重庆理工大学/电气学院
CHONGQING UNIVERSITY OF TECHNOLOGY

嵌入式Linux系统开发教程
—基于ARM处理器通用平台（arm9-arm11-contexA系列）

丛超

2025年4月

1 C语言基础回顾

2 数组与字符串

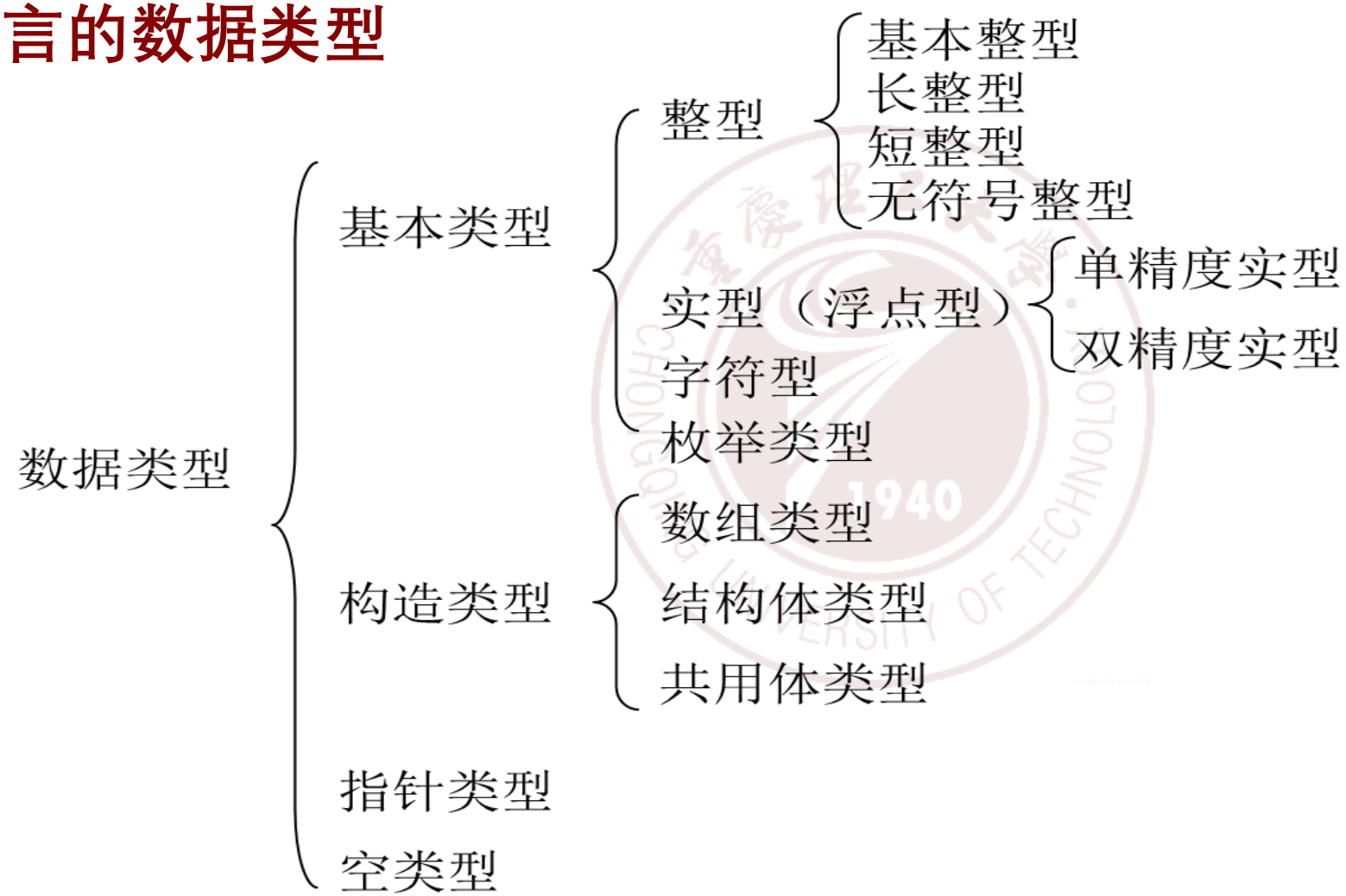
3 函数与指针

4 预处理命令

01

C语言基础回顾

C语言的数据类型


数据类型
基本类型
整型
基本整型
长整型
短整型
无符号整型
实型（浮点型）
单精度实型
双精度实型
字符型
枚举类型
构造类型
数组类型
结构体类型
共用体类型
指针类型
空类型

C语言的关键字

auto	default	float	long	sizeof	unsigned void
break	do	for	register	static	volatile while
case	double	goto	restrict	struct	_Bool
char	else	if	return	switch	_Complex
const	enum	inline	short	typedef	_Imaginary0
continue	extern	int	signed	union	

C语言的标识符（变量命名）

标识符命名规范如下：

（1）标识符只能由字母、数字和下划线组成。

（2）标识符不能以数字作为第一个字符。

（3）标识符不能使用关键字。

（4）标识符区分大小写字母，如add、Add和ADD是不同的标识符。

以下标识符，哪些是合法的?

