

Системное и прикладное программное обеспечение

Лабораторный практикум.

Новосибирск 2023

Лабораторный практикум содержит методические указания к выполнению лабораторных работ по курсу «Системное и прикладное программное обеспечение».

Составитель: канд. физ.-мат. наук В. М. Неделько.

Работа подготовлена кафедрой ПС и БД.

© Новосибирский государственный техниче-
ский университет, 2001 г.

© В. М. Неделько, 2023 г.

Содержание.

Правила аттестации.....	
Раздел 1. Программные средства для математических расчётов.....	
1.1. Базовые конструкции языка Python.....	
1.2. Использование библиотек numpy, scipy, matplotlib (Python).....	
1.3. Анализ данных средствами Python, pandas.....	
1.4. Разбор бинарных файлов средствами Python на примере образа диска	
1.5. Написание компилируемых (напр. на C++) библиотек для Python.....	
1.6. Вызов Python из приложений (в т.ч. на C++).....	
Раздел 2. Инструментарий пользователя ЭВМ.....	
2.1. Командная строка. Файловые менеджеры. Архивация данных.....	
2.2. Программный интерфейс ОС. Средства GDI.....	
2.3. Издательская система TeX. (согласовать задание).....	
Раздел 3. Управление ресурсами.....	
3.1. Файловая система (FAT, ext2fs).....	
3.2. Производительность функций чтения и записи файлов.....	

Правила аттестации

В рамках курса требуется выполнить и защитить 7 лабораторных работ (на 90 баллов — 6 работ, на 50 баллов — 3 работы), а также сдать зачёт в форме устного опроса по билетам.

Лабораторные работы выполняются в бригадах по 1-2 человека.

Работы, которые выделены шрифтом в оглавлении, являются рекомендованными. Их замена на другие работы допускается только по согласованию с преподавателем.

Оставшиеся работы выбираются на усмотрение учащегося из любых работ, в т.ч. из предлагавшихся в прошлые годы (27 работ).

Как минимум, одна работа должна быть выполнена в jupyter notebook.

Работы могут выполняться в среде Windows или Linux, при этом по крайней мере одна работа должна быть выполнена под Linux.

Одна и та же работа не может выполняться и под Windows, и под Linux. Следует обратить внимание, что некоторые работы имеют отдельные описания для Windows и Linux, но эти варианты считаются одной работой.

Оценивание за семестр проводится по балльно-рейтинговой системе.

При этом качество (и полнота) выполнения работ оценивается по традиционной шкале: от 2 до 5+.

Также каждая работа имеет вес. Итоговая оценка по курсу есть средневзвешенная оценка по всем заданиям. Оценка в Болонской системе и БРС вычисляется на основе традиционной оценки по таблице соответствия: 2,5 = 50 баллов, 3,0 = 65 баллов, 3,5 = 73 балла, 4,0 = 80 баллов, 4,5 = 87 баллов, 5,0 = 95 баллов, 5,5 = 100 баллов.

Зачёт выставляется при наборе не менее 50% баллов.

Оценка A+ ставится при условии безупречного выполнения всех заданий.

Материал к каждой лабораторной работе разделен на четыре части:

- обзорная информация,
- вопросы для изучения,
- задание к выполнению,
- контрольные вопросы.

Защита работ предполагает демонстрацию выполненного задания, а также готовность практически продемонстрировать владение материалом и инструментарием, указанным в вопросах для изучения и дать ответы на контрольные вопросы.

Требования к отчету.

Отчёт оформляется по одной лабораторной работе на выбор.

Все отчёты сдаются в электронном виде.

Отчет должен начинаться с заголовка, где указывается тема, исполнитель и цель работы.

Основную часть составляет раздел «Ход работы», в котором должны быть описаны проделанные действия с объяснением результатов и обязательным приведением конкретных значений (имен файлов, путей, размеров архивов, номеров кластеров и т. п.). Порядок выполнения шагов может отличаться от рекомендованного в задании.

Вывод может содержать, например, (аргументированное) заключение о степени удобства и эффективности изученного программного продукта и возможной области его применения.

Раздел 1. Программные средства для математических расчётов

Лабораторная работа 1.1. Базовые конструкции языка Python

Цель. Освоение базовых средств языка Python.

Вопросы для изучения.

1. Структура кода в Python. Основные операторы.
2. Встроенные типы и базовые операции. Целочисленное деление.
3. Строки и операции над ними.
4. Семантика присваивания.
5. Список, кортеж. Операции над списками. Срезы. Генераторы списков.
6. Словарь, множество.
7. Функции. Свойство `__doc__`.
8. Классы.
9. Модули.
10. Ввод-вывод для консоли и текстовых файлов.
11. Обработка исключений.
12. Функции – генераторы. Оператор `yield`.

Задание.

