实验一总结

By 肝到快穿越的助教

首先

首先我重申一遍：助教与各位同学是合作关系，而不是敌对关系。如果有什么问题，完全可以找我们argue，但没必要大发雷霆——我们充分理解你的处境。如果确实是我们的问题，我们会立即改正。如果不是我们的问题，就没有必要为了零点零几分的总评而大动干戈了。

实验分析

线性表有什么不能理解的吗？那肯定没有。这次实验的目的就是帮水平不够的小朋友们补一下基础。

——Cindy

在分析实验内容之前，希望各位能明白，本次实验主要锻炼的是各位的代码/工程能力。各位既然选了这门课，我们就应该尽力让各位达到这门课要求的代码能力：一个信院大二学生应该具备的代码能力。

个人认为，所谓的代码能力包括但不限于：

1. 分析问题，将问题转化为抽象的数据结构及其逻辑操作；
2. 统筹规划代码框架，使得功能的添加/删除变得容易；
3. 拥有良好的变量命名/缩进/换行习惯，让自己和别人看得舒服；
4. 调试程序，修改编写时遇到的 bug。

好，讲了这些，我们开始看实验的基本要求。

首先，实验题目是“链表多项式计算器”，所以按照规则，我们的数据结构应该是单链表。既然是多项式，那么自然地，我们应该用一个结构体来表示多项式的每一个项。我在检查代码的时候发现有人用了两个链表，一个用作系数链表，一个用作指数链表。这么做其实问题很大：首先，就逻辑而言，每一个系数与指数是绑定在一起的，两个链表与一个链表的逻辑含义就不一样；其次，假如你在某个操作中忘了改两个链表中的某一个链表，整个多项式就会发生混乱，而且你也很难查出原因。

然后我们来分析一个链表计算器需要哪些操作。创建删除加减乘除等各个“显式”操作当然都是必不可少的；但一些“隐式”操作实际更为重要，比如“多项式整理函数”。这个函数应该对链表进行排序、合并同类项，并删除系数为 0 的节点。可以想见，只要有了这个函数，我们在插入的时候就可以随便无序插入，最后整理一下就行了。

接下来分析功能复用问题。很明显，减法可以用“加一个负多项式”来实现，定积分可以用“不定积分求值”实现。乘方可以用“多次乘法”实现。取模和取余实际可以用一个函数完成。

最后是细节问题。既然要求了禁止内存泄露，且 m1，m2，m3 三个操作数可能会相同，那么合适的操作流程应该是：

1. 先申请一个空的多项式指针（我们设这个指针为 ptr）作为结果的存储位置；
2. 进行运算，结果放到 ptr 里；
3. 删除 m3；
4. 用 ptr 替换掉 m3。

如果用 C++，建议把构造函数、拷贝构造函数、析构函数、等号的运算符重载都写一遍。

当然，不同人的实现方法也不一样，具体实现方法就看各位自己了。

我们为啥不推荐大家用 Devcpp?

原因很简单：Devcpp 的报错、缩进简直就是一坨**。有返回值的函数不返回东西，不报错；访问未初始化的变量，不报错；访问空指针/野指针，不报错。相比之下，VS 连循环时下标越界都能给你画上波浪线，这种新手必备的东西为啥不用呢？其实高版本的 gcc 也会告诉你这些东西，但是对于新手不好配置，所以我们统一推荐大家用 VS。devcpp 一时爽，oj re 火葬场。

再提醒一点，无论编译时报了多少个 warnings，都要挨个看完。Warnings 里经常会把经典错误报出来，比如 if(i=0)。如果你不看，最后花更多时间 debug，就很得不偿失。

代码风格

请各位时隔半个月再来看一下你的代码。你还能记得那些函数里的 a, b, c, p, q, r 的含义吗？我相信，很多人看了自己之前写的代码都会觉得很不舒服，因为这已经成了天书。

