C语言课件知识点整理

第一章:编程的基本概念

一.什么是程序

```
为了让计算机执行某些操作或解决某个问题而编写的一系列有序指令集合
之后整个程序案例均在Visual Studio2019开发工具中完成
```

二.C语言及特点

```
C语言是贝尔实验室的Dennis Ritchie在1973年设计的

结构化的程序设计语言:层次清晰,便于按模块化方式组织程序,易于调试和维护

语句简洁:学习时入门相对容易,C语言很好地总结了其他语言提出的程序库概念

功能强大:即可用于系统软件的开发,也适合于应用软件的开发

移植性好:只要对这种语言稍加修改,便可以适应不同型号机器或者各类操作系统

特点总结:
    1.简洁紧凑,灵活方便
    2.运算符丰富
    3.数据结构丰富
    4.C语言是结构式语言
    5.程序设计自由度大
    6.可以对硬件进行操作
```

三.C语言中的注释

```
注释:介绍说明用的 可以理解为标注
//单行注释:只能注释掉双斜杠后的当前行,被注释的内容不会被编译执行
/*多行注释*/:只要写在/*开头*/结尾的中间部分都可以被注释掉
```

四.程序的基本结构

```c
#include<stdio.h> //以#开头的语句称为预处理器指令
//以.h结尾的文件称为头文件,可以是C程序中现成的标准库也可以是自定义的库文件
//stdio.h文件包含了有关输入输出语句的函数
int main() {
    //main()函数是C程序处理的起点,程序的入口
    //大括号表示函数的主体用来包裹语句块
    printf("Hello World\n");
    //printf在屏幕上输出并换行
    return 0;
}
```

五.C语言文件扩展名

```
C语言源程序以.c为扩展名保存
经过编译后编译器输出的目标文件常见扩展名为.o或.obj
连接器输出可执行文件的扩展名为.exe
连接器连接过程中C语言函数库中的头文件扩展名为.h
```

六.C语言执行流程

```
                                        |-C语言函数库
执行流程为:C源程序-->编译-->目标文件-->连接-->可执行程序
```

第二章:数据类型\常量\变量

一.数据类型(基本数据类型)

1. 数值类型

```
整型:     短整型 short   整型 int  长整形 long
浮点型:   单精度型 float  双精度型 double
```

2. 非数值类型

```
字符型:    char
```

3. 所占的字节数:(1个字节8位)

```
short:2  int:4  long:8  float:4  double:8  char:1
   16      32     64      32        64       8
```

4. 所能表示的范围:

```
short:(-32768~32767)
int:(-2147483648~2147483647)
long:(-9223372036854775808~-9223372036854775807)
float:(1.2E-38~3.4E+38 6~7位小数)
double:(2.3E-308~1.7E308 15~16位小数)
char:(-128~127)
```

5. 基本数据类型所使用的占位符:

```
short int long  ->%d
float           ->%f
double          ->%lf
char            ->%c
十六进制         ->%x
八进制           ->%o
字符串           ->%s
```

6. 类型转换

```
自动类型转换 short int long float double
强制类型转换 (类型)表达式
```

7. 转义字符表及ASCII码表

转义序列	名称	描述
\a	警告	产生一则警告。
\b	退格	将光标回退一格。
\f	换页	将光标移到下一页的第一格。
\n	换行	将光标移到下一行的第一格。
\r	回车	将光标移到当前行的第一格。
\t	水平制表	将光标移到下一个水平制表位置。
\v	垂直制表	将光标移到下一个垂直制表位置。
\'	单引号	产生一个单引号。
\"	双引号	产生一个双引号。
\?	问号	产生一个问号。
\\	反斜线	产生一条反斜线。
\0	空	产生一个空字符。

函数不能return array！！！
% mod (signed/unsigned integer only!!)
逗号运算符取右边
注意bitwise operations
注意break！
char*t字符串常量与字符数组
pointer decay
struct相互赋值要进行复制
enum赋值递增（配合switch）
改变stdin&stdout
string.h!!!!
strcmp返回值，一样就是0
strstr
文件操作！！！！！

值	符号	值	符号	值	符号
0	**空字符**	44	,	91	[
32	**空格**	45	-	92	\
33	!	46	.	93	]
34	"	47	/	94	^
35	#	48 ~ 57	0 ~ 9	95	_
36	$	58	:	96	`
37	%	59	;	97 ~ 122	a ~ z
38	&	60	<	123	{
39	'	61	=	124	|
40	(	62	>	125	}
41	)	63	?	126	~
42	*	64	@	127	DEL (Delete**键**)
43	+	65 ~ 90	A ~ Z		

8. 代码演示

```c
#include<stdio.h>
int main() {
    short s = 1;
    int i = 2;
    long l = 3;
    char c = 'a';
    printf("%d,%d,%d,%c\n", s, i, l,c);
    i = 2200000000;//选用数据类型时 注意是否超出范围
    printf("%d\n", i); //如果超出范围则输出错误的值

    int c11, d; //--> int c11; int d;
    int e = 1, f = 2;//-->int e = 1; int f = 2;

    float ff = 3.14f; //浮点数类型后面加f
    double dd = 3.14;
```

二.常量

1.
```
**概念:**固定不变的值,分为直接常量和符号常量
```

2.
```
**直接常量:**使用具体数据表达的一种形式,直接常量又分为
    整型常量,浮点型常量,字符常量字符串常量如1 2 3 4  a b c d等

**符号常量:**可以使用一个标识符代表一个常量
    宏定义形式:`#define PI 3.1415926`
    const关键字:`const float PI=3.14`
```

3. 代码演示

```c
#include<stdio.h>
//定义常量
#define PI 3.14  //宏定义形式
const double pai = 3.14; //const关键字形式
int main() {
    printf("常量输出:%f\n",PI);
    printf("常量输出:%f\n", pai);
    //pai = 3.1415; //常量的值不允许修改
    //PI = 3.1415;  //常量的值不允许修改
    return 0;
}
```