area

3a　　//标识符不能以数字开头

DATE

_name

ab.c　　//标识符智能由字母、数字、下划线组成

lesson_1

long　　//标识符不能使用关键字

abc#　　//标识符只能由字母、数字和下划线组成

C语言的常量

（1）整形常量

整形常量是整数类型的常量，又称整常数。根具不同的技术方法，整数常量可记为二进制整数、八进制整数、十进制整数和十六进制整数。

①二进制整数，如0b100、0B101011。

②八进制整数，如0112、056。

③十进制整数，如2、-100。

④十六进制整数，如0x108、0X29。

C语言的常量

（2）实型常量

实型也称为浮点型，实型常量也称为实数或浮点数，也就是在数学中用到的小数。在C语言中，实型常量有两种表示形式：十进制小数形式和指数形式，具体示例如下：

①十进制小数形式：由数字和小数点组成（注意：必须有小数点），如1.23、-45.6、100.0等。

②指数形式：又称科学计数法，规定以字母e或E表示以10为底的指数，如12.3e3（代表12.3×10^3）、-34.8e-2（代表-34.8×10^{-2}）、0.1E4（代表0.1×10^4）等。

C语言的常量

（3）字符常量

C语言中的字符常量是由单引号括起来的一个字符，如'a'、'2'、'#'等。字符常量两侧的一对单括号是必不可少的，如'B'是字符常量，而B是一个标识符。

（4）字符串常量

字符串常量是用一对双引号括起来的字符序列，例如“hello”、“123”、“itcast”等。字符串的长度等于字符串中包含的字符个数，例如“hello”的长度为5个字符。

C语言的常量

（5）符号常量

C语言也可以用一个标识符来表示一个常量，称为符号常量。符号常量在使用前必须先定义，其语法格式如下所示：

#define 标识符 常量

上述语法格式中，define是关键字，前面加符号“#”，表示这是一条预处理命令（预处理命令都以符号“#”开头），称为宏定义。

例如，将圆周率用PI表示，可写成：

#define PI 3.14

C语言的变量

变量（Variable）是在程序执行过程中可以改变、可以赋值的量。在C语言中，变量必须遵循“先定义，后使用”的原则，即每个变量在使用之前都要用变量定义。语句将其定义为：某种具体的数据类型，其语法格式如下：

类型关键字 变量名1[，变量名2，….]

其中，方括号内的内容为可选项，当同时定义多个相同类型的变量时，它们之间需用逗号分隔。

C语言的进制

表示形式	特点	举例
十进制	由0~9的数字序列组成，数字前可以带正负号。	如256、-128、0、+9是合法的十进制数。
八进制	以8为基的数值系统称为八进制。八进制数由以数字0开头，后跟0~7的数字序列组成。八进制数010相当于十进制数8。	如021、-017是合法的八进制数。
十六进制	以16为基的数值系统称为十六进制。十六进制数由数字0加字母x（大小写即可）开头，后跟0~9，a~f（大小写均可）的数字序列组成。十六进制数0x10相当于十进制16。	如0x12、-0x1F是合法的十六进制数，他们分别代表十进制数18、-31。

C语言的字符变量

字符型变量用于存储一个单一字符，在C语言中用char表示，其中每个字符变量都会占用1个字节。char表示1字节，本质是无符号整型数，范围0-255。

字符是用单引号括起来的一个符号。字符串是用双引号括起来的字符序列，以\0结束，"asdf"有5个字符，\0也是其中的一个字符，但不显示。

C语言的转义字符

除了可以直接从键盘上输入的字符以外，还有一些字符是无法用键盘直接输入的，此时需要采用一种新的定义方式——转义字符，它以“\”开头，随后接特定的字符。

C语言的字符集

字符集（Character set）是多个字符的集合，字符集种类较多，每个字符集包含的字符个数不同，常见字符集名称：ASCII字符集、GB2312字符集、BIG5字符集、GB18030字符集、Unicode字符集等。

C语言的转义字符

① 单引号 '\''
② 双引号 '\"'
③ tab键 '\t'
④ 退格键 '\b'
⑤ 警铃 '\a'
⑥ 回车 '\r'
⑦ 换行 '\n'
⑧ 8进制 '\ddd'，必须是三个位，d代表一个8进制位的数
⑨ 16进制 '\xhh'，必须是2个位，h代表一个16进制位的数

C语言的类型转换

C语言程序中的类型转换可分为隐式类型转换和显式类型转换两种。

1.隐式类型转换

隐式类型转换即系统自动进行的类型转换。分两种情况，下面分别介绍。

01

不同类型数据进行运算

不同类型的数据进行运算时，系统会自动将低字节数据类型转换为高字节数据类型，即从下往上转换。

02

赋值转换

在赋值类型不同时，即变量的数据类型与所赋值的数据类型不同，系统就会将“=”右边的值转换为变量的数据类型，再将值赋给变量。

示例4.2.6-1 隐式类型转换

(1) 'c' +12;最终结果是什么类型?
解析：结果是int类型，字符'c'先变为int类型，然后和12相加。

(2)23f+12;最终结果是什么类型?
解析：结果是float类型，23先变为float类型，然后和12相加。

示例4.2.6-1 隐式类型转换

(1) 'c' +12;最终结果是什么类型?
解析：结果是int类型，字符'c'先变为int类型，然后和12相加。

(2)23f+12;最终结果是什么类型?
解析：结果是float类型，23先变为float类型，然后和12相加。

2.显式类型转换

显式类型转换是指使用强制类型转换运算符，将一个变量或表达式转换成所需的类型，其基本语法格式如下：

（类型名）表达式;

示例4.2.6-2 显式类型转换，判断强制转换后的值

(1)(int)3/2;

解析：值是1，两个整数相除，值为整数。

(2)(double)3/2;

解析：值是1.500000，先将3强转为double类型为3.00000，然后除以2结果为1.500000。

3.字符串与数值之间的转换

字符串转型成数据的函数。

(1)函数返回值为int类型：int atoi(const char *nptr);

