## 池塘夜降彩色雨

任务：设计一个程序，演示美丽的“池塘夜雨”景色：色彩缤纷的雨点飘飘洒洒地从天而降， 滴滴入水有声，溅起圈圈微澜。

要求：(1) 雨点的空中出现位置、降范过程的可见程度、入水位置、颜色、最大水圈等，都是随机确定的 ； (2) 多个雨点按照各自的随机参数和存在状态，同时演示在屏幕上。(3) 增加“电闪雷鸣”景象。(4) 增加风的效果，展现“风雨飘摇”的情景。(5) 增加雨点密度的变化：时而“和风细雨”， 时而“暴风骤雨”。(6) 将“池塘”改为“荷塘”，雨点滴在荷叶上的效果是溅起四散的水珠，响声也不同。

测试数据：适当调整控制雨点密度、最大水圈和状态变化的时间间隔等参数。

实现提示：

(1) 每个雨点的存在周期可分为三个阶段：从天而降、入水有声和圈圈微澜，需要一

个记录存储其相关参数、当前状态和下一状态的更新时刻。

(2) 在图形状态编程。雨点下降的可见程度应是断断续续、依稀可见；圈圈水波应是由里至外逐渐扩大和消失。

(3) 每个雨点发生时，生成其记录，并预置下一个雨点的发生时间。

(4) 用一个适当的结构管理当前存在的雨点，使系统能利用它按时更新每个雨点的状态，一旦有雨点的水圈全部消失，就从结构中删去。

## 停车场管理

任务：设停车场内只有一个可停放n辆汽车的狭长通道，且只有一个大门可供汽车进出。汽车在停车场内按车辆到达时间的先后顺序，依次由北向南排列（大门在最南端，最先到达的第一辆车停放在车场的最北端），若车场内已停满n辆汽车，则后来的汽车只能在门外的便道上等候，一旦有车开走，则排在便道上的第一辆车即可开入；当停车场内某辆车要离开时，在它之后开入的车辆必须先退出车场为它让路，待该辆车开出大门外，其它车辆再按原次序进入车场，每辆停放在车场的车在它离开停车场时必须按它停留的时间长短交纳费用。（1）试为停车场编制按上述要求进行管理的模拟程序。（2） 两个栈共享空间，思考应开辟数组的空间是多少？（3） 汽车可有不同种类，则它们的占地面积不同，收费标准也不同，如1辆客车和1.5辆小汽车的占地面积相同，1辆十轮卡车占地面积相当于3辆小汽车的占地面积。（4） 汽车可以直接从便道上开走，此时排在它前面的汽车要先开走让路，然后再依次排到队尾。（5） 停放在便道上的汽车也收费，收费标准比停放在停车场的车低，请思考如何修改结构以满足这种要求。

测试数据：设n=2,输入数据为：（‘A’，1，5），（‘A’，2，10），（‘D’，1，15），（‘A’，3， 20）， （‘A’，4，25），（‘A’，5，30），（‘D’，2，35），（‘D’，4，40），（‘E’，0，0）。每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻，其中，‘A’表示到达；‘D’表示离去，‘E’表示输入结束。

要求：以栈模拟停车场，以队列模拟车场外的便道，按照从终端读入的输入数据序列进行模拟管理。每一组输入数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码及到达或离去的时刻，对每一组输入数据进行操作后的输出数据为：若是车辆到达，则输出汽车在停车场内或便道上的停车位置；若是车离去；则输出汽车在停车场内停留的时间和应交纳的费用（在便道上停留的时间不收费）。栈以顺序结构实现，队列以链表实现。

实现提示：需另设一个栈，临时停放为给要离去的汽车让路而从停车场退出来的汽车，也用顺序存储结构实现。输入数据按到达或离去的时刻有序。栈中每个元素表示一辆汽车，包含两个数据项：汽车的牌照号码和进入停车场的时刻。

## 魔王语言解释

任务：有一个魔王总是使用自己的一种非常精练而抽象的语言讲话，没有人能昕得懂，但他的语言是可以逐步解释成人能听懂的语言，因为他的语言是由以下两种形式的规则由人的语言逐步抽象上去的：

(1)



(2)



在这两种形式中，从左到右均表示解释。试写一个魔王语言的解释系统，把他的话解释成人能听得懂的话。

要求：用下述两条具体规则和上述规则形式(2)实现。设大写字母表示魔王语言的词汇；小写字母表示人的语言词汇；希腊字母表示可以用大写字母或小写字母代换的变量。魔王语言可含人的词汇。

(1)



(2)



测试数据：B(ehnxgz)B解释成tsaedsaeezegexenehetsaedsae

若将小写字母与汉字建立下表所示的对应关系，则魔王说的话是“天上一只鹅地上一只鹅鹅追鹅赶鹅下鹅蛋鹅恨鹅天上一只鹅地上一只鹅”。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | d | s | a | e | z | g | x | n | h |
| 天 | 地 | 上 | 一只 | 鹅 | 追 | 赶 | 下 | 蛋 | 恨 |

实现提示：将魔王的语言自右至左进栈，总是处理栈顶字符。若是开括号，则逐一出栈，将字母顺序入队列，直至闭括号出栈，并按规则要求逐一出队列再处理后入栈。其他情形较简单，请读者思考应如何处理。应首先实现栈和队列的基本操作。

## JAVA中常见的数据结构

任务：分析并实现JAVA中常见的数据结构：1．ArrayList 2．Vector3．LinkedList 4．HashMap5．HashTable 6 TreeMap 等。

要求：实现上述结构。

## 算术表达式求值演示

任务：表达式计算是实现程序设计语言的基本问题之一，也是栈的应用的一个典型例子。设计一个程序，演示用算符优先法对算术表达式求值的过程。

要求：以字符序列的形式从终端输入语法正确的、不含变量的整数表达式。利用教科书表3.1给出的算符优先关系，实现对算术四则混合运算表达式的求值，并仿照教科书的例3-1演示在求值中运算符械、运算数校、输入字符和主要操作的变化过程。

## 银行业务模拟

任务：客户业务分为两种。第一种是申请从银行得到一笔资金，即取款或借款。第二种是向银行投入一笔资金，即存款或还款。银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队。处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，则立刻排入第二个队等候，直至满足时才离开银行；否则业务处理完后立刻离开银行。每接待完一个第二种业务的客户，则顺序检查和处理(如果可能)第二个队列中的客户，对能满足的申请者予以满足，不能满足者重新排到第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查(因为此时已不可能还有能满足者)转而继续接待第一个队列的客户。任何时刻都只开一个窗口。假设检查不需要时间。营业时间结束时所有客户立即离开银行。

