2019 年“数据结构与 C 语言程序设计”考试内容包括“数据结构”与“C 语言程序设计”两门课程的内容，各占比例 50%。试卷满分为 150 分。

“数据结构”部分

一、概述

1.数据的逻辑结构与存储结构的基本概念;

2.算法的定义、基本性质以及算法分析的基本概念，包括采用大 O 形式表示时间复杂度和空间复杂度。

二、线性表

1.线性关系、线性表的定义，线性表的基本操作;

2.线性表的顺序存储结构与链式存储结构(包括单(向)链表、循环链表和双向链表)的构造原理;

3.在以上两种存储结构的基础上对线性表实施的基本操作，包括顺序表的插入与删除、链表的建立、插入与删除、查找等操作对应的算法设计(含递归算法的设计)。

三、数组

1.一维数组和二维数组的存储;

2.矩阵的压缩存储的基本概念;

3.对称矩阵、对角矩阵以及三角矩阵的压缩存储。

四、堆栈与队列

1.堆栈与队列的基本概念与基本操作;

2.堆栈与队列的顺序存储结构与链式存储结构的构造原理;

3.在不同存储结构的基础上对堆栈与队列实施插入与删除等基本操作的算法设计;

4.堆栈和队列在解决实际问题中应用。

五、树与二叉树

1.树与二叉树的基本概念，基本特征、名词术语;

2.完全二叉树与满二叉树的基本概念，二叉树的基本性质及其应用;

3.二叉树的顺序存储结构与二叉链表存储结的基本原理;

4.二叉树的前序遍历、中序遍历、后序遍历和按层次遍历，重点是二叉树在以二叉链表作为存储结构基础上各种遍历算法(包括非递归算法)的设计与应用;

5.二叉排序树的基本概念、建立(插入)、查找以及平均查找长度 ASL 的计算。

六、图

1.图的基本概念、名词术语;

2.图的邻接矩阵存储方法和邻接表(含逆邻接表)存储方法的构造原理及特点;

3.图的深度优先搜索与广度优先搜索;

4.最小(代价)生成树、最短路径、AOV 网与拓扑排序的基本概念。

七、文件及查找

1.顺序查找法以及平均查找长度(ASL)的计算;

2.折半查找法以及平均查找长度(ASL)的计算，包括查找过程对应的“判定树”的构造;

3.散列(Hash)表的构造、散列函数的构造，散列冲突的基本概念、处理散列冲突的基本方法以及散列表的查找和平均查找长度的计算。

八、内排序

1.排序的基本概念，各种内排序方法的基本原理和特点，包括排序过程中进行的元素之间的比较次数，排序总趟数、排序稳定性以及时间复杂度与空间复杂度计算;

2.插入排序法(含折半插入排序法);

3.选择排序法;

4.(起)泡排序法;

5.谢尔(Shell)排序法;

6.快速排序法;

7.堆积(Heap)排序法，包括堆积的定义与构造;

“C 语言程序设计”部分

一、C 语言基本知识

1.C 语言的特点以及 C 语言程序的组成;

2.数据类型，包括整型、实型、字符型等常量与变量和变量的赋值;用 typedef 定义类型;

3.各种类型数据之间的混合运算;

4.算术表达式、关系表达式和逻辑表达式，表达式 sizeof 的含义。

二、基本语句

1.赋值语句(含条件赋值语句)、条件语句(含 if、if-else、switch)、循环语句(含 while、do-while、for 语句，包括循环嵌套和 break 语句与 continue 语句);

2.输入/输出语句，包括整型、实型、字符型(含字符串)等类型数据的格式输入函数 scanf 和格式输出函数 printf。

三、数组

1.一维数组和二维数组的定义、引用与初始化;

2.字符数组的定义、引用与初始化，字符数组的输入与输出，字符串和字符串处理函数的应用。

四、函数

1.函数的定义，函数参数(形参和实参)与函数的返回值;

2.函数的调用，包括函数的嵌套调用和函数的递归调用;

3.命令行参数的基本概念，带参数的主函数的概念和应用。

五、指针

1.指针的基本概念，包括定义、使用、指针变量作为函数参数和函数返回值以及函数指针;

2.数组与指针，包括指向数组的指针变量的定义与赋值、通过指针引用数组元素、数组名作为函数参数;

3.字符串与指针，指向字符串的指针变量。

六、预处理指令

1.预处理指令的基本概念，文件包含和条件包含预处理指令;