1. Выполните следующие операции над списком.
 - Создать список. Вывести срезы, а также список в обратном порядке.
 - Вывести список в строку через `;`.
 - Продемонстрировать результат операции `zip`.
 - Примеры генератора списка.
2. Напишите программу, которая читает файл и выдаёт частоту вхождений каждого слова.
3. Написать функцию и `lambda`-функцию.
4. Написать функцию, которая выдаёт на консоль все сочетания (модуль `itertools`) для заданных `m` и `n`. Сохранить функцию в отдельном модуле.
5. Продемонстрировать обработку исключения при обращении к несуществующему элементу словаря.

6. Написать примеры кода, демонстрирующие возможности языка, перечисленные в вопросах для изучения. В качестве функции-генератора реализовать генератор чисел Фибоначчи или простых чисел.
7. Создать GUI на tkinter.

Контрольные вопросы.

1. Когда происходит присваивание по ссылке, а когда по значению?
2. Как осуществляется работа с матрицами?
3. Арифметический условный оператор.
4. Почему циклы в Python используются реже, чем в классических языках?
5. Что возвращает функция-генератор?

Лабораторная работа 1.2. Использование библиотек numpy, scipy, matplotlib (Python)

Цель. Освоение базовых средств Python.

Вопросы для изучения.

1. Назначение библиотек numpy, scipy, matplotlib.
2. Основные операции над матрицами.
3. Импорт данных из текстовых файлов, вывод в текстовые файлы.
4. Двумерные и трёхмерные графики.
5. Символьные вычисления.

Задание.

1. Выполните следующие вычисления.
 - Пользуясь непосредственным вводом значений, задайте матрицу A , размером 3 на 3, и вектор свободных коэффициентов b .
 - Вызвав соответствующую функцию, получите решение системы линейных уравнений $Ax = b$.
 - Убедитесь в правильности решения путём подстановки, вычислите невязку (меру точности решения).
2. Освойте использование файлов для хранения данных.
 - В Excel задайте матрицу, размером 5 на 5, сохраните файл в формате csv.
 - Импортируйте файл в Python.
 - Найдите обратную матрицу, сохраните её в файл в формате csv.

— Откройте файл в Excel.

3. Двумерные графики.

— Постройте график некоторого полинома третьей степени.

— Найдите корни полинома.

— Постройте в одном окне два графика.

— Постройте спираль, пользуясь параметрическим заданием функции в декартовых координатах.

— Постройте график в полярных координатах.

4. Трёхмерные графики.

— Постройте графики двумерной плотности вероятности, представляющей собой смесь двух нормальных распределений. В качестве альтернативы можно взять любую функцию двух переменных с неединственным локальным экстремумом.

— Постройте контурную диаграмму.

— Постройте график пространственной кривой.

4. Символьные вычисления.

— Пользуясь символьными вычислениями, найдите производную выбранной функции. Вычислите неопределённый интеграл.

— Создайте изображение формулы.

Контрольные вопросы.

1. Чем различаются вектор-строка и вектор-столбец, как их задать?

Лабораторная работа 1.3. Анализ данных средствами Python, pandas

Цель. Получение навыков работы с pandas.

Вопросы для изучения.

1. Библиотека pandas.

2. Средства статистического анализа данных.

Задание.

1. Загрузить таблицу Iris из репозитория UCI (можно из любого доступного источника: UCI, sklearn, kaggle). Поместить данные в объект DataFrame библиотеки pandas. Вычислить корреляции между признаками на всей та-

блице и отдельно по классам (использовать `pd.groupby`). Визуализировать распределения классов на всех парах переменных.

2. Загрузить таблицу Mushroom из репозитория UCI (любого источника) в объект DataFrame. Вычислить распределение значений категориальных признаков по классам (использовать `pd.groupby`). Визуализировать распределения. Закодировать значения числами.

Альтернативное задание: На портале kaggle.com выбрать любую из учебных задач. Запустить на выбор один из скриптов, предлагаемых для её решения. Добиться совпадения полученных результатов с опубликованными.

Лабораторная работа 1.4. Разбор бинарных файлов средствами Python на примере образа диска

Цель. Получение навыков извлечения данных из структурированных двоичных файлов средствами Python.

Вопросы для изучения.

1. Двоичные файлы в Python.
2. Работа с байтовыми последовательностями.
3. Битовые операции.

Задание.

1. Открыть образ диска как бинарный файл.
2. Прочитать область FAT в массив `numpy`. Вычислить объём, занимаемый файлами.
3. Выдать список (имена) всех файлов с диска.
4. Прочитать из загрузочной записи параметры ФС.
5. Считать содержимое одного из файлов.

Лабораторная работа 1.5. Взаимодействие Python с приложениями на C++

Цель. Получение навыков программирования в Python. Изучение средств взаимодействия Python с кодом на C++.

Вопросы для изучения.

1. Создание новых функций.
2. Подключение C++ модулей в Python.
3. Вызов Python из программ на C++.

Задание.

1. Создать функцию для вычисления площади эллипса, задаваемого уравнением $1 = x'Ax$.
2. Реализовать эту же функцию на C++ и осуществить её вызов из среды Python. Убедиться в правильности полученного результата.
3. Вызывая Python из программы на C++, построить график одномерной функции по своему выбору.
4. Реализовать на C функцию сортировки. Сравнить быстродействие со встроенной.