三.变量

1. 概念:

```
可以改变的值,实质是内存的别名,
想要知道内存中存放变量的地址可以在变量名前加&符号
```

2. 命名规则:

```
变量名可以由字母,数字,下划线组合而成必须以字母或下划线开头
C语言中的某些词(main include等)成为保留字,具有特殊意义,不能用作变量名
C语言区分大小写price和PRICE是两个不同的变量
```

3. 代码演示

```
 案例1
```

```c
#include<stdio.h>
int main() { //定义在函数体当中的变量为局部变量 只能在当前函数体内使用
    //定义变量
    int a;
    a = 10;
    printf("变量a输出:%d\n", a);
    int b = 20;
    printf("变量b输出:%d\n", b);
    b = 40;
    printf("变量b修改后输出:%d\n", b);
    int c, d;
    c = 3;
    d = 4;
    printf("变量c的值为:%d,变量d的值为%d\n", c, d);
    int e = 1,f = 2;
```

```
案例2
```

```c
#include<stdio.h>
int i; //定义在函数体外的变量为全局变量 可以在当前文件中使用
int main() { //定义在函数体当中的变量为局部变量 只能在当前函数体内使用
    //定义变量
    int a = 10;
    printf("变量a输出:%d\n", a);

    i = 30;
    printf("变量i输出:%d\n", i);
    return 0;
}
void eat() {
    //a = 11; //提示未定义访问不到
    i = 31; //不提示错误证明可以访问到
}
```

第三章: 运算符

一.运算符

1. 算数运算符

```
    + -  *  /  %  ++  --
    %:取模运算只能用于整数
    要想求浮点数的余数可以使用`#include<math.h>`中的`fmod(f,f2)`函数
```

2. 关系运算符

```
    ==(比较是否相等)  !=  >  <  >=  <=
```

3. 逻辑运算符

```
    &&  ||  !
```

4. 赋值运算符

```
    = (赋值)  +=   -=   *=  /=  %=
```

5. 位运算

```
&(按位与)    |(或)    ^(异或)    ~(非)    <<(左移)    >>(右移)
```

？？？？

6. 条件运算符(三元运算符)

```
表达式1 ? 表达式2 : 表达式3    ->    3>2?3:2
```

1. 代码演示

```
案例一:算数运算符
```

```c
#include<stdio.h>
    int main() {
        int A = 10, B = 5;
        //算数运算符的各种运算
        printf("A=10,B=5\n");
        printf("A+B=%d\n", A + B);
        printf("A-B=%d\n", A - B);
        printf("A*B=%d\n", A * B);
        printf("A/B=%d\n", A / B);
        //取模,表达式中书写了A%%B,其中两个%表示输出一个%
        printf("A%%B=%d\n", A % B);

        int a = 1, b = 2, c = 3, d = 0;
        printf("a,b,c的值为:%d,%d,%d\n", a, b, c);
        printf("a++的值为:%d\n", a++);
```

```
案列二:关系运算符
```

```c
#include <stdio.h>
int main()
{
    int a = 10;
    int b = 5;
    printf("a=%d,b=%d\n", a, b);
    printf("a>b:%d\n", a > b);
    printf("a<b:%d\n", a < b);
    printf("a>=b:%d\n", a >= b);
    printf("a<=b:%d\n", a <= b);
    printf("a==b:%d\n", a == b);
    printf("a!=b:%d\n", a != b);
    return 0;
}
```

```
案例三:逻辑运算符
```

```c
#include <stdio.h>
int main()
{
    int d = 1;
    int e = 2;
    int f = 3;
    printf("d > e && f < e%d\n",d > e && f < e);//false
    printf("e<f || d>f%d\n",e<f || d>f);//true
    printf("!(f > d)%d\n",!(f > d));//false
    return 0;
}
```

```
案例四:赋值及复合赋值运算符
```

```c
#include <stdio.h>
int main()
{
    int a = 10;
    int b = 20;
    int c = 0;
    c = a + b;
    printf("a+b结果为:%d\n",c);//30
    c += a;
    printf("a+=b结果为:%d\n", c);//40
    c -= a;
    printf("a-=b结果为:%d\n", c);//30
```

```
    c /= 1;
    printf("a/=b结果为:%d\n", c);//30
    b %= 2;
```

```
案例五:位运算符
```

```
/*
 * 位运算符
 *      &(按位与)       |(或)       ^(异或)  ~(非)    <<(左移) >>(右移)
 *
 *      二进制
 *          十进制转换二进制
 *
 *      13  二进制   1101
 *
 *      2048    1024    512   256   128  64  32   16   8  4  2  1 权次方
 *                                                    1  1  0  1
 *      154                         1    0   0    1   1  0  1  0
 *      685             1       0   1    0   1    0   1  1  0  1
 *      1250    1       0       0   1    1   1    0   0  0  1  0
 *      243                         1    1   1    1   0  0  1  1
```

```
案例六:三元运算符
```

```
#include<stdio.h>
int main() {
    int a = 10, b = 5,c = 0;
    c = a > b ? a : b;
    printf("c的值为:%d\n", c);//10
    return 0;
}
```

第四章:输入输出函数

一.输出函数:printf()

1. 格式化输出函数,一般用于向标准输出设备按规定格式输出信息 基本数据类型所使用的占位符(格式字符):

```
short int long  ->%d
float           ->%f
double          ->%lf
char            ->%c
十六进制          ->%x
八进制            ->%o
字符串            ->%s

例如:int a = 10; printf("%d",a);
```

