**C语言编码规范**

**1. 基本要求**  
1.1 程序结构清析，简单易懂，单个函数的程序行数不得超过100行。   
1.2 打算干什么，要简单，直接了当，代码精简，避免垃圾程序。   
1.3 尽量使用标准库函数和公共函数。   
1.4 不要随意定义全局变量，尽量使用局部变量。   
1.5 使用括号以避免二义性。   
  
**2.可读性要求**  
2.1 可读性第一，效率第二。   
2.2 保持注释与代码完全一致。   
2.3 每个源程序文件，都有文件头说明，说明规格见规范。   
2.4 每个函数，都有函数头说明，说明规格见规范。   
2.5 主要变量（结构、联合、类或对象）定义或引用时，注释能反映其含义。   
2.7 常量定义（DEFINE）有相应说明。   
2.8 处理过程的每个阶段都有相关注释说明。   
2.9 在典型算法前都有注释。   
2.10 利用缩进来显示程序的逻辑结构，缩进量一致并以Tab键为单位，定义Tab为 6个   
字节。   
2.11 循环、分支层次不要超过五层。   
2.12 注释可以与语句在同一行，也可以在上行。   
2.13 空行和空白字符也是一种特殊注释。   
2.14 一目了然的语句不加注释。   
2.15 注释的作用范围可以为：定义、引用、条件分支以及一段代码。   
2.16 注释行数（不包括程序头和函数头说明部份）应占总行数的 1/5 到 1/3 。   
  
**3. 结构化要求**  
3.1 禁止出现两条等价的支路。   
3.2 禁止GOTO语句。   
3.3 用 IF 语句来强调只执行两组语句中的一组。禁止 ELSE GOTO 和 ELSE RETURN。   
3.4 用 CASE 实现多路分支。   
3.5 避免从循环引出多个出口。   
3.6 函数只有一个出口。   
3.7 不使用条件赋值语句。   
3.8 避免不必要的分支。   
3.9 不要轻易用条件分支去替换逻辑表达式。   
  
**4. 正确性与容错性要求**  
4.1 程序首先是正确，其次是优美   
4.2 无法证明你的程序没有错误，因此在编写完一段程序后，应先回头检查。   
4.3 改一个错误时可能产生新的错误，因此在修改前首先考虑对其它程序的影响。   
4.4 所有变量在调用前必须被初始化。   
4.5 对所有的用户输入，必须进行合法性检查。   
4.6 不要比较浮点数的相等，   
如： 10.0 \* 0.1 == 1.0 ， 不可靠   
4.7 程序与环境或状态发生关系时，必须主动去处理发生的意外事件，如文件能否   
逻辑锁定、打印机是否联机等。   
4.8 单元测试也是编程的一部份，提交联调测试的程序必须通过单元测试。   
  
**5. 可重用性要求**  
5.1 重复使用的完成相对独立功能的算法或代码应抽象为公共控件或类。   
5.2 公共控件或类应考虑OO思想，减少外界联系，考虑独立性或封装性。   
5.3 公共控件或类应建立使用模板。

代码

学生成绩管理的系统，成绩的录入，分数的排序，删除，修改。

struct \_\_stu{

 int stuNum;

 char name[20];

 int ScoreMath;

};

\_\_stu stu[50];

int numberSTU = 0;

void printinfo(void)

{

 printf("1 成绩录入\n");

 printf("2 成绩显示\n");

 printf("3 成绩修改\n");

 printf("4 [成绩统计](https://www.baidu.com/s?wd=%E6%88%90%E7%BB%A9%E7%BB%9F%E8%AE%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)\n");

 printf("5 成绩排序\n");

 printf("6 显示菜单\n");

 printf("7 退出\n");

}

void inputinfo(int n)

{

 printf("第%d个学生的学号：",n);

 scanf("%d",&stu[n-1].stuNum);

 getchar();

 printf("第%d个学生的姓名：",n);

 gets(stu[n-1].name);

 printf("第%d个学生数学成绩：",n);

 scanf("%d",&stu[n-1].ScoreMath);

 getchar();

}

void displayinfo(int n)

{

 int i = 0;

 printf("学号\t\t姓名\t\t分数\n");

 for (i = 0; i < n; i++)

 {

  printf("%d\t\t%s\t\t%d\n",stu[i].stuNum,stu[i].name,stu[i].ScoreMath);

 }

}

void countNum(int n)

{

 int i = 0;

 int tempNum = 0;

 for (i =0 ; i < n; i++)

 {

  if(stu[i].ScoreMath < 60)

  {

   tempNum++;

  }

 }

 printf("及格的人数为 ：%d\n不及格的人数为 ：%d\n",n-tempNum,tempNum);

}

void modify(int stuNumber)

{

 int i = 0;

 for (i = 0; i < numberSTU;i++)

 {

  if (stu[i].stuNum == stuNumber)

  {

   printf("你想修改的分数为：");

   scanf("%d",&stu[i].ScoreMath);

   getchar();

   printf("修改成功\n");

   return;

  }

 }

 printf("学号错误，修改失败\n");

}

void soucemath\_from\_higt\_to\_low(int n)

 int indexi;

 int indexj;

 \_\_stu \_temp;

 for (indexi = 0; indexi < n;indexi++)

 {

  for (indexj = indexi+1; indexj < n; indexj++)

  {

   if (stu[indexi].ScoreMath < stu[indexj].ScoreMath)

   {

    \_temp = stu[indexj];

    stu[indexj] = stu[indexi];

    stu[indexi] = \_temp;

   }

  }

 }

 printf("数学分数从高到低排列：\n");

 displayinfo(numberSTU);

}

int main(int argc, char\* argv[])

{

 char casechar = 0;

 int studentNUm = 1;

 printinfo();

 while(1)

 {

  printf("输入指令:");

  scanf("%c",&casechar);

  getchar();

  switch (casechar)

  {

   case '1':

    numberSTU++;

    inputinfo(numberSTU);

   // printf("输入信息\n");

    break;

   case '2':

    displayinfo(numberSTU);

    //printf("显示成绩\n");

    break;

   case '3':

    printf("输入你想修改的学生学号:");

    scanf("%d",&studentNUm);

    getchar();

    modify(studentNUm);

    break;

   case '4':

    countNum(numberSTU);

    break;

   case '5':

    soucemath\_from\_higt\_to\_low(numberSTU);

    break;

   case '6':

    printinfo();

    break;

   case '7':

    return 0;

  }

 }

 return 0;

}