

# 《高级语言程序设计A》

## (课程号：B0300311S；课时：48学时)

### 2025/26第一学期期末总复习

课程的全部内容可分为五大模块：

- (1) 基础知识
- (2) 三种基本结构及流程控制
- (3) 函数
- (4) 数据类型
- (5) 文件

下面按这五大模块分别作回顾

#### 模块一：基础知识部分

需要掌握的知识要点：

(1) 计算机的组成，重点是内存知识，内存容量及内存地址，每个存储单元为一个字节，按字节编址，一个字节由8个二进制位组成，内存容量取决于地址总线数量，**n根地址总线最大存储容量为 $2^n$ B。**

(2) 计算机是电子设备，计算机中的所有信息都是以二进制编码的形式存储在机器内部相应的存储空间中。进制间的相互转换：

- a) N进制数转成十进制数：将各位数符所代表的值乘以对应位的位权再累加求和，则得到对应的十进制数结果，该方法简称为“按权求和”。
- b) 十进制数转成N进制数：除以N取余至商为零再逆序输出所有余数，简称为“除N取余法”。
- c) 二进制数与八进制数的相互转换：二到八，3合1；八到二，1分3。
- d) 二进制数与十六进制数的相互转换：二到十六，4合1；十六到二，1分4。
- e) 八进制数与十六进制数的相互转换：通过二进制数作为中间数进行转换。

(3) 从C的源程序到可执行的文件的全过程：**编辑**（生成扩展名为.c的源文件）、**编译**（生成扩展名为.obj的目标文件）、**链接**（生成扩展名为.exe的可执行文件）最后才能运行。

(4) C语言程序的组成：由一个或多个文件组成，每个文件由一个或多个函数组成，函数是C程序的基本单位，一个程序有且只有一个main()函数，这是程

序的入口函数；每个函数的组成：函数首部 and 函数体，函数中可有注释行/\* ... \*/

(5) 编译预处理有三种：文件包含、宏定义、条件编译；预处理命令行都必须以“#”号开始,最后不加分号

无参宏定义#define 宏名 串

带参数的宏可以完成一些简单的功能，例如：

#define MUL(x,y) x\*y  $3+2 \times 4+5 = 16$

★ 主函数中c= MUL(3+2,4+5)，则变量c的值为？正确进行宏替换（技巧：对实参表达式不事先进行计算，直接代替参数位置）

区分：#define MUL(x,y) (x)\*(y)

主函数中c= MUL(3+2,4+5)，则变量c的值为？ $6822 \times 445 = 45$

(6) C程序中的六种符号：关键字、标识符、运算符、分隔符、其它符号、数据（包括常量和变量）

(7) 熟记所有常见关键字，关键字中最多的是流程控制语句中的关键字，如：for、while、do、if、else、switch、case、default、break、continue、goto、return等，还有一大类是变量类型及存储类别的，如：void、int、short、long、signed、unsigned、char、double、float、enum、union、struct、typedef、static、register、auto、extern,还有一些如：sizeof等；

(8) 能正确定义标识符并会判断合法的标识符（在程序中，符号常量名、变量名、函数名、类型名都是用户自定义标识符），自定义标识符的正确与否可从以下几方面判断（也可以认为是判断的三个步骤）：①是否以字母或下划线开头；②是否是只有字母、数字、下划线的序列③是否选用了关键字

(9) 掌握所有的运算符及其优先级、结合方向及特殊的运算规则：  
34种运算符的优先级共15个级别，从高到低依次为：[]、()、->、.---单目运算符--->乘除类算术运算符--->加减类算术运算符--->移位运算符--->有大小关系的关系运算符--->等与不等的关系运算符--->位运算符--->逻辑与--->逻辑或--->条件运算符--->各种赋值及复合赋值运算符--->逗号运算符。

结合方向：单目运算符、条件运算符、赋值运算符这3种为自右向左结合，其它均为自左向右结合

能改变变量值的运算符：赋值、++、--，注意前后++、--的区别

具有特殊运算规则的运算符：

①逻辑与&&（短路功能，第一运算对象为假则第二对象不运算）；

②逻辑或||（短路功能，第一运算对象为真则第二对象不运算）；

③条件运算符（总是先运算第一运算对象，根据其真或假的值从第二或第三对象中取一个作为整个表达式的值），常常可以代替一些简单的双分支if,例如：if (x>y) z=x; else z=y; 可以用条件表达式写成：z=(x>y)?x:y;

④%运算符要求两个运算对象均为int型，余数的符号与被除数相同。

/运算符包含整数除和实数除，整数和整数的结果还是整数，实型和整数的结果是实型。

⑤注意两个运算符的区别：=（赋值）与从==（相等）

⑥逗号表达式的结果是最后一个子表达式的结果；关系表达式的值是逻辑值。

注意表达式的值和变量的值是两个概念，很多情况下不是一回事儿。

例如：int a=1,b; 表达式b=a++使得变量a的值最终为2，b的值最终为1，该赋值表达式的值为1，因为赋值表达式是以最后一个被赋值变量的值作为整个表达式的结果的；

