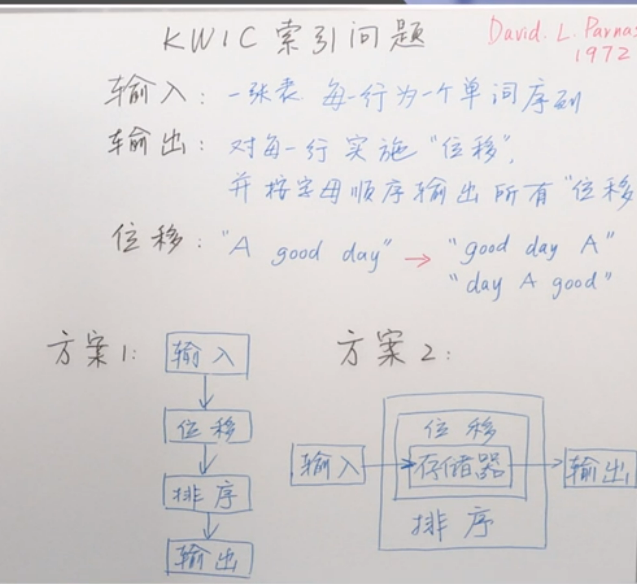
第一章

* 什么是系统
  1. 模块：库函数、独立的系统……
  2. 独立运行的，组成软件的一个独立模块
* 系统的特性：
  1. 通用性：功能用途广
  2. 隔离性：模块间关系简单
  3. 精巧型：性能好、易于使用
* 数据管理系统的功能：
  1. 数据存放：安全性
  2. 数据组织：有序、高效
  3. 数据正确性
  4. 处理平台
* 课程内容安排：
  1. 如何使用：构建软件，组织、索引……
  2. 构建思想
  3. 实现原理

RDBMS NoSQL Search Engine Hadoop

* 程序模块化设计：
  1. 隔离/封装性
  2. 
  3. 信息隐藏
* 模块合作的方式：
  1. API调用
  2. 库函数Library
  3. 框架Framework

测验：

**第*1*题：如课程中提到，由于大部分应用程序都需要数据管理功能，因此人们将数据管理功能抽象出来，形成可独立运行的系统。然而并非所有的功能抽象都有必要成为独立的系统。比如“排序”这一功能，只需要构造成一个库函数，供各种程序调用即可。那么，为什么数据管理功能需要被构造成独立运行的系统？以下那种说法不成立？**

A：数据管理功能很复杂，复杂的功能需要成为独立的系统。√

不是因为功能复杂才将数据管理作为独立的系统，而是因为数据管理功能通途很广

B：数据管理需要消耗大量资源（包括CPU、内存、硬盘），独立系统可实现独立的资源调配，有利于资源的高效利用。

C：同一份数据有时候需要被多个应用程序共享，独立的系统有利于共享。

D：作为独立系统的一个好处是：当应用程序出错时，不容易牵连数据管理系统，造成数据损坏。

**第*2*题：关于为什么要强调软件系统中模块与模块之间的隔离性，哪种说法不对？**

A：便于各个模块的独立开发

B：当一个模块的实现需要改变时，不至于对其他模块造成较大影响

C：减少模块之间的交互，提升性能√

减少模块间的交互不一定能提升性能，主要是为了降低模块耦合度，便于模块内功能开发和模块间功能调度等

D：有利于程序的可读性

**第*3*题：关于如何让数据管理系统具备较高的隔离性，哪种说法不对？**

A：让数据管理系统的职责明确

B：让数据访问的方式尽可能简单

C：尽可能将所有的数据都交给数据管理系统管理 ？

D：让数据管理系统运行在独立的机器上√

隔离性要求构建软件的模块间关系简单，主要体现在模块间的接口逻辑上，和是否运行在独立的机器上关系不大

作业：

/\* 思考题：请为以下程序做模块设计

\*

\* 输入：若干文件，每个文件是一篇文章，由若干英文单词组成。

\* 例如：file1.txt, file2.txt, file5.txt，....

\*

\* 输出：一个单词文件列表，记录了出现在文件中的每一个单词以及包含这个单词的所有文件的文件名。

\* 例如： apple 出现于 file1.txt, file5.txt, ....

\* fish 出现于 file2.txt, file5.txt, ....

\* computer 出现于 file1.txt, file8.txt, ...

\* ... ...

\*/

#ifndef READTEXTS\_H

#define READTEXTS\_H

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <set>

#include <map>

using namespace std;

enum ERROR\_CODE {success, overflow, resource\_access\_failed, unkown\_error};

class readTexts

{

private:

vector<pair<string, string>> files; // 用于存储文件名和文件内容

map<string, set<int>> table; // 用于存储单词和其出现的文件之间的关系

public:

readTexts();

ERROR\_CODE inputFile(const string& fileName, const string& rootPath = "./"); // 读入文件

ERROR\_CODE PrintAllWord(); // 输出每一个单词以及包含这个单词的所有文件的文件名

private:

ERROR\_CODE saveFile(const string& file, const string& fileName); // 依据文件名存文件到files中

ERROR\_CODE loadFile(string& file, const string& fileName); // 依据文件名读文件

ERROR\_CODE deleteFile(const string& file); // 依据文件名删除文件

ERROR\_CODE parseWord(const string fileName); // 将某个文件内所有的词存储到table中

ERROR\_CODE findWord(vector<string>& output, string word); // 寻找某个单词出现的所有文件的名字

};

#endif