Раздел 2. Инструментарий пользователя ЭВМ

Лабораторная работа 2.1. Работа в режиме командной строки. Переменные окружения. Командные файлы. Файловые процес- соры. Архиваторы

Цель. Изучение принципов организации диалога пользователя с ОС средствами командной строки. Приобретение практических навыков по управлению ОС из командной строки. Ознакомление с операциями переадресации и фильтрации (шаблон имени файла). Получение начальных навыков по созданию пакетных файлов и использованию переменных окружения. Освоение возможностей файловых процессоров. Освоение типовых операций по работе с архивами. Изучение распространенных архиваторов.

2.1.а. Вариант для ОС Windows

Вопросы для изучения.

1. Структура и основные элементы командной строки DOS.
2. Основные команды для манипулирования файлами.
3. Шаблон (маска) имени файла. Его использование.
4. Переменные окружения. Команда set.
5. Утилита start.
6. Пакетные (командные) файлы.
 - Принцип организации.
 - Дополнительные возможности, по сравнению с командной строкой.
 - Ввод параметров. Использование переменных окружения.
7. Операции с каталогами и файлами в FAR–manager.
 - Создание, переименование, перемещение, копирование, просмотр, удаление файлов и каталогов. Просмотр и изменение атрибутов файла. Просмотр информации об устройстве.
 - Поиск файлов и каталогов.
 - Связывание файлов с приложениями.
 - Подключение сетевых дисков.
 - Запуск приложений из FAR в отдельной области памяти (Shift-Enter).
8. Управление конфигурацией и режимы панелей.
 - Выделение файлов (files highlighting).
 - Сохранение конфигурации (Shift-F9).

- Сортировка файлов на панели.
- Сравнение каталогов.

9. Редактирование в редакторе FAR.

- Кодировки. Изменение кодировки текста.
- Использование буфера обмена для копирования блоков текста внутри документа и для обмена текстом между документами: между двумя файлами в FAR; между FAR и Word.

10. Работа с архиваторами из командной строки.

- Типовые операции с архивом: создание, обновление, пополнение, удаление, просмотр оглавления.
- Параметры архивирования: сохранение структуры каталогов, архивация с паролем.
- Создание многотомных архивов.

Задание.

1. Запустив консоль (сеанс MS DOS, cmd.exe), выполните следующие действия.

- Сделайте текущим каталог d:\work (команды d: и cd \work).
- Создайте каталог my.
- Скопируйте все файлы с расширением txt из work в my.
- Выдайте содержимое my.
- Измените системное приглашение (prompt).
- Создайте текстовый файл (COPY CON file ... Ctrl-Z).

При необходимости пользуйтесь справкой.

Следующие действия можно выполнять из командной строки оболочки FAR-manager.

2. Используйте средства перенаправления вывода.

- Создайте файл, содержащий список файлов текущего каталога с подкаталогами (требуется только список полных имён без другой информации).
- Создайте файл со справкой для команды rag.

3. Выдайте значения переменных окружения. Выясните смысл нескольких переменных.

4. Создайте командные файлы с перечисленными ниже функциями.

- Выдается сначала список файлов с расширением txt, затем с расширением doc.
- Выдается содержимое каталога, указанного в переменной окружения TEMP (или TMP).

- При наличии двух параметров производится копирование файла с именем, заданным первым параметром, во второй. Если параметр один, то файл выдается на терминал. При другом числе параметров выдается ошибка.
5. Найти в реестре раздел, описывающий созданную ассоциацию для расширения `.pas` (найти в `HKEY_CLASSES_ROOT` раздел `.pas`, запомнить значение по-умолчанию — это идентификатор типа, затем найти раздел с таким именем в корне `HKEY_CLASSES_ROOT`). Создать по аналогии раздел, связывающий `.cpr` с `bc.exe` (или с `notepad.exe`).
 6. Открыть на панелях FAR два каталога с разных дисков (устройств). Выделить на первой панели (не подряд) несколько файлов (клавиша `Ins`), скопировать в личный каталог.
 7. Связать `*.pas` с `x:\bp\bin\bp.exe` (или, например, с блокнотом) так, чтобы в окне редактирования появлялся открываемый файл (использовать `!!` в качестве параметра команды).
 8. Вставить произвольный текст из документа Word в текстовый файл в редакторе FAR. Добиться, чтобы в тексте одновременно присутствовали фрагменты в разных кодировках.
 9. Вызывая `gar` (или другой архиватор) из командной строки, поместить заданные файлы в архив. Заархивировать файлы каталога соответствующие маске, задав дополнительно пароль архива. Извлечь файлы из архива.
 10. С помощью `gar` создать многотомный архив, сохраняя структуру каталогов. Разархивировать файлы в другой каталог.

Контрольные вопросы.

