综合实验 系统设计与应用

**班别：**2021级计算机类1班  **学号： 2021463030114**  **姓名： 侯文辉**

1. **实验目的**
2. 通过该综合性项目学习，使学生完成一个具有一定难度的管理系统的编写、调试、运行工作。
3. 进一步掌握结构化程序设计的基本方法和编程技巧。
4. 巩固所学理论知识，使理论与实际相结合，从而提高自我综合分析问题、解决问题的能力。
5. **实验内容**

|  |  |
| --- | --- |
| 第1题 （文件名：zhonghe.c） | |
| 1、题目要求 | 使用本学期所学习的结构化程序设计思想，设计并实现一个“学生信息管理系统”，系统功能与技术要求如下：  （1）系统基本功能：增加、修改、删除、浏览、查找、排序、统计等。  （2）系统必须使用链表等数据结构表示系统中管理的数据对象。  （3）数据约束功能：对输入数据要有一定的有效性检测，不合法的输入数据应有出错提示且要求重新输入。  （4）界面要求：至少分两层菜单。提供良好的人机界面，方便用户进行相关操作。  （5）如有余力，请把数据的操作的结果以文件形式保存起来。 |
| 2、源代码 | （此处分功能逐一贴出每个功能函数的代码）   1. 增加   struct student \*addlinker(int i)  {        struct student \*head1 = NULL;      for (int stuNum; stuNum < i; stuNum++)      {          int flag = 1;          struct student \*prev;          struct student \*current;          current = addmem(current);          if (head1 == NULL) //将第一部分的结构放入头地址/头结构              head1 = current;          else //后续的结构地址都会放入              prev->next = current;          current->next = NULL;          puts("请输入学号、姓名、总成绩(回车结束)");          do          {              scanf("%d %s %d", &current->id, current->name, &current->sumMark);              if (seachTheSame(head1, current->id))              {                  puts("已经存在该学生了喔,重新输入吧");              }              else                  flag = 0;          } while (flag);          prev = current; //尾插法 暂存的current正式放入prev      }      return head1;  }   1. 修改   void change(void)  {      int flag = 1;      struct student \*current;      current=head;      int stu=0;      puts("请输入该学生的学号");      scanf("%d",&stu);      while(stu!=current->id&&current->next!=NULL)      {          current=current->next;      }      if(stu!=current->id)      {          system("cls");          printf("没有找到学生信息喔");          Delay(0xffffffff);          system("cls");          return 0;      }      else      {          puts("请输入学号、姓名、总成绩(回车结束)");          scanf("%d %s %d", &current->id, current->name, &current->sumMark);      }      return 0;  }   1. 删除   void delete (void)  {      struct student \*current;      struct student \*before;      current=head;      int stu=0;      puts("请输入该学生的学号");      scanf("%d",&stu);      if (head->id==stu)      {          before=head;          head=head->next;          free(before);      }      else      {          while(stu!=current->id&&current->next!=NULL)          {              before=current;              current=current->next;          }          if(stu!=current->id)          {              printf("没有找到学生信息喔\n");              puts("输出#以退出");              while(getchar()!='#');              system("cls");              return 0;          }          else          {              before->next=current->next;              free(current);              (stuNum)--;          }          puts("删除成功!\n");          puts("输出#以退出");          while(getchar()!='#');      }  }   1. 查找   void search(void)  {      struct student \*current;      current=head;      int stu=0;      puts("请输入该学生的学号");      scanf("%d",&stu);      while(stu!=current->id&&current->next!=NULL)      {          current=current->next;      }      if(stu!=current->id)      {          system("cls");          printf("没有找到学生信息喔\n");          puts("输出#以退出");          while(getchar()!='#');          system("cls");          return 0;      }      else      {          printOneLinker(current);          puts("输出#以退出");          while(getchar()!='#');      }      return 0;  }   1. 排序   struct student \*sortLinked(struct student \*head, int stuNum1)  {      struct student \*current;      current = head;      /\*这里没有采用冒泡,因为比较麻烦所以没用，就直接暴力遍历了,直接swap\*/      for (int i = 0; i < stuNum1; i++)      {          current = head;          for (int k = 0; k < stuNum1 - 1; k++)          {              if ((current->sumMark < current->next->sumMark) && current->next != NULL)              {                  /\*current copy到temp\*/                  int temp1 = current->id, temp3 = current->sumMark;                  char temp2[15];                  strcpy(temp2, current->name);                  /\*next copy到current\*/                  current->id = current->next->id;                  strcpy(current->name, current->next->name);                  current->sumMark = current->next->sumMark;                  /\*temp copy到current.next\*/                  current->next->id = temp1;                  strcpy(current->next->name, temp2);                  current->next->sumMark = temp3;              }              current = current->next;          }      }      return head;  }   1. 统计   void statistics(void)  {      struct student \*current;      current=head;      int sum=0,i=0;      while(current->next!=NULL)      {          sum+=current->sumMark;          i++;          current=current->next;      }      double avg=sum/i;      printf("平均分= %.2f\n",avg);      puts("输出#以退出");      while(getchar()!='#');  }   1. 浏览   void printAllLinked(struct student \*head)  {      struct student \*current = head;      if (head == NULL)          puts("没有学生信息喔？");      else      {          puts("这里是学生信息");          while (current != NULL)          {              printOneLinker(current);              //printf("学号: %d\t 姓名: %s\t 总成绩: %d\n", current->id, current->name, current->sumMark);              current = current->next;          }      }  }   1. 其它   void wannasay(void)  {      puts("  其实你综合看代码看下来其实会很乱，因为linker,c里写了一些关于链表的操作，但是\n主的c文件zonghe.c又有一些是关于链表的操作，少量，但是也多。\n");      puts("  一开始有着对程序架构的完美蓝图，但是在具体写程序的时候你会发现，不\n实际。还是要多加练习，真的就是不熟练，对变量的生命周期一点都不熟悉，调试了两\n个下午，直到周六晚上突破了变量的问题才开始写主函数。\n");      puts("  写完这个项目之后不禁感叹，想去搞项目了，只有项目才能摆脱很多坏习惯，比\n如变量名乱写（看我的变量，是不是全是什么head/prev/before/current之类的，虽\n然说方便我移植，但是要是在项目里说不定会被打死，太多重名的了。而且注释也不\n写完全=>   By pansyhou.\n");      puts("输出#以退出");      while(getchar()!='#');  }  void saveData(void)  {      struct student \*current;      current=head;      FILE \*fp;      int i;      fp=fopen("data.txt","w");      if(fp==NULL)      {          printf("文件打开失败...");          exit (0);      }      for(i=0;i<stuNum;i++)      {          fprintf(fp,"学号: %d\t 姓名: %s\t 总成绩: %d\n", current->id, current->name, current->sumMark);          current=current->next;      }      fclose(fp);  } |
| 3、运行结果  (贴运行结果截图) | （此处分功能逐一贴出每个功能的运行界面截图）   1. 增加      1. 修改      1. 删除      1. 查找      1. 排序      1. 统计      1. 浏览 2. 其它 |
|  |  |

