第3部分　课程设计

3.1　概　　述

课程设计是对学生的一种全面综合训练，是不可缺少的一个教学环节。通常，课程设计中的问题比平时的习题复杂得多，也更接近实际。课程设计着眼于原理与应用的结合点，使学生学会如何把书上学到的知识用于解决实际问题，培养软件工作所需要的动手能力；另一方面，能使书上的知识变“活”，起到深化理解和灵活掌握教学内容的目的。平时的习题较偏重于如何编写功能单一的“小”算法，局限于一个或两个知识点，而课程设计题是软件设计的综合训练，包括问题分析、总体结构设计、用户界面设计、程序设计基本技能和技巧、多人合作、以至一整套软件工作规范的训练和科学作风的培养。此外，还有很重要的一点是：计算机是比任何教师更严厉的检查者。

为达到上述目的，使学生更好地掌握面向过程程序设计的基本方法和C语言的应用，本教材安排了课程设计环节，提供了6个题目供读者选择。每个实习题采取了统一的格式，由问题描述、基本要求和选做内容等部分组成。问题描述旨在为学生建立问题提出的背景，指明问题“是什么”。基本要求则对问题进一步求精，划出问题的边界，指出具体的参量或前提条件，并规定该题的最低限度要求。选做部分向那些尚有余力的读者提出了更高的要求，同时也能开拓其他读者的思路，在完成基本要求时力求避免就事论事的不良思想方法，尽可能寻求具有普遍意义的解法，使得程序结构合理，容易修改、扩充和重用。

3.2　总 体 要 求

3.2.1　系统分析与系统设计

1. 软件生存周期

软件生存周期是指一个软件从提出开发要求开始直到该软件报废为止的整个时期。通常，软件生存周期包括可行性分析和项目开发计划、需求分析、概要设计、详细设计、编码、测试、维护等活动，可以将这些活动以适当方式分配到不同阶段去完成。

(1) 可行性分析和项目开发计划。明确“要解决的问题是什么”，“解决问题的办法和费用”，“解决问题所需的资源和时间”。要回答这些问题，就要进行问题定义、可行性分析，制定项目开发计划。

(2) 需求分析。需求分析的任务是准确地确定软件系统必须做什么，确定软件系统具备哪些功能。写出软件需求规格说明书。

(3) 概要设计。概要设计的任务是把软件需求规格说明书中确定的各项功能转换成

需要的体系结构。

(4) 详细设计。详细设计就是为每个模块完成的功能进行具体描述,要把功能描述转变为精确的、结构化的过程描述。

(5) 编码。编码就是把每个模块的控制结构转换成计算机可接受的程序代码。

(6) 测试。测试是保证软件质量的重要手段,其主要方式是在设计测试用例的基础上检验软件的各个组成部分。测试分为单元测试、集成测试、确认测试。

(7) 维护。软件维护是软件生存周期中时间最长的阶段。已交付的软件投入正式使用后,便进入软件维护阶段,它可以持续几年甚至几十年。

2. 系统分析

"分析就是在采取行动之前,对问题的研究"。系统分析在软件开发过程中是非常重要的。系统分析是采用系统工程思想方法,对项目的实际情况进行分析综合,制定各种可行方案,为系统设计提供依据。其任务包括对用户进行需求调查,在明确系统目标的基础上,开展用户结构设置、业务关系、数据流程等方面的深入研究和分析,提出系统的结构方案和逻辑模型。系统分析是使系统设计达到合理、优化的重要步骤,该阶段的工作深入与否,直接影响到将来新系统的设计质量和实用。

