学生学号	0121710880223	实验课成绩	

武漢理工大學

学 生 实 验 报 告 书

实验课程名称	数据结构
开 课 学 院	计算机科学与技术学院
指导教师姓名	胡燕
学 生 姓 名	刘佳迎
学生专业班级	计算机类 m1702 班

2018 -- 2019 学年 第 一 学期

实验教学管理基本规范

实验是培养学生动手能力、分析解决问题能力的重要环节；实验报告是反映实验教学水平与质量的重要依据。为加强实验过程管理，改革实验成绩考核方法，改善实验教学效果，提高学生质量，特制定实验教学管理基本规范。

1、 本规范适用于理工科类专业实验课程，文、经、管、计算机类实验课程可根据具体情况参照执行或暂不执行。

2、 每门实验课程一般会包括许多实验项目，除非常简单的验证演示性实验项目可以不写实验报告外，其他实验项目均应按本格式完成实验报告。

3、 实验报告应由实验预习、实验过程、结果分析三大部分组成。每部分均在实验成绩中占一定比例。各部分成绩的观测点、考核目标、所占比例可参考附表执行。各专业也可以根据具体情况，调整考核内容和评分标准。

4、 学生必须在完成实验预习内容的前提下进行实验。教师要在实验过程中抽查学生预习情况，在学生离开实验室前，检查学生实验操作和记录情况，并在实验报告第二部分教师签字栏签名，以确保实验记录的真实性。

5、 教师应及时评阅学生的实验报告并给出各实验项目成绩，完整保存实验报告。在完成所有实验项目后，教师应按学生姓名将批改好的各实验项目实验报告装订成册，构成该实验课程总报告，按班级交课程承担单位（实验中心或实验室）保管存档。

6、 实验课程成绩按其类型采取百分制或优、良、中、及格和不及格五级评定。

附表：实验考核参考内容及标准

	观测点	考核目标	成绩组成
实验预习	1. 预习报告 2. 提问 3. 对于设计型实验，着重考查设计方案的科学性、可行性和创新性	对实验目的和基本原理的认识程度，对实验方案的设计能力	20%
实验过程	1. 是否按时参加实验 2. 对实验过程的熟悉程度 3. 对基本操作的规范程度 4. 对突发事件的应急处理能力 5. 实验原始记录的完整程度 6. 同学之间的团结协作精神	着重考查学生的实验态度、基本操作技能；严谨的治学态度、团结协作精神	30%
结果分析	1. 所分析结果是否用原始记录数据 2. 计算结果是否正确 3. 实验结果分析是否合理 4. 对于综合实验，各项内容之间是否有分析、比较与判断等	考查学生对实验数据处理和现象分析的能力；对专业知识的综合应用能力；事实求实的精神	50%

实验课程名称：数据结构

实验项目名称	实验三			实验成绩	
实 验 者	刘佳迎	专业班级	计算机类 m1702 班	组 别	
同 组 者				实验日期	2018 年 12 月 31 日

一部分：实验预习报告（包括实验目的、意义，实验基本原理与方法，主要仪器设备及耗材，实验方案与技术路线等）

实验目的及方案:

问题描述

大学的每个专业都要制订教学计划。假设任何专业都有固定的学习年限，每学年含两学期，每学期的时间长度和学分上限均相等。每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的，可以有任意多门，也可以没有。每门课恰好占一个学期。试在这样的前提下设计一个教学计划编制程序。

1、基本要求

（1）.输入参数包括：学期总数，一学期的学分上限，每门课的课程号（固定占 3 位的字母数字串）、学分和直接先修课的课程号。

（2）允许用户指定下列两种编排策略之一：一是使学生在各学期中的学习负担尽量均匀；二是使课程尽可能地集中在前几个学期中。

（3）若根据给定的条件问题无解，则报告适当的信息；否则，将教学计划输出到用户指定的文件中。计划的表格格式自行设计。

2、测试数据

学期总数：6；

学分上限：10；

该专业共开设课数：12

课程号：从 C01 到 C12；

学分顺序：2，3，4，3，2，3，4，4，7，5，2，3。先修关系见书图 7.26。

3、输出输入要求

输入参数包括：学期总数，一学期的学分上限，每门课的课程号（固定占 3 位的字母数字串）、学分和直接先修课的课程号。

输出要求输出各门课程所对应的学分，以及每学期各门课程的安排。

4、实现提示

可设学期总数不超过 12，课程总数不超过 100。如果输入的先修课程号不在该专业开设的课程序列中，则作为错误处理。应建立内部课程号与课程号之间的对应关系。

实验问题概要设计

1、抽象数据类型图的定义如下:

ADT Graph{

数据对象 V:V 是具有相同特性的数据元素的集合,称为顶点集.