2. 标志:

```
-:结果左对齐,右边填空格
+:输出符号(正负符号)
空格:输出值为正(包括0)时为空格,为负时为负号
设置宽度及精度  见案例
```

3. 代码演示:

```
案例一:
```

```
#include<stdio.h>
int main() {
    int a = 50;
    float b = 156.255f;
    printf("格式字符和标志字符的使用==================\n");
    printf("原样输出(%%d):        a=%d\n", a);
    printf("左对齐(%%-d):         a=%-d\n", a);
    printf("输出±(%%+d):     a=%+d\n", a);
    printf("留出符号位(%% d): a=% d\n", a);
    printf("八进制输出(%%#o): a=%#o\n", a);
    printf("十六进制输出(%%#x):   a=%#x\n", a);
    printf("浮点数输出(%%#f): a=%#f\n", b);
    printf("设置输出最小宽度=========================\n");
    int c = 20000;
    printf("有负号左对齐,右边留空:%-9d\n", c); //设置宽度为9
```

二.输入函数:scanf()

1. 格式化输入函数,它从标准输入设备(键盘)读取输入的信息scanf("%d",&num);

```
%d:转换字符串(与printf函数相同)
&:地址符号(附在读取的每个变量上)用于指明变量在内存中的位置 num:变量的名称
```

2. 注意事项:

```
1.scanf函数中没有精度控制
2.scanf函数中要求给出变量地址,如果给出变量名会出错
3.scanf函数在输入多个数值时,若格式控制串中没有非格式字符作
输入数据之间的间隔则可用空格.tab或回车作间隔
4.多个scanf函数连续给多个字符变量输入时的问题
```

gets （\n,EOF）

3. 代码演示

```
案例一:
```

```
#include<stdio.h>
int main() {
    //在每个最后不带\n的printf后面加fflush(stdout);
    //在每个不想受接收缓冲区旧内容影响的scanf前面加rewind(stdin);
    //连续输入字符例子会和上一个影响所以把这个例子移动到上面来测试
    char c1, c2;
    printf("连续输入字符:\n");
    scanf_s("%c", &c1, 1);   //多一个参数表示最多读取多少位字符
    rewind(stdin);
    //fflush(stdin);              //清空输入缓冲  在这里暂时没用
    scanf_s("%c", &c2, 1);
    printf("您输入的值为:c1=%c,c2=%c\n", c1, c2);

    //格式为scanf(格式化字符串,&变量名);    &变量名表示变量的地址
    int a, b;
```

三.字符输入函数: getchar ()

```
从键盘上读入一个字符后将得到的字符显示在屏幕上 char c = getchar();
注意:getchar()函数只能接受单个字符,输入数字也按字符处理
```

四.字符输出函数: putchar ()

```
像标准输出设备输出一个字符 putchar(ch);
ch为一个字符变量或常量
putchar函数的作用等同于printf("%c",ch);
```

- 案例一:

```
#include<stdio.h>
int main() {
    printf("使用getchar函数==============================\n");
    char ch;
    printf("请输入一个字符:");
    ch = getchar(); //从键盘读入字符直道回车结束  使用本函数时必须包含stdio.h头文件
    printf("你输入的字符为:");
    printf("%c\n", ch); //显示输入的第一个字符

    printf("使用putchar函数==============================\n");
    char c;          //定义字符变量
    c = 'B';         //给字符变量赋值
    putchar(c);      //输出该字符
    putchar('\x42');//输出字母B
    putchar(0x42);  //直接用ASCII码值输出字母B
```

第五章:条件语句

一.条件语句

1. 单分支if语句

```
    如果...就...
    if(条件){条件成立后我们要做的事}
```

2. if-else语句

```
    如果...就...否则...
    if(条件){条件成立后我们要做的事}else{条件不成立我们要做的事}

需求:判断用户年龄      如果年龄满18我们就输出   欢迎光临   否则输出   谢绝入内
```

```c
#include<stdio.h>
    int main() {
        //需求:判断用户年龄
        //如果年龄满18我们就输出   欢迎光临
        //否则输出   谢绝入内
        printf("请输入年龄:");       //给用户一个提示
        int age;
        scanf_s("%d", &age);      //获取用户在控制输入的值,并给age赋值
        if (age >= 18) {
            printf("欢迎光临");
        }
        else {
            printf("谢绝入内");
        }
        return 0;
```

3. if-esle if语句

```
    如果...就...否则   如果(可能有多个)...就 ...否则...
     if(条件1){条件1成立后我们要做的事}
     else if(条件2){条件2成立后我们要做的事}
            .可以有多个
     else(前面所有条件都不满足我们要做的事)
需求:判断春夏秋冬
```

```c
#include<stdio.h>
    int main() {
        //需求:判断春夏秋冬
        // 1 2 3春
        // 4 5 6夏
        // 7 8 9秋
        // 10 11 12冬
        // 要求 : 用户输入的形式
        // 至少三种方式实现
        printf("请输入月份:");
        int month;
        scanf_s("%d",&month);
        if (month <= 3) {
            printf("春天在这里");
        }
```

4. if嵌套

```
需求:如果分数大于60输出及格
      否则如果小于60输出挂科
      否则输出  谢天谢地
```

```c
#include<stdio.h>
int main() {
    //需求:如果分数大于60输出及格
    //否则如果小于60输出挂科
    //否则输出  谢天谢地
    int score;
    scanf_s("%d",&score);
    if (score > 60 && score <= 100) {
        printf("及格");
    }
    else if (score < 60 && score >= 0) {
        printf("挂科");
    }
    else {
        if (score > 100 || score < 0) {
```

5. switch case

```
switch (值) {//整型 字符 Enum枚举
    case 值:语句; break;
    ...
    ...
    ...
```

```
    default:语句; break;
}
需求:判断春夏秋冬
    1 2 3春
    4 5 6夏
    7 8 9秋
    10 11 12冬
```

```c
#include<stdio.h>
    int main() {
        printf("请输入月份:");
        int month;
        scanf_s("%d",&month);
        switch (month) {//整型 字符 Enum枚举
            case 1:
            case 2:
            case 3:printf("春"); break;
            case 4:
            case 5:
            case 6:printf("夏"); break;
            case 7:
            case 8:
            case 9:printf("秋"); break;
```

