**燕 京 理 工 学 院**

**课程设计报告**

课程名称 数据结构课程设计

设计题目 8.学生成绩管理系统的设计与实现（单链表实现）

专业、班级 计科2101

学 号 20210210020

姓 名 王世超

指导教师 孙丽云

设计时间 2021-2022学年第二学期

2022年 6 月 24 日

|  |  |
| --- | --- |
| **一、引言**  设计题目的目的：为用户提供学生信息简单管理的程序  设计题目的意义：巩固数据结构知识，熟练掌握C语言基本知识和技能，掌握数据结构与算法的设计方法，具备初步的独立分析和设计能力  主要任务：   1. 创建新的链表， 2. 在表头位置或表尾位置添加信息， 3. 按学号，或按姓名、得分1、得分2修改信息， 4. 按学号，或按姓名、得分1、得分2删除信息， 5. 按学号修改信息或选择修改单项信息， 6. 选择位置插入信息，输出全部信息， 7. 对一项信息排序， 8. 计算：平均值、统计得分高于90的个数、统计得分低于60的个数， 9. 更改语言、编码、科目， 10. 保存。 | |
| **二、正文**   1. **模块划分**   主要功能划分如左图所示，由Check选择需要执行的功能，主要的功能由创建，（头部）添加，查找，删除，修改，插入，列出，排序，以及其他功能（如尾部添加，计算，按其他项目查找，删除，修改等）组成。  程序中的主要全局变量：   1. char[]类型的可自定义科目; 2. char[]类型的学号和姓名； 3. float类型的几个得分 4. 含有②和③的学生结构体 5. 含有④和⑤指针的结点结构体   //对数据输入赋值,对非法输入赋值,返回节点的数据  stu Scan();  //创建一个空链表，返回链表头指针  node \*Creat();  //在头部添加元素  void HeadAdd(node \*head, stu data);  //在尾部添加元素  void TailAdd(node \*head, stu data);  //默认的添加方式，调用HeadAdd  void Add(node \*head);  //需要用户选择的添加方式，调用TailAdd  void AddH(node \*head);  //为Insert提供插入点和数据，调用Insert  int Preinsert(node \*head);  void Insert(node \*head, int i, stu e);  //列出节点p的信息  void Print(node \*p);  //列出head的所有节点，调用Print  void List(node \*head);  //以学号查找  node \*Find(node \*head);  //自定义查找  node \*FindPlus(node \*head);  //自定义删除  int DelPro(node \*head);  //以学号删除  void Delete(node \*head);  //以学号修改  int Alter(node \*head);  //以学号修改单个数据  int AlterPlus(node \*head);  //Check与Menu检查指令并执行  int Cheak(char str[]);  int Menu();  //询问保存并退出  void Quit();  //判断排序依据  int SortCHECK(node \*head);  //自定义排序  int SortAdaption(node \*head, int subj);  //计算：平均值，合格，不及格  float Average(node \*head);  int More(node \*head);  int Less(node \*head);   1. **功能及其关键代码**   创建表    // 分配空间      node \*head = (node \*)malloc(sizeof(node));      // 令头节点指向空      head->next = NULL;    在头部添加数据  node \*new = Creatnode(data);      new->next = head->next;      head->next = new;  输入    scanf("%s", str);  // 对num类的输入赋值  strcpy(data.num, str);  // 对score类的输入赋值      data.score = (float)atof(str);    查找（以学号）  scanf("%s", searchdata.num);  // p从头节点开始遍历      node \*p = head->next;      // 当p不为空时循环      while (p)      {      if (!strcmp(p->data.num, searchdata.num))         {  // 找到后在此处输出当前节点          }        p = p->next;      }    输出（单个节点）  // 输出当前节点p的信息      printf(" %s\t%s\t%.2f\t%.2f\n", p->data.num, p->data.name, p->data.score, p->data.score2);  输出（全部内容）  node \*p = head->next;  while (p != NULL)      {   // 此处输出当前节点          p = p->next;      }    删除（通过学号）  //判断是否为空表      if (!head->next)          return;  // 输入删除项      scanf("%s", sr);      // 待删除从头节点开始遍历，delete为被删除的节点，front为它的前一个节点      node \*delete = head->next;      node \*front = head;      // 查找结点,删除节点不等于目标节点向后遍历,等于退出循环      while (strcmp(delete->data.num, sr))  {  // front与delete都向后一位          front = delete;          delete = delete->next;  // 判断是否未找到，直到最后一个节点数据都与输入不等          if (delete == NULL)              return;      }  // 