需求分析是对用户要求和用户情况进行调查分析,确定系统的用户结构、工作流程、用户对应用界面和程序接口的要求,以及系统应具备的功能等,是系统开发的准备阶段。

需求分析的基本任务如下。

(1) 问题识别。

① 功能需求:明确所开发的软件必须具备什么样的功能。

② 性能需求:明确待开发的软件的技术性能指标。

③ 环境需求:明确软件运行时所需要的软、硬件的要求。

④ 用户界面需求:明确人机交互方式、输入输出数据格式。

(2) 分析与综合,导出软件的逻辑模型。

分析人员对获取的需求,进行一致性的分析检查,在分析、综合中逐步细化软件功能,划分成各个子功能。用图文结合的形式,建立起新系统的逻辑模型。

(3) 编写文档。

① 编写"需求规格说明书",把双方共同的理解与分析结果用规范的方式描述出来,作为今后各项工作的基础。

② 编写初步用户使用手册,着重反映被开发软件的用户功能界面和用户使用的具体要求,用户手册能强制分析人员从用户使用的观点考虑软件。

③ 编写确认测试计划,作为今后确认和验收的依据。

④ 修改完善软件开发计划。在需求分析阶段对待开发的系统有了更进一步的了解,所以能更准确地估计开发成本、进度及资源要求,因此对原计划要进行适当修正。

3. 系统设计

系统设计的任务是将系统分析阶段提出的逻辑模型转化为相应的物理模型,设计内

容随系统目标、数据性质和系统功能的不同而存在很大差异。首先应根据系统研制的目标，确定系统功能；其次是数据分类和编码，完成空间数据的存储和管理；最后是系统的建模和产品的输出。

系统设计是整个软件开发的核心，不但要完成逻辑模型所规定的任务，而且要使所设计的系统达到最优化。一个优化的软件系统必须具有运行效率高、控制性能好和可变性强等特点。要提高系统的运行效率，应尽量避免中间文件的建立，减少文件扫描的次数，尽量采用优化的数据处理算法。为了提高系统的可变性，最有效的方法是采用模块化的结构设计方法，首先将系统作为统一整体，然后按功能逐步分解为若干个模块，这样设计出来的系统才能做到可变性好和具有生命力。

3.2.2 详细设计与编码

1. 详细设计的基本任务

(1) 为每个模块进行详细的算法设计。用某种图形、表格、语言等工具将每个模块处理过程的详细算法描述出来。

(2) 为模块内的数据结构进行设计。对于需求分析、概要设计确定的概念性的数据类型进行确切的定义。

(3) 对数据结构进行物理设计，即确定数据库的物理结构。物理结构主要指数据库的存储记录格式、存储记录安排和存储方法，这些都依赖于具体所使用的数据库系统。

(4) 其他设计。根据软件系统的类型，还可能要进行以下设计：

① 代码设计。为了提高数据的输入、分类、存储、检索等操作，节约内存空间，对数据库中的某些数据项的值要进行代码设计。

② 输入/输出格式设计。

③ 人机对话设计。对于一个实时系统，用户与计算机频繁对话，因此要进行对话方式、内容、格式的具体设计。

(5) 编写详细设计说明书。

(6) 评审。对处理过程的算法和数据库的物理结构都要评审。

2. 结构化程序设计方法

详细设计是软件设计的第二阶段，主要确定每个模块具体执行过程，也称“过程设计”，详细设计的目标不仅是逻辑上正确地实现每个模块的功能，而且要使设计出的处理过程清晰易读。过程设计中采用的典型方法是结构化程序设计(简称 SP)方法，最早是由 E. W. Dijkstra 在 20 世纪 60 年代中期提出的，它是实现详细设计目标的关键技术之一。

结构化程序设计方法的基本要点是：

(1) 采用自顶向下，逐步求精的程序设计方法。在需求分析、概要设计中，都采用了自顶向下，逐层细化的方法。

(2) 使用 3 种基本控制结构构造程序。任何程序都可由顺序、选择、循环 3 种基本控制结构构造。

① 用顺序方式对过程分解，确定各部分的执行顺序。

② 用选择方式对过程分解，确定某个部分的执行条件。

③ 用循环方式对过程分解，确定某个部分进行重复的开始和结束的条件。

④ 对处理过程仍然模糊的部分反复使用以上分解方法，最终可将所有细节确定下来。

3. 编码

编码即程序设计，是对详细设计的结果的进一步求精，用相应的程序设计语言表达出来。在充分理解和把握语言运行机制的基础上，编写出正确的、清晰的、易读易改和高效率的程序。另外，在标识符的命名、代码的长度（一个方法长度一般不超过 40 行，否则应划分为两个或多个方法）、程序书写的风格[如缩进格式、空格（空行）的应用、注释等]方面也应注意，遵循统一的规范。