写一个上述银行业务的事件驱动模拟系统，通过模拟方法求出客户在银行内逗留的平均时间。

要求：利用动态存储结构实现模拟。

测试数据：一天营业开始时银行拥有的款额为10000(元)，营业时间为600(分钟)。其他模拟参量自定，注意测定两种极端的情况：一是两个到达事件之间的间隔时间很短，而客户的交易时间很长，另一个恰好相反，设置两个到达事件的间隔时间很长，而客户的交易时间很短。

实现提示：事件有两类：到达银行和离开银行。初始时银行现存资金总额为total。开始营业后的第一今事件是客户到达，营业时间从0到closetime。到达事件发生时随机地设置此客户的交易时间和距下一到达事件之间的时间间隔。每个客户要办理的款额也是随机确定的，用负值和正值分别表示第一类和第二类业务。变量total，closetime以及上述两个随机量的上下界均交互地从终端读入，作为模拟参数。

两个队列和一个事件表均要用动态存储结构实现。注意弄清应该在什么条件下设置离开事件，以及第二个队列用怎样的存储结构实现时可以获得较高的效率。注意：事件表是按时间顺序有序的。

## 学生信息管理系统

任务：设计一个学生信息管理系统，实现对学生基本信息的添加、删除、修改和查询等操作。

要求： 程序采用图形界面下进行交互的工作方式，完成如下功能：

多种方式建立学生信息；每个学生信息由学号、姓名、数学、英语和语文组成；可以通过手工录入每个学生信息，并在C盘下以StudentFile.txt保存；也可以导入某个路径下存放学生信息的文本文件。浏览所有学生信息。按照学号对所有学生信息进行升序、降序排列，并输出；选用冒泡、选择、快速排序等算法；不仅输出屏幕显示，还需要写入存放学生信息的文件。按姓名、学号等方式，实现对学生信息精确查询、模糊查询，并输出屏幕显示

精确查询结果演示

查询“姓名是刘梅”同学信息，则输出

学号 姓名 数学 英语 语文 ………..

2004112011 刘梅 88 90 78 ……....

模糊查询结果演示

查询“姓刘”的同学信息，则输出

学号 姓名 数学 英语 语文 ………..

2004112011 刘梅 88 90 78 ……....

2004112011 刘强 87 80 98 ……....

2004112011 刘星 86 70 58 ……....

能够实现连续多次查询

学生信息的插入、删除、修改。

通过插入、删除和修改后，保持所有学生信息的有序性；

插入、删除和修改后，对存放所有学生信息的文件及时更新。

数据的统计功能

统计每个学生的平均分和总分；

统计每个科目的平均分和最高分、最低分；

将上述统计结果，写入存放学生信息的文件。

## 电梯模拟

任务：设计一个电梯模拟系统。这是一个离散的模拟程序，因为电梯系统是乘客和电梯等“活动体“构成的集合，虽然他们彼此交互作用，但他们的行为是基本独立的。在离散的模拟中，以模拟时钟决定每个活动体的动作发生的时刻和顺序，系统在某个模拟瞬间处理有待完成的各种事情，然后把模拟时钟推进到某个动作预定要发生的下一个时刻。

要求：

(1) 模拟某校五层教学楼的电梯系统。该楼有一个自动电梯，能在每层停留。五个楼层由下至上依次称为地下层、第→层、第二层、第三层和第四层，其中第一层是大楼的迸出层，即是电梯的“本垒层“，电梯“空闲“时，将来到该层候命。 (2) 乘客可随机地进出于任何层。对每个人来说，他有一个能容忍的最长等待时间，一旦等候电梯时间过长，他将放弃。(3) 模拟时钟从0开始，时间单位为0.1秒。人和电梯的各种动作均要耗费一定的时间单位(简记为t)，比如：

有人进出时，电梯每隔40t测试一次，若无人进出，则关门；

关门和开门各需要20tg

每个人进出电梯均需要25h

如果电梯在某层静止时间超过300t，则驶回1层候命。

(4) 按时序显示系统状态的变化过程：发生的全部人和电梯的动作序列。

测试数据： 模拟时钟Time的初值为0，终值可在500~10000范围内逐步增加。

实现提示：

(1)楼层由下至上依次编号为0，1，2，3，4。每层有要求Up(上)和Down(下〉的两个按钮，对应10个变量CaliUp[0..4]和CallDOWEl[0..4]。电梯内5个目标层按钮对应变量Caucar[0..4]。有人按下某个按钮时，相应的变量就置为1，一旦要求满足后，电梯就把该变量清为0。

(2)电梯处于三种状态之-zGoingUp(上行)、GoingDown(下行)和Idle(停候)。如果电梯处于Idle状态且不在1层，则关门并驶回1层。在1层停候时，电梯是闭门候命。一旦收到往另一层的命令，就转入GoingUp或GoingDown状态，执行相应的操作。

(3)用变量Time表示模拟时钟，初值为0，时间单位。)为0。l秒。其他重要的变量有：

Floor ——电梯的当前位置(楼层)；

DI ——值为0，除非人们正在进入和离开电梯；

D2 ——值为0，如果电梯已经在某层停候30Ot以上；

D3 ——值为0，除非电梯门正开着又无人迸出电梯；

State ——电梯的当前状态(GoingUp，GoingDOWEl，Idle)。

系统初始时，Floor=1，Dl=D2=D3=0，State=Idle。

(4)每个人从进入系统到离开称为该人在系统中的存在周期。在此周期内，他有6种可能发生的动作：

M1. [进入系统，为下一人的出现作准备]产生以下数值：

InFloor ——该人进入哪层楼；

011tFloor ——他要去哪层楼；

GiveupTime ——他能容忍的等候时间；

Inter-Time ——下一人出现的时间间隔，据此系统预置下一人进入系统的时刻。

M2. [按电钮并等候]此时应对以下不同情况作不同的处理：

①Floor=InFloor且电梯的下一个活动是E6(电梯在本层，但正在关门〉；

②Floor=InFloor且D3手0(电梯在本层，正有人迸出)；

③其他情况，可能D2=0或电梯处于活动El(在1层停候)。

M3. [进入排队]在等候队列Queue[InFloor]末尾插入该人，并预置在GiveupTime个t之后，他若仍在队列中将实施动作M4。

M4. [放弃]如果Floor手InFloor或Dl=0，则从Quem[InFloor]和系统删除该人。如果Floor=InFloor且D1学0，他就继续等候(他知道马上就可进入电梯〉。