2.宏替换，带参数的宏。

七、结构体与共用体

1.结构体的基本概念和特点，结构体变量的初始化与引用，结构体指针的使用;

2.结构体数组，包括结构体数组的定义、初始化及应用;

3.共用体的基本概念，共用体变量的引用。

八、位运算

1.位运算和位运算符;

2.位运算的应用。

九、文件

1.文件的基本概念，包括文件类型指针 FILE 与文件的使用方式;

2.文件的打开函数 fopen 与关闭函数 fclose;

3.文件的状态，包括 feof 函数和 ferror 函数;

4.文件的读/写，包括 fread 和 fwrite 函数、fputc 和 fgetc 函数、fgets 与 fputs 函数的应用;

5.文件的输入函数 fscanf 和输出函数 fprintf 的应用;

6.文件的定位，包括 rewind 函数和 fseek 函数以及 ftell 函数的应用。

参考用书：

1.《数据结构教程第 3 版》唐发根编著北京航空航天大学出版社 2017

2.《C 程序设计》谭浩强编著清华大学出版社（版次不限）

一．选择题

1．C

解释：（教材 P12）O（n）表示运行时间与 n 成正比，O（n^2）表示运行时间与 n^2 成正比。

2．A

解释：（教材 P28）与链表相比，顺序表可以随机存取。

	顺序表	链表
A	1	（1+n）/2
B	n	n
C	n	n
D	1	1

3．D

解释：（教材 P31）在已知链表中结点的插入或删除位置明确的情况下，插入一个结点或删除一个结点时，仅需要修改指针而不需要移动元素。

4．A

解释：队列遵循“先进先出”的规则，从队头删除，从队尾插入，因此最好有队头指针和队尾指针。另外，循环单链表可以解决队列的假溢出问题。

5．C

解释：（教材 P152）完全二叉树的叶结点只可能在二叉树的最下面两层上出现，A 错误。完全二叉树最多只有最下面两层结点的度可以小于 2，BD 错误。

6．D

解释：课后习题原题

7．D

解释：（教材 P221）图的遍历通常采用两种方式，分别是深度优先搜索和广度优先搜索，A 正确。一个各个边的权值都相等的环（是带权连通图），它的最小生成树就不唯一，B 正确。（教材 P231）迪杰斯特拉提出了按路径长度递增的次序产生最短路径的算法，C 正确。（教材 P236）测试 AOV 网是否具有回路的方法就是构造拓扑序列。AOV 网是有向图，不是一般的图。在拓扑排序过程中，需要选择没有前驱的顶点。所以，拓扑排序是检测 AOV 网（有向图）是否存在回路的方法之一，D 错误。

8．A

解释：教材课后习题原题

9．C

解释：教材课后习题原题

10. B

解释：（教材 P297）泡排序方法不一定要进行 n-1 趟排序。判断泡排序方法结束的一个条件是“在一趟排序过程中没有进行过元素交换的动作”。若原序列本身就是有序序列，那么第一趟排序过程中就没有进行元素交换的动作，泡排序到此结束，不会等到 n-1 趟，B 错误。

二．简答题

1．　答：（教材 P92）采用顺序存储结构，在重新调整存储空间时，元素的移动量比较大，尤其是在分配的存储空间即将充满时。而采用链式存储结构就可以避免这种情况。另外，由于采用了链式存储结构，不必事先声明一片存储区作为堆栈的存储空间，因而不存在因为栈满而产生溢出的问题，只要从存储库中可以动态申请到可用链结点空间，就可以认为堆栈没有满。在实际问题中，若不知道或难以估计将要进栈元素的最大数量时，采用链式存储结构比采用顺序存储结构更为合适。

2. 答：（教材 P170）根据定义，二叉树的前序遍历是先访问根结点，其次再按前序遍历方式递归地遍历根结点的左子树和右子树。显然，在得到的前序序列中，第一个结点一定是二叉树的根结点。根据二叉排序树的特点，每个结点的值都大于它的左子树所有结点的值（若存在的话），都小于它的右子树所有结点的值（若存在的话）。二叉排序树的这一特点自然而然地将前序序列分割成前后两个子序列，即左右两个子树。如此递归下去，当取尽前序序列中的所有结点时，便得到了一棵二叉排序树。