但是如果以上表达式改为：b=++a则变量a、变量b以及赋值表达式的结果均为2，请大家一定要注意理解，做题时看清题目问你表达式的值还是某个变量的值，注意区分。

这一部分要多刷题，熟悉各种运算符的使用，两本教材和慕课上的题目每题都要理解会做。

(10) 常量是其值不可变化的量，有两种表现形式：字面常量和符号常量，C是定义符号常量只有一种方法：#define 常量名 串  
C中的字面常量有五种：整型常量、实型常量、字符常量、字符串常量、枚举常量（不作要求）

① 整型常量有三种合法进制表达：十进制数、八进制数（如012，以0为前缀）、十六进制数(如0xa3，以0x或0X为前缀)

注意：068不是正确的整型数，0开头表示是用八进制表示，但是数字8不是八进制的合理数符

② 实型常量只有十进制数，有两种表示形式：小数形式、指数形式，注意：小数形式要求一定要有小数点，小数点前面或后面的数字可以缺省；指数形式一定要有E（或e），并且在其左边一定是合法的十进制整数或小数形式的实数，右边必须是合法的十进制整数，E左右两边表达都不可以缺省。

③ 字符常量是以一对单引号括起的单个字符，如果以双引号作为定界符则一定不是正确的字符常量。需要特别注意的是以右\开头的转义字符的三种形式：'\n'（右斜杠开头后跟一个特定字母）、'\121'（右斜杠开头后跟不超过3位八进制数所表示的ASCII码）、'\xa5'（右斜杠开头后跟x或X，再跟不超过2位十六进制数所表示的ASCII码），记住'A'、'a'、'0'、'\n'的ASCII码分别为：65、97、48、10，字符常量的大小就是其ASCII码值的大小

小写转大写：ch=ch-32；大写转小写：ch=ch+32；

④ 字符串常量是以一对双“”括起的0个或多个字符，需要特别注意字符串的长度指双引号中实际字符的个数；而字符串所占的字节数为字符串长度+1，有一个'\0'作为字符串结尾的标志需要占用1字节空间

(11) 变量：在程序运行过程中其值可变的量，需要先定义后使用。

每一个变量都具有：变量名、变量的当前值、变量的地址、变量的存储属性。

指针变量中存的是地址，其余变量中存放的是值。

变量的输入和输出控制需要熟练掌握，在scanf的格式控制串中出现的所有内容都需要运行时从键盘读入，例如：对应于输入语句scanf(“x=%d”,&x); 从键盘必须输入：x=2回车才能让变量x获得值2

在所有函数之外定义的变量称为全局变量，其作用域为定义点开始到程序结束（去掉含同名局部量的作用区域），如果未做初始化，则默认初值自动为0；

在函数内部定义的变量称为局部变量（形式参数也是局部变量），局部变量分auto（默认）和static两种，前者是在函数被调用时分配空间，每次函数执行完毕后将不再存在，但是static型的局部变量在程序刚运行时便分配空间并且做初始化（如果未指定初值则默认初值为0），初始化只做一次，每次函数执行完毕后将继续存在，但是只在本函数中才起作用，因此这种变量的读程序题一定要

注意！练习：

```
#include <stdio.h>
int cal(int n)
{
    static int f=20;
    f+=n;
    return f;
}
int main ()
{
    int i;
    for(i=1;i<7;i++)
    if (i%2) printf("%d ",cal(i));
    return 0;
}
```

答案：21 24 29

在做读题时要区分各种变量，哪些是全局变量，哪些是局部变量，是自动局部变量还是静态局部变量，全局变量有没有初始化，有没有同名局部量等问题，确定好每一种变量的生命期与作用域，从而保证能得到正确结果。

## 模块二：三种基本结构及流程控制

C程序的三种基本结构是顺序结构、选择结构、循环结构。这三种结构分别需要借助于特定的语句控制实现。

(1) 顺序结构：赋值语句和函数调用语句是控制该结构的主要语句。函数调用语句中最常用的是：scanf()、printf()、getchar()、putchar()等用于控制输入输出的语句。注意用scanf()输入变量值时要严格按照格式控制串的形式要求输入，用printf()输出时要进行相应的格式控制，需要熟记几种格式转换说明符：%c、%s、%d、%f、%lf（当变量为double类型时，在scanf中必须用%lf，但输出时用%f和%lf都可以）。

注意%s的特殊性，无论scanf还是printf中，与之对应的都是地址，其他格式控制串，如：%d、%c、%lf、%f等，在scanf中用地址对应，但是在printf中用值对应