M5. [进入电梯]从Queue[InFloor〕删除该人，并把他插入到lElevator(电梯)校中。置Cancar[011tFloor]为1。

M6. [离去]从Elevator和系统删除该人。

(5)电梯的活动有9种：

E1. [在1层停候]若有人按下一个按钮，则调用Controler将电梯转入活动E3或E60。

E2. [要改变状态?]如果电梯处于GoingUp(或GoingDown〉状态，但该方向的楼层却无人等待，则要看反方向楼层是否有人等候，而决定置State为GoingDown(或GoingUp〉还是Idle。

E3. [开门]置DI和D2为非0值，预置300个t后启动活动E9和76个t后启动E5，然后预置20个t后转到目。

E4. [让人出入]如果Elevator不空且有人的011tFloor=Floor，则按进入的倒序每隔25个t让这类人立即转到他们的动作M6。Elevator中不再有要离开的人时，如果Queue[Floor]不空，则以25个t的速度让他们依次转到MLQueue[Floor]空时，置Dl为0，D3手0，而且等候某个其他活动的到来。

E5. [关门]每隔40个t检查D1，直到是D1=O(若D1手0，则仍有人出入〉。置D3为0并预置电梯再20个t后启动活动E6(再关门期间，若有人到来，则如M2所述，门再次打开)。

E6. [准备移动]置Caucar[Floor]为0，而且若State手GoingDown，则置CallUP

CFloor]为05若State手GoingUp，则置CallDownCFloor]为0。调用Controler函数。

如果State=Idle，则即使已经执行了Controler，也转到E1。否则，如果D2手0，则取消电梯活动E9。最后，如果State=GoingUp，则预置15个t后(电梯加速〉转到E7；如果State=GoingDown，则预置15个t后(电梯加速)转到E8。

E7. [上升一层]置Floor加1并等候51个人如果现在Caucar[Floor〕=1或CallUp[Floor]=1，或者如果((Floor=1或CallDOWEl[Floor]=1)且CallUp[j]=CallDOWEl[j]=CallCar-0]=0对于所有j>Floor)，则预置14个t后(减速)转到E2；否则重复E7。

E8. [下降一层]除了方向相反之外，与E7类似，但那里的51和14个t，此时分别改为61和23个t(电梯下降比上升慢)。

E9. [置不活动指示器]置D2为0并调用Controler函数(E9是由E3预置的，但几乎总是被E6取消了训。

(6)当电梯须对下一个方向作出判定时，便在若干临界时刻调用Controler函数。该

函数有以下要点：

C1. [需要判断?]若State手Idle，则返回。

C2. [应该开门?]如果电梯处于E1且CallUp[1]，CallDown[1]或Caucar[1]非0，则

预置20个t后启动E3，并返回。

C3. [有按钮按下?]找最小的j手Floor，使得CallUp[j]，CallDOWEl〔j]或Caucar[j]非0，并转到C4。但如果不存在这样的j，那么，如果Controler正为E6所调用，则置j为1，否则返回。

C4. [置State]如果Floor>j，则置State为GoingDOWEl；如果Floor<j，则置State为

GoingUp。

C5. [电梯静止?]如果电梯处于E1而且j手1，则预置20个t后启动E6。返回。

(7)由上可见，关键是按时序管理系统中所有乘客和电梯的动作设计合适的数据结构。

【选作内容】

(l) 增加电梯数量，模拟多梯系统。

(2) 某高校的一座30层住宅楼有三部自动电梯，每梯最多载客15人。大楼每层8户，每户平均3。5人，每天早晨平均每户有3人必须在7时之前离开大楼去上班或上学。模拟该电梯系统，并分析分别在一梯、二梯和三梯运行情况下，下楼高峰期间各层的住户应提前多少时间候梯下楼?研究多梯运行最佳策略。

## 迷宫问题

任务：以一个m×n的长方阵表示迷宫，0和1分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求出一条从入口到出口的通路，或得出没有通路的结论。

要求：编写一个求解迷宫的非递归程序。求得的通路以三元组(i，j，d)的形式输出，其中：(i，j)指示迷宫中的一个坐标，d表示走到下一坐标的方向。如：对于下列数据的迷宫，输出的一条通路为z(1，1，1)，(1，2，2)，(2，2，2)，(3，2，3)，(3，1，2)，··。(2) 编写递归形式的算法，求得迷宫中所有可能的通路；(3) 以方阵形式输出迷宫及其通路。

测试数据：迷宫的测试数据如下：左上角(1，1)为入口，右下角(8，9)为出口。

1 2 3 4 5 6 7 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

实现提示：

计算机解迷宫通常用的是“穷举求解“方法，即从人口出发，顺着某一个方向进行探索，若能走通，则继续往前进；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。

可以二维数组存储迷宫数据，通常设定入口点的下标为(1，1)，出口点的下标为(n，n)。为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。

## 文学研究助手

任务：文学研究人员需要统计某篇英文小说中某些形容词的出现次数和位置。试写一个实现这一目标的文字统计系统,称为"文学研究助手"。

要求：英文小说存于一个文本文件中。待统计的词汇集合要一次输入完毕,即统计工作必须在程序的一次运行之后就全部完成。（1）程序的输出结果是每个词的出现次数和出现位置所在行的行号,格式自行设计。(2)模式匹配要基于KMP算法。(3)整个统计过程中只对小说文字扫描一遍以提高效率。（4）假设小说中的每个单词或者从行首开始,或者前置一个空格符。利用单词匹配特点另写一个高效的统计程序,与KMP算法统计程序进行效率比较。(5)推广到更一般的模式集匹配问题,并设待查模式串可以跨行(提示:定义操作GetAChar)。

测试数据：以你的C源程序模拟英文小说,C语言的保留字集作为待统计的词汇集。

实现提示：约定小说中的词汇一律不跨行。这样,每读入一行,就统计每个词在这行中的出现次数。出现位置所在行的行号可以用链表存储。若某行中出现了不止一次,不必存多个相同的行号。

如果读者希望达到选做部分(1)和(2)所提出的要求,则首先应把KMP算法改写成如下的等价形式,再将它推广到多个模式的情形。

i=1;j=1;

while(i!=s.curlen+1&&j!=t.curlerl十1)