(2)函数返回值为long int类型：long atol(const char *nptr);

(3)函数返回值为double类型：double atof(const char *nptr)。

示例4.2.6-3 判断输出结果是什么

```
#include <stdio.h>
int atoi(const char *nptr);   //函数声明
int main(int argc,char **argv){
int a=atoi(argv[1]);  //将atoi函数中argv[1]参数值赋值给a，并定义为int类型
    printf("%d\n",a);    //打印输出a的值
    int b=atoi("8");     //将atoi函数中字符串“8”值赋值给b，并定义为int类型
    printf("%d\n",b);    // 打印输出b的值
    int c='8'-'0';        //将字符'8'所对应ASCII值减去字符'0'所
                          //对应ASCII值赋值给c，并定义为int类型
    printf("%d\n",c);    //打印输出c的值
    }
```

数字转字符串，所需函数如下：

```
int  sprintf(char *str, const char *format，...);
```

参数：str是指向一个字符数组的指针，该数组存储了C字符串；format是字符串，包含要被写入到字符串str的文本。它可以包含嵌入的format标签，format标签可被随后的附加参数中指定的值替换，并按需求进行格式化。

format标签属性是
%[flags][width][.precision][length]specifier。

%-md	整型
%-ml	长整型
%mu	无符号
%mo	八进制
%mx	十六进制
%mp	十六进制前面有0x
%m.nf	单精度
%m.nlf	双精度
%mc	字符
%ms	字符串
m	代表占多少个字符的宽度
.n	代表占小数点的位数
-	代表左对齐
...	用逗号分隔的值(常量、变量、表达式)替换前面的格式符

示例4.2.6-4 判断输出结果

```
#include <stdio.h>
int main(int argc,char **argv){
        int a=23;
        char str[256];
        sprintf(str,"%d",a);
        printf("%s\n",str);
        sprintf(str,"0x%x，  0%o",a,a);
        printf("%s\n",str);
        double b=3.13241324;
        sprintf(str,"%.2lf",b);
        printf("%s\n",str);
}
```

输入输出

C语言中没有提供专门的输入/输出语句，输入/输出操作是通过调用C的标准库函数来实现的。C的标准函数库中提供用于标准输入/输出操作的库函数，使用这些标准输入/输出函数时，只要在程序的开始位置加上如下一行编译预处理命令即可：

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

它的作用是：将输入/输出函数的头文件stdio.h包含到用户源文件中。其中，h为head之意，std意为standard之意，i为input之意，o为output之意。

字符输入/输出

getchar()和putchar()是专门用于字符输入/输出的函数。其中，getchar()用于从键盘读一个字符，待用户击键后，将读入值返回，并自动将用户击键结果回显在屏幕上；putchar()则把字符写到屏幕的当前光标位置。这两个函数的使用格式如下：

```
变量=getchar();
putchar(变量);
```

示例4.2.7-1 getchar()和putchar()函数的使用

```
#include <stdio.h>
int main(int argc,char **argv){
    char ch;
    printf("Press a key and then press Enter:");
    ch=getchar();        //从键盘输入一个字符，并将该字符存入变量ch
    printf("You pressed ");
    putchar(ch);         //在屏幕上显示变量ch中的字符
    putchar('\n');       // 输出一个回车换行符
}
```

格式输入/输出

C语言提供了一对格式化输入/输出函数：scanf()函数与printf()函数。scanf()函数用于读取用户输入数据，printf()函数用于向控制台输出数据。

（1）scanf()函数

scanf()函数用于读取用户从键盘输入的数据，它可以灵活接收各种类型的数据，如字符串、字符、整型、浮点数等，scanf()函数也可以通过格式控制字符控制用户的输入，但它只使用类型（%d、%c、%f等）格式控制，并不使用宽度、精度、标志等格式控制。scanf()函数一般格式如下：

scanf(格式控制字符串,参数地址列表);

scanf()函数一般格式如下：

scanf(格式控制字符串,参数地址列表)；

其中，格式控制字符串是用双引号括起来的字符串，它包括格式转换说明符和分隔符两部分。

格式转换说明符	用法
%d或%i	输入十进制整数
%o	输入八进制整数
%x	输入十六进制整数
%c	输入一个字符，空白字符（空格、回车、制表符）也作为有效输入
%s	输入字符串，遇到一个空白字符（空格、回车、制表符）时结束
%f或%e	输入实数，以小数或指数形式输入均可
%%	输入一个百分号

示例4.2.7-2 判断输出结果

```
#include<stdio.h>
int main(int argc,char **argv)
{
    int a;                    //定义a为整型数据
    scanf("%d",&a);           //接收一个从键盘输入的整型数据
    printf("--- %d\n",a);  //输出a的值
}
```

（2）printf()函数

printf()函数为格式化输出函数，该函数最后一个字符f表示“格式（format）”的意思，其功能是按照用户指定的格式将数据输出到屏幕上。printf()函数的一般格式如下：

printf(格式控制字符串,输出值列表);

格式控制字符串是用双引号括起来的字符串，也简称为格式字符串，输出值参数列表中可有多个输出值，也可以没有。一般，格式控制字符串包括两个部分：格式转换说明符和需原样输出的普通文本。