1. Как сравнить два файла (по содержимому)?
2. Файлы с какими расширениями могут выполняться DOS?
3. Чем отличаются пути `\my` и `my`?
4. Для чего используется переменная окружения `path`?
5. Приведите примеры команд, которые работают в пакетных файлах, но не могут использоваться в командной строке.
6. Какие символы используются в маске имени файла?
7. Перечислите символы перенаправления ввода–вывода.
8. Что означают в DOS имена `CON`, `PRN`?
9. Для чего нужен реестр Windows?
10. Чем отличается связывание файлов с приложениями в FAR и в Windows?
11. По какому правилу выделяются файлы при выполнении сравнения каталогов?
12. Как перенести в командную строку имя текущего файла?

13. Как поместить в буфер обмена путь к текущему каталогу?
14. Какие кодировки наиболее употребительны?
15. Как запустить из-под FAR приложение DOS в отдельной области памяти (в отдельном окне)?
16. Вы не помните имя файла, но знаете фрагмент содержимого, как можно найти требуемый файл?
17. Как можно результаты поиска сохранить в файле?
18. На чем основывается сжатие информации?
19. Что такое архивный файл?
20. В чем отличие команд *e* и *x* для *rar*?
21. Что такое многотомный архив?
22. Что означает самораспаковывающийся архив?
23. Что содержит оглавление архива?
24. Как осуществляется архивация с паролем?

2.1.b. Вариант для ОС Linux

Общие сведения.

Для взаимодействия с пользователем Unix-подобные системы поддерживают консоль и некоторое количество виртуальных или удаленных терминалов. После регистрации пользователя на терминале для него запускается интерпретатор команд, предоставляющий полный интерфейс к ОС.

В задачу командного процессора входит также интерпретация и выполнение командных файлов, предоставляя для их написания развитый скриптовый язык программирования.

Вопросы для изучения.

1. Структура и основные элементы командной строки. Специальные символы.
2. Основные команды для манипулирования файлами.
3. Операторы `;`, `&`, `&&`, `||`.
4. Перенаправление ввода-вывода (`<`, `>`, `>>`, `|`), каналы и фильтры.
5. Переменные окружения. Команда `set`.
6. Команда `export`.
7. Раскрытие выражений. Шаблон имени файла.
8. Командные файлы.
 - Принцип организации.
 - Условные операторы и циклы.
 - Ввод параметров.

- Команда `source` или `.'`.
- 9. Команда `su`.
- 10. Монтирование и размонтирование файловых систем.
- 11. Операции с каталогами и файлами в `mc` (Midnight commander).
 - Создание, переименование, перемещение, копирование, просмотр, удаление файлов и каталогов. Просмотр и изменение атрибутов файла. Просмотр информации об устройстве.
 - Поиск файлов и каталогов.
 - Связывание файлов с приложениями.
 - Подключение сетевых дисков.
- 12. Управление конфигурацией и режимы панелей.
 - Выделение файлов (files highlighting).
 - Сохранение конфигурации (Shift-F9).
 - Сортировка файлов на панели.
 - Сравнение каталогов.
- 13. Редактирование во встроенном редакторе `mc`.
- 14. Запуск приложений.
- 15. Работа с архиваторами из командной строки.
 - Типовые операции с архивом: создание, обновление, пополнение, удаление, просмотр оглавления.
 - Параметры архивирования: сохранение структуры каталогов, архивация с паролем.
 - Создание многотомных архивов.
- 16. Работа с архивами в `mc`.

Задание.

1. Открыв консоль, выполните следующие действия.
 - Сделайте текущим домашний каталог (`cd ~`).
 - Создайте каталоги `Work` и `Work/my`.
 - Скопируйте все файлы с расширением `txt` из `Work` в `my`.
 - Выдайте содержимое `my`.
 - Измените системное приглашение (`ps1`), так чтобы оно стало похожим на приглашение DOS.
 - Создайте текстовый файл (`cat >file ... Ctrl-D`).При необходимости пользуйтесь справкой.
Следующие действия можно выполнять из командной строки оболочки `mc`.
2. Используйте средства перенаправления вывода.

- Создайте файл, содержащий список файлов текущего каталога с подкаталогами.
 - Создайте файл со справкой для команды `tag`.
3. Выдайте значения переменных окружения. Выясните смысл нескольких переменных.
 4. Создайте командные файлы с перечисленными ниже функциями.
 - Выдается сначала список файлов с расширением `txt`, затем с расширением `dos`.
 - Выдается содержимое каталога, указанного в переменной окружения `TEMP` (или `TMP`).
 - При наличии двух параметров производится копирование файла с именем, заданным первым параметром, во второй. Если параметр один, то файл выдается на терминал. При другом числе параметров выдается ошибка.
 5. На первую панель поместить любой доступный каталог с файлами небольшого размера, на вторую `~/work/my`. Выделить на первой панели (не подряд) несколько файлов (клавиша `Ins`), скопировать в каталог `~/work/my`.
 6. Связать `*.pas` с `freepascal` (или с подходящим редактором) так, чтобы в окне редактирования появлялся открываемый файл.
 7. Создать виртуальный диск. Смонтировать его ФС и скопировать на неё несколько файлов.
 8. Вызывая `tag` из командной строки, поместить заданные файлы в архив. Заархивировать файлы каталога соответствующие маске, задав дополнительно пароль архива. Извлечь файлы из архива.
 9. С помощью `tag` создать многотомный архив, сохраняя структуру каталогов. Разархивировать файлы в другой каталог.
 10. Сжать файл, используя `gzip`.
 11. Работая в `ms`, выделить заданные файлы и поместить их в архив. Выдать архив на панель (работа, как с каталогом). Извлечь заданные файлы из архива.