数据关系 R: R={VR}

VR={(v,w)|v,w∈V,(v,w)表示 v 和 w 之间存在直接先修关系}

基本操作 P:

LocateVex（ALGraph G,VertexType u）;//图的邻接表存储的基本操作

CreateGraph（ALGraph *G）;//构造生成树

Display（ALGraph G）;//输出图的邻接矩阵

FindInDegree（ALGraph G,int indegree）;//求顶点的入度

TopologicalSort（ALGraph G）;//有向图 G 采用邻接表存储结构。若 G 无回路,则输出 G 的顶点的一个拓扑序列并返回 OK,否则返回 ERROR。

}ADT Graph

2、栈的定义:

ADT Stack{

数据对象:D={ai|ai∈ElemSet,i=1,2,…n,n>=0}

数据关系:R1={<ai-1 ai>|ai-1,ai∈D,i=2,…,n}

基本操作 D：

InitStack (SqStack *S); //初始化一个空栈 S

StackEmpty(SqStack S); //若栈 S 为空栈，则返回 TRUE，否则返回 FALSE

Push(SqStack *S,SElemType *e); //插入元素 e 为新的栈顶元素

Pop(SqStack *S,SElemType *e); //若栈不空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；否则返回 ERROR

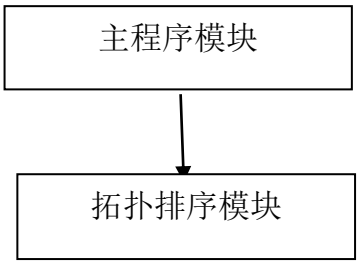
}ADT Stack

3、主程序

```
void main()
{
ALGraph f;
printf("教学计划编制问题的数据模型为拓扑排序 AOV-网结构。\n");
printf("以下为教学计划编制问题的求解过程:\n");
printf("请输入学期总数:");
scanf("%d",&xqzs);
printf("请输入学期的学分上限:");
scanf("%d",&xfsx);
CreateGraph(&f);
Display(f);
```

```
    TopologicalSort(f);
    }
```

4、本程序只有两个模块,调用关系简单.



第二部分：实验过程记录（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

实验主要内容

1、顶点、边、图类型

```
#define MAX_VERTEX_NUM 100
typedef int pathone[MAXCLASS];
typedef int pathtwo[MAXCLASS];
 typedef enum{DG}GraphKind; /* {有向图,有向网,无向图,无向网} */
 typedef struct ArcNode
 {
   int adjvex; /* 该弧所指向的顶点的位置*/
   struct ArcNode *nextarc; /* 指向下一条弧的指针*/
   InfoType *info; /* 网的权值指针）*/
 }ArcNode; /* 表结点*/
 typedef struct
 {
   VertexType data; /* 顶点信息*/
   ArcNode *firstarc; /* 第一个表结点的地址,指向第一条依附该顶点的弧的指针*/
 }VNode,AdjList[MAX_VERTEX_NUM]; /* 头结点*/
 typedef struct
 {
   AdjList vertices,verticestwo;
   int vexnum,arcnum; /* 图的当前顶点数和弧数*/
   int kind; /* 图的种类标志*/
 }ALGraph;
```

2、图的基本操作

int InitGraph(ALGraph &G);初始化邻接多重表，表示一个空图。

int LocateVex(ALGraph G,VertexType u)； 若 G 中存在顶点 u,则返回该顶点在图中位置;否则返回-1。

Status CreateGraph(ALGraph *G)；采用邻接表存储结构,构造没有相关信息的图 G(用一个函数构造种图。

void Display(ALGraph G)；输出图的邻接矩阵 G。

void FindInDegree(ALGraph G,int indegree[])；求顶点的入度，算法调用。

Status TopologicalSort(ALGraph G)；有向图 G 采用邻接表存储结构。若 G 无回路,则输出 G 的顶点的一个拓扑序列并返回 OK, 否则返回 ERROR