{

while(j!=0&&s.ch[i]!=t.ch[j])

j=next[j]; //j==O或s.ch[i]==t.ch[j]

j++;i++;//每次进入循环体,i只增加一次

}

【选作内容】

## 文本格式化

任务：输入文件中含有待格式化〈或称为待排版〉的文本,它由多行的文字组成,例如一篇英文文章。每一行由一系列被一个或多个空格符所隔开的字①组成,任何完整的字都没有被分割在两行(每行最后一个字与下一行的第一个字之间在逻辑上应该由空格分开),每行字符数不超过80。除了上述文本类字符之外,还存在着起控制作用的字符:符号"@"指示它后面的正文在格式化时应另起一段排放,即空一行,并在段首缩入8个字符位置。"@"自成一个字。

一个文本格式化程序可以处理上述输入文件,按照用户指定的版面规格重排版面z实现页内调整、分段、分页等文本处理功能,排版结果存入输出文本文件中。

试写一个这样的程序。

要求：(1)输出文件中字与字之间只留一个空格符,即实现多余空格符的压缩。

(2)在输出文件中,任何完整的字仍不能分割在两行,行尾不齐没关系,但行首要对齐(即左对齐)。

(3)如果所要求的每页页底所空行数不少于3,则将页号印在页底空行中第2行的中间位置上,否则不印。

(4)版面要求的参数要包含：

页长(PageLength)一一每页内文字(不计页号)的行数。

页宽(PageWedth)一一每行内文字所占最大字符数。

左空白(LeftMargin)-一一每行文字前的固定空格数。

头长(HeadingLength)一一每页页顶所空行数。

脚长(FootingLength)一一每页页底所空行数(含页号行)。

起始页号(StartingPageNumber)一一首页的页号。

(5)输入文件名和输出文件名要由用户指定。

(6)允许用户指定是否右对齐,即增加一个参量"右对齐否"(RightJustifying),缺省值可设为"y"(yes)。右对齐指每行最后一个字的字尾要对齐,多余的空格要均匀分布在本行中各字之间。

(7)实现字符填充(characterstuffing)技术。"@"作为分段控制符之后,限制了原文中不能有这样的字。现在去掉这一限制:如果原文中有这样的字,改用两个"@"并列起来 表示一个"@"字。当然,如果原文中此符号夹在字中,就不必特殊处理了。

(8)允许用户自动按多栏印出一页。

实现提示：

可以设:左空白数×2+页宽<=160,即行印机最大行宽,从而只要设置这样大的一个行缓冲区就足够了,每加工完一行,就输出一行。

如果输入文件和输出文件不是由程序规定死,而是可由用户指定,则有两种做法:一是像其他参量一样,将文件名交互地读入字符串变量中;更好的方式是让用户通过命令行指定,具体做法依机器的操作系统而定。

应该首先实现GetAWord(w)这一操作,把诸如行尾处理、文件尾处理、多余空格符压缩等一系列"低级"事务留给它处理,使系统的核心部分集中对付排版要求。

每个参数都可以实现缺省值②设置。上述排版参数的缺省值可以分别取56,60,10,5,5和1。

## 简单行编辑程序

任务：文本编辑程序是利用计算机进行文字加工的基本软件工具,实现对文本文件的插入、删除等修改操作。限制这些操作以行为单位进行的编辑程序称为行编辑程序。

被编辑的文本文件可能很大,全部读入编辑程序的数据空间(内存)的作法既不经济,也不总能实现。一种解决方法是逐段地编辑。任何时刻只把待编辑文件的一段放在内存,称为活区。试按照这种方法实现一个简单的行编辑程序。设文件每行不超过320个字符,很少超过80个字符。

要求：实现以下4条基本编辑命令:

(1) 行插入。格式:i<行号><回车><文本>.<回车>

将<文本>插入活区中第<行号>行之后。

(2) 行删除。格式:d<行号l>[<空格><行号2>]<回车>

删除活区中第<行号l>行(到第<行号2>行)。例如:"d10"和"d1014"。

(3) 活区切换。格式n<回车>

将活区写入输出文件,并从输入文件中读入下一段,作为新的活区。

(4) 活区显示。格式:p<回车>

逐页地(每页20行)显示活区内容,每显示一页之后请用户决定是否继续显示以后备页(如果存在)。印出的每一行要前置行号和一个空格符,行号固定占4位,增量为1。

各条命令中的行号均须在活区中各行行号范围之内,只有插入命令的行号可以等于活区第一行行号减1,表示插入当前屏幕中第一行之前,否则命令参数非法。

(5)对于命令格式非法等一切错误作严格检查和适当处理。

(6)加入更复杂的编辑操作,如对某行进行串替换；在活区内进行模式匹配等，格式可以为S<行号>@<串1>@<串2><回车>和m<串><回车>。

实现提示：

(1)设活区的大小用行数ActiveMULen(可设为100)来描述。考虑到文本文件行长通常为正态分布,且峰值在60到70之间,用320×ActiveMULen大小的字符数组实现存储将造成大量浪费。可以以标准行块为单位为各行分配存储,每个标准行块可含81个字符。这些行块可以组成一个数组,也可以利用动态链表连接起来。一行文字可能占多个行块。行尾可用一个特殊的ASCII字符(如(012)8)标识。此外,还应记住活区起始行号。行插入将引起随后各行行号的顺序下推。

(2)初始化函数包括:请用户提供输入文件名(空串表示无输入文件)和输出文件名,两者不能相同。然后尽可能多地从输入文件中读入各行,但不超过ActiveMULen-LX的值可以自定,例如20。

(3)在执行行插入命令的过程中,每接收到一行时都要检查活区大小是否已达ActiveMaxLen。如果是,则为了在插入这一行之后仍保持活区大小不超过ActiveMaxLen应将插入点之前的活区部分中第一行输出到输出文件中;若插入点为第一行之前,则只得将新插入的这一行输出。

(4)若输入文件尚未读完,活区切换命令可将原活区中最后几行留在活区顶部,以保持阅读连续性;否则,它意味着结束编辑或开始编辑另一个文件。

(5)可令前三条命令执行后自动调用活区显示。

## 串基本操作的演示

任务：如果语言没有把串作为一个预先定义好的基本类型对待,又需要用该语言写一个涉及串操作的软件系统时,用户必须自己实现串类型。试实现串类型,并写一个串的基本操作的演示系统。

要求：（1）在教科书4.2.2节用堆分配存储表示实现HString串类型的最小操作子集的基础上,实现串抽象数据类型的其余基本操作(不使用C语言本身提供的串函数)。参数合法性检查必须严格。

(2) 串头表改用单链表实现。

(3) 对命令的格式(即语法)作严格检查,使系统既能处理正确的命令,也能处理错误的命令。注意,语义检查(如某内部名对应的串已被删除而无定义等)和基本操作参数合法性检查仍应留给基本操作去做。

(4) 支持串名。将串名(可设不超过6个字符)存于串头表中。命令(1)(3)(5)要增加命令参数<结果串名>;命令(7)中的<串标识1> 改为<串名>,并用此名作为结果串名,删除原被替串标识,用<串名>代替<串标识>定义和命令解释中的内部名。每个命令执行完毕时立即自动删除无名串。

利用上述基本操作函数构造以下系统:它是一个命令解释程序,循环往复地处理用户键入的每一条命令,直至终止程序的命令为止。命令定义如下:

(1)赋值。 格式: A <串标识> <回车>

用<串标识>所表示的串的值建立新串,并显示新串的内部名和串值。例:A ‘Hi!’