Контрольные вопросы.

1. Как сравнить два файла (по содержимому)?
2. Как отличить исполняемый файл?
3. Чем отличаются пути `/my` и `my`?
4. Для чего используется переменная окружения `PATH`?
5. Как передать вывод одной программы другой программе в качестве параметров?
6. Какие символы используются в маске имени файла?

7. Перечислите символы перенаправления ввода–вывода.
8. Как заставить команду `sr` вывести файл на терминал?
9. Чем отличается связывание файлов с приложениями в `ms` и в `Windows`?
10. По какому правилу выделяются файлы при выполнении сравнения каталогов?
11. Как перенести в командную строку имя текущего файла?
12. Вы не помните имя файла, но знаете фрагмент содержимого, как можно найти требуемый файл?
13. Как можно результаты поиска сохранить в файле?
14. На чем основывается сжатие информации?
15. Что такое архивный файл?
16. Что такое многотомный архив?
17. Что означает самораспаковывающийся архив?
18. Что содержит оглавление архива?
19. Как осуществляется архивация с паролем?

Лабораторная работа 2.2. Программный интерфейс MS Windows: Win API. Средства GDI

Цель. Изучение основ взаимодействия приложений с ОС `Windows`.
Средства `GDI`

Все взаимодействие `Windows` с приложением основано на передаче сообщений. Каждое сообщение представляется 32-разрядным кодом.

Основным интерфейсным элементом `Windows` являются окна, поэтому именно окна выступают в качестве конечных адресатов сообщений. И действительно, большая часть сообщений логически привязано к определенному окну: перемещение, изменение размеров окна, щелчок мыши в области окна. Ввод с клавиатуры также привязывается к окну, являющемуся в данный момент активным.

Для реакции на сообщения каждому окну приложение должно сопоставить функцию обработки. Эта функция вызывается операционной системой автоматически для каждого пришедшего сообщения.

Однако первоначально сообщения попадают не в функцию окна-адресата, а в головной модуль приложения. Причем головной модуль `windows`–приложения обязан сам регулярно запрашивать сообщения, чтобы затем... от-

правлять их обратно операционной системе (обычно головной модуль содержит цикл, который вплоть до завершения программы выполняет две указанных операции). Такая идеология выглядит переусложненной, однако в ней есть определенная логика: функции окон играют роль обработчиков событий, а головной модуль имеет возможность контроля всех сообщений.

Ниже приводится пример минимальной программы для Windows.

Для понимания данного примера практически не требуется знание языка C, как такового, однако необходимо представление об основных элементах программ и общих принципах программирования.

```
#include <windows.h>

LRESULT CALLBACK WindowFunc(HWND, UINT, WPARAM, LPARAM);
char szWinName[]="MyWin";

// Головной модуль
int PASCAL WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst,
                  LPSTR lpszArgs, int nWinMode){
    HWND hwnd,hwnd1;
    MSG msg;
    WNDCLASS wcl;

    wcl.hInstance=hThisInst;
    wcl.lpszClassName=szWinName;
    wcl.lpfnWndProc=WindowFunc;
    wcl.style=0;

    wcl.hIcon=LoadIcon(NULL, IDI_APPLICATION);
    wcl.hCursor=LoadCursor(NULL, IDC_ARROW);
    wcl.lpszMenuName=NULL;

    wcl.cbClsExtra=NULL; wcl.cbWndExtra=NULL;

    wcl.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);

    // Регистрация класса окна
    if(!RegisterClass(&wcl)) return 0;

    // Создание главного окна приложения
    hwnd=CreateWindow(szWinName, "Window Skeleton",
                     WS_OVERLAPPEDWINDOW,
                     CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
                     HWND_DESKTOP, NULL, hThisInst, NULL);

    // Отрисовка окна
```

```

ShowWindow(hwnd, nWinMode);
UpdateWindow(hwnd);

// Выдача диалогового окна
MessageBox(hwnd, "Message", "test", MB_OK);

// Цикл обработки сообщений
while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
return msg.wParam;
}

// Функция окна
LRESULT CALLBACK WindowFunc(HWND hwnd, UINT message,
                             WPARAM wParam, LPARAM lParam) {
    switch (message) {
        case WM_DESTROY:
            PostQuitMessage(0); break;
        default:
            return DefWindowProc(hwnd, message, wParam, lParam);
    }
    return 0;
}

```

Данная программа содержится в одном файле. Она может быть оттранслирована практически любым компилятором (например, Borland C 3.11).