？？？？？

第六章:循环结构

一.循环

```
循环语句是由循环体及终止语句两部分组成,循环体就是要重复执行的操作
```

1. while循环

```
    while(循环条件){循环操作}
    特点:先判断,再执行
```

```c
#include<stdio.h>
    int main() {
        //需求:输出10次我要成功!
        int i = 0;
        while (i<10)
        {
            printf("我要成功!");
            i++;
        }
        return 0;
    }
```

2. do while循环

```
    do{循环操作}while(循环条件);
    特点:先执行一次,再判断
```

```c
#include<stdio.h>
    int main() {
        //需求:输出10次我要成功!
        int i = 0;
        do{
            printf("我要成功!");
            i++;
        } while (i < 10);
        return 0;
    }
```

3. for循环

```
for(1;2;3){4}
    1.初始值(从几开始循环)
    2.条件(boolean类型的结果)
    3.每次循环对初始值的改变
    4.循环体(重复要做的事情)
    执行顺序:1 2 4 3 2 4 3 2 4 3 ...直到2条件不成立   循环结束
```

```c
#include<stdio.h>
    int main() {
        //需求:输出1~100的偶数
```

```
        for (int i = 1; i <= 100; i++) {
            if (i % 2 == 0) {//打出偶数个
                printf("当前i的值为:%d\n" + i);
            }
        }
        return 0;
    }
```

4. 案例练习

案例一:计算100以内数字之和

```
#include<stdio.h>
int main() {
    //需求1:计算1+2+3+4+...100的结果
    int x = 0;
    for (int i = 1; i <= 100; i++) {
        x = x + i;//x+=i;
    }
    printf("%d",x);
    return 0;
}
```

案例二:找出100到999之间所有的水仙花数

```
#include<stdio.h>
int main() {
    //需求2:找出100到999之间所有的水仙花数(153 = 1*1*1 + 5*5*5 +3*3*3)
    int g,s,b;
    for (int i = 100; i < 1000; i++) {
        g = i % 10;
        s = i % 100 / 10;
        b = i / 100;
        if (i == g * g * g + s * s * s + b * b * b) {
            printf("%d\n",i);
        }
    }
    return 0;
}
```

案例三:打印99乘法表

```
#include<stdio.h>
int main() {
    //需求:打印99乘法表
    for (int y = 1; y <= 9; y++) {
        for (int x = 1; x <= y; x++) {
            printf("%d*%d=%d ",x,y,x * y); //%2d表示固定输出2位
        }
        printf("\n"); //打印到行尾换行
    }
    return 0;
}
```

案例四:循环输入5个学生的成绩　计算5个学生的总分及平均分

```
#include<stdio.h>
int main() {
    //需求:循环输入5个学生的成绩　计算5个学生的总分及平均分
    int sum = 0;
    for (int i = 1; i <= 5; i++) {                  //5个学生成绩的循环输入
        printf("请输入第%d个学生的成绩:\n",i);
        double score;                               //获取输入的成绩
        scanf_s("%lf",&score);
        sum += score;                               //计算总分
    }
    printf("总分为:%d\n" , sum);
    printf("平均分为:%lf\n" , sum / 5.0);           //这里lf需要double类型所以写了5.0
    return 0;
}
```

5. continue和break(通常与条件语句同时使用)

```
continue:继续下一次循环
break:跳出整个循环
```

```
#include<stdio.h>
int main() {
    //1:50层楼 模拟电梯 第8楼和第9楼位同一家公司电梯只停8楼,9楼不停
    //2:40层及以上为私人住所电梯不上去
    for (int i = 1; i <= 50; i++) {
        if (i == 9) {
            continue;
        }
        else if (i > 39) {
            break;
        }
        printf("电梯当前在第%d层\n",i);
    }
    return 0;
}
```

6. 各循环的无限循环

```
while(true){循环操作}
do{循环操作}while(true);
for(;;){循环操作}
```

```
#include<stdio.h>
int main() {
    /*for (;1==1;) {
        printf("a");
    }*/
    /*for (;true;) {
        printf("a");
    }*/
    /*for (;;) {
        printf("a");
    }*/
    /*while (1==1) {
        printf("a");
    }*/
    /*while (true) {
```

```
综合案例:猜拳游戏
```

```
#include<stdio.h>
//#include<time.h>
#include<stdlib.h> //获取随机数所用到的预处理文件
int main() {
    while (true) {
        int numWj = 0;//玩家出的拳(数字)
        //srand(time(NULL));
        int numDn = rand() % 3 + 1; //电脑随机出拳(数字)
        printf("请出拳:1-石头 2-剪刀 3-布 0-退出\n");
        scanf_s("%d", &numWj); //&取地址运算符
        if (numWj > 3 || numWj < 0) {
            printf("请正确出拳! \n");
        }
        else if (numWj == 0) {
            break;
```

第七章:数组

一.数组

1. 概念:连续等大的存储空间,类型相同的数据集合

2. 定义:类型 数组名 [数组大小]

```
int arr[3];             //只定义不赋值
int arr1[3]={1,2,3};    //完全列举法
int arr2[3]={1,2};      //部分列举法
int arr3[]={1,2,3,4};   //省略大小列举法
```

3. 如何访问数组元素:

通过数组下标来访问,数组下标从0开始

4. 代码演示

案例一:数组基本操作

```c
#include<stdio.h>
int main() {
    int arr[3];             //只定义不赋值
    int arr1[3] = { 1,2,3 };    //完全列举法
    int arr2[3] = { 1,2 };      //部分列举法
    int arr3[] = { 1,2,3,4 };   //省略大小列举法
    //通过下标获取数组中的元素
    printf("获取数组中的元素:%d\n",arr1[1]);
    //给数组赋值
    arr[0] = 1;
    arr[1] = 2;
    printf("arr数组赋值后的第一个元素为:%d\n", arr[0]);
    //未被赋值的数组元素 数组在全局时默认为0 局部时为随机值
    printf("arr数组未赋值的空间元素为:%d\n", arr[2]);
    //数组遍历
```