格式转换说明符	用法
%d或%i	输出带符号的十进制整数，正数的符号省略
%u	以无符号十进制整数形式输出
%o	以无符号八进制整数形式输出，不输出前导符0
%x	以无符号十六进制整数形式（小写）输出，不输出前导符0x
%X	以无符号十六进制整数形式（大写）输出，不输出前导符0x
%c	输出一个字符
%s	输出字符串
%f	以十进制小数形式输出实数（包括单、双精度），整数部分全部输出，隐含输出6位小数，输出的数字并非全部是有效数字，单精度实数的有效位数一般是7位，双精度实数的有效位数一般为16位
%e	以指数形式（小写e表示指数部分）输出实数，要求小数点前必须有且仅有1位非0数字
%E	以指数形式（大写E表示指数部分）输出实数
%g	根据数据的绝对值大小，自动选取f或e格式中输出宽度较小的一种，且不输出无意义的0
%G	根据数据的绝对值大小，自动选取f或E格式中输出宽度较小的一种，且不输出无意义的0
%p	以主机的格式显示指针，即变量的地址
%%	显示百分号%

示例4.2.7-3 判断输出结果是什么

```
#include <stdio.h>
int main(int argc,char **argv){
    int a[23];                  //定义整型数组a
    scanf("%d",a);              //接收一个从键盘输入的整型数据a
    printf("--- %d\n",*a);      //输出a[0]中的数据
    int b;                      //定义b为int类型
    scanf("%d",&b);             //接收一个从键盘输入的整型数据b
    printf("--- %d\n",b);        //输出b值
    char c[256];                //定义字符型数组c
    scanf("%s",c);              //接收键盘输入字符串，存储于字符型数组c中
    printf("--- %s\n",&c[0]);   //输出数组中的字符串
    }
```

运算符

1.算术运算符

(1) +：加法运算符，用于两个变量或者两个表达式之间的加法运算。
(2) -：减法运算符，用于两个变量或者两个表达式之间的减法运算。
(3) *：乘法运算符，用于两个变量或者两个表达式之间的乘法运算。
(4) /：除法运算符，用于两个变量或者两个表达式之间的除法运算。
(5) %：求余运算符，用于计算两个变量相除之后的余数。
(6) ++：自增运算符。
(7) --：自减运算符。

C语言中有两个很有用的运算符：自增运算符“++”和自减运算符“--”，其中，“++”是运算数加1，而“--”是运算数减1，换句话说：x=x+1同++x；x=x-1同--x。自增和自减运算符可用在运算数之前，也可放在其后，例如：x=x+1可写成++x或x++。

但在表达式中这两种用法是有区别的。如自增或自减运算符放在运算数之前，则执行时先加1或减1运算，再进行赋值；运算符放在运算数之后，则执行时先赋值，再进行加1或减1运算。

2.赋值运算符

赋值运算符的含义是将一个数据赋给一个变量，虽然书写形式与数学中的等号相同，但两者的含义截然不同。由赋值运算符及相应操作数组成的表达式称为赋值表达式。其一般形式为：

变量名=表达式;

涉及算数运算符的复合赋值运算符共有5个，即+=、-=、*=、/=、%=。

+=这个运算符比较常用。如：

```
int a=23;
a+=1;//相当于a先加1，再赋值给a
```

3.关系运算符

关系运算符用于对两个数据进行比较，其结果是一个逻辑值（“真”或“假”），如“5>3”，其值为真。

C语言的比较运算中，“真”用非“0”数字来表示，“假”用数字“0”来表示。关系运算符有>、>=、<、<=、==、!=。

4.逻辑运算符

逻辑运算符也称布尔运算，C语言提供了三种逻辑运算符：&&、||、！。用逻辑运算符连接操作数组成的表达式称为逻辑表达式。逻辑表达式的值只有真和假两个值，用1表示真，用0表示假。

“&&”运算的特点是：只有两个操作数都为真，结果才为真；只要有一个为假，结果就为假。因此，当要表示两个条件必须同时成立时，可用“逻辑与”运算符连接这两个条件。

“||”运算的特点是：只要有一个操作数为真，结果就为真；只有两个操作数都为假，结果才为假。因此，当需要表示“或者…或者…”这样的条件时，可用“逻辑或”运算符连接这两个条件。

“！”运算的特点是：当操作数为非0值时，取反为0；当操作数为0值时，取反为1。

示例4.2.8-1 计算输入的日期是否为闰年

```
#include<stdio.h>
int main(int argc,char **argv){
int y;
scanf("%d",&y);
printf("%s\n"，((y%4==0 )&& (y%100！=0)) || (y%400==0)?"闰年"：  "平年");
//闰年判断条件
}
```

5.条件运算符

C语言提供了一个条件运算符，即“?:”，其语法格式如下：

a?b:c;

如果a是真，则结果为b，否则结果为c。

如：

```
int a=23,b=34,c=12;
a>b?a:b; //如果a>b，则表达式结果为a的值，否则，为b的值
a>b?a:b>c?b:c;  //右结合，相当于a>b?a:(b>c?b:c);
```

6.逗号运算符

逗号运算符可将多个表达式连接在一起，构成逗号表达式，其作用是实现对各个表达式的顺序求值，因此逗号运算符也称为顺序求值运算符。其一般形式如下：

表达式1,表达式2,…,表达式n;

7.长度运算符

sizeof()运算符是最容易混淆的。sizeof()本身是关键字并不是函数。用途是用来计算数据类型以及变量的长度。

示例4.2.8-2 判断打印输出的结果

```
#include<stdio.h>
#include<string.h>
int main(int argc,char **argv){
printf("%ld\n",sizeof(int));//代表用int类型分配的内存是4字节，打印输出的结果为4
int a;
printf("%ld\n",sizeof(a)); //代表a的内存是4字节，打印输出的结果为4
char buf[200];
printf("%ld\n",sizeof(buf)); //200代表buf内存长度为200字节，打印输出的结果为200
char buf1[200]="abc";
printf("%ld\n",strlen(buf1)); //buf1中字符串的长度为3，打印输出的结果为3
}
```