(2)判相等。格式: E <串标识1> <串标识2> <回车>

若两串相等,则显示"EQUAL",否则显示"UNEQUAL"。

(3)联接。 格式:C <串标识1> <串标识2> <回车>

将两串拼接产生结果串,它的内部名和串值都显示出来。

(4)求长度。格式:L〈串标识> <回车>

显示串的长度。

(5)求子串。格式:S <串标识> +<数1>+<数2><回车>

如果参数合法,则显示子串的内部名和串值。<数>不带正负号。

(6)子串定位。格式:I <串标识1> <串标识2> <回车>

显示第二个串在第一个串中首次出现时的起始位置。

(7)串替换。格式: R <串标识1> <串标识2> <串标识3> <回车>

将第一个串中所有出现的第二个串用第三个串替换,显示结果串的内部名和串值,原串不变。

(8)显示。格式:P <回车>

显示所有在系统中被保持的串的内部名和串值的对照表。

(9)删除。格式:D <内部名> <回车>

删除该内部名对应的串,即赋值的逆操作。

(10)退出。格式:Q <回车>

结束程序的运行。

在上述命令中,如果一个自变量是串,则应首先建立它。基本操作函数的结果(即函数值)如果是一个串,则应在尚未分配的区域内新辟空间存放。

测试数据：自定。但要包括以下几组:

(1)E ‘’ ‘’<回车>,应显示"EQUAL"。

(2)E ‘abc’ ‘abcd’<回车>,应显示"UNEQUAL"。

(3)C ‘ ‘ ‘ ‘ <回车>,应显示"。

(4)I ‘a’ ‘’ <回车>,应报告:参数非法。

(5)R ‘aaa’ ‘aa’ ‘b’<回车>,应显示'ba’

(6)R ‘aaabc’ ‘a’ ‘aab’<回车>,应显示’aabaabaabbc’。

(7)R ‘Faaaaaaaa’ ‘aaaa’ ‘ab’,<回车>,应显示’abab’。

## 程序分析

任务：读入一个C程序,统计程序中代码、注释和空行的行数以及函数的个数和平均行数,并利用统计信息分析评价该程序的风格。

要求：

(1) 把 C 程序文件按字符顺序读入源程序;

(2) 边读入程序,边识别统计代码行、注释行和空行,同时还要识别函数的开始和结束,以便统计其个数和平均行数。

(3) 程序的风格评价分为代码、注释和空行三个方面。每个方面分为 A,B,C 和 D 四个等级 , 等级的划分标准是 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A级 | B级 | C级 | D级 |
| 代码(函数平均长度) | 10~15行 | 8~9或16~20行 | 5~7或21~24行 | <5或>24行 |
| 注释(占总行数比率) | 15~25% | 10~14或26~30% | 5~9或31~35% | <5%或>35% |
| 空行(占总行数比率) | 15~25% | 10~14或26~30% | 5~9或31~35% | <5%或>35% |

(4) 报告函数的平均长度。

(5) 找出最长函数及其在程序中的位置。

(6) 允许函数的嵌套定义,报告最大的函数嵌套深度。

测试数据：

先对较小的程序进行分析。当你的程序能正确运行时,对你的程序本身进行分析。

实现提示：

为了实现的方便,可作以下约定:

(1) 头两个字符是 FFF 的行称为注释行(该行不含语句)。除了空行和注释行外,其余均为代码行(包括类型定义、变量定义和函数头)。

(2) 每个函数代码行数(除去空行和注释行)称为该函数的长度。

(3) 每行最多只有一个"{" 、"}"、"switch" 和"struet"(便于识别函数的结束行)。

## C#数据结构原理解析实现

任务：分析C#的ArrayList，LinkedList，Queue，Stack，Dictionary的实现原理，并模拟实现上述数据结构。

要求：实现类型定义与基本操作。

## 多维数组

任务：设计并模拟实现整型多维数组类型。

要求：尽管 C 等程序设计语言已经提供了多维数组 , 但在某些情况下 , 定义用户所需的多维数组很有用的。通过设计并模拟实现多维数组类型 , 可以深刻理解和掌握多维数组。整型多维数组应具有以下基本功能 :

(1) 定义整型多维数组类型 , 各维的下标是任意整数开始的连续整数 ;

(2) 下标变量赋值 , 执行下标范围检查 ;

(3 〉同类型数组赋值 ;

(4) 子数组赋值 , 例如 ,a[1..n]=a [2..n+1];a[2..4][3..5]=b[1..3][2..4];

(5) 确定数组的大小。

(6) 各维的下标是任意字符开始的连续字符。

(7) 数组初始化。

(8) 可修改数组的下标范围。

实现提示：

各基本功能可以分别用函数模拟实现 , 应仔细考虑函数参数的形式和设置。 定义整型多维数组类型时 , 其类型信息可以存储在如下定义的类型的记录中：

＃define MaxDim 5

Typedef struct

int dim ,

BoundPtr lower ,

Upper;

ConstPtr constants;

)NArray,\*NarrayPtr;

整型多维数组变量的存储结构类型可定义为:

Typedef struct{

ElemType \*elem;

Int num;

NArrayType TypeRecord;

}NArrayType;

## 识别广义表的头或尾

任务：写一个程序,建立广义表的存储结构,演示在此存储结构上实现的广义表求头/求尾操作序列的结果。

要求：(1) 设一个广义表允许分多行输入,其中可以任意地输入空格符,原子是不限长的仅由字母或数字组成的串。

(2) 广义表采用如教科书中图5.8所示结点的存储结构,试按表头和表尾的分解方法编写建立广义表存储结构的算法。

(3) 对已建立存储结构的广义表施行操作,操作序列为一个仅由"t"或"h"组成的串,它可以是空串(此时印出整个广义表),自左至右施行各操作,再以符号形式显示结果。