案例二:循环给数组赋值

```c
#include <stdio.h>

int main()
{
    char a[10];
    int i;
    printf("请输入十个字符:");
    for (i = 0; i < 10; i++)
    {
        scanf_s("%c", &a[i],1);
    }
    printf("输出:");
    for (i = 9; i >= 0; i--)
    {
        printf("%c", a[i]);
```

案例三:对数组元素求最大值、最小值

```c
/**
 * 对数组元素求最大值、最小值
 */
#include <stdio.h>

int main()
{
    //num[5]存放要排序的数字
    //max,min 最大值，最小值
    //i 循环变量
    int num[5], max, min, i;

    /*循环输入5个数字*/
    printf("请输入5个数字:");
    for (i = 0; i < 5; i++)
```

案例四:线性查找

```c
/**
 *  线性查找
 */
#include <stdio.h>
#define N 10
int main()
{
    //i作为循环变量，也作为数组元素下标
    //num为要查找的数组
    //search为要查找的元素
    int i, num[N], search;

    /*循环录入数组元素*/
    printf("请输入%d个数组元素:\n", N);
```

```
    for (i = 0; i < N; i++)
```

案例五:插入算法

```
/*插入算法*/
#include <stdio.h>
#define N 9
int main() {
    int i, j; //循环变量
    //num 已排好序的数组
    //数组的大小比数组当前大小多1是为了给插入的数预留位置
    //in 要插入的数字
    int num[N + 1] = { 1,4,7,13,16,19,22,25,28 }, in;
    //打印要插入的数组的所有元素
    printf("插入前的数组元素:");
    for (i = 0; i < N; i++) {
        printf(" %d ", num[i]);
    }
    //从键盘读取一个要插入的数
```

案例六:冒泡排序

```
/**
 * 冒泡排序:相邻的两个数字两两比较    较大的放在后面
 *  int [] x = {6,3,8,2,9,1};
 *  从小到大排序:6 3 8 2 9 1-->1 2 3 6 8 9
 *
 *  第一趟:6 3 8 2 9 1
 *      第一次:3 6 8 2 9 1
 *      第二次:3 6 8 2 9 1
 *      第三次:3 6 2 8 9 1
 *      第四次:3 6 2 8 9 1
 *      第五次:3 6 2 8 1 9
 *  第二趟:3 6 2 8 1 9
 *      第一次:3 6 2 8 1 9
 *      第二次:3 2 6 8 1 9
 *      第三次:3 2 6 8 1 9
```

二.多维数组

二维数组(数组的数组,数组里面存数组)

```
#include<stdio.h>
int main() {
    int arr[1][2];
    int arr1[2][3] = { {1,2,3},{4,5,6} };
    int arr2[2][3] = { {1,2,3},{1} };
    //int arr3[][] = { {1,2},{3,4} };  //不可以
    //int arr3[2][] = { {1,2},{3,4} }; //不可以
    int arr3[][2] = { {1,2},{3,4} };
    //通过下标获取数组中的元素
    printf("获取数组中的元素%d\n",arr1[0][0]);
    //给数组赋值
    arr[0][0] = 1;
    //未赋值的空间存放随机值
    printf("获取数组中的元素%d\n", arr[0][1]);
    //数组遍历
```

第八章:函数

一.函数

1. 理解:函数相当于一系列逻辑的集合

2. 为什么使用函数:

```
使程序变得更简短而清晰
有利于程序为维护
可以提高程序开发的效率
提高代码的复用性
```

3. 函数类型:

```
库函数
    由C语言系统提供
```

```
    用户无须定义,也不必在程序中作类型说明
    只需在程序前包含有该函数定义的头文件即可使用
自定义函数
    用户在程序中根据需要而编写的函数
```

4. 常用的库函数

```
    函数                          头文件              用途
double sqrt(double x)               math.h            计算x平方根
double pow (double x,double y)      math.h            计算x的y次幂
double ceil(double x)               math.h            求不小于x最小整数double显示
double floor(double x)              math.h            求不大于x最大整数double显示
int toupper(int x)                  ctype.h           将x转换为全大写
int tolower(int x)                  ctype.h           将x转换为全小写
int rand(void)                      stdlib.h          产生一个随机数
void exit(int retval)               stdlib.h          终止程序
```

5. 函数的定义

```
    返回值类型    函数名    参数列表    函数体
       void       eat        ()        {}
```

6. 函数原型声明

```c
//在形式上与函数头部类似,最后加一个分号
//原型声明中参数表里的参数名可以不写(只写参数类型)
#include<stdio.h>
int count(int,int);
int main(){
    return 0;
}
int count(int x,int y){
    return x+y;
}
```

7. 函数的调用

```
    通过在程序中使用函数名称,可以执行函数中包含的语句,称为函数调用
    函数名(参数列表)
```

8. 函数的形参和实参

```
    形参:定义函数时的参数
    实参:调用函数时传过来的参数
```

9. 代码演示

```
案例一:函数的定义及使用
```

```c
#include<stdio.h>
void test1();
void test2(int a);
void test3(int a, int b);
int test4();
int test5(int, int); //声明可以不写参数名
int main() {
    //函数调用
    test1();
    test2(2);
    test3(1,2);
    int a = test4();
    printf("调用test4函数得到的返回值为:%d\n",a);
    int b = test5(2,3);
    printf("调用test5函数得到的返回值为:%d\n", a);
```

```
案例二:求最大值函数
```

```c
#include <stdio.h>

/*求最大值函数*/
int max(int a, int b)
{
    if (a > b)
    {
```

```
        return a;
    }
    return b;
}
void main()
{
    int m = max(5, 8);
    printf("最大值：%d\n", m);
```

案例三:计算长方形的周长及面积

```
/*
12_4、根据边长计算长方形的周长及面积，用函数完成
函数定义:double getArea(double c){}
double getGirth(double c){}
*/
#include "stdio.h"

//定义函数,计算面积
double getArea(double c,double k)
{
    //计算，并返回面积
    return c * k;
}

//定义函数，计算周长
```

案例四:函数打印直角三角形

```
/*
12_1、用函数打印直角三角形，用户输入几行，就打印几行
如：输入：6
*
**
***
****
*****
******
函数定义：void getSquare(int num){}
*/
#include "stdio.h"

//定义函数
void getSquare(int num)
```

第九章:指针

一.什么是指针?