1. **评分表（必填）**

|  |  |  |  |
| --- | --- | --- | --- |
| 功能模块 | 分值 | 是否完成 | 得分 |
| 1、增加 | 10 | 10 | 10 |
| 2、修改 | 15 | 15 | 15 |
| 3、删除 | 10 | 10 | 10 |
| 4、查找 | 10 | 10 | 10 |
| 5、排序 | 10 | 10 | 10 |
| 6、统计 | 15 | 15 | 15 |
| 7、浏览 | 10 | 10 | 10 |
| 8、文档编写 | 20 | 20 | 20 |
| 总计 | | | 100 |

1. 实验心得

其实你综合看代码看下来其实会很乱，因为linker,c里写了一些关于链表的操作，但是主的c文件zonghe.c又有一些是关于链表的操作，少量，但是也多。

一开始有着对程序架构的完美蓝图，但是在具体写程序的时候你会发现，不实际。还是要多加练习，真的就是不熟练，对变量的生命周期一点都不熟悉，调试了两个下午，直到周六晚上突破了变量的问题才开始写主函数。

写完这个项目之后不禁感叹，想去搞项目了，只有项目才能摆脱很多坏习惯，比如变量名乱写（看我的变量，是不是全是什么head/prev/before/current之类的，虽然说方便我移植，但是要是在项目里说不定会被打死，太多重名的了。而且注释也不写完全=>