所以我们建议：

1. 变量的命名要有意义，a, b, c, p, q, r 就好像电脑里的“新建文件夹”一样，没有任何意义。
2. i, j, k 只用作循环变量。为啥非得是 i？我认为原因在于 i 是“iterator（迭代器，循环变量）”的简写。j 和 k 属于“爱屋及乌”。
3. 在用 for 循环时，除非特殊要求，i 请从 0 开始而不是 1 开始。数组下标减一难道不累吗？
4. 注意缩进，代码的层次很重要，在这方面请做一个强迫症。如果你实在不会缩进，下个 vscode，打开代码文件，右键格式化文档。
5. 在一段逻辑的多个步骤之间加空行，以示分隔。
6. 写好注释。注意：注释未必是必须的。如果你的变量命名的很好，一下就能看出逻辑，不加注释也无妨。“好的代码不需要注释”。
7. 注意英文拼写。“数据”的英文是“data”而不是“date”。“coefficient”的缩写应当为“coef”而不是“ceof”。

公开处刑

接下来是喜闻乐见的（匿名）公开处刑环节。在下面我会贴上一些经典的让人看着不舒服的/错误的代码。

迷惑代码	迷惑点
`p[i]=malloc(1024);`	malloc 请进行强制类型转换。 malloc 里面请用 sizeof。1024 是哪来的？
`if (p=NULL)`	经典错例
`fprintf(fp, ".4lf", p->coef)`	我敢保证这位肯定没看过自己输出的是啥
`else{` `  while(p&&s->data>p->data){` `  q=q->next;p=p->next;` `  }` `  if(p) {` `  if(p->data==s->data){` `  p->c=p->c+s->c;` `  }` `  else{` `  s->next=q->next;` `  q->next=s;` `  }}` `  else{` `  s->next=q->next;` `  q->next=s;` `  }` `}` `for(q=L->next,p=L;q!=NULL;){` `if(q->c==0){` `  p->next=q->next;` `  free(q);q=p->next;` `}` `else{` `p=p->next;q=q->next;` `} }`	你能分清哪个 else 属于哪一层 if 吗？
`#define LEN sizeof(Node)//这是上面` `p=(Node*)malloc(sizeof(LEN));//这是下面`	怪不得你在 free 某个节点的时候影响了其他节点的值
`q->next = (linkitem)malloc(sizeof lnode);`	sizeof 是咋用的？
`fout=fopen("polyn.out","r\w");`	？？？
`fscanf("%d", &m);`	？？？
`fscanf(fin,'%f%d', &a1&k1);`	单引号？

`` while (true) { if (rear->coef==0 && rear->exp==0) { fprintf(fp, "0.0000"); } //下面省略 } ```	于是你就无限循环输出了 0.0000
`` if (a[n] && a[n]->next) { q = a[n]; s = a[n]; while (q->next) { //销毁原多项式 q = q->next; free(s); s = q; }//end while free(q); free(s); }//end if ```	s 与 q 永远相等，所以你每次 free 都会 free 两遍
`` TREE preorder(TREE s, int flag) { if (!s); if (s) { ```	这是实验三某人的代码，你在干嘛？
`` TREE init(char g) { int a, b; char c, ch; tree* p, * q; tree* z; int flag = 0; ```	这是实验三某人的代码，您不能换个行吗？看着不累吗？

我相信大家在看了以上的代码都会觉得：只要他开 debug 看了一下，甚至只要跑一遍程序，甚甚至只要编译了一遍，就会立刻发现这些低级错误。

总结

本次实验中大多数同学都能很好地完成任务。但也有一部分人由于时间紧张（但我们给了三周啊，还加上了国庆假期!）、对文件操作不熟等原因完成速度较慢，导致最后分数不理想。为提升各位的代码能力，鼓励大家完整地完成一个工程，本次实验允许补交，补交细则如下：

补交细则

资格认定：只有登记表上“是否提交”一栏为 TRUE，且没有给分/总分低于 80 的同学才能补交。

给分细则：补交的分数=min(补交后的基础分*0.9+附加分, 80)。不能补交需要单独检查的附加功能。

实验要求：与课程主页上发布的实验要求完全相同，没有修改。

测试用例：与已经发布的 67 个测试样例完全相同，没有修改。各位可以照着这个 debug。

截止日期：2019-11-19 23:59 (GMT+8:00)。

补交方式：请将.c/.cpp 文件命名为姓名-学号-Lab1.c/.cpp 的格式（如“张三-PB20000000-Lab1.cpp”）,发至 dsa2019@126.com。邮件标题为姓名-学号-Lab1-补交（如“张三-PB20000000-Lab1-补交”）。邮件标题不对的没有自动回复，且视为无效提交。