1. 指针:专门存放变量地址的变量

```
指针变量声明的一般形式为：数据类型 *变量名=初始地址值;
指针运算符有&和*运算符
    &:取变量的地址
    *:取指针指向变量的内容
两者互为逆运算
```

2. 星号是用来指定一个变量是指针。以下是有效的指针声明:

```
int i;
int *p = &i;  指针的类型必须与指针所指向的类型一致

int i;
int *p = &i;
int *q = p; 用已初始化指针变量做初值

int    *ip;    /* 一个整型的指针 */
double *dp;    /* 一个 double 型的指针 */
float  *fp;    /* 一个浮点型的指针 */
char   *ch     /* 一个字符型的指针 */
```

注意:指针变量需要先赋值,再使用,不允许把一个数赋予指针变量

所有指针的值都是一个代表内存地址的长的十六进制数。
不同数据类型的指针之间唯一的不同是，指针所指向的变量或常量的数据类型不同。

二.为什么要使用指针?

```
1.方便的使用字符串
2.有效地表示复杂的数据结构
3.动态分配内存
4.得到多于一个的函数返回值
```

1. 代码演示

```
案例一:指针的定义及使用
```

```c
#include <stdio.h>
int  main(){
    int a;
    int *b = &a;  //指针的类型必须与指针所指向的类型一致

    int c;
    int *d = &c;
    int *e = d; //用已初始化指针变量做初值

    int *ip;    /* 一个整型的指针 */
    double *dp;    /* 一个 double 型的指针 */
    float *fp;    /* 一个浮点型的指针 */
    char *ch;     /* 一个字符型的指针 */

    //printf("未赋值的指针变量:%d",ip);
```

```
案例二:指针处理一维数组
```

```c
#include <stdio.h>
int main(){
    /*用传统数组形式*/
    /*int i, a[10];
    printf("请输入十个数字组成数组a: \n");
    for (i = 0; i < 10; i++){
        scanf_s("%d", &a[i]);
    }
    printf("输出a数组中的所有元素:  \n");
    for (i = 0; i < 10; i++){
        printf("%4d", a[i]);
    }
    printf("\n");*/

    /*采用指针变量表示的地址法输入输出数组各元素*/
```

```
案例三:指针处理成绩输出
```

```c
#include<stdio.h>
#define N 5
int main() {
    int score[N];
    int* p = score;
    for (int i = 0; i < N;i++) {
        printf("请输入第%d个学生的成绩:\n",i+1);
        scanf_s("%d",p+i);
    }
    printf("\n");
    for (int i = 0; i < N; i++) {
        printf("成绩为:%d ", score[i]);
    }
    printf("\n");
    for (int i = 0; i < N;i++) {
```

第十章:函数高级

一.掌握函数参数的传递方式

1. C语言中函数参数的传递方式有两种

```
1.值传递:实参传递给形参  对形参的操作  不影响实参
2.地址传递:实参地址传递给形参  对形参的操作  影响实参
```

见案例:

```
#include<stdio.h>
void test1(int);
void test2(int*);
int main() {
    int a = 0;
    //test1(a);
    test2(&a);
    printf("%d",a);
    return 0;
}
void test1(int a) {//值传递
    a = 10; //给参数赋值看a的值会不会影响
}
void test2(int *a) {//地址传递
    *a = 10; //给参数赋值看a的值会不会影响
```

二. 使用数组作为函数参数

1. 数组形式

2. 指针形式

```
见案例:
```

```
#include<stdio.h>
int getMin(int[]);
int getMax(int*);
int main() {
    int arr[5];
    int min;
    int max; //存放找到的最大值
    printf("请输入5个不同的值,存储在数组中\n");
    for (int i = 0; i < 5;i++) {
        scanf_s("%d",&arr[i]);
    }
    min = getMin(arr);
    max = getMax(arr);
    printf("数组中最小的值为:%d\n",min);
    printf("数组中最大的值为:%d\n",max);
```

三.自定义头文件

```
**使用自定义头文件来存储自己的常用函数, 从而提高程序的重用性,提高效率**
```

1. 自定义头文件:headerFile.h

```
//文件中实现功能求三个数的立方和
int cube(int a,int b,int c){
    int ans = (a*a*a)+(b*b*b)+(c*c*c);
    return ans;
}

//引用自定义头文件
#include<stdio.h>
#include<headerFile.h>
int main(){
    int a,b,c;
    printf("请输入第一个数:");
    scanf_s("%d",&a);
    printf("请输入第二个数:");
    scanf_s("%d",&a);
```

第十一章:字符串

#include<string.h>

一.理解字符串常量

```
字符串常量是双引号括起来的任意字符序列
如:"HelloWorld" "Hello\"Accp\"" 可以包含转义字符
字符串结束符\0  helloWorld\0
```

二.理解字符数组和字符串的异同

```
C语言中没有专门的字符串变量,
通常用一个字符数组来存放一个字符串
字符数组和字符串的区别是字符串的末尾有一个空字符\0
```