3、栈类型

```
#define STACK_INIT_SIZE 10 /* 存储空间初始分配量*/
 #define STACKINCREMENT 2 /* 存储空间分配增量*/
 typedef struct SqStack
 {
```

```
        SElemType *base; /* 在栈构造之前和销毁之后，base 的值为 NULL */
        SElemType *top; /* 栈顶指针*/
        int stacksize; /* 当前已分配的存储空间，以元素为单位*/
     }SqStack; /* 顺序栈*/
```

4、栈的基本操作

Status InitStack(SqStack *S)；初始化

void ClearStack(SqStack *S)；清空栈的操作

Status StackEmpty(SqStack S)； 若栈 S 为空栈，则返回 TRUE，否则返回 FALSE

Status Pop(SqStack *S,SElemType *e)；若栈不空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；否则返回 ERROR

Status Push(SqStack *S,SElemType e)；插入元素 e 为新的栈顶元素

5、部分操作代码如下：

```
int LocateVex(ALGraph G,VertexType u)
{ /* 初始条件: 图 G 存在,u 和 G 中顶点有相同特征*/
   /* 操作结果: 若 G 中存在顶点 u,则返回该顶点在图中位置;否则返回-1 */
   int i;
   for(i=0;i<G.vexnum;++i)
     if(strcmp(u,G.vertices[i].data)==0)
       return i;
   return -1;
}
Status CreateGraph(ALGraph *G)
 { /* 采用邻接表存储结构,构造没有相关信息的图 G(用一个函数构造种图) */
   int i,j,k;
   VertexType va,vb;
   ArcNode *p;

   printf("请输入教学计划的课程数: ");
   scanf("%d",&(*G).vexnum);
   printf("请输入拓扑排序所形成的课程先修关系的边数: ");
   scanf("%d",&(*G).arcnum);
   printf(" 请 输 入 %d 个 课 程 的 代 表 值 (<%d 个 字符):\n",(*G).vexnum,MAX_NAME);
   for(i=0;i<(*G).vexnum;++i) /* 构造顶点向量*/
   { scanf("%s",(*G).vertices[i].data);
     (*G).vertices[i].firstarc=NULL;
   }
   printf(" 请 输 入 %d 个 课 程 的 学 分 值 (<%d 个 字符):\n",(*G).vexnum,MAX_NAME);
   for(i=0;i<(*G).vexnum;++i) /* 构造顶点向量*/
   {scanf("%s",(*G).verticestwo[i].data);
   }
   printf("请顺序输入每条弧(边)的弧尾和弧头(以空格作为间隔):\n");
```

```
        for(k=0;k<(*G).arcnum;++k) /* 构造表结点链表*/
        { scanf("%s%s",va,vb);
          i=LocateVex(*G,va); /* 弧尾*/
          j=LocateVex(*G,vb); /* 弧头*/
          p=(ArcNode*)malloc(sizeof(ArcNode));
          p->adjvex=j;
          p->info=NULL; /* 图*/
          p->nextarc=(*G).vertices[i].firstarc; /* 插在表头*/
          (*G).vertices[i].firstarc=p;
         }
        return OK;
     }
    void Display(ALGraph G)
    { /* 输出图的邻接矩阵 G */
        int i;
        ArcNode *p;
        switch(G.kind)
        {case DG: printf("有向图\n");
        }
        printf("%d 个顶点：\n",G.vexnum);
        for(i=0;i<G.vexnum;++i)
          printf("%s ",G.vertices[i].data);
        printf("\n%d 条弧(边):\n",G.arcnum);
        for(i=0;i<G.vexnum;i++)
        {
          p=G.vertices[i].firstarc;
          while(p)
          {printf("%s                                →                                %s
",G.vertices[i].data,G.vertices[p->adjvex].data);
           p=p->nextarc;
          }
          printf("\n");
        }
     }
    void FindInDegree(ALGraph G,int indegree[])
     { /* 求顶点的入度，算法调用*/
        int i;
        ArcNode *p;
        for(i=0;i<G.vexnum;i++)
          indegree[i]=0; /* 赋初值*/
        for(i=0;i<G.vexnum;i++)
        {
          p=G.vertices[i].firstarc;
          while(p)
```

```
          { indegree[p->adjvex]++;
            p=p->nextarc;
          }
        }
    Status InitStack(SqStack *S)
     { /* 构造一个空栈 S */
       (*S).base=(SElemType *)malloc(STACK_INIT_SIZE*sizeof(SElemType));
       if(!(*S).base)
         exit(OVERFLOW); /* 存储分配失败*/
       (*S).top=(*S).base;
       (*S).stacksize=STACK_INIT_SIZE;
       return OK;
    }

    void ClearStack(SqStack *S)  //清空栈的操作
    {
       S->top=S->base;
    }

    Status StackEmpty(SqStack S)
     { /* 若栈 S 为空栈，则返回 TRUE，否则返回 FALSE */
       if(S.top==S.base)
         return TRUE;
       else
         return FALSE;
    }