### (2) 选择结构：

用if...else、switch语句配合复合语句、break语句控制实现。

if语句有三种形式：单分支if、双分支if、嵌套if，else应与其前最靠近的未匹配过的if相对应，而与对齐的形式无关，每一个else实际上都隐含了一个条件，要充分利用这些条件分析程序；if后的条件通常是关系或逻辑表达式，但是实际语法上可以是任何表达式，仅以该表达式计算的值是0或非0来判断条件的假或真。注意：if()及else后面都只能控制一条语句，经常需要用到复合语句；注意不要误用空语句导致语法错。

注意：if(!x) ... 此处!x相当于x==0

if(x) ... 此处x相当于x!=0

注意：if(x=y)和if(x==y)是完全不同的表达

注意：if(x>y); z=x; else z=y;存在语法错误，原因就是if后多打了分号，使得else无匹配项。

例如：int x=1,y=2,z=3;

语句if (x=y) z=10; else z=20;执行后，z的值为10  
而语句if (x==y) z=10; else z=20;执行后，z的值为20

if (x%2) 就是 if (x%2!=0) 也就是 if (x%2==1) 代表x是一个奇数  
if (!(x%2)) 就是 if (x%2==0), 代表x是一个偶数

在阅读和书写程序时注意if后条件的表达，每个分支所控制的语句。  
特别是注意理解嵌套的if，其中每一个else总是与其前最靠近的且未与其它else  
匹配过的if相对应，注意每个if或else分别对应于什么情况的处理。

switch语句中特别要注意的是：

- ①switch后面的表达式类型应当为整型、字符型、枚举型，而不可以是float或double型，因为表达式的取值必须是离散值；case后面必须是常量
- ②找到与表达式匹配的常量后开始执行，直到遇到break或switch语句结束才会停止。因此每一个分支后如果没有及时用break，语法上是正确的，会继续向下执行，这是**读程题考试的一个重要考点**。
- ③default分支是可选项，可以有，也可以没有，根据实际编程需要而定
- ④嵌套的switch在阅读的时候要注意每一层的起止位置，break的作用是结束本层的switch语句的执行。

(3) 循环结构：常用while、do...while、for三种流程控制语句实现，注意while和for属于当型循环，条件满足才进入循环体，因此循环体可能一次都不执行；do...while属于直到型循环，先执行循环体再判断条件，若while后的条件为非0则继续循环，若为0则结束，因此循环体至少执行一次。三种循环是可以相互转化的。这里特别强调的是：循环的退出第一种情况是循环控制条件为0；如果表示条件的表达式值永远不可能为0，则该循环为死循环，这是编程中需要避免的。

循环条件：for语句的表达式2，如果表达式2缺省，则表示永真条件；while或do ... while后的表达式表示条件。

循环退出的第二种情况是：循环体内满足某条件执行break语句退出本层循环。在循环体中可以用continue表示忽略本次循环体中的后续语句，重新开始下一次循环条件的判断，以便决定是否要继续循环。

**循环内有break语句和continue语句的程序阅读题反复刷题。**

掌握循环体执行的次数的计算及循环之后相关变量的结果。

对于循环嵌套，理解每一层循环的循环体内容，理解两重循环的控制及执行方式。

循环体只有一条语句，注意复合语句的正确使用。

特别注意循环体是不是空语句，例如：for (i=0;i<3;i++); s+=i; 这里循环体为空语句，语句s+=i;在循环终止后只执行一次，是普通的赋值语句。

某些情况下循环体就是什么也不需要，这个时候就一定要使用空语句。

注意：程序中有关退出的几个操作：

- ①在循环结构中用break结束本层循环、在switch语句中用break结束本层switch中当前分支的执行并退出switch语句；
- ②在函数体内的return语句结束本函数的调用，返回到调用点；
- ③库函数exit结束程序的执行，返回到操作系统，该函数在<stdlib.h>头文件中

练习：

1、若已定义int i，则对于如下循环语句，循环体执行的次数分别为\_\_\_\_\_。

①for( i=0 ; ; i++)

```

    printf("%d ",i); //次数：无穷
②for( ; i=0 ; i++)
    printf("%d ",i); //次数：零
③for( i=0 ; i<0 ; i++)
    printf("%d ",i); //次数：零
④ i=0;
while( i=1)
    i++; //次数：无穷

```

2、下列程序运行后，x和y的值分别是？

```

#include<stdio.h>
int main()
{
    int x=1,y=2;
    for(;x<10;x++)
    {
        x+= 2;
        if ( x>7 )
        {
            printf("x=%d y=%d\n",x,y);
            break;
        }
        if ( x==6 ) continue;
        y *= x;
    }
    printf("x=%d,y=%d\n",x,y);
    return 0;
}