Головной модуль программы сначала определяет и регистрирует в системе класс окна. Класс представляет собой совокупность свойств окна, в том числе, с ним связывается функция окна. Таким образом, два окна одного класса будут «близнецами».

Затем создается окно данного класса и вызывается функция его отображения.

Следующим шагом выдается окно сообщения (это, разумеется, необязательный момент, который добавлен лишь для «оживления» картинки).

Наконец, последний блок – это цикл обработки сообщений.

Функция окна сама обрабатывает лишь одно сообщение: **WM_DESTROY**, которое генерируется при закрытии окна (вызывается функция **PostQuitMessage()**, которая генерирует сообщение, завершающее приложение). Для остальных сообщений вызывается функция обработки по умолчанию: **DefWindowProc()**.

Вопросы для изучения.

1. Основные элементы взаимодействия приложения с ОС Windows.
 - Сообщения.
 - Цикл обработки сообщений.
 - Функция окна.
2. Этапы создания окна.
3. Элементы оконного интерфейса. Виды окон.

Задание.

1. Отредактировать и оттранслировать пример, приведенный выше.
2. Проверить работоспособность. Проследить выполнение шагов по тексту программы.
3. Модифицировать код, добавив создание второго окна, дочернего, с помощью вызова:

```
hwnd1=CreateWindow(szWinName, "Window 1",  
    WS_OVERLAPPEDWINDOW | WS_CHILD ,  
    CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,  
    hwnd, NULL, hThisInst, NULL);
```

Сравнить поведение окон.

4. Добавить выдачу дополнительного сообщения, например,
`MessageBox(hwnd,"extra message","text",MB_YESNO);`
5. Реализовать функцию отрисовки геометрической фигуры в заданном месте по клику мыши.
6. Исследовать ассемблерный листинг кода системного вызова. Найти инструкции, непосредственно относящиеся к вызову.

Контрольные вопросы.

1. Какие типы окон Вы знаете?
2. Какую особенность имеют модальные окна диалога (сообщения)?
3. В чем особенность дочерних окон?
4. Для чего нужен цикл сообщений?

Лабораторная работа 2.3. Издательская система TeX.

Цель. Освоение использования редактора TeX для оформления математических текстов.

Общие сведения.

Несмотря на предоставляемые в MS Word мощные средства по созданию документов, этот продукт имеет недостатки и не вполне удобен, напри-

мер, для написания математических текстов. В связи с этим актуальным является использование системы TeX, которая надежно работает с большими текстами и, кроме того, предоставляет значительно более удобный (при некотором навыке) инструмент по набору математических формул. Кроме того, TeX позволяет делать оформление более профессионального издательского уровня.

Вопросы для изучения.

1. Создание текстовых документов в LaTeX.
 - Заголовок и тело документа.
 - Форматирование текста.
 - Компиляция и просмотр.
2. Включение формул.
3. Обработка и включение графики.

Задание.

1. Оформите в TeX на выбор фрагмент (1 стр.) лекции по математическому анализу.
2. Возьмите из Интернет подходящую картинку. С помощью графического редактора преобразуйте ее в формат eps (для pdf_latex нужен формат png), задав подходящие размеры, и вставьте в текст.
3. Создайте dvi и pdf файлы (утилиты ...MiKTeX 2.5\miktex\bin\latex.exe и ...pdf_latex.exe). Оцените результат.
4. Выберите любой математический журнал с требованиями представления работ в TeX. Оформите произвольный математический текст в соответствии с требованиями журнала.
5. В качестве альтернативы предыдущим заданиям оформить отчёт по РГР.
6. Скомпилируйте пример презентации с использованием пакета beamer.

Контрольные вопросы.

1. Чем отличаются математический и текстовый режимы?
2. Каковы особенности «выключенных» формул?
3. Назовите способы устранения ошибки "**Overfull hbox**".
4. Как задать требуемый размер полей?
5. В чем отличие `\linebreak` от `\newline`?
6. Что содержит «шапка»?
7. Какие графические форматы поддерживает TeX?

Раздел 3. Управление ресурсами ВС

Лабораторная работа 3.1. Файловая система (FAT, ext2fs).

Цель. Ознакомление с файловой системой. Приобретение навыков по восстановлению файловой структуры средствами низкого уровня.

3.1.a. Вариант для ОС Windows

Вопросы для изучения.

1. Физическая модель диска. Физический адрес сектора.
2. Логическая модель диска. Основные элементы файловой системы.
3. Загрузочная запись.
4. Организация таблицы размещения файлов.
5. Структура каталога.
6. Интерфейс программы diskeditor.

Задание.

Работа выполняется под эмулятором в виртуальной DOS.