(4) 将hsub和tsub这两个操作合为一个(用变量参数h和t分别返回各自的结果),以便提高执行效率。

(5) 设原子为单个字母。广义表的建立算法改用边读入边建立的自底向上识别策略实现,广义表符号串不整体地缓冲。

实现提示：

(1) 广义表串可以利用 C 语言中的串类型或者利用实习三中已实现的串类型表示。

(2) 输入广义表时靠括号匹配判断结束,滤掉空格符之后,存于一个串变量中。

(3) 为了实现指定的算法，应在上述广义表串结构上定义以下4个操作:

Test(s):当 S 分别为空串、原子串和其他形式串时，分别返回‘N’，‘E’和‘O’(代表Null,Element和Other)。

hsub(s,h):s 表示一个由逗号隔开的广义表和原子的混合序列,h为变量参数,返回时为表示序列第一项的字符串。如果s为空串,则h也赋为空串。

tsub(s,t):s的定义同hsub操作；t为变量参数,返回时取从S中除去第一项(及其之后的逗号,如果存在的话)之后的子串。

strip(s,r)：s 的定义同 hsub 操作；r为变量参数。如果串S以"("开头和以")"结束,则返回时取除去这对括号后的子串,否则取空串。

(4) 在广义表的输出形式中,可以适当添加空格符,使得结果更美观。

## 简单LISP算术表达式计算器

任务：设计一个简单的 LISP 算术表达式计算器。

简单 LISP 算术表达式(以下简称表达式)定义如下:

(1) 一个 0..9 的整数;或者

(2)(运算符 表达式 表达式)

例如,6,(+45),(+(+25)8) 都是表达式,其值分别为6,9和15。

要求：（1）实现LISP加法表达式的求值。

(2) 标准整数类型的 LISP 加法表达式的求值。

(3) 标准整数类型的 LISP 四则运算表达式的求值。

(4) LISP 算术表达式的语法检查。

测试数据：

6,(+45),(+(+25)8),(+2(+58)),(+(+(+12)(+34))(+(+56)(+78)))

实现提示：

写一个递归函数:

int Evaluate(FILE \* CharFile)

字符文件 CharFile 的每行是一个如上定义的表达式。每读入CharFile 的一行,求出并返回表达式的值。

可以设计以下辅助函数

status isNumber(char ReadInChar); //视ReadInchar 是否是数字而返回 TRUE 或 FALSE 。

int TurnToInteger(char IntChar); // 将字符’0’..’9’ 转换为整数 0..9

## 重言式判别

任务：一个逻辑表达式如果对于其变元的任一种取值都为真，则称为重言式；反之，如果对于其变元的任一种取值都为假，则称为矛盾式；然而，更多的情况下，既非重言式，也非矛盾式。试写一个程序，通过真值表判别一个逻辑表达式属于上述哪一类。

要求：(1) 逻辑表达式从终端输入，长度不超过一行。逻辑运算符包括 "|"，"&" 和 "~"，分别表示或、与和非，运算优先程度递增，但可由括号改变，即括号内的运算优先。逻辑变元 为大写字母。表达式中任何地方都可以含有多个空格符。

(2) 若是重言式或矛盾式，可以只显示"True forever"，或"False forever"，否则显示 "Satisfactible" 以及变量名序列，与用户交互。若用户对表达式中变元取定一组值，程序就求出并显示逻辑表达式的值。

测试数据：

(1) (A|~A)&(B|~B)

(2) (A&~A)&C

(3) A|B|C|D|E|~A

(4) A&B&C&~B

(5) (A|B)&(A|~B)

(6) A&~B|~A&B;O ,0;0,1;1,0;1,1 。

## 哈夫曼编/译码器

任务：利用哈夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成 本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端将传来的数据进行译码(复原)。对于双工信道(即可以双向传输信息的信道)，每端都需要一个完整的编 /译码系统。试为这样的信息收发站写一个哈夫曼码的编/译码系统。

要求：

a.一个完整的系统应具有以下功能：

(1)I：初始化(Initialization)。从终端读入字符集大小n , 以及n个字符和n个权值，建立哈夫曼树，并将它存于文件hfmTree中。

(2)E：编码(Encoding)。利用已建好的哈夫曼树(如不在内存，则从文件hfmTree中读人)，对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。

(3)D: 译码(Decoding)。利用已建好的哈夫曼树将文件 CodeFile 中的代码进行译码，结果存入文件TextFile中。

(4)P：打印代码文件(Print)。将文件CodeFile以紧凑格式显示在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件 CodePrin 中。

(5)T：打印哈夫曼树(Tree printing)。将已在内存中的哈夫曼树以直观的方式(树或凹入表形式)显示在终端上，同时将此字符形式的哈夫曼树写入文件TreePrint中。

b.上述文件CodeFile中的每个"0"或"1"实际上占用了一个字节的空间，只起到示意或模拟的作用。为最大限度地利用编码存储能力，试改写你的系统，将编码结果以二进制形式存放在文件CodeFile中。

c.修改你的系统，实现对你的系统的原程序的编码和译码(主要是将行尾符编/译码问题)。

d.实现各个转换操作的源/目文件，均由用户在选择此操作时指定。

测试数据:

(1)利用教科书例 6-2 中的数据调试程序。

(2)用下表给出的字符集和频度的实际统计数据建立哈夫曼树 , 并实现以下报文的编码和译码："THIS PROGRAM IS MY FAVORITE"。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字符 |  | A | B | C | D | E | F | G | H | I | J | K | L | M |
| 频度 | 186 | 64 | 13 | 22 | 32 | 103 | 21 | 15 | 47 | 57 | 1 | 5 | 32 | 20 |
| 字符 | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |  |
| 频度 | 57 | 63 | 15 | 1 | 48 | 51 | 80 | 23 | 8 | 18 | 1 | 16 | 1 |  |

## 图遍历的演示

任务：很多涉及图上操作的算法都是以图的遍历操作为基础的。试写一个程序（1）演示在连通的无向图上访问全部结点的操作。(2)借助于栈类型(自己定义和实现)，用非递归算法实现深度优先遍历。(3)以邻接表为存储结构，建立深度优先生成树和广度优先生成树，再按凹入表或树形打印生成树。

要求：以邻接多重表为存储结构，实现连通无向图的深度优先和广度优先遍历。以用户指定的结点为起点，分别输出每种遍历下的结点访问序列和相应生成树的边集。

测试数据：

教科书图7.33。起点为北京。

实现提示：设图的结点不超过30个，每个结点用一个编号表示(如果一个图有n个结点，则它们的编号分别为1，2，…，n)。通过输入图的全部边(存于数据文件中，从文件读写)输入一个图，每个边为一个数对，可以对边的输入顺序作出某种限制。注意，生成树的边是有向边，端点顺序不能颠倒。