调试提示

您可以百度一下 freopen 函数，这个函数可以把文件输入/输出弄到 stdin 里，这样就可以用 printf/scanf 了。(这也是我在批改这次实验的时候学到的)

批改花絮（助教吐槽）

开始的设想是这样的：一个 polyn.in 里有所有的 67 个测试点，顺序地跑完这 67 个，最后给分。

结果架不住大家纷纷崩溃的程序啊……

程序跑了一半，最后崩溃，你说剩下的我该怎么给分？

于是我们把测试样例分解了，简而言之就是每一个测试点都有一个独立的 polyn.in，批改的时候程序会自动把需要的 in 文件复制到目录下，运行，再删除，换成下一个 in 文件。

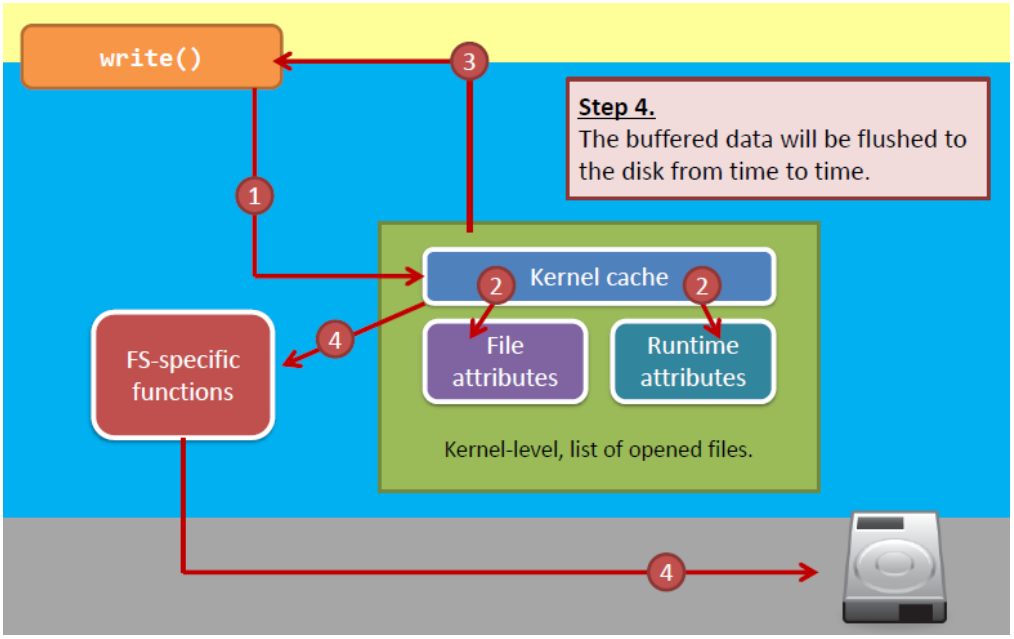
然而这也架不住一批输入输出有问题的人啊……负号前面的加号留着，你说评测程序该怎么办？

还有程序卡死的，最严重的一个程序，十个测点会卡死三个，你说这怎么办……

对于一些编译不过的、明显是输入输出有问题的，大部分测点全是 fail，能直接给 1 分吗？不能，这不人道，所以要尝试看一下 bug 在哪。如果只是小错，稍微改改，起码能给个分。抱着这种心态，被迫看了一坨坨的代码，最后也改不过来，只有时间在不停的流逝……

还有那些用 VS 的，只要在第一行#define _CRT_SECURE_NO_WARNINGS 关掉这个警告就行了，为啥非得用 scanf_s 呢？评测程序用的是 makefile+gcc/g++，就像 oj 一样不支持 scanf_s 那玩意。遇到了怎么办？手动改，真就是手动改……vscode 打开，ctrl+H，替换……fopen_s 也要换……

还有人问，为啥我用 testprog 里的测试样例一点问题没有，用了后来发的测试样例就啥也不能输出？大概率是程序半路崩溃了。借用一张操作系统 PPT：



为了保证性能，所有的写请求都会首先存在系统的一个 cache 里，攒够一波之后再写到磁盘上。如果程序半路崩溃了，cache 里的东西就有可能直接丢失而不是写到磁盘上，因此你的输出就是一片空白。所以我们把测试样例的 in 文件分成了 67 个来解决这个问题。

吐槽完毕，祝各位同学能够开心地度过本门课程。