```

答案：x=9,y=6

这一部分请大家要多做第四章的主教材和补充习题，阅读程序写答案及选择题部分，多练！

这一模块中有很多经典算法，下面几个是同学们必须非常熟悉的：

#### (1) 质数问题

① 判断一个整数是否为质数（用一层循环可以解决）

教材中的例子请完全掌握，结合第5章也必须掌握定义函数来实现，函数原型：int prime(int n); 判断参数n是否为质数，如果是，返回1，如果不是，返回0。

② 找出一定范围内所有的质数（用两层循环可以解决），结合实验题目看

③ 能按一定的格式输出

#### (2) 求和问题：

掌握求和问题中条件的控制方式、累加的方式，每一项的特点：

① 可以根据求和的项数要求来求解，例如：求100项之和；

② 可以根据每一项的精度进行求解，例如：求到这一项的绝对值小于1E-6为止，注意这时的循环控制为：

```

do
{
    求当前项的语句；
    将当前项累加到和里的语句；
    根据求和要求改变分子、分母、符号位的语句；
}while(fabs(term)>=1E-6);

```

③ 每一项实际上都可以表达成： $1.0 * \text{分子} * \text{符号位} / \text{分母}$ ，（假设分子分母都是int型）有的求和不存在正负项交替，则不需要考虑符号位；有的分子或分母是某一个常量，则上式更简化。求和需要设一个累加器首先清零，然后在循环体内加上表示项的那个变量。具体参考教材中的求和示例以及实验1中数列求和问题。

④ 典型求和案例：

$$s = \left\{ 1, -\frac{1}{3}, \frac{1}{5}, \dots, \frac{(-1)^{i+1}}{2i-1} \dots \right\} \quad i = 1, 2, \dots$$

教材例4.8

再举几个求和示例：重点分析当前项item的表达

例1： $s = 1 - 2/3 + 3/5 - 4/7 + \dots + i/2^i - 1$  该数列与例4.8非常类似，区别在于分子，所以这题的 $\text{item} = \text{sign} * i / (2.0 * i - 1)$

例2： $s = 1 + 2 + 3 + \dots + N$  这是例4.6的求和问题，本题分母恒为1，无正负交替问题，只有分子，分子正好就是当前项，因此 $\text{item} = i$ ，所以在编程时不需要定义item变量，直接用 $\text{sum} += i$ 表示加当前项

例3： $s = 1 - 3/5 + 5/7 - 7/9 + \dots + (2*i-1) / (2*i+1)$ ，这个数列与本例接近，只是分子分母与i之间的函数关系有变化， $\text{item} = \text{sign} * (2.0*i-1) / (2.0*i+1)$

例4： $s = 1 + 1/2! + 2/3! + 3/4! + \dots$ ，这个数列没有正负项交替，所以不需要sign变量，设f存放阶乘，初始化为1.0，充分利用相邻阶乘之间的关系，因此 $f * i; \text{item} = (i-1)/f$ ，和值sum初值为1.0，将第一项1直接作为sum初值，循环从i=2开始

例5： $s = 1/2 - 2/3 + 3/5 - 5/8 + \dots$ ，这个数列很有意思，分子分母分别是一个斐波那契数列，有正负项交替。用a,b分别代表分子分母，初值分别是1,2，于是 $\text{item} = \text{sign} * 1.0 * a / b$ 且 $b = a + b; a = b - a$ ， $\text{sign} = -\text{sign}$

通过以上几个例子，对数列求和问题既要掌握通用方法，又要会根据实际的需要正确表达每一项的三个要素：正负号、分子、分母。

(3) 符合一定特征的数据寻找问题，例如水仙花数和完数、完全平方数问题。  
水仙花数：一个三位数如果恰好等于它每一位数字的立方和，这个数就叫做“水仙花数”。例如， $153 = 1^3 + 5^3 + 3^3$ ，因此153是“水仙花数”。

求解思路：将三位数字分离出来：个位： $a = n \% 10$ ；十位： $b = n / 10 \% 10$ ；或： $b = n \% 100 / 10$ ；百位： $c = n / 100$ ；然后符合 $a^3 + b^3 + c^3 = n$ 的数就是水仙花数

完数：一个数如果恰好等于它的因子（不包含自身）之和，这个数就叫做“完数”。例如，6的因子为1、2、3，而 $6 = 1 + 2 + 3$ ，因此6是“完数”

完全平方数：完全平方数是一个可以表示为某个整数的平方的数。 $9 = 3^2$ ，可以使用条件运算符来判别， $n = (\text{int}) \sqrt{n} * (\text{int}) \sqrt{n}$ 。