1. Скопировать на виртуальную дискету произвольный текстовый файл, размером не менее 2-х кб. Удалить его. Восстановить файл в diskedit-е "вручную", т. е. непосредственно корректируя запись в каталоге (dir) и таблицу размещения файлов (FAT).
2. Скопировать на дискету два текстовых файла. Объединить их в один, не перемещая данные, а корректируя цепочку кластеров в FAT и информацию в dir.
3. Создать в diskedit новый текстовый файл (выбрать свободный кластер, записать в него текст, создать запись в каталоге и цепочку в FAT).
4. Освоить операцию поиска в diskedit (найти кластер с записанным текстом).
5. Проверить дискету NDD или ScanDisk. Исправить ошибки, если таковые допущены.

В отчете описать последовательность действий (указывая конкретные размеры файлов, в какие элементы FAT какие числа записываются и т. д.).

Контрольные вопросы.

1. В каком кластере находится корневой каталог?
2. Перечислите элементы записи каталога.
3. Каков принцип хранения длинных имен файлов в каталоге системы FAT?

4. Чем отличается физический и логический адрес сектора?
5. Каков размер сектора? Кластера?
6. Какие записи являются обязательными для некорневого каталога?
7. Что содержит загрузочный сектор?

3.1.b. Вариант для ОС Linux

Общие сведения.

Сокращение ext2fs расшифровывается как "вторая расширенная файловая система" (second extended filesystem). При этом имеются в виду расширения по отношению к ФС minix, использованной в первых версиях Linux.

Рассматриваемая файловая система состоит из следующих элементов:

- начальный (загрузочный) блок,
- суперблок,
- область индексных дескрипторов (INOD-s),
- область данных.

Весь объем логического диска разбит на блоки фиксированного размера, обычно по 1024 байта. Первый блок является резервным и служит для хранения загрузчика, если такой имеется.

Суперблок содержит общую информацию о ФС, в частности размеры ее элементов.

Область индексных дескрипторов является массивом записей, описывающих файлы. Каждая запись (INOD) хранит следующую информацию о файле:

- тип файла,
- время создания и модификации,
- идентификаторы владельца и группы владельца,
- права доступа и атрибуты,
- номера блоков.

В большинстве файловых систем поддерживается семь типов файлов

- обычные файлы,
- каталоги,
- файлы байт-ориентированных (символьных) устройств,
- файлы блок-ориентированных (блочных) устройств,
- сокеты,
- именованные каналы (FIFO),
- символические ссылки.

В данной работе нас будут интересовать только обычные файлы и каталоги. Каталог представляет собой файл, представляющий собой список

файлов и состоящий из массива записей, включающих имя файла и номер индексного дескриптора. Заметим, что один и тот же файл может быть описан в разных каталогах (в том числе под разными именами).

В INOD хранится информация о размещении файла на диске. Имеется массив из 13 элементов, первые 10 из которых являются непосредственно номерами блоков, занятых файлом, 11-й элемент указывает на блок 1-го уровня косвенности, 12-й и 13-й — на блоки 2-го и 3-го уровней соответственно.

Блок 1-го уровня содержит продолжение массива блоков, занятых файлом (если файл имеет размер более 10 блоков). Если одного блока 1-го уровня недостаточно для описания всех блоков, занятых файлом, то используется блок 2-го уровня, содержащий массив номеров дополнительных блоков первого уровня. аналогичным образом при необходимости используется блок 3-го уровня.

Учитывая, что один блок может содержать до 256 указателей на блоки, получаем, что максимально допустимый размер файла в ext2fs составляет: $1024 \cdot (10 + 256 + 256 \cdot 256 + 256 \cdot 256 \cdot 256) = 17,247,250,432$ байт.

Для выполнения работы необходим доступ на запись к устройству с ext2fs, это может быть виртуальный диск или FDD. В последнем случае диску необходимо отформатировать командой: `mkfs -t ext2 /dev/fd0`

Также для работы потребуется дисковый редактор. Это может быть, например lde (Linux Disk Editor) и утилита debugfs из пакета e2fsprogs.

Вопросы для изучения.

1. Физическая модель диска. Физический адрес сектора.
2. Логическая модель диска. Основные элементы файловой системы.
3. Загрузочная запись (резервный блок).
4. Суперблок.
5. Структура INOD.
6. Структура каталога.
7. Интерфейс дискового редактора.

Задание.

Работа выполняется под эмулятором в виртуальном Linux.

1. Скопировать на дискету (виртуальный диск) произвольный текстовый файл, размером не менее 2-х кб. Удалить его. Восстановить файл, пользуясь дисковым редактором, непосредственно корректируя запись в каталоге и соответствующий INOD.

2. Скопировать на дискету два текстовых файла. Объединить их в один, не перемещая данные, а корректируя цепочку блоков в индексных дескрипторах и информацию в каталогах.
3. Создать новый текстовый файл (выбрать свободный блок на диске, записать в него текст, создать запись в каталоге и INOD).
4. Освоить операцию поиска (найти блок с записанным текстом).
5. Проверить целостность ФС. Исправить ошибки, если таковые допущены.
В отчете описать последовательность действий (указывая конкретные размеры файлов, содержимое INOD и т. д.).