3.2.3　上机调试和测试

软件测试的目的是尽可能多地发现程序中的错误，而调试则是在进行了成功的测试之后才开始的工作。调试的目的是确定错误的原因和位置，并改正错误，因此调试也称为纠错。

1. 调试技术

(1) 简单的调试方法。

① 在程序中插入打印语句。该办法的优点是能显示程序的动态过程，较易检查源程序的有关信息。缺点是效率低。

② 运行部分程序。

测试某些被怀疑有错的程序段，只执行需要检查的程序段，提高效率。

(2) 归纳法调试。

归纳法是一种从特殊到一般的思维过程，从对个别事例的认识当中，概括出共同特点，得出一般性规律的思考方法。

(3) 演绎法调试。

演绎法是一种从一般的推测和前提出发，运用排除和推断过程做出结论的思考方法。演绎法调试是列出所有可能性的错误原因的假设，然后利用测试数据排除不适当的假设，最后再用测试数据验证余下的假设确实是出错的原因。

(4) 回溯法调试。

该方法从程序产生错误的地方出发，人工沿程序的逻辑路径返回搜索，直到找到错误的原因为止。

2. 软件测试的目的及原则

(1) 软件测试的目的。

① 软件测试是为了发现错误而执行程序的过程。

② 一个好的测试用例能够发现至今尚未发现的错误。

③ 一个成功的测试是发现了至今尚未发现的错误的测试。

因此，测试阶段的基本任务应该是根据软件开发各阶段的文档资料和程序的内部结构，精心设计一组“高产”的测试用例，利用这些实例执行程序，找出软件中潜在的各种错误和缺陷。

(2) 测试方法。

软件测试方法一般分为两大类：动态测试方法与静态测试方法。动态测试方法中又根据测试用例的设计方法不同，分为黑盒测试与白盒测试两类。

静态测试是指被测试程序不在机器上运行，而是采用人工检测和计算机辅助静态分析的手段对程序进行检测。人工检测是不依靠计算机而是靠人工审查程序或评审软件。利用静态分析工具对被测试程序进行特性分析，从程序中提取一些信息，以便检查程序逻辑的各种缺陷和可疑的程序构造。

一般意义上的测试大多是指动态测试。有两种方法，分别是黑盒测试法和白盒测试法。

黑盒法把被测试对象看成一个黑盒子，测试人员完全不考虑程序的内部结构和处理过程，只在软件的接口处进行测试，依据需求规格说明书，检查程序是否满足功能要求。因此，黑盒测试又称为功能测试或数据驱动测试。通过黑盒测试主要发现以下错误：

① 是否有不正确或遗漏了的功能。

② 在接口上，能否正确地接受输入数据，能否产生正确的输出信息。

③ 访问外部信息是否有错。

④ 性能上是否满足要求等。

白盒法把测试对象看作一个打开的盒子，测试人员须了解程序的内部结构和处理过程，以检查处理过程的细节为基础，对程序中尽可能多的逻辑路径进行测试，检查内部控制结构和数据结构是否有错，实际的运行状态与预期的状态是否一致。