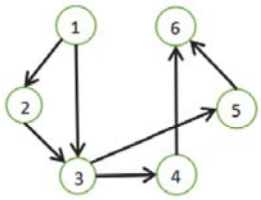
3. 答：（教材 P288）至少要进行 0+1+……+(n-1)=n(n-1)/2 次探测。因为，散列表初始为空，第一次向散列表插入关键字时无冲突，无需进行探测；第二次时有冲突，需要探测一次；第三次时有冲突，需要探测两次。以此类推。因此至少要进行 n(n-1)/2 次探测。

4. 答：（教材 P319）不一定是，因为后续的排序过程中可能还有更小的元素会插入到它的前面。

三．综合题

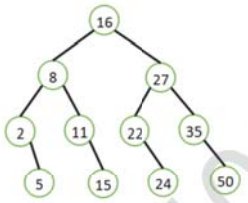
1.这段代码的功能是删除链表中值域为 x 的结点。

2.如图，



拓扑序列有：1,2,3,4,5,6；1,2,3,5,4,6

3. 判定树如图所示，



比较一次的分别有：16；

比较两次的分别有：8,27；

比较三次的分别有：2,11,22,35；

比较四次的分别有：5,15,24,50

4. 该序列为：（65,60,63,50,36,40,25,38）

四．算法设计题

```
#define M 50
void ExchangeChild(BTREE T)
{
    BTREE STACK[M],p=T,q=NULL;
    int top=-1;
    if(T!=NULL)
    {
        STACK[++top]=p;
        while(!(p==NULL&&top==-1))
        {
```

```
            p=STACK[top--];
            if(p!=NULL)
            {
                if(p->data==item)
                {
                    q=p->lchild;
                    p->lchild=p->rchild;
                    p->rchild=q;
                    break;
                }
                STACK[++top]=p->rchild;
                STACK[++top]=p->lchild;
            }
        }
    }
}
```

五．单项选择题

1. D

解释：-做负号运算符时，是单目运算符；*做指针运算符时，是单目运算符；&做取地址运算符时，是单目运算符。

2. B

解释：逻辑表达式也可以，不是必须转换成关系表达式，A 错误；break 语句是跳出循环语句的，不是 if 语句，C 错误；当 if 中的条件为假时，如果没有 else 部分，就执行 if 语句之后的语句，不会出错，D 错误。

3. C

解释：初始条件是 i=0；第一次执行表达式 2（i=0<3 成立，i 加 1 变为 1）；第二次执行表达式 2（i=1<3 成立，i 加 1 变为 2）；第三次执行表达式 2（i=2<3 成立，i 加 1 变为 3）；第四次执行表达式 2（i=3<3 不成立，i 加 1 变为 4）循环结束。

4. A

解释：p+1 所指的整数变量用*(p+1)表示，B 错误；指针 p 向后移动用 p+1 表示，不需要*，CD 错误。

5. D

解释：str 是数组名，本身就是一个地址，用 scanf 输入时不需要再使用&，A 错误；scanf 函数将空格视为输入结束的标志，无法继续输入 World，B 错误；gets 函数也不需要用&，C 错误；

6. A

解释：for 语句的三个表达式都可以省略，但是()中的两个分号;不可以省略。

7. D

解释：数组的大小是通过占用的内存空间大小判断的，并没有存到数组的第一个元素前面，D 错误。

8. C

解释：按位与运算。当 x 取 3 时，x&(x-1)=3&2 不等于 0，A 错误；当 x 取 6 时，x&(x-1)=6&5 不等于 0，B 错误；2^n 的二进制表示是一个 1 后边若干个 0，减 1 之后，原来的 1 变为 0，后边的 0 变为 1，经过按位与运算，得到的结果为 0，故选 C。

9. C

解释：name 是数组名，是该数组的首地址，是一个常量，不能出现在=左边，A 错误；pc 是一个指针，要用->才能访问成员变量，B 错误；Point 是成员变量的类型名，放完成员变量时要使用成员变量的变量名而不是类型名，D 错误。

10. B

解释：当程序关闭某一个文件后，是可以再次打开的，不然无法进行之后对该文件的操作，B 错误。

六．综合题

1. a=(x-x%100)/100;b=(x-x%10-a*100)/10;c=x-100*a-10*b;