Контрольные вопросы.

1. В каком блоке находится корневой каталог?
2. Перечислите элементы записи каталога.
3. Каков принцип хранения информации о блоках, занятых файлом?
4. Чем отличается физический и логический адрес сектора?
5. Каков размер сектора? Блока?
6. Какие записи являются обязательными для каталога?
7. Что содержит загрузочный сектор?
8. Что содержит суперблок?
9. Как хранится информация о свободных блоках?
10. Как отличить свободный INOD ?

Лабораторная работа 3.2. Измерение производительности функций работы с файлами.

Цель. Выработка навыков эффективной работы с файлами программными средствами.

Общие сведения.

Для выполнения работы можно взять за основу следующий код.

```
#include <fstream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/timeb.h>

static int delta_time(const _timeb &t1, const _timeb &t2)
{
    return (int) ((t2.time - t1.time) * 1000
        + ((int) t2.millitm) - ((int) t1.millitm));
}

using namespace std;

static int test_fprintf()
{
    _timeb time0;
    _timeb time1;

    FILE* t = fopen("test_fprintf.txt", "wt");

    _ftime(&time0);

    for (int i=0; i<500000; i++)
        fprintf(t, "%i %i %i %i\n", i*4, i*4+1, i*4+2, i*4+3);

    _ftime(&time1);

    fclose(t);
    return delta_time(time0, time1);
}

static int test_ofstream()
{
    _timeb time0;
    _timeb time1;

    ofstream outfile("test_ofstream.txt");

    _ftime(&time0);

    for (int i=0; i<100000; i++)
        outfile << i*4 << ' ' << i*4+1 << ' ' << i*4+2 << ' ' << i*4+3 << '\n';

    _ftime(&time1);
}
```

```

    outfile.close();
    return delta_time(time0,time1);
}

static int test_ifstream()
{
    _timeb time0;
    _timeb time1;

    ifstream infile("test_ofstream.txt");

    _ftime(&time0);

    int x1,x2,x3,x4;

    for (int i=0; i<100000; i++){
        infile >> x1 >> x2 >> x3 >> x4;
        //printf("%i %i %i %i\n",x1,x2,x3,x4);
    }

    _ftime(&time1);

    infile.close();
    return delta_time(time0,time1);
}

static int test_fscanf()
{
    _timeb time0;
    _timeb time1;

    FILE* t = fopen("test_fprintf.txt","rt");

    _ftime(&time0);

    int x1,x2,x3,x4;

    for (int i=0; i<500000; i++){
        fscanf(t,"%i %i %i %i\n",&x1,&x2,&x3,&x4);
        //printf("%i %i %i %i\n",x1,x2,x3,x4);
    }

    _ftime(&time1);

    fclose(t);
    return delta_time(time0,time1);
}

static long filesize(FILE *stream)
{
    const long curpos = ftell(stream);
    fseek(stream, 0L, SEEK_END);
    const long length = ftell(stream);
    fseek(stream, curpos, SEEK_SET);
    return length;
}

static int test_fread()
{
    _timeb time0;
    _timeb time1;

```

```

FILE* b = fopen("test_fprintf.txt","rb");
const long size = filesize(b);

char* buf = new char[size];

_ftime(&time0);

const int N = (int) fread(buf,1,size,b);

_ftime(&time1);

fclose(b);
delete [] buf;
return delta_time(time0,time1);
}

int main(int argc, char* argv[])
{
    const int dt_fprintf = test_fprintf();
    printf( "\ntest_fprintf takes %i milliseconds\n", dt_fprintf);

    const int dt_ofstream = test_ofstream();
    printf( "\ntest_ofstream takes %i milliseconds\n", dt_ofstream);

    const int dt_ifstream = test_ifstream();
    printf( "\ntest_ifstream takes %i milliseconds\n", dt_ifstream);

    const int dt_fscanf = test_fscanf();
    printf( "\ntest_fscanf takes %i milliseconds\n", dt_fscanf);

    const int dt_fread = test_fread();
    printf( "\ntest_fread takes %i milliseconds\n", dt_fread);

    getch();
    return 0;
}

```

Вопросы для изучения.

1. Библиотека stdio.h
2. Библиотека iostream.
3. Поэлементное и буферизованное чтение.
4. Работа с текстовыми файлами.
5. Фрагментация файлов.
6. Кэширование.

Задание.

1. Сравните быстродействие fprintf и <<.
2. Сравните быстродействие >> и комбинации fgets+sscanf.
3. Сравните быстродействие поэлементного и буферизованного вывода. Проследите зависимость от размера буфера.
4. Оцените эффект кэширования.

5. Сравните скорость чтения фрагментированных и нефраgmentированных файлов. Задание не является обязательным, за выполнение даются дополнительные баллы.

Контрольные вопросы.

1. Чем `iostream` лучше `stdio.h`?
2. Чем `stdio.h` лучше `iostream`?
3. Сформулируйте основные рекомендации по увеличению скорости работы с файлами.