黑盒法和白盒法都不能使测试达到彻底。为了用有限的测试发现更多的错误，需精心设计测试用例。

3.2.4 课程设计报告

1. 课程设计报告的内容及要求

(1) 需求和规格说明。简述题目要解决的问题是什么，规定软件做什么。

(2) 设计(算法分析、具体实现)。

① 设计思想：阐述程序结构(如类图)、重要的数据结构和主要算法思想(文字描述，不要画框图)。

② 设计表示：包括类名及其作用、类中数据成员名称及其作用、类中成员函数原型及其功能，可以用表格形式表达。

③ 实现注释：包括各项要求的实现程度、在完成基本要求的基础上还实现了什么功能。

④ 详细设计表示：包括主要算法的框架及实现此算法的成员函数接口。

(3) 用户手册。即使用说明(包括数据输入时的格式要求)。

(4) 测试与思考。调试过程中遇到的主要问题是如何解决的,对设计和编码的回顾讨论和分析,程序运行的时空效率分析,测试数据集,运行实例,改进设想,经验和体会等。

2. 附录

源程序清单:打印文本和磁盘文件,磁盘文件是必需的。源程序要加注释,除原有注释外,再用钢笔加一些必要的注释和断言。

测试数据:列出测试数据集。

运行结果:上面测试数据输入后程序运行的结果。

3.3　课程设计样例——学生成绩管理系统

3.3.1　目标与要求

1. 目标

(1) 掌握和利用 C 语言进行程序设计的能力。
(2) 理解和运用结构化程序设计的思想和方法。
(3) 掌握开发一个小型实用系统的基本方法。
(4) 学会调试一个较长程序的基本方法。
(5) 掌握书写程序设计开发文档的能力(书写课程设计报告)。

2. 要求

(1) 用 C 语言实现系统。
(2) 利用结构体数组实现学生成绩的数据结构设计。
(3) 系统具有增加、查询、插入、排序等基本功能。
(4) 系统的各个功能模块要求用函数的形式实现。
(5) 完成设计任务并书写课程设计报告。
(6) 将学生成绩信息存在文件中。

3.3.2　分析

1. 学生成绩管理系统的主要功能

(1) 输入学生记录。
(2) 查看数据。
(3) 插入数据。
(4) 查找数据。
(5) 更新数据。
(6) 保存数据。

(7) 显示或打印数据。

(8) 退出系统。

2. 题目分析

该题主要考查学生对结构体、指针、文件的操作,以及C语言算法的掌握,所以完成此道题目要求较强的设计能力,尤其是要有大局意识。如何调试程序也非常重要,通过这个程序可学到以前调试短程序没有的经验。菜单中的每一个选项都对应一个子程序。

3.3.3 实现步骤

1. 系统需求

(1) 当前学生信息:通过结构体 struct student 来保存学生的姓名、学号、性别以及语文、数学、英语、计算机等课程的相关信息,且通过 cin 函数来给当前学生输入初始信息。

(2) 学生成绩查询:输入一个学号,在文件中查找此学生,若找到则输出此学生的全部信息和成绩;若找不到则输出查找失败的信息。同时也可以全部把各科的平均成绩,最高和最低分输出。

(3) 新生插入:通过该生的学号和该班上的学生的学号比较大小,若大就在后,若小则靠前排,将此学生的信息保存下来。

(4) 输出全部学生信息和全部学生成绩。

(5) 退出系统。

2. 总体设计

仔细阅读系统要求,首先将此系统划分为如下模块(即如下函数)。

(1) 输入初始的学生信息:其中包括学生的姓名、学号和性别以及学生的语文、数学、英语、计算机等课程的相关信息;可用函数 cin(stu * p1)来实现此操作。

(2) 查询模块:可用函数 stu * lookdata(stu * p1)来实现。找到就输出此学生的全部信息,包括学生的语文、数学、英语、计算机等成绩。

(3) 插入模块:可用函数 insert()来实现。其中通过学号的大小比较,并以此来排序。

(4) 输出学生的信息以及成绩:通过学生的姓名来查看学生的语文、数学、英语、计算机等相关成绩,同时也可以分别通过函数 caverage()、maverage()、eaverage()和comaverage()来输出语文、数学、英语、计算机等成绩的平均分数、最高和最低分数。