2. （1）s[n]==' '||s[n]=='\t'||s[n]=='\n' （2）n=n-1 （3）n

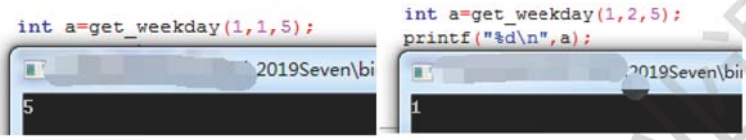
3. 静态局部变量采用静态存储方式，在程序运行期间由系统分配固定的存储空间。普通局部变量采用动态存储方式，在程序运行期间根据需要进行动态地分配存储空间。如果在定义局部变量时不赋初值的话，对于静态局部变量来说，编译时自动赋初值 0 或空字符。而对普通局部变量来说，它的值是一个不确定的值。

4. 预处理指令是由 ANSI C 统一规定的用以改进程序设计环境，提高编程效率的指令。它不是 C 语言本身的组成部分，不能直接对它们进行编译。必须在对程序进行编译之前根据预处理指令对程序作相应的处理。预处理指令的典型应用场景有宏定义和文件包含。

七．程序设计题

```
int get_weekday(int year,int month,int day)
{
    int y=year,m=month,d=day;
    int leaps=0,year_leap=0,days=0;
    int year_isleap[12]={31,29,31,30,31,30,31,31,30,31,30,31};
    int year_isnotleap[12]={31,28,31,30,31,30,31,31,30,31,30,31};
    for( ;y>=1;y--)
    {
        if(y%4==0)
        {
            if(y%100==0)
            {
                if(y%400==0)
                {
                    if(y==year)
                        year_leap=1;
                    else
                        leaps++;
                }
            }
            else
            {
                if(y==year)
                    year_leap=1;
                else
                    leaps++;
```

```
            }
        }
    }
    days+=leaps*366+(year-1-leaps)*365;
    if(year_leap)
        for(m=m-2;m>=0;m--)
            days+= year_isleap[m];
    else
        for(m=m-2;m>=0;m--)
            days+= year_isleap[m];
    days+=d;
    return (days%7);
}
```



八．程序设计题

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main()
{
    char item[20],
    filename[20],
    ch[20],/*存放待比较的单词*/
    com_item[20],/*用于比较过程中做标记*/
    output_ch[20];/*已经输出过的单词*/
    FILE *fp;/*指向进行对比的文件*/
    FILE *output_fp=fopen("output","w+");/*指向暂时存储已经输出的字符串的文件*/
    int i=0,j=0,is_output=0,is_finded=0;
    printf("请输入您要查找的单词：\n");
    scanf("%s",item);
    for(i=0;item[i]!='\0';i++)/*转为小写*/
        if(item[i]>=65&&item[i]<=90)
            item[i]=item[i]+32;
    printf("请输入目标文件：\n");
    scanf("%s",filename);
    if((fp=fopen(filename,"r"))==NULL)
    {
        printf("cannot open file");
    }
    fscanf(fp,"%s",ch);
    while(!feof(fp))
```

```
{
    for(i=0;ch[i]!='\0';i++)/*转为小写*/
        if(ch[i]>=65&&ch[i]<=90)
            ch[i]=ch[i]+32;
    if((strlen(item)!=strlen(ch))||strcmp(item,ch)==0)
    {
        fscanf(fp,"%s",ch);
        continue;/*若长度不相等或为原字符串，则跳过，节省时间*/
    }

    for(i=0;item[i]!='\0';i++)
        com_item[i]=item[i];/*复制到另一个数组中，便于做标记*/
    for(i=0;ch[i]!='\0';i++)
    {
        for(j=0;com_item[j]!=ch[i]&&com_item[j]!='\0';j++);
        if(com_item[j]=='\0')
            break;/*若未找到对应字母，则跳出本次检查*/
        else
            com_item[j]='*';/*若找到对应字母，则做标记*/
    }
    if(i==strlen(ch))/*若该单词是变位单词，则判断是否输出*/
    {
        is_finded=1;
        fseek(output_fp,0,0);/*从头开始搜索存放已经输出的单词的文档*/
        fscanf(output_fp,"%s",output_ch);
        while(!feof(output_fp))
        {
            if(strcmp(output_ch,ch)==0)
            {
                is_output=1;
                break;/*若已经输出过相同的单词，则标记为 1，且不再继续查找*/
            }
            fscanf(output_fp,"%s",output_ch);
        }
        if(!is_output)
        {
            fprintf(output_fp,"%s\n",ch);
            printf("%s\n",ch);
        }
        is_output=0;
    }
fscanf(fp,"%s",ch);
}
if(is_finded==0)
```

```
            printf("Not find the right word");
        fclose(fp);
        fclose(output_fp);
}
```