C语言的控制语句

分支语句

1．if

if语句主要有3种格式，分别是：

（1）if (表达式1)

```
{
        语句1
}
```

（2）if (表达式1)

```
{
        语句1
}
else
{
        语句2
```

（3）if (表达式1)

```
{
    语句1
}
else if (表达式2){
    语句2
}
....
else{           //可省略
    语句n
}
```

示例4.3.1-1 判断输入的年份是否为闰年，重新编写4.2.8-1代码。

```
#include <stdio.h>
#include <string.h>
int main(int argc,char **argv){
    int y;
    scanf("%d",&y);   //输入判断的年份
    if (y%4==0 && y%100!=0 || y%400==0)   //闰年判断条件
    printf("闰年\n");   //满足条件，输出闰年
    else
    printf("平年\n");   //不满足条件，输出平年
}
```

示例4.3.1-2 输入学分，打印优良中差不及格

```
#include <stdio.h>
#include <string.h>
int main(int argc,char **argv){
   int s;
    scanf("%d",&s);    //输入成绩
    if (s>100 || s<0)    //如果s小于0或者s大于100
       printf("输入错误\n");
    else if (s>=90)     //如果s大于等于90
       printf("优\n");
    else if (s>=80)     //如果s大于等于80
       printf("良\n");
    else if (s>=70)     //如果s大于等于70
       printf("中\n");
    else if (s>=60)     //如果s大于等于60
       printf("差\n");
    else
       printf("不及格\n");
}
```

2．switch

格式：

```
switch(表达式){
case 常量1：
    语句1;
    break;
case 常量2：
    语句2;
    break;
.......
default：
    语句n;
    break;
}
```

首先表达式和case后面的常量进行匹配，如果匹配成功，则开始向下执行代码，如果case中无法找到匹配的值，则进入default，开始向下执行代码。

在所有的case中，如果匹配成功，则不再继续匹配。break用来跳出switch和循环。

示例4.3.1-3 设计一个简单的计算器程序，要求根据用户从键盘输入的下列表达式计算结果。

```
#include <stdio.h>
#include <string.h>
int main(int argc,char **argv){
        int data1,data2;   //定义data1，data2为int类型
        char op;                                        //定义op为字符型
        printf("请输入表达式:");
        scanf("%d%c%d",&data1,&op,&data2);   //输入运算表达式
        switch(op)    //根据输入的运算符确定执行的运算
        {
                case'+':printf("%d+%d=%d\n",data1,data2,data1+data2);break;  //执行加法运算
                case'-':printf("%d-%d=%d\n",data1,data2,data1-data2);break;  //执行减法运算
                case'*':printf("%d*%d=%d\n",data1,data2,data1*data2);break;  //执行乘法运算
                case'/':                                        //执行除法运算
                if(0==data2)                            //检查除数是否为0，避免出现除零溢出错误
                        printf("Division by zero!");
                else
                        printf("%d/%d=%d\n",data1,data2,data1/data2);break;
                default:printf("Unknown operator!\n");
        }
}
```

4.3.2循环语句

1．do ... while

格式：

```
do {
                执行语句
}while(条件);
```

执行过程如下：在检查条件是否为真之前，该循环首先会执行一次语句，然后检查条件是否为真，如果条件为真的话，就会重复这个循环。

示例4.3.2-1 用do...while语句求0~9数值间的累计和

```
#include <stdio.h>
int main(int argc,char **argv){
        int i=0;              //初始化循环变量i值为0
        int sum=0;            //初始化sum值为0
        do{
        sum+=i;               //实现累加
        i++;                  //循环控制变量i增1
        }while(i<10);         //当i小于10时执行循环体
        printf("--- %d\n",sum);
    }
```

2．while

格式：

```
while(条件){
    执行语句
}
```

执行过程如下：首先判断表达式的值是否为真，如果为真，则执行循环体内的语句，然后再判断表达式是否为真，如果为真，再执行循环体内的语句，如此循环往复，直到表达式的值为假为止。

示例4.3.2-2 用while语句求0~9数值间的累计和

```
#include <stdio.h>
int main(int argc,char **argv){
    int i=0;          //初始化循环变量i值为0
    int sum=0;        //初始化sum值为0
    while(i<10){      //当i小于10时执行循环体
    sum+=i;           //实现累加
    i++;              //循环控制变量i增1
    }
    printf("--- %d\n",sum);
}
```

3．for循环

格式

```
for(表达式1;表达式2;表达式3){
    执行语句;
}
```

执行过程：首先求解表达式1的值，然后判断表达式2是否为真，如果为真，则执行循环体语句，然后求解表达式3的值；接下来再判断表达式2是否为真，如果为真，继续执行循环体语句及求解表达式3的值，直到表达式2为假为止。

示例4.3.2-3 用for语句求0~9数值间的累计和

```
#include<stdio.h>
int main(int argc,char **argv){
        int sum=0;   //初始化sum值为0
        for(i=0;i<10;i++){  //依次处理0~9数据
          sum+=i;         //实现累加
        }
        printf("--- %d\n",sum);
    }
```