## 教学计划编制问题

任务：大学的每个专业都要制定教学计划。假设任何专业都有固定的学习年限，每学年含两学期，每学期的时间长度和学分上限值均相等。每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的，可以有任意多门，也可以没有。每门课恰好占一个学期。试在这样的前提下设计一个教学计划编制程序。

要求：(1)输入参数包括：学期总数，一学期的学分上限，每门课的课程号(固定占3位的字母数字串)、学分和直接先修课的课程号。

(2)允许用户指定下列两种编排策略之一：一是使学生在各学期中的学习负担尽量均匀；二是使课程尽可能地集中在前几个学期中。

(3)若根据给定的条件问题无解，则报告适当的信息，否则将教学计划输出到用户指定的文件中。计划的表格格式自行设计。

测试数据：学期总数：65；学分上限：103；该专业共开设12门课，课程号从CO1到C12，学分顺序为2，3，4，3，2，3，4，4，7，5，2，3。先修关系见教科书图7.26。

## 校园导游咨询

任务：设计一个校园导游程序，为来访的客人提供各种信息查询服务。

要求：

(1) 设计你所在学校的校园平面图,所含景点不少于10个。以图中顶点表示校内各景点,存放景点名称、代号、简介等信息;以边表示路径,存放路径长度等相关信息。

(2)为来访客人提供图中任意景点相关信息的查询。

(3)为来访客人提供图中任意景点的问路查询,即查询任意两个景点之间的一条最短的简单路径。

(4) 提供图中任意景点问路查询 , 即求任意两个景点之间的所有路径。

(5) 提供校园图中多个景点的最佳访问路线查询 , 即求途经这多个景点的最佳 ( 短 )路径。

(6) 扩充道路信息 , 如道路类别 ( 车道、人行道等 ) 、沿途景色等级 , 以至可按客人所需分别查询人行路径或车行路径或观景路径等。

(7) 扩充每个景点的邻接景点的方向等信息 , 使得路径查询结果能提供详尽的导向信息。

实现提示：一般情况下,校园的道路是双向通行的,可设校园平面图是一个无向网。顶点和边均 含有相关信息。

## 最小生成树问题

任务：若要在n个城市之间建设通信网络,只需要架设n-1条线路即可。如何以最低的经济代价建设这个通信网,是一个网的最小生成树问题。

要求：(1) 利用克鲁斯卡尔算法求网的最小生成树。(2) 实现教科书 6.5 节中定义的抽象数据类型 MFSet 。以此表示构造生成树过程中的连通分量。(3)以图形和文本两种形式输出生成树中各条边以及他们的权值。（4）利用堆排序 ( 参见教科书 10.4.3 节 ) 实现选择权值最小的边。

实现提示：

通信线路一旦建立,必然是双向的。因此,构造最小生成树的网一定是无向网。设图 的顶点数不超过 30 个 , 并为简单起见、网中边的权值设成小于 100 的整数 , 可利用 C 语言 提供的随机数函数产生。

图的存储结构的选取应和所作操作相适应。为了便于选择权值最小的边 , 此题的存储 结构既不选用邻接矩阵的数组表示法 , 也不选用邻接表 , 而是以存储边 ( 带权 ) 的数组表示 图。

## 表达式类型的实现

任务：一个表达式和一棵二叉树之间,存在着自然的对应关系。写一个程序,实现基于二叉树表示的算术表达式Expression的操作。

要求：假设算术表达式 Expression 内可以含有变量 (a~z) 、常量 (0~9) 和二元运算符 (+,- ,\*, /,^(乘幂))。实现以下操作：

(1)ReadExpk(E) 一一以字符序列的形式输入语法正确的前缀表示式并构造表达式E。(2)WriteExpk(E) 一一用带括弧的中缀表示式输出表达式 E 。

(3)Assign(V,c) 一一实现对变量 V 的赋值 (V=c), 变量的初值为 0 。

(4)Value(E) 一一对算术表达式E求值。

(5)CompoundExpr(P,E1,E2) 一一构造一个新的复合表达式(E1)P(E２)．

测试数据：

(1) 分别输入 0;　a;-91;+a\*bc;+\*5^x2 \*8x;+++ \*3^ X3\*2^ x2x6并输出。

(2) 每当输入一个表达式后,对其中的变量赋值,然后对表达式求值。

实现提示：

(1) 在读入表达式的字符序列的同时,完成运算符和运算数(整数)的识别处理以及相应的运算。

(2) 在识别出运算数的同时,要将其字符形式转换成整数形式。

(3) 用后根遍历的次序对表达式求值。

(4) 用中缀表示输出表达式E时 ,适当添加括号,以正确反映运算的优先次序。

## redis数据结构（内存模型）

任务：redis常用的数据类型主要有五种：String、List、Hash、Set和Sorted Set。分析上述数据结构，查阅相关文献，例如<http://www.cnblogs.com/hjwublog/p/5639990.html>。

要求：实现上述结构和基本操作。

## 192 银行家算法

任务：假设一家银行拥有资金2000万，现有10家公司向其贷款进行筹建，每家均需300万才能建成。如果这家银行将2000万的资金平均贷给这10家公司，则每家公司将得到200万的贷款，都不能筹建成功，也就不能还贷，那么这10家公司都将“死锁”。若这家银行给其中的4家各贷300万，另4家各贷 200万，这样将还有2家公司得不到贷款，不能开工建设，但有4家可筹建完成，这4家公司运营所得利润可向该银行还贷，银行可以利用还贷的资金继续向其他 的公司贷款，从而保证所有公司筹建成功投入运营。

银行家算法是为了把一定数量的资金供多个用户周转，并保证资金的安全。银行家算法可归纳为：

（1）当一个用户对资金的最大需求量不超过银行家现有的资金时，就可接纳该用户。

（2）用户可以分期贷款，但贷款总数不能超过最大需求量。

（3）当银行家现有的资金不能满足用户的尚需贷款数时，可以推迟支付，但总能使用户在有限的时间里得到贷款。

（4）当用户得到所需的全部资金后，一定能在有限时间里归还所有的资金。

要求：实现对上述算法的模拟。

## 193 文件索引

任务:已知职工文件中包括职工号、职工姓名、职务和职称等若干数据项(见下表)。职务有校长、系主任、室主任和教员；校长领导所有系主任，系主任领导他所在系的所有室主任，室主任领导他所在室的全体教员；职称有教授、副教授和讲师3种。