1. 字符串声明:

```
//手工加空字符
char name[15]={'W','a','n','g','L','i','\0'};
char name[15]="WangLi";     //系统自动加空字符
//省略数组大小,系统自动计算,
//大小为字符总数+1,最后一位存入空字符
char password[]="12345678";
```

```
案例一:字符串的使用
```

```
#include<stdio.h>
#include<string.h>
int main() {
    //手工加空字符
    char name1[15] = { 'W','a','n','g','L','i','\0' };
    char name2[15] = "WangLi";      //系统自动加空字符
    char name3[15];
    //省略数组大小,系统自动计算,
    //大小为字符总数+1,最后一位存入空字符
    char password[] = "12345678";
    printf("请输入姓名:");
    //scanf_s("%s",name3,15); //使用scanf时不能输入空格
    //printf("输入姓名为:%s",name3);
    //gets(buffer) //函数对输入长度不加以限制
    //所以在vs中用gets会有警告信息This function or variable may be unsafe.
```

三.使用字符串输入输出函数

1. 字符串输入输出函数的使用

```
//scanf_s("%s",name3,15); //使用scanf时不能输入空格
//printf("输入姓名为:%s",name3);
gets_s(name3);//从键盘读入一行存入name,并用\0去带行尾\n
puts(name3);//把字符串输出
//按一定格式输出是通常使用printf
```

四.使用字符串处理函数

1. 代码演示

```
案例二:字符串处理函数
```

```
#include<stdio.h>
#include<string.h>
int main() {
    char arr[] = "BeiJing";
    int len1 = strlen(arr);
    int len2 = strlen("ShangHai");
    printf("len1的长度为:%d,len2的长度为:%d\n",len1,len2);

    char source[] = "字符串复制";
    char target[20];
    strcpy_s(target,source);
    printf("源字符串为:%s,目标字符串为:%s",source,target);

    //gets(buffer) //函数对输入长度不加以限制
    //所以在vs中用gets会有警告信息This function or variable may be unsafe.
```

五.理解指针与字符串的关系

1. 代码演示

```
案例三:指向字符串的指针
```

```c
#include<stdio.h>
#include<string.h>
int main() {
    char uname[30] = "wahaha";
    char* p;
    int count=0;
    for (p = uname; *p != '\0';p++) {
        if (*p=='a') {
            count++;
        }
    }
    printf("共计%d个a",count);
    return 0;
}
```

六.使用字符指针数组

1. 代码演示

```
案例四:字符指针数组
```

```c
#include<stdio.h>
#include<string.h>
int main() {
    int a = 1, b = 2, c = 3;
    //定义一个指针数组
    int* arr[3] = { &a, &b, &c };//也可以不指定长度，直接写作 int *arr[]
    printf("%d, %d, %d\n", *arr[0], *arr[1], *arr[2]);
    //定义一个指向指针数组的指针
    /*int** parr = arr;
    printf("%d, %d, %d\n", **(parr + 0), **(parr + 1), **(parr + 2));*/
    //指针数组还可以和字符串数组结合使用
    const char* str[3] =    { "java", "C语言", "C#"};
    printf("%s,%s,%s\n", str[0], str[1], str[2]);
    return 0;
}
```

七.使用字符串作为函数参数

1. 代码演示

```
案例五:字符串作为函数参数
```

```c
#include<stdio.h>
#include<string.h>
void mystrcpy(char[],char[]);
int mystrlen(char*);
int main() {
    char arr1[] = "lanzhou";
    char arr2[20];
    int len = mystrlen(arr1);
    printf("arr1数组长度为:%d\n", len);
    mystrcpy(arr2, arr1);
    printf("arr2字符串为:%s", arr2);
    return 0;
}
//自定义字符串复制函数
void mystrcpy(char dest[],char src[]) {
```

第十二章:结构

一.结构体类型标识符的定义

```
结构:是一种构造类型,它由若干成员组成,
每一个成员可以是一个基本数据类型或构造类型
```

二.结构体和数组的区别

```
    数组的每一个成员类型都相同,结构体可以拥有不同类型的成员
```

一般的,定义一个有n个成员的结构体类型可以采用如下形式:
struct 结构体名{类型 成员名;类型 成员名;类型 成员名;…};

定义结构体名同时定义变量:
struct 结构体名{类型 成员名;类型 成员名;类型 成员名;...}变量名表;

定义无名称的结构体:
struct{类型 成员名;类型 成员名;类型 成员名;...}变量名表;

1. 示例

```
struct date{
    int year;
    int month;
    int day;
}
struct student{
    int num;
    char name[10];
    char sex;
    struct date birthday;
    char addr[20]
}
```

2. 结构体类型变量的定义:

```
结构体名定义好后,该标识符的使用就如其他类型标识符
(int,double等)一样使用,用这种类型也能定义相应的变量.
struct stu{
    int age;
    char name[10];
}
struct 结构体名  变量名列表;
如:struct stu stu1,stu2;
    struct student s1,s2;
```

3. 代码描述

```
案例:结构体的定义
```

```
#include<stdio.h>
//定义结构体变量的同时初始化值
struct stu1 {
    char name[10];
    int age;
}s1 = {"Tom",30};
struct stu2 {
    char name[10];
    int age;
    char* add;
}st1 = { "ZhangSan",30,"上海" },st2;
int main() {
    st2 = st1;//stu2例子 某些情况可以对结构体变量整体操作
    //通过.来引用结构体的某个成员变量
    st2.add = "兰州";
```

4. 结构体数组

```
案例:结构体数组
```

```
#include <stdio.h>
struct {
    char* name;  //姓名
    int num;  //学号
    int age;  //年龄
    char group;  //所在小组
    float score;  //成绩
}student[5] = {  //也可以不赋值也可以不写出大小
    {"Li ping", 5, 18, 'C', 145.0},
    {"Zhang ping", 4, 19, 'A', 130.5},
    {"He fang", 1, 18, 'A', 148.5},
    {"Cheng ling", 2, 17, 'F', 139.0},
    {"Wang ming", 3, 17, 'B', 144.5}
};
int main() {
```