示例4.3.2-4 求1-100之间的能被3整除但不能被7整除的数的和

```
#include<stdio.h>
int main(int argc,char **argv)
{
      int i,sum;
      i=1,sum=0;
      while(i<=100)
      {
         if (i%3==0&&i%7!=0){   //判断能被3整除但不能被7整除
         sum+=i; }
          i++;
        }
printf("sum=%d\n",sum);
}
```

4．循环跳转

通常计算机中的语句是按照它们的编写顺序、逐条执行的。C语言提供了4种跳转语句：goto语句、break语句、continue语句和return语句。

- goto的作用是改变程序的流向，转去执行语句标号所标识的语句。通常与条件语句配合使用，用来实现条件转移、构成循环、跳出循环体等功能。
- return结束当前函数，如果在循环里面，同时结束循环；如果在main函数中，可结束程序。
- exit(0)结束程序，在任何函数中或任何.c文件中都可结束程序。
- break结束switch和循环，不能用在goto中向后跳出。
- continue向前跳到条件处。中止当前这一次的循环。

示例4.3.2-5 计算半径为1-100之间的圆的面积，要求面积小于100

```
#include <stdio.h>
int main(int argc,char **argv){
int i;
    double s=0;
    for(i=1;i<=100; i++){       //定义循环变量i初值与终值
       s=3.14*i*i;              //计算圆的面积
       if (s>=100) break;       //若s值大于等于100，则跳出循环
       printf("--- s=%lf\n",s);
    }
}
```

5．循环嵌套

如果将一个循环语句放在另一个循环语句的循环体中，就构成了嵌套循环。while、do...while、for这三种循环语句均可以相互嵌套，即在每种循环体内，都可以完整地包含另一个循环结构。嵌套的循环，也称多重循环，常用于解决矩阵运算、报表打印等问题。

示例4.3.2-10 输出打印九九乘法表（下三角）

```
#include <stdio.h>
int main(int argc,char **argv){
   int i,j;
   for(i=1;i<=9;i++){  //外层循环控制被乘数i从1变化到9
      for(j=1;j<=i;j++){   //内层循环控制被乘数j从1变化到i
         printf("%d*%d=%-4d ",j,i,i*j);  //输出i行j列的值
         }
         printf("\n");  //控制输出换行，准备输出下一行
    }
}
```

输出结果为:

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

02

数组与字符串

数组的定义

数组是相同类型数据的集合，内存空间连续。

格式：

数据类型 数组名 [元素的个数];

例如：

int a [5];

上式定义了一个数组，数组名为a，数组里有5个元素，元素数据类型为整型。

数组本质探讨

数组的本质就是一种最简单的数据结构。所谓的数据结构狭义上来讲就是数据的存储的方式。在实际应用中数组的应用相当广泛，基本每个程序都用数组，有的是字符串数组，有的是纯数字的。数组还经常和指针联系在一起，数组的指针，指针数组等。

一维数组和二维数组

任意一种数据类型在内存都有它的存在形式，对于数组，可以把它抽象为一块内存条。也就是说二维数组，三维数组在内存中是不同的内存条。所以所谓的二维数组或者更高维度的数组都是一维数组，只不过需要不同的抽象模型，所以用到不同维度数组。

二维数组的定义

一维数组是数据的数组，每个单元至少是一个数据。多维数组是数组的数组，每个元素是一个数组。

格式：

类型 数组名[数组的个数][元素个数];

如：

(1)int a[2][5]; //a是二维数组名，包含2个一维数组

(2)int b[3][5][6]; //3个平面，5个线，6个点

(3)int c[8][3][5][6]; //8个立方，3个平面，5个线，6个点

3.二维数组的初始化

方法一：把二维数组当成一维数组，例如：

int a[3][4]={1，2，3，4，5，6，7，8，9，10，11，12}; //定义二维数组，3行4列

int a[][4]={1，2，3，4，5，6，7，8，9，10，11，12}; //定义二维数组，3行4列，行数可省略，列数不可省略

int a[][4]={1，2，3}; //定义二维数组1行4列，没有初始化的默认为0

方法二：把二维数组当成多组一维数组，例如：

int a[3][4]={{1，2，3，4}，{5，6，7，8}，{9，10，11，12}}; //{}为一个组

int a[][4]={{1，2}，{6，7，8}，{9}};

//a[0][2]和a[0][3]为0，a[1][3]为0，a[2][1]、a[2][2]和a[2][3]为0，没有初始化的默认为0。

二维字符数组的初始化

```
//a[0][0]、a[0][1]、和a[0][2]数据为'1'、'2'、'3'，其它默认为0
char a[3][4]={'1'，'2'，'3’ };

// a[0][0]、a[0][1]、和a[0][2]数据为'1'、'2'、'3'，其它默认为0
char a[3][4]={"123"};

//没有初始化默认为0
char a[][4]={"123"，"ad"，"YYZ"};
```

4.4.4 字符串

1.字符

2.字符串与数组

C语言无字符串变量，而是用字符数组处理字符串，字符串的结束标志为'\0'。

字符串的本质是字符数组，我们先来看看怎么初始化一个字符串。例如：

```
1.    char s[5]={"asdf"}; //字符串长度为4，数组长度是5
2.    char s[5]="asdf"; //同上
3.    char s[]="asdf"; //数组长度是5，字符串长度为4
```

3.字符串操作的常用函数

在C语言本身的库中，<string.h>这个头文件里封装了对字符串操作的函数，可以直接用，这些函数也可以自己实现。

常用操作字符串函数列表及重点函数讲解：

（1）复制字符串

char *strcpy(char *dest,const char *src);

（2）计算字符串长度，size_t类型是int

size_t strlen(const char *s);