(4) 最大公约数和最小公倍数问题，第四章课后读程题有，理解辗转相除法。

(5) 整数的逆置问题，例如将 $n = 123$ 转变成321。方法：设一个累加器变量存转变后的结果，初始化为0，然后用循环，每次首先将个位数求出，然后对n降阶，再将累加器\*10+个位数，下面的代码供参考：

```
int s = 0, r;
while (n)
{
    r = n % 10;
    n /= 10;
    s = s * 10 + r;
}
循环终止时的s就是n的逆置整数
```

(6) 利用循环可以控制输入数据的范围。  
例如：实现输入的a,b范围是：10<=a<=b<=1000  
int a,b;  
do  
{ scanf("%d%d",&a,&b);  
} while (a<10||a>b||b>1000);//条件满足要重新输入a 和b  
这里while后的条件也可以表达成：  
while (!(a>=10 && b<=1000 && a<=b));

## 模块三：函数

C语言的源程序由一个或多个函数组成，其中必须有且只能有一个主函数main()，程序执行总是从main函数开始的，也是因main函数的结束而结束整个程序。

函数只可以嵌套调用，而不能嵌套定义（即：一个函数的函数体内不可以定义另一个函数）

这一部分需要搞清楚以下问题：

(1) 函数定义：包括函数首部和函数体。

函数首部又包括了函数返回值类型 函数名（形式参数表），如果返回值类型缺省，则默认为int型；函数的形式参数名字无关紧要，重要的是形参的变量类型，每一个形参都是自动局部变量，在形参表中每个形参前必须有类型标识

若函数无需返回值，则将返回值类型可以定义为void，函数体内一般不需要return语句。

函数返回值类型不是void类型时，函数体内一定要有return语句。

函数的返回值由首部的函数返回值类型决定，return后的表达式类型若与函数返回值类型不一致，则自动转换，将return后表达式的类型转成函数返回值类型。

函数通过return只能返回一个值，如果一个函数需要返回多个信息，则可以考虑借助于指针形式参数实现（参考例7.8）。

例如：int f(int a[],int n,int \*max); 由函数返回数组a的前n个元素之和，而元素的最大值通过设定max指针，在函数体内求\*max实现，调用时，要用一个实参变量的地址对应于指针形式参数max。

函数体是由一对大括号括起的语句系列组成，如果函数返回值为void，则可以没有return语句

(2) 函数原型声明：函数应当先定义后被调用，如果出现调用先于定义的情况，则必须对后定义的函数在调用点之前进行原型声明，函数原型声明就是函数定义的首部最后加一个分号，在函数原型声明中形式参数名字可以缺省。

函数原型声明语句要求在调用点之前，声明位置有两种：可以在所有函数定义的最前面，这时该函数可被程序中所有函数调用；也可以在某个函数体内部，这时该函数未必能被所有函数调用。

(3) 函数调用：形式为：函数名（实际参数表），实际参数与形式参数对应的要求是：**个数相同、对应位置的类型最好一致；调用时实际参数表无需写数据类型**

如果函数的返回值类型为void，则只能作为独立的函数调用语句来调用，而不



能将调用结果作为表达式的运算对象使用；  
如果函数的返回值不是void类型，则可以作为独立的函数调用语句，更多情况下是作为运算对象来调用的

(4) 形式参数：有值形式参数和指针形式参数两种，值形式参数是实在参数将值赋值到值形式参数变量中，这叫做单向值传递；指针形式参数，这时将实参的地址赋值到指针形式参数中，这叫单向传地址。  
虽然都是单向传递，但是可以通过在被调函数中修改指针所指向的内容达到修改实参变量的目的。如：swap(int \*x,int \*y),在该函数体内交换的是\*x和\*y的值；对swap的几种正确与错误实现需要能正确理解（具体参考实验3的实验题2）。如果函数原型为：swap(int \*x,int y),在函数体内交换\*x和y的值，则主函数中的调用形式应该为：swap(&a,b);只有实参a可能会得到改变，无法完成a、b互换的目的。

(5) 变量的作用域问题：取决于变量的定义位置  
全局变量——在所有函数之外定义的变量，作用于定义点到程序结束处，但应去除掉同名局部量的作用域部分。

局部自动变量——在某函数内部定义的变量，有三个定义位置：形式参数表、函数体刚开始的位置、复合语句刚开始的位置。其作用域只在本函数内的当前语句块（即包含其定义的最近的一对大括号），生命期为：所在函数被调用时（或语句块开始时）分配空间，该函数当次结束时（或退出语句块时）空间不复存在。下次再调用则重新分配空间。在不同函数中定义的局部量可以同名，互不影响，因此形式参数可以与实在参数变量同名，因为处于不同的函数。