删除front的后一节点指向删除点的后一阶段，删除点被闲置出来      front->next = delete->next;      free(delete);  删除（通过学号）  // 判断是否为空表      if (!head->next)          return;  // 输入修改项  scanf("%s", sr);  // 待修改从头节点开始遍历      node \*modify = head->next;  // 查找结点,修改节点不等于目标节点向后遍历,等于退出循环      while (strcmp(modify->data.num, sr))      {          // 查找要修改的节点          modify = modify->next;  // 若没有找到，modify会后移到表的末尾指向的空          if (modify == NULL)              return 1;  }  // 对找到的修改节点赋值  modify->data = Scan();    插入(在head插入节点，i为插入位置，e为插入数据，i与e在之前赋值后调用此模块)    // 若插入位置为第一个节点      if (i == 1)      {          // 创建新节点          node \*new = Creatnode(e);          // 插入节点          new->next = head->next;          head->next = new;      }      else      {          // 设置指针p指向head的第i-1个节点          node \*p = head->next;          for (int j = 1; j < i - 1; j++)          {              p = p->next;          }          // 创建新节点          node \*new = Creatnode(e);          // 插入节点          new->next = p->next;          p->next = new;      }    排序  // 创建一个新表      node \*newhead = (node \*)malloc(sizeof(node));      newhead->next = NULL;      // 遍历令新表的每一个score都等于原表的每一个score      node \*p = head->next;      // 若原链表不为空，创建节点，使p自增，对节点数值判断      while (p)      {          // 创建一个节点等于原表的data，后继将会指向新表一个位置的后继          node \*newnode = (node \*)malloc(sizeof(node));          // 新节点的data与被排序表的data相等          newnode->data = p->data;          // 新表头节点为q,令新节点插入          node \*q = newhead;          // 若新表长大于1，判断大小，q的score小于newnode的score,q后移        while (q->next && q->next->data.score < newnode->data.score)          // 新表自增，到达合适位置，此前的数据比此后的数据小              q = q->next;  // 先令newnode指向新表q位置的后继，将newnode插入到新表的q位置          newnode->next = q->next;          q->next = newnode;          // 原表自增          p = p->next;  }     1. **遇到的问题以及解决方法**   参考到了git和dotnet的命令操作，我为很多功能都定义了"/某一命令"和!strcmp(strncmp)，用于执行其功能的调用。  在程序编写之处，所有的输入都是直接赋值，一旦输入过长或类型不一致就会发生错误，对此我添加了一个长数组str[128]接受输入的字符串，并对它验证合法性合法再赋给原数值，合法规则，例如：长度是否超过10，得分是否包含其他符号，小数点个数是否大于1，取值范围是否正确，否则将赋值为预定的0或Unknow，对于字符串对浮点的转换则使用了atof()。  起初采用的编码为UTF-8，但当我使用UTF-8进行输入和输出时发现，字符中的中文输出时会替换为"",我则是直接设置了在程序运行开始就设置活动代码页为GBK，并允许用户自己更改编码。  在编写插入时，由于链表不同数组不再可用数组下标操作，我添加了一个计数i，再次对i自增搜索需要插入的位置。但出现了第二个问题，如果用户确定一个信息需要插入到i位置，再尾部插入时位置将会变化，我为Menu添加了一个标志c，当插入被执行时将会被标记，再次进行尾部插入将会提示可能造成数据错位。  在编写排序时，考虑到用户可能不需要排序后的列表替换为原来的列表，于是新增了newhead的新表和newnode的转接点，令newnode与原表比较，确定newhead的插入位置，newnode指向插入位置后面的全部数据，再插入到newhead。为优化用户对升序或是降序的选择，添加了一个标志f，令其可以帮助用户切换排序方向。 | |
| **三、结论**  我为此程序增加了一些或许更贴合用户体验的功能，如用户只要记住各个功能的代码就可以快速执行功能。用户可通过只按下一个按键就可以快速切换功能的执行方式，暂时只支持自定义排序的按C切换排序方向。另外在用户启动时提示帮助选单，退出时也会提醒保存，当进行过插入项目后在再尾部添加时也会做出会提示。一些模块的输入操作在输入不合法时会自动赋值。  此程序存在一些问题：存在需要用户输入的模块当用户输入了非法字符可能会导致程序无法正常运行，这些模块您可在/Beta中查看。  目前只做出了能够自定义的查找，删除，排序操作，还有模块的功能需要将功能拓展。  另外，还可以增加一些新的模块，如读取、管理保存的文件，当增加科目和得分或其他信息时可以自适应地增加输出。 | |
| **四、参考文献**  [1] 严蔚敏．数据结构（C语言版第2版）[M]．北京：人们邮电出版社，2015. | |
| **五、指导教师评语**  **签名：**    **年 月 日** | |
| **课程设计成绩（五级分制）** |  |