（3）比较两个字符串

int strcmp(const char *s1,const char *s2);

s1=s2，返回0；s1>s2 返回正数；s1<s2 返回负数。

（4）将字符串src连接在dest后面

char *strcat (char *dest,const char *src);

例:

```
int main()
{
char str1[256]=" asdf ";    //定义字符型数组str1并初始化初值为"asdf"
char str2[]="dsfasdf";     //定义字符型数组str2并初始化初值为"dsfasdf"
strcat(str1,str2);          //将字符串str2连接在str1后面
printf("%s\n",str1);        //输出新字符串的值为" asdfdsfasdf"
}
```

示例4.4.1-1 字符串格式输出综合应用

```
#include<stdio.h>
int atoi(const char *nptr);   //函数声明
int main(int argc,char **argv){
   int a=23;
   double f=23.2;
   char buf[256];
   sprintf(buf,"%d",a);
   printf("%s\n",buf);
   sprintf(buf,"%lf",f);
   printf("%s\n",buf);
   sprintf(buf,"aaa：%s%s%d%x%o%d：%d","a","b",4,20,20,2,20);
   printf("%s\n",buf);
}
```

03

函数与指针

4.5函数

C语言是一种通用的高级编程语言，拥有丰富的函数库和强大的系统编程能力。在C语言中，函数是模块化和可重用的代码段，用于实现特定的功能。函数将一系列的语句组织在一起，可以接受输入参数并返回值。

函数的定义包括函数名、参数列表、返回类型和函数体:

- 函数名是唯一的标识符，用于调用函数。
- 参数列表指定了函数接受的参数，可以是任意数量和类型的参数。
- 返回类型指定了函数返回值的类型，可以是基本数据类型、指针类型或结构体类型。
- 函数体是实现具体功能的语句块。

C语言中的函数可以分为以下几类：

1. 标准库函数（Standard Library Functions）：输入输出函数（如scanf和printf）、字符串处理函数（如strcpy和strlen）、数学库函数（如sqrt和sin）等。这些函数被包含在标准库中，可以直接调用。
2. 用户自定义函数（User-defined Functions）
3. 递归函数（Recursive Functions）：递归函数是一种特殊的函数，它在函数体内部调用自身。
4. 内联函数（Inline Functions）：内联函数是一种将函数的代码嵌入到调用点的编译器特性。内联函数的主要目的是减少函数调用的开销，提高程序的执行速度。
5. 回调函数（Callback Functions）：回调函数是一种通过函数指针作为参数传递的函数。

无论是标准库函数、用户自定义函数还是其他类型的函数，都可以根据需要进行组合和调用，实现复杂的程序逻辑和功能。

C语言中的函数可以分为以下几类：

标准库函数（Standard Library Functions）	输入输出函数（如scanf和printf）、字符串处理函数（如strcpy和strlen）、数学库函数（如sqrt和sin）等。这些函数被包含在标准库中，可以直接调用。
递归函数（Recursive Functions）	递归函数是一种特殊的函数，它在函数体内部调用自身。通过递归调用，函数可以解决涉及重复计算的问题，实现更简洁和优雅的算法。但需要注意递归函数应当包含条件语句以避免无限递归。
回调函数（Callback Functions）	回调函数是一种通过函数指针作为参数传递的函数。它可以在程序运行时动态地指定回调函数，以实现特定的行为。常见的例子是在图形界面编程中，通过回调函数响应用户的操作。
用户自定义函数（User-defined Functions）	在程序中，我们可以自定义函数，通过函数名调用并执行特定的功能。用户自定义函数可以接受参数并返回值，可以将程序分解成多个函数，提高代码的模块化和可读性。
内联函数（Inline Functions）	内联函数是一种将函数的代码嵌入到调用点的编译器特性。内联函数的主要目的是减少函数调用的开销，提高程序的执行速度。通过使用关键字inline来声明内联函数。

4.5.1函数的定义

在编程中，函数定义是指定义一个函数的结构、功能和操作的代码块。函数定义包含函数的名称、参数列表、返回类型、函数体等组成部分。

以下是一个函数定义的一般格式：

PLAINTEXT

```
返回类型 函数名(参数列表)
{
    // 函数体
    // 执行的操作和逻辑
    // 可能包括返回语句
}
```

- 函数名：函数的名称（最好见名知义），用于调用和引用函数。
  - 合法的函数名由字母、数字和下划线组成
  - 一般约定以字母开头

```c
// 有意义的函数名
int max(int a, int b){}          // 求最大值
int sum(int a, int b, int c){}   // 求和
// 没有意义的函数名
void zaaaaaaaa(){}
void xyc123(char c){}
```

形式参数与实际参数

在使用函数时，经常会用到形式参数和实际参数。两者都叫做参数，那么二者有什么关系？

- 通过名称理解
  - 形式参数，按照名称进行理解就是形式上存在的参数。
  - 实际参数，按照名称进行理解就是实际存在的参数。

```c
// 没有参数
void show(){}
void print(void){}
// 有参数
int max(int a, int b){}           // 2个参数, a, b 都是形参
int sum(int a, int b, int c){}    // 3个参数, a, b, c 都是形参
// 给形参初始化 -- error
int show(int a = 9){}             // error, 语法错误
```

形式参数与实际参数

下面是一个简单的函数定义，其中 `a` 和 `b` 就是形式参数：

```c
int add(int a, char b)
{
    int sum = a + b;  // b 会进行隐式类型转换 char->int
    return sum;
}
```