5. 结构体指针

```
案例:结构体指针
```

```c
#include <stdio.h>
int main() {
    struct {
        char* name;  //姓名
        int num;  //学号
        int age;  //年龄
        char group;  //所在小组
        float score;  //成绩
    } stu1 = { "Tom", 12, 18, 'A', 136.5 }, * pstu = &stu1;
    //读取结构体成员的值
    printf("%s的学号是%d，年龄是%d，在%c组，今年的成绩是%.1f! \n",
        (*pstu).name, (*pstu).num, (*pstu).age, (*pstu).group, (*pstu).score);
    printf("%s的学号是%d，年龄是%d，在%c组，今年的成绩是%.1f! \n",
        pstu->name, pstu->num, pstu->age, pstu->group, pstu->score);
    return 0;
```

第十三章:文件

一.文件

概念:是存储在外部硬盘上数据的集合,操作系统是以文件为单位对数据进行管理的

1. 文件类型有两种

   文本文件和二进制文件

二.文件的基本操作步骤

```
1.创建文件指针变量 FILE* fp;
    只有通过文件指针,才能调用相应文件
2.打开文件        fp = fopen("文件","打开方式");
    打开失败返回NULL指针
    文本文件的打开方式:
        rt:打开文件进行读操作
        wt:创建文件进行写操作
        at:向文件追加数据
        rt+:打开文件进行读写操作
        wt+:创建文件进行读写操作
3.读写文件
    读文件        fscanf()和fread()  文件必须存在
    写文件        fprintf()和fwrite()
        int fprintf("文件指针变量","格式字符串"."输入列表");
        将缓冲区的内容写入文件
```

三.为什么要使用文件

```
1.程序运行是可以使用内存存储数据,但是内存中的数据在程序退出.断电等操作后就会全部清除
2.将数据存储在内存中,每次运行程序都需要重新录入信息
3.有实际意义的程序都需要永久保存数据
```

四.文件操作常用函数

1. 常用函数

```
fopen()   打开文件
fclose()  关闭文件
fread()   读取文件到缓冲区
fwrite()  将数据从缓存去写入文件
fprintf() 类似于printf()
fscanf()  类似于scanf()
Feof()    判断文件活动指针是否到达文件末尾,到达返回true
Rewind()  将文件位置指示器从新置于文件开头
Remove()  删除文件
fflush()  将内部缓冲区数据写入指定文件
```

2. fopen 和 fopen_s

```
fopen用法: fp = fopen(filename,"w")。
fopen_s用法:，须定义另外一个变量errno_t err,然后err = fopen_s(&fp,filename,"w")。
返回值:  fopen打开文件成功，返回文件指针（赋值给fp），  打开失败则返回NULL值;
fopen_s打开文件成功返回0，失败返回非0。
```

3. 在vs编程中，经常会有这样的警告：

```
warningC4996: 'fopen': This function or variable may be unsafe.
Considerusing fopen_s instead. To disable deprecation,
use_CRT_SECURE_NO_WARNINGS. See online help fordetails.
是因为 fopen_s比fopen多了溢出检测，更安全一些。
（在以后的文章里还有get与get_s的比较,strcpystrcpy_s的比较，
他们的共同点都是用来一些不可预料的行为，以后将进行详尽解释）

在 Visual Studio 中关闭项目的警告
      若要在 Visual Studio IDE 中关闭整个项目的警告：
      打开项目的 "属性页" 对话框。
      选择 "配置属性" " > c/c + + > 高级" 属性页。
      编辑 "禁用特定警告" 属性以添加 4996 。
      选择 "确定" 以应用所做的更改。
```

4. 案例演示

案例1:写入学生信息

```c
#include<stdio.h>
int main() {
    FILE* fp; //定义文件指针变量
    char name[20]; //存储姓名
    char sex[2];  //存储性别
    int age; //存储年龄
    char school[20]; //存储学校名称

    //从键盘输入一个学生的基本信息
    printf("输入姓名：");
    scanf_s("%s", &name,21);
    printf("输入性别：");
    scanf_s("%s", &sex,3);
    printf("输入年龄：");
    scanf_s("%d", &age);
```

案例2:输出学生信息

```c
#include<stdio.h>
int main() {
    FILE* fp; //定义文件指针变量
    char name[20]; //存储姓名
    char sex[2];  //存储性别
    int age; //存储年龄
    char school[20]; //存储学校名称

    fp = fopen("d:\\demo1.txt", "rt"); //以只读的方式打开C:\\student.txt
    //从d:\\demo.txt中读取学生的信息
    fscanf_s(fp, "%s", &name,21); //从文件中读取姓名
    fscanf_s(fp, "%s", &sex,3);  //从文件中读取性别
    fscanf_s(fp, "%d", &age); //从文件中读取年龄
    fscanf_s(fp, "%s", &school,21); //从文件中读取学校名称

```

案例3:写入多个学生信息

```c
#include <stdio.h>
//定义一个结构体变量
struct Student
{
    char name[20]; //姓名
    char sex[2];  //性别
    int age; //年龄
    char school[20]; //学校名称
};
#define N 2 //学生数目
int main()
{
    FILE* fp; //定义文件指针变量
    struct Student s[50]; //定义学生结构体数组
    int i; //循环变量
```

案例4:输出多个学生信息

```c
#include <stdio.h>
//定义一个结构体变量
struct Student
{
    char name[20]; //姓名
    char sex[2];  //性别
    int age; //年龄
    char school[20]; //学校名称
};
#define N 2
int main(){
    FILE* fp; //定义文件指针变量
    struct Student s[50];
    int i;  //循环变量
    fp = fopen("D:\\studentbin.txt", "rt");
    //以只读的方式打开C:\\studentbin.txt
    //从C:\studentbin.txt中读取N个学生的信息存入s数组中
    fread(s, sizeof(struct Student), N, fp);
    //输出所有学生基本信息
    printf("姓名\t年龄\t性别\t在读学校\n");
    printf("==================================\n");
    for (i = 0; i < N; i++){
        printf("%s\t%d\t%s\t%s\n", s[i].name, s[i].age, s[i].sex, s[i].school);
    }
    fclose(fp); //关闭fp所指向的文件
    return 0;
}
```