессами
+ **каждый из N процессов связан с одним процессом**
92. Топология связей между процессами:
- между собой связаны три процесса
- между собой связаны четыре процесса
+ **между собой связаны только два процесса**
- один процесс связан с N-1 процессами
+ **один процесс связан с N процессами**
- один процесс связан с N+1 процессами
- нет связи между процессами
+ **каждый из N процессов связан с одним процессом**
93. Тип вместимости связи между процессами (буферизации):
- ограниченная связь
- открытый тип вместимости связи
+ **нулевая вместимость связи**
- закрытая вместимость связи
+ **ограниченная вместимость связи**
+ **неограниченная вместимость связи**
- каждый из процессов связан с одним процессом
- первичный вместимость связи
94. Размер значения именованного канала, посылающий сообщение к серверу:
+ **нулевая вместимость связи**
- каждый из процессов связан с одним процессом
- первичный вместимость связи
- закрытая вместимость связи
- параллельная связь вместимости
+ **неограниченная вместимость связи**

- каждый из N процессов связан с одним процессом
+ **ограниченная вместимость связи**
95. Анонимные каналы:
+ **не имеют имени**
- имеют названия
+ **полудуплексные**
- дуплексные
+ **передают данные потоком**
- не передает данные
- устанавливают связи
- доступны
96. Функции соединения клиентов анонимными каналами:
- CreatePipe
+ **DuplicateHandle**
- WriteFile
- hWritePipe
- hReadPipe
+ **SetHandleInformation**
- ReadFile
+ **CreateProcess**
97. Способы передачи наследуемого дескриптора процессу-клиенту анонимного канала:
+ **через командную строку**
- синхронный обмен
+ **через файл**
- топологическая связь
- основе wm_copydata
- с помощью сервер
- через канал
+ **посредством сообщения wm_copydata**
98. Функции описанные в файле stdio.h, обеспечивают ввод-вывод в стандартные файлы:
- printf
+ **stdin**
+ **stdout**
- scanf
- stdopen
+ **stderr**
- stdcreate
- cin
99. Прототип именованных каналов:
- stdin
+ **lpName**
+ **dwOpenMode**
- scanf

- stdout
+ **dwPipeMode**
- stdcreate
- cin

100. Флаги, для определения направления передачи данных:
+ **pipe_access_duplex**
- pipe_name
- nOutBufferSize
- nDefaultTimeOut
+ **pipe_access_inbound**
+ **pipe_access_outbound**
- dwOpenMode
- dwPipeMode

Методические материалы, определяющие процедуры оценивания знаний

Ключи к тестовым заданиям.

Шкала оценивания *(за правильный ответ дается 1 балл)*

«неудовлетворительно» – 50% и менее

«удовлетворительно» – 51-80%

«хорошо» – 81-90%

«отлично» – 91-100%

Критерии оценки тестового материала по дисциплине

«Системное программирование»:

✓ 5 баллов - выставляется студенту, если выполнены все задания варианта, продемонстрировано знание фактического материала (базовых понятий, алгоритма, факта).

✓ 4 балла - работа выполнена вполне квалифицированно в необходимом объёме; имеются незначительные методические недочёты и дидактические ошибки. Продемонстрировано умение правильно использовать специальные термины и понятия, узнавание объектов изучения в рамках определенного раздела дисциплины; понятен творческий уровень и аргументация собственной точки зрения

✓ 3 балла – продемонстрировано умение синтезировать, анализировать, обобщать фактический и теоретический материал с формулированием

конкретных выводов, установлением причинно-следственных связей в рамках определенного раздела дисциплины;

✓ 2 балла - работа выполнена на неудовлетворительном уровне; не в полном объёме, требует доработки и исправлений и исправлений более чем половины объема.