(5) 退出系统:可用一个函数 exit()来实现,首先将信息保存到文件中,释放动态创建的内存空间,再退出此程序。

3. 详细设计

(1) 界面设计。

此系统界面采用图形和数字化菜单设计。主界面设计如下。

学生成绩管理系统

请选择相应的数字执行相应的功能。

1：输入其他数据

2：查看数据

3：插入数据

4：查找数据

5：更新数据

6：保存数据

7：显示或打印数据

8：语文成绩状况

9：数学成绩状况

10：英语成绩状况

11：计算机成绩状况

12：退出系统

(2) 数据结构设计。

程序设计中用到的学生信息结构体类型：

```
typedef struct student
    {
        char name[MAX];
        int num[MAX];
        char sex[MAX];
        int chinese;
        int mathematic;
        int english;
        int computer;
        struct student * next;
    }
```

(3) 程序代码。

```
/* 原始密码是 123456 */
#include "stdio.h"
#include "stddef.h"
#include "stddef.h"
#include "string.h"
#define MAX 10
typedef struct student                    /* 定义结构体 */
{
    char name[MAX];                       /* 姓名 */
    int num[MAX];                         /* 学号 */
    char sex[MAX];                        /* 性别 */
    int chinese;                          /* 语文 */
    int mathematic;                       /* 数学 */
```

```
    int english;                                /*英语*/
    int computer;                               /*计算机*/
    struct student *next;                       /*结构体指针*/
}stu;
stu *head;                                      /*头指针*/
void print()                                    /*显示或打印函数*/
{
    system("cls");
    printf("\t\t\tScore Manage System\n");      /*成绩管理系统*/
    printf("<1>Enter Record\t");                /*输入数据*/
    printf("<2>Display\t");                     /*显示*/
    printf("<3>Insert\t");                      /*插入数据*/
    printf("<4>Quest\t");                       /*访问数据*/
    printf("<5>Update\t");                      /*以前数据*/
    printf("<6>Save\t");                        /*保存数据*/
    printf("<7>Fresh\t");                       /*更新数据*/
    printf("<8>Chinese Average\t");             /*语文平均成绩*/
    printf("<9>Math Average\t");                /*数学平均成绩*/
    printf("<10>English Average\t");            /*英语平均成绩*/
    printf("<11>Computer Average\t");           /*计算机平均成绩*/
    printf("<12>Quit\t\n");                     /*退出*/
}
void cin(stu *p1)                               /*输入相关数据的函数*/
{
    printf("Enter name:\n");
    scanf("%s",&p1->name);
    printf("Enter num:\n");
    scanf("%d",&p1->num);
    printf("Enter sex:\n");
    scanf("%s",&p1->sex);
    printf("Enter score:\n");
    printf("Enter chinese:\n");
    scanf("%d",&p1->chinese);
    printf("Enter math:\n");
    scanf("%d",&p1->mathematic);
    printf("Enter English:\n");
    scanf("%d",&p1->english);
    printf("Enter Computer:\n");
    scanf("%d",&p1->computer);
}
stu *cindata()                                  /*其他数据是否继续输入的函数*/
{
    stu *p1,*p2;
    int i=1;
    char ch;
```

```
    p1=(stu *)malloc(sizeof(stu));
    head=p1;
    while(i)
    {
        cin(p1);
        printf("Do you Want to Continue?yes or no");     /*是否继续输入数据*/
        ch=getchar();
        ch=getchar();
        if(ch=='n'||ch=='N')
        {
            i=0;
            p1->next=NULL;
        }
        else
        {
            p2=p1;
            p1=(stu *)malloc(sizeof(stu));
            p2->next=p1;
        }
    }
    return(p1->next);
}
stu *lookdata(stu *p1)                          /*查看数据的函数*/
{
    while(p1!=NULL)
    {
        printf("Num:%d\t",p1->num);
        printf("Name:%s\t",p1->name);
        printf("Sex:%s\t",p1->sex);
        printf("\n");
        printf("Chinese:%d\t",p1->chinese);
        printf("Math:%d\t",p1->mathematic);
        printf("English:%d\t",p1->english);
        printf("Computer:%d\t",p1->computer);
        printf("\n");
        p1=p1->next;
        }
    return p1;
}
void insert()                                   /*通过比较学号来插入数据的函数*/
{
    stu *p1,*p3,*p2;
    char ch;
    p1=head;
```