静态局部变量——在某函数内部定义的变量，有static关键字，其作用域还是只在本函数内，生命期为整个程序运行期间。本次函数调用结束后仍然存在，其值保留，但是在函数之外的其余部分均不可见，等该函数再次被调用时，该变量就在原来值的基础上继续变化。静态局部变量只初始化一次，如果不给定初始化的值，将自动初始化为0。

(6) 变量的生命期：取决于变量的存储类别：extern,auto(缺省)，static，register（一般很少使用了）。形参都只能是auto存储类别的局部变量  
extern通常用于多文件结构中，声明来自于另一个文件的全局变量或函数。

练习：多做书后及配套的补充习题，帮助题解

1、若用一维数组名作为函数调用时的实在参数，则实际上传递给形参的是\_\_\_\_\_。

**答案：数组首地址**

2、若在主函数中定义了数组变量：int arr[20]，并调用了函数f(arr),则下列关于函数f的原型声明，哪几个是完全等效的\_\_\_\_\_。

- A、void f(int x[20]);
- B、void f(int x[]);
- C、void f(int x[10]);
- D、void f(int x);
- E、void f(int \*x);

**答案：ABCE**

3、静态局部变量的读题要刷题练习

```
#include <stdio.h>
```

```
int fun1() {
```

```
    static int count = 0;
```

```

int local = 0;
count++;
local++;
return count + local;
}
int main() {
    printf("%d", fun1()); // 第一次调用
    printf("%d", fun1()); // 第二次调用
}

```

4、有函数原型为void f(int,int \*);主函数中有变量定义：int a, \*p=&a,n; 则下列几种调用正确的是\_\_\_\_\_？

A、f(a,p); B、f(\*p,&a); C、f(a,\*p); D、f(\*p,a);  
 E、f(a,&a); F、f(\*p,p); G、f(a,a); H、f(p,p); I、n=f(a,p)

**答案：ABEF**

**重点函数：判断一个整数是否质数、求最大公约数、判断一个数是否为水仙花数等**

## 模块四：数据类型

C语言源程序中所处理的任何数据（常量或变量）都是属于某一种数据类型的，数据类型将决定该类型的值在内存中占用空间的大小、数据值的表现形式及数据范围、可进行的运算及操作、变量值的组成情况等。

C中数据类型分为：基本类型、构造类型、指针类型、空类型四大类，其中基本类型包括整型、实型、字符型；构造类型有：枚举型、数组类型、结构体类型、共同体类型，指针类型的特殊性在于其变量中存放的是内存地址信息而不是内存中的值。

这一模块的重点是数组、指针类型、结构体、字符串（用数组和指针实现）。

### (1) 数组：

基本知识：

一维及二维数组的定义，注意定义数组时必须是在方括号是给整型常量或常量表达式，例：int a[4];（注意：此时元素的有效下标范围为0~3，特别注意：起始下标为0，不可以下标越界访问）；定义数组时元素个数不能用一个已有值的变量，例如：int n=10; int a[n];就是错误的定义。

一维数组名和二维数组名都是代表数组的首地址，可以理解为数组中第一个元素的起始地址，是一个地址常量，因此不能对数组名进行赋值、++、--这三种运算

一维及二维数组的初始化，当初始化时可以缺省元素个数，对于二维数组只能缺省行数。初始化遵循的规则是从左至右依次，且初值个数不能超过数组容量；当初值少于元素个数时其余元素的默认初值为0

访问元素有两种形式，结合指针的知识，有a[i]和\*(a+i)两种等效表示；元素的地址也有两种等效形式：a+i或&a[i]，这里的下标i处可以是常量、变量或表达式，范围不超过数组允许的范围就可以了

一维字符数组操作字符串、会正确输入输出数组的元素、正确输入输出字符串（例如：char str[20]="ABCDE"，则printf("%s\n",str)；的输出结果是ABCDE，与%s对应的是表示字符串的首地址，输出遇到0结尾标志就结束）。

掌握string.h函数中的一些常用函数的使用，如：

strlen求长度（有效字符个数，不包括结尾标记）

strcmp进行两个字符串的比较（例如：判断串s1是否大于s2，不能用s1>s2,而应该用strcmp(s1,s2)>0）

strcpy实现字符串的赋值、两个字符数组内容的交换

strcat实现字符串的连接，

会编程实现字符串的逆置、字符串中空格的删除、字符串中指定形式字符的抽取（例如只保留数字字符）、回文的判断等算法，掌握字符串处理函数的函数体实现。

必须熟练掌握在一维数组中的以下算法：

教材中基本都有对应的函数定义及调用方法：读入一维数组中的所有元素、输出所有元素、求和、求平均、求最大或最小元素、排序（冒泡或简单选择排序）、查找等

二维数组掌握一些最基本的方法，例如：按矩阵形式输出、转置、求主对角线或次对角线元素之和。

算法：首先要理解一维数组名作实参和形式参数传地址的实质；理解void f (int a[]) 中的形式参数a实质上是一个指针形式参数，也就是该函数首部可以写成：void f (int \*a) 或void f (int a[10]) 这里的10可以是任意整数都是完全等效的