    Status Pop(SqStack *S,SElemType *e)
     {/* 若栈不空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；否则返回
ERROR */
       if((*S).top==(*S).base)
         return ERROR;
       *e=*--(*S).top;
       return OK;
     }
    Status Push(SqStack *S,SElemType e)
     { /* 插入元素 e 为新的栈顶元素*/
       if((*S).top-(*S).base>=(*S).stacksize) /* 栈满，追加存储空间*/
       {
         (*S).base=(SElemType
*)realloc((*S).base,((*S).stacksize+STACKINCREMENT)
         *sizeof(SElemType));
         if(!(*S).base) exit(OVERFLOW); /* 存储分配失败*/
         (*S).top=(*S).base+(*S).stacksize;
         (*S).stacksize+=STACKINCREMENT;
```

```
        }
        *((*S).top)++=e;
        return OK;
      }
    Status TopologicalSort(ALGraph G)
     { /* 有向图 G 采用邻接表存储结构。若 G 无回路,则输出 G 的顶点的一个拓
扑序列并返回 OK, */
        /* 否则返回 ERROR。*/
        int i,k,j=0,count,indegree[MAX_VERTEX_NUM];
        bool has=false;
        SqStack S;
        pathone a;
        pathtwo b;
        ArcNode *p;
        FindInDegree(G,indegree); /* 对各顶点求入度 indegree[0..vernum-1]
*/
        InitStack(&S); /* 初始化栈*/
        for(i=0;i<G.vexnum;++i) /* 建零入度顶点栈 S */
          if(!indegree[i])
         { Push(&S,i);
           //cout<<*G.vertices[i].data<<endl;
         }
         /* 入度为者进栈*/
        count=0; /* 对输出顶点计数*/
        while(!StackEmpty(S))
        { /* 栈不空*/
          Pop(&S,&i);
          a[i]=*G.vertices[i].data;
          b[i]=*G.verticestwo[i].data;
          printf("      课      程      %s      →      学      分      %s
",G.vertices[i].data,G.verticestwo[i].data);
       /* 输出 i 号顶点并计数*/
          ++count;
          for(p=G.vertices[i].firstarc;p;p=p->nextarc)
          { /* 对 i 号顶点的每个邻接点的入度减*/
            k=p->adjvex;
            if(!(--indegree[k])) /* 若入度减为,则入栈*/
           {Push(&S,k);
            //cout<<*G.vertices[i].data<<endl;
           }
          }
        }
        if(count<G.vexnum)
        {printf("此有向图有回路\n");
```

```
        return ERROR;
      }
      else
      {printf("为一个拓扑序列。\n");
      has=true;
      }

       FindInDegree(G,indegree);   /*   对   各   顶   点   求   入   度
indegree[0..vernum-1] */
       ClearStack(&S);
       cout<<"=====================   课   程   计   划   如   下
==============================="
        <<endl;
       int qq=1; //学期数
       int xxf;
       while(qq<=xqzs)
       {
          int result[20];
          int rtop=0;
          int nn=0;
          //int ccount=0;
            // 学期学分计算
          xxf=0;
          for(i=0;i<G.vexnum;++i) /* 建零入度顶点栈 S */
          {   if(0==indegree[i])
              {
                 Push(&S,i);
              }
          }
          while(!StackEmpty(S))
          {
             int bb;
             Pop(&S,&i);
             bb=atoi(G.verticestwo[i].data);
             xxf=xxf+bb;
             if(xxf>xfsx)
             {
                 break;
             }
             indegree[i]--;
             for(p=G.vertices[i].firstarc;p;p=p->nextarc)
             { /* 对 i 号顶点的每个邻接点的入度减*/
                k=p->adjvex;
                indegree[k]--;
```

```
                /* if(!(--indegree[k])) 若入度减为,则入栈
                {
                    Push(&S,k);
                }*/
            }
            result[rtop]=i;
            rtop++;
        }
        cout<<"第"<<qq<<"个"<<"学期的课程为："<<endl;
        for(nn=0;nn<rtop;nn++)
        {
            cout<<"课程"<<G.vertices[result[nn]].data<<endl;
        }
        qq++;
    }
return OK;
}
```

