|  |  |
| --- | --- |
| **专题二：基础数据结构-链表**  **实验报告** | |
| 报告小组：47 | 小组成员：郭书廷(3170104871),叶帆(3170102410) | 指导老师：张引，田沈晶，熊海辉 | |
| 项目：共享子串、多项式计算、通讯录管理 | |
| **Project1-共享子串** | |
| 问题描述 | 存储字符串，将相同的后缀用同一个条链表存储。 |
| 算法分析与代码设计 | **程序目的：**  1.如原题所述。  2.题目原意为输入两个字符串，找出他们的公共后缀，并用链表存储公共后缀。但仅存储两个字符串缺乏实际意义，我们注意到这个题目与一种数据结构——trie树非常相似，所以本程序的目的是用trie树实现一个字典。  **结构设计：**  1.使用链表，将题述结构分为三个部分：一个公共子串，两个非公共子串，将三者链接即可。  2.我们设计了一个前缀trie树的模板，树的节点设计如下：  struct node{ struct node\* s[95]; int leaf; char\* mean; };  其中s为子节点数组(assci字符中共有95个可打印字符)，leaf表示该节点是不是一个单词的结束，mean为以该节点结束的单词的释义。  **模板设计：**  struct node{ struct node\* s[95]; int leaf; char\* mean; };  typedef struct node Prefix\_Trie\_base;  typedef struct node\* Prefix\_Trie\_iterator;  struct Prefix\_Trie\_ { struct node\* root; };  用此类型定义一颗树:  typedef struct Prefix\_Trie\_\* Prefix\_Trie;  新建一棵树:  Prefix\_Trie New\_Prefix\_Trie();  向树t添加含义为mn的单词str:  void Insert(Prefix\_Trie t, const char\* str, const char\* mn);  在树t中查找单词str，返回释义(char\*):  char\* Find(Prefix\_Trie t, const char\* str);  将树t的节点释放至内存池：  void Free\_Prefix\_Trie(Prefix\_Trie t);  释放内存池：  void Free\_all(); |
| 使用方法 | 1.执行strstr.exe，输入两个不含空格的字符串即可。  2.将可执行文件与字典文件Dict.txt放在同一目录，输入单词即可。 |
| **Project2-多项式计算** | |
| 问题描述 | 计算多项式的和、差、积。 |
| 算法分析与代码设计 | **功能：**  程序采用命令式交互，执行代码，可以使用一下命令进行多项式计算(系数和求值支持高精度)。  定义变量var,变量名符合C语言标准 set <var> or s <var>  将变量var1的值拷贝到var2 copy <var1> <var2> or cp <var1> <var2>  手动修改变量var，输入一个多项式temp，使var=var+temp add\_to <var> or m <var>  输出多项式var print <var> or p <var>  输出多项式var的项数size <var>  清除所有变量，重置计算器 clear or ca  将两个变量相加/减/乘，并将答案放入预置变量ans add/minus/mul <var1> <var2>  计算var的值（输入x的值，答案可以保存到文件） calc <var> or cc <var>  求导 diff <var> 清屏 cls 帮助 help or h 退出 exit or q  **算法：**  1.使用第一题的**trie树**模板存储、查找变量名。  2.多项式存储使用**链表**。  3.使用**高精度**算法，高精度数的存储使用**不定长数组**vector(非C++模板)。  4.高精度乘法使用**快速傅里叶变换**。（多项式的乘法仍使用n^2方法）  5.求值乘方运算使用**倍增法**。  6.多项式按指数大小排序使用**随机化快排**。  **方法简述：**  对于两个多项式如果进行加法，则执行三个步骤：  1.将两个多项式拼接  2.对拼接后的多项式进行排序  3.合并同类项  如果进行减法，则将多项式各项的符号全部翻转，然后进行加法。  如果进行乘法，则将两个多项式的项两两相乘，将乘积放入新的链表，在对新链表排序，合并同类项。  **注：**  由于代码总长度较大(约1300+行)，关于vector、倍增法求乘方（快速幂）、链表与简易内存池的实现细节详见ppt、代码及注释。 |
| 使用说明 | 运行可执行文件，按程序提示及上述指令，输入计算即可。 |
| **Project3-通讯录管理** | |
| 问题描述 | 实现通讯录管理器，满足添加、删除、查找、导入、导出等功能。 |
| 算法分析与代码设计 | **代码设计：**  使用第一题的trie树模板，将单词释义换成字符串的vector即可。  Trie树节点和vector结构体定义如下：  struct node{ struct node\* s[256]; int leaf; vector vec; };  typedef char\* \_\_tp\_vector;  const \_\_tp\_vector \_\_INIT\_DATA\_VECTOR=0;  typedef \_\_tp\_vector vec\_base;  typedef \_\_tp\_vector\* vector\_iterator;  struct vector\_  {  /\*vec->end is a virtual element, you may not access it.\*/  vector\_iterator data;/\*实际存储空间\*/  vector\_iterator begin;/\*首指针\*/  vector\_iterator end;/\*尾指针\*/  int size,mx\_size;/\*当前元素个数和最大元素个数\*/  };  **UI设计：**  i <name> <info> 插入/新建信息  d <name> <num> 删除name的第num条信息，如果num为0则删除整个联系人  f <name> 查找联系人  e <file\_name> 导出至file\_name  p 输出所有联系人  q 退出 |
| 使用说明 | 运行可执行文件，输入要导入的联系人文件名，按上述命令操作即可，输入h获取帮助。  **注：**导入后的所有操作都在内存中进行，除非导出，否则程序不会自动保存。 |