练习：

1、若有说明：double a[5]; int k=4;，则对 a 数组元素的不合法引用是\_\_\_\_\_。

A. a[k-1]      B. a[0]      C. a[4]      D. a[5]

答案：D

2、以下一维数组的定义正确的是\_\_\_\_\_。

A、int n=3, a[n];    B、#define N=3    int a[N];  
C、const int n=3; int a[n];    D、#define N 3    int a[N\*3];

答案：D

3、以下一维数组定义及初始化正确的是\_\_\_\_\_

A、int a[ ];      B、int a[]={1,2,3}  
C、int a[3]={1,2,3,4}    D、int a[3]={,2,}

答案：B

4、以下选择中，能正确定义二维数组并初始化是\_\_\_\_\_。

A、int a[2][3] = { {1,2,3}, {4,5} };  
B、int a[][3] = { {1,2}, {3,4}, {5,6} };  
C、int a[2][] = { {1,2,3}, {4,5,6} };  
D、int a[2][3] = { {1,2}, {3,4,5}, {6} };  
答案：B

5、若定义int m[10]={9,1,15,6,2,11,8,7,19,3};则m[m[1]+2\*m[4]]的值是\_\_\_\_\_，\*m+m[8]的值为\_\_\_\_\_。答案：11 28

6、下列字符数组定义与初始化不正确的是\_\_\_\_\_答案：D

A、char s[ ]="abc";    B、char s[]={ "abc" };  
C、char s[3]={ 'a','b','c' };    D、char s[3]="abc";

7、下列关于数组的描述中不正确的是\_\_\_\_\_。答案：B

A、可以对字符型数组进行整体输入、输出  
B、可以对整型数组进行整体输入、输出  
C、利用字符型数组可以实现字符串操作  
D、不能通过赋值运算符“=”对字符型数组进行整体赋值

(2) 指针类型：定义形式为：数据类型 \*指针变量名，例：int \*p; 注意p与\*p的区别。指针变量若未赋值，不能对\*p操作，若对指针赋值为空值，也不能对\*p操作。该指针变量可赋值为普通int变量地址、一维数组名等

练习：

1、已知：char \*s = "student"; 则printf("%s\n", s+3)输出为\_\_\_\_\_

答案：dent

2、若有说明：int a, b=9, \*p=&a;，则能完成a=b赋值功能的语句是\_\_\_\_\_

A、a=\*p; B、\*p=\*b; C、a=&b; D、\*p=&\*b;

答案：B

3、设有定义：int \*p, y; int b[6] = {2, 4, 6, 8, 10, 12}; p = b + 2; 下列语句中，能使y的值为8的是\_\_\_\_\_。

A. y = \*(p + 2); B. y = \*p; C. p++; y = \*p; D. y = b[5];

答案：C

指针与数组的关系：对指针操作一维数组要熟练掌握，理解数组名是地址常量的概念，指针指向数组的时候执行p++、p--的意义，以及数组元素的下标法访问及指针运算符访问方法：例：int a[4]={12,23,3,47}, \*p=a; 执行语句 ++p; 后，p[2]或\*(p+2)都是表示元素a[3], 值为47；而p+1或a+2都是表示元素的地址，即&a[2]。

本例中不可以做a++或a--或a=a+2这些改变a的操作，因为数组名a是地址常量，不可以被改变。

区别\*p++和(\*p)++这两个表达式的不同含义

掌握元素访问的多种表达，例如：a[a[2]]、\*(p+(p+1))的值为47

指向同一数组空间中的指针可以做减法，表示二者之间相隔的元素个数

用字符指针操作字符串：比用字符数组操作字符串具有更大的灵活性，注意二者的区别

例：char \*s="ABC"; char \*s; s="ABC"; 都是正确的

char p[4]="ABC"; 正确，但是char p[4]; p="ABC"; 却错误

s=p; 是正确的赋值，但是s=\*p, \*s="AB"都是错误的

(3) 结构体类型：

会正确定义结构体类型，以及嵌套的结构体类型

掌握三种结构体变量定义的方式（先定义类型再定义变量；定义类型的同时定义变量，类型名可以缺省；用类型别名定义变量）；掌握结构体变量的初始化方式，对其成员依次赋初值。理解结构体变量内存占用情况（各成分依次存放，故结构体变量所占空间至少为所有成员需要的空间之和），会正确访问结构体变量的成员（点运算符的前面必须是结构体类型的变量；箭头运算符前面必须是结构体类型的地址或指针变量），会正确定义结构体数组，并掌握结构体数组元素各个成员的读入、访问方式；掌握用结构体指针正确访问结构体或结构体数组元素的成员。