6、主函数：

```
int main()
{  ALGraph f;
   printf("教学计划编制问题的数据模型为拓扑排序 AOV-网结构。\n");
   printf("以下为教学计划编制问题的求解过程:\n");
   printf("请输入学期总数:");
   scanf("%d",&xqzs);
   printf("请输入学期的学分上限:");
   scanf("%d",&xfsx);
   CreateGraph(&f);
   Display(f);
   TopologicalSort(f);
}
```

教师签字_____

第三部分　结果与讨论（可加页）

一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）

实验过程中出现的问题及解决方法

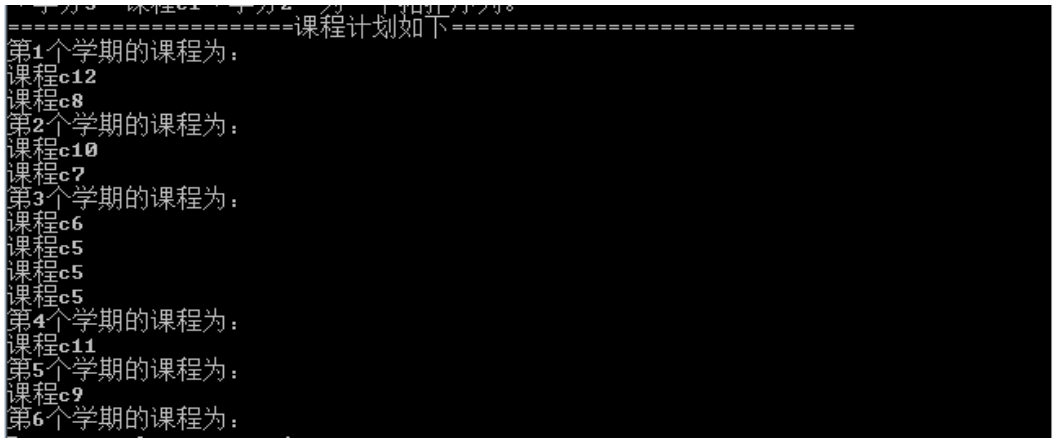
我们在实验过程中遇到的最大难题是两个课程排序算法的编写。刚开始的时候没有任何的思路，网上也只有拓扑排序的算法，对于课程设计要求的排序算法没有任何头绪。经过请教同学以及翻阅了一些相关书籍，并在网上的搜索有了排序算法的大体思路。经过三天的修改，终于写出了符合要求的排序算法。

数据处理和现象

输入学期总数，学分上限，课程数，先修关系边数，课程代表符号，相对学分值

```
教学计划编制问题求解程序:
请输入学期总数:6
请输入学期的学分上限:10
请输入教学计划的课程数: 12
请输入拓扑排序所形成的课程先修关系的边数: 15
请输入12个课程的代表值(<10个字符):
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
c11
c12
请输入12个课程的学分值(<10个字符):
请顺序输入每条弧(边)的弧尾和弧头(以空格作为间隔):
c4 c1
c5 c4
c2 c1
c3 c2
c5 c3
c3 c1
c7 c3
c8 c3
c12 c1
c12 c9
c12 c10
c11 c9
c6 c11
c8 c6
c10 c9
```

输入完成后执行可得到每个学期的课程结果



二、小结、建议及体会

经过此次课程设计，我认识到了理论与实践结合的重要性，仅仅只是从课本上学到算法原理是远远不够的。在实践中，我们总会出现许多错误。这就要求我们以一个脚踏实地的态度来处理问题。我们深刻地认识到自己写程序的不足，使我们学到了好多有用的知识，让我们明白了 C 语言的语句用法。