要求：请给该文件建立索引,通过该索引文件,能做到:

(1)能够检索出全体职工间领导与被领导的情况；

(2)能够分别检索出全体教授、全体副教授、全体讲师。要求指针数量尽可能少，给出各指针项索引的名称及含义即可

职工号 职工姓名 职务 职称

1 张军 教员 讲师

2 沈灵 系主任 教授

3 叶明 校长 教授

4 张莲 室主任 副教授

5 叶宏 系主任 教授

6 周芳 教员 教授

7 刘光 系主任 教授

8 黄兵 教员 讲师

9 李民 室主任 教授

10 赵松 教员 副教授

## 194 城市公交线网优化调查分析

任务：建立公交线路调查统计数据结构，包括每站的上车或下车人数，起点终点等信息，具体要求不限，可根据实际情况设定。

要求：（1）统计出行要求。（2）优化公交线路。

## 195 AVLree的实现及分析

任务：AVL 树是平衡的二元查找树。一株平衡的二元查找树就是指对其每一个节点，其左子树和右子树的高度只差不超过1.

编写程序实现AVL树的判别；并实现AVL树的ADT，包括其上的基本操作；节点的加入和删除。BSt和AVL的差别就在平衡性上，所以AVL的操作关键要考虑如何在保持二元查找树定义条件下对二元树进行平衡化。

要求：（1） 编写AVL树的判别程序，并判别一个人元查找数是否为AVL树。二元查找树用其先序遍历结果表示，如：5,2,1,3,7,8.

（2） 实现AVL树的ADT，包括其上的基本操作：节点的加入和删除，另外包括将一般二元查找树转变为AVL树的操作。

## 196 Red-Black Tree 的分析和实现

任务：一棵具有红黑特征的树是一棵特殊的二元查找树，它在每个结点上增加一种额外属性——颜色（只能是红或黑）。

红黑树（Red-Black Tree）是一棵具有如下特征的二元查找树： 1． 每一个结点或者是红色或者是黑色。 2． 每个叶结点都是黑色。

3． 如果一个结点是红色，那么它的两个孩子结点都是黑色。

4． 从任何一个结点到所有以该结点为子树的叶结点的简单路径上拥有相同的黑色结点。

一般叶结点是不存放数据的，它作为哨兵用来表示已经到达叶结点。

要求：实现红黑树的基本操作，并演示其应用。

## 197 CD管理

任务：数据库文件用来存储歌曲，每首歌曲包含三个属性：歌曲作者、歌曲题目和歌曲创作的年份，即（Artist,Title,year）。数据库中的歌曲被存放在一个数据存储区中，且这些歌曲按照三个属性组织成三个有序链表，目的是提高诸如getByYearRange等算法的效率。

在数据库基础上实现Playlist接口。在CD数据库基础上，一些应用软件，如mp3播放器自己有一个playlist结构，playlist是一个按特定顺序播放的歌曲列表。将上部分的playlist扩展成一个sortable playlist，该sortable playlist支持以下排序。

要求：在上述基础上编写一个小型应用程序，该程序能够测试（演示）上述接口操作，如sortable playlist sortByYear是否完成了其预定的功能。同时该应用程序还应完成诸如存入磁盘、从磁盘载入等功能。

1、CD数据库能在内存中实现，同时也应该能够将该数据库存储在磁盘上。

2、同样的，playlist也应该能在内存中实现，并能够存储在磁盘上。

3、最好是用可视界面说明和验证这些操作的执行效果，即实现上述问题描述中的要求 。

## 公交线路上优化路径的查询

任务：最短路径问题是图论中的一个经典问题，其中的 Dijkstra 算法一直被认为是图论中的好算法，但有的时候需要适当的调整 Dijkstra 算法才能完成多种不同的优化路径的查询。

对于某城市的公交线路，乘坐公交的顾客希望在这样的线路上实现各种优化路径的查询。设该城市的公交线路的输入格式为：

线路编号：起始站名 ( 该站坐标 ) ；经过的站点 1 名 ( 该站坐标 ) ；经过的站点 2 名 ( 该站坐标 ) ；……；经过的站点 n 名 ( 该站坐标 ) ；终点站名 ( 该站坐标 ) 。该线路的乘坐价钱。该线路平均经过多少时间来一辆。车速。

例如： 63 ： A(32,45) ； B(76,45) ； C(76,90) ；……； N(100,100) 。 1 元。 5 分钟。 1/ 每分钟。

假定线路的乘坐价钱与乘坐站数无关，假定不考虑公交线路在路上的交通堵塞。

对这样的公交线路，需要在其上进行的优化路径查询包括：任何两个站点之间最便宜的路径；任何两个站点之间最省时间的路径等等。

要求：① 根据上述公交线路的输入格式，定义并建立合适的图模型。

② 针对上述公交线路，能查询获得任何两个站点之间最便宜的路径，即输入站名 S ， T 后，可以输出从 S 到 T 的最便宜的路径，输出格式为：线路 x ：站名 S ，…，站名 M1 ；换乘线路 x ：站名 M1 ，…，站名 M2 ；…；换乘线路 x ：站名 MK ，…，站名 T 。共花费 x 元。

③ 针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（不考虑在中间站等下一辆线路的等待时间），即输入站名 S ， T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x ：站名 S ，…，站名 M1 ；换乘线路 x ：站名 M1 ，…，站名 M2 ；…；换乘线路 x ：站名 MK ，…，站名 T 。共花费 x 时间。

④ 针对上述公交线路，能查询获得任何两个站点之间最省时间的路径（要考虑在中间站等下一辆线路的等待时间），即输入站名 S ， T 后，可以输出从 S 到 T 的考虑在中间站等下一辆线路的等待时间的最省时间的路径，输出格式为：线路 x ：站名 S ，…，站名 M1 ；换乘线路 x ：站名 M1 ，…，站名 M2 ；…；换乘线路 x ：站名 MK ，…，站名 T 。共花费 x 时间。

实现提示：需深入考虑，应根据不同的应用目标，即不同的优化查询来建立合适的图模型。

## A \*寻路算法

任务：A\*启发式搜索算法常用于游戏中寻路。参考相关文献，实现并演示寻路算法。

要求：应用A\*启发式搜索算法实现寻路操作。

## 对10亿人的年龄进行排序

任务：现有10亿人的数据，要求对数据进行排序。

要求：实现上述算法。

思路：年龄的范围可以认为是0~200间，这样可以用200个桶，采用基数排序。