会用typedef给出一个类型的别名：typedef 原类型名 新类型名；

练习：

1、定义struct Point{ int x,y; }pos[] = {{1,2},{3,4},{5,6}}, \*pt=pos; 则表达式(++pt)->y的值为\_\_\_\_\_, ++(pt->x) 的值为\_\_\_\_\_, ++pt->x 的值为\_\_\_\_\_, (\*pt).y 的值为\_\_\_\_\_。

答案：4 4 5 4

2、以下对结构体变量stu中成员的非法引用是\_\_\_\_\_。

```
struct Point
{
    int x;
    int y;
} pos, *p=&pos;
```

A、pos.x    B、(\*p).y    C、p->x    D、Point.y

答案：D

作为函数参数，通常定义结构体的指针作为形式参数，将实参结构体变量的地址传入，这样省时效率高，而且可在被调函数中修改对应实参结构体成员的值

练习：

下列程序的运行结果是？

```
#include <stdio.h>
typedef struct Student
{
    char name[10];
    double score;
} STU;
```

//注意：这里STU是结构体类型名，如果前面去掉typedef则为结构体变量名

```
void f (STU *s)
{
    (*s).score*=20;
    printf("%s %.1f\n", s->name, (*s).score);
}

int main()
{
    STU stu[]={{"LiuBo", 4.5}, {"JiWei", 4.6}, {"Lijie", 5.0}};
    f (stu+1);
    printf("%s %.1f\n", stu->name, (*stu).score);
    return 0;
}
```

答案： JiWei 92.0

LiuBo 4.5

## 模块五：文件

C语言中的文件的组成及分类：

C中的文件由数据流形式组成，可按数据的存放形式分为二进制文件和文本文件

C文件打开的方式：

基本方式：r(只读) w(只写)、a(追加)

后面可以加上+表示可读可写（注意，如果打开一个新文件可读可写，则一定是w+；如果打开一个旧文件可读可写，则可能是r+或a+）、加上b表示打开二进制文件，否则默认是对文本文件操作

注意：

文件的操作步骤：

- (1) 定义FILE\*文件指针，一个文件对应一个指针
- (2) 利用文件指针打开文件fp=fopen(文件名, 打开方式)，注意文件名要用双引号括起，并且路径之间用\\，打开方式也要用双引号括起
- (3) 进行文件读写操作，共四对函数：

单字符读写：fgetc、fputc（重点掌握）

字符串读写：fgets、fputs

格式化读写：fscanf、fprintf

块数据读写：fread、fwrite //均有4个参数

一般用c=fgetc(fp)来从已有文件中获取一个字符，用fputc(c,fp)向fp所对应的文件中写入一个字符，该操作往往要用循环条件控制执行。

用fwrite和fread实现数据块的读写，这在内存变量为结构体类型时特别有用，

(4) 文件操作完成后要用fclose(fp)关闭文件。

(5) 判断一个旧文件是否结束，常用的有两种方式：

feof(fp)或((c=fgetc(fp))!=EOF)

重点掌握:

教材中的例11.1 (fputc) ,11.2(fgetc) ,11.4(fscanf/fprintf)这几个例子。

期末复习要点：掌握教材的例题、书后习题；慕课上的单元测验、编程练习、网上期末测试题要再次看看做做，确保掌握；四次实验的编程题再编写一遍；适当做补充习题或你们自己找来的题目，题目关键是掌握方法。

□

附：期末考卷题型、分值、主要知识点分布

题型	题量	考查章	主要知识点
一、单选题	20 ( 1 分 *20=20)	1--11	信息的存储及表示、进制转换、用户自定义标识符、常量、变量定义及使用、运算符的优先级、表达式的计算、各种控制语句、函数定义及调用、变量生命期与作用域、数组定义及初始化、元素的访问、指针的定义及运算、指针访问数组、指针或字符数组管理字符串、字符串处理函数应用、strlen与sizeof用于字符串的区别、结构体类型的定义及变量及其成员的访问、文件打开方式、带参宏定义及求解、文件包含
二、填空题	10空 ( 1 分 *10=10)	1--11	
三、读程题	5题 ( 5 分 *5=25)	4--10	if、switch、循环、break、continue、函数调用、静态局部变量、全局变量、局部变量、一维数组、一级指针、字符串、带参宏定义、结构体
四、程序填空	3 题 10 空 4--11		if、循环及嵌套、函数定义及调用、数组定

题	( 2 分 *10=20)	义及元素求解和访问、数组排序、字符串、 结构体、文件读写
五、编程1	1 (10分) 4---5	流程控制、函数定义及调用
六、编程2	1 (15分) 4---7	数组的常见操作、流程控制、函数