下面是一个函数调用的示例，其中 整数 `2` 和 字符 `a` 就是实际参数：

```c
int result = add(2, 'a');
```

main的参数

在前面介绍函数定义的内容中，曾在讲解函数体时提到过主函数main的有关内容，下面在此基础上对main函数的参数进行介绍。

在运行程序时，有时需要将必要的参数传递给主函数。主函数main的形式参数如下：

```
main (int argc, char* argv[] )
```

两个特殊的内部形参argc和argv是用来接收命令行实参的，这是只有主函数main才能具有的参数。

argc参数

argc参数保存命令行的参数个数，是整型变量。这个参数的值至少是1，因为至少程序名就是第一个实参。

argv参数

argv参数是一个指向字符指针数组的指针，这个数组里的每一个元素都指向命令行实参。所有命令行实参都是字符串，任何数字都必须由程序转变成为适当的格式

内联函数

在C语言中，内联函数（Inline Function）是一种用于优化代码执行效率的机制。内联函数在编译时将函数的代码直接插入到调用它的地方，而不是通过函数调用的方式执行，从而减少了函数调用的开销，提高了代码的执行速度。

C语言的内联函数使用 `inline` 关键字来声明。将函数声明为内联函数只是给编译器一个建议，告诉它将函数的代码插入到调用的地方。编译器可以选择忽略内联函数的建议，继续将函数编译为常规函数。

以下是内联函数的一般格式：

```c
inline 返回类型 函数名(参数列表)
{
    // 函数体
    // 执行的操作和逻辑
    return 表达式; // 可选的返回语句
}
```

具体来说，内联函数的使用规则和特点包括：

C

```c
#include <stdio.h>
inline int multiply(int a, int b)
{
    return a * b;
}

int main()
{
    int result = multiply(10, 20);
    printf("result = %d\n", result);
    return 0;
}
```

5. 递归函数

在C语言中，递归函数是一种函数调用自身的技术。递归函数可以用于解决需要重复执行相同操作的问题，将问题不断分解为更小的子问题，并通过函数调用自身来处理这些子问题，直到达到最简单的情况。

递归函数的基本原理是把大问题分解成一个或多个相同的小问题，然后通过调用自身来解决这些小问题。递归函数包括两部分：基本情况和递归调用:

- 基本情况指的是递归函数终止的条件，当函数遇到基本情况时，不再调用自身，直接返回结果。
- 递归调用指的是在函数内部调用自身来解决规模较小的子问题。通过递归调用，问题的规模不断缩小，直到达到基本情况，然后逐层回溯，获得最终的结果。

需要注意的是，递归函数必须具备终止条件，否则会陷入无限递归的循环中，导致程序异常。

4.6.1指针本质概述

程序运行过程中产生的数据都保存在内存中，内存是以字节为单位的连续存储空间，每个字节都有一个地址，这个地址就称为指针。通过指针就可以获取保存在内存中的数据的地址。例如：

```
int a=10;
```

编译器会根据变量a的类型int，为其分配4个字节地址连续的存储空间。假如这块连续空间的首地址为0x0037FBCC，那么这个变量占据0x0037FBCC~0x0037FBCF这四个字节的空间，0x0037FBCC就是变量a的地址。

在C语言中，内存单元的地址称为指针，专门用来存放地址的变量。

指针的定义形式如下：

变量类型 *变量名;

（1）变量类型指的是指针指向的变量的数据类型。即指针类型在内存中的寻址能力，如char类型决定了指针指向1个字节地址空间，int类型决定了指针变量指向4个字节地址空间。

（2）*表示了定义的变量是一个指针变量类型。

（3）变量名是存储内存地址的名称。即指针变量，其命名遵循标识命名规则。

例如：

```
char *i;                    //char类型的指针变量i
int *t;                 //int类型的指针变量t
double *c;                  //double类型的指针变量c
long *a;                    //long类型的指针变量a
long double *s;             //long double类型的指针变量s
unsigned int *T;            //unsinged int类型的指针变量T
```

4.6.2连续空间的内存地址

int a[8]={9，2，3，4，5，6，7，8};

4.6.3指针变量

1.栈区内存

栈内存由编译器自动分配和释放。栈具有先入后出的特点，由于栈由高地址向低地址增长，因此先入栈的数据地址较大，后入栈的数据地址较小。数据入栈后栈顶地址减小，数据出栈时栈顶地址增大。栈内存主要用于数据交换，如函数传递的参数、函数返回地址、函数的局部变量等。

```
int main(){
int a=12;    //局部变量
int* p=&a;  //给p赋值，而不是给*p赋值
   }
```

*p就是变量名，值是12； p[0]也是变量名，值是12；p[1]的表达方式是错的，因为此时p开始的内存中只有一个4字节。

2.数据段内存

数据段内存通常是指用来存放程序中已初始化的全局变量的一块内存区域。数据段属于静态内存分配。

```
1. int a=12;   //全局变量
2. int main(){
3. int* p=&a; //给p赋值，而不是给*p赋值
4. p[0]=23;
5.     }
```

3.堆区内存

堆是不连续的内存区域，各块区域由链表将它们串联起来。该区域一般由程序员分配或释放，若程序员不释放，程序结束时可能由操作系统回收（程序不正常结束则回收不了）。堆区的上限是由系统中有效的虚拟内存决定的，因此获得的空间较大，而且获得空间的方式也比较灵活。

例如：

```
1. int main(){
2. int* p=(int*)malloc(4);
3. *p=23;
4.     }
```

整型的指针类型有以下含义：

（1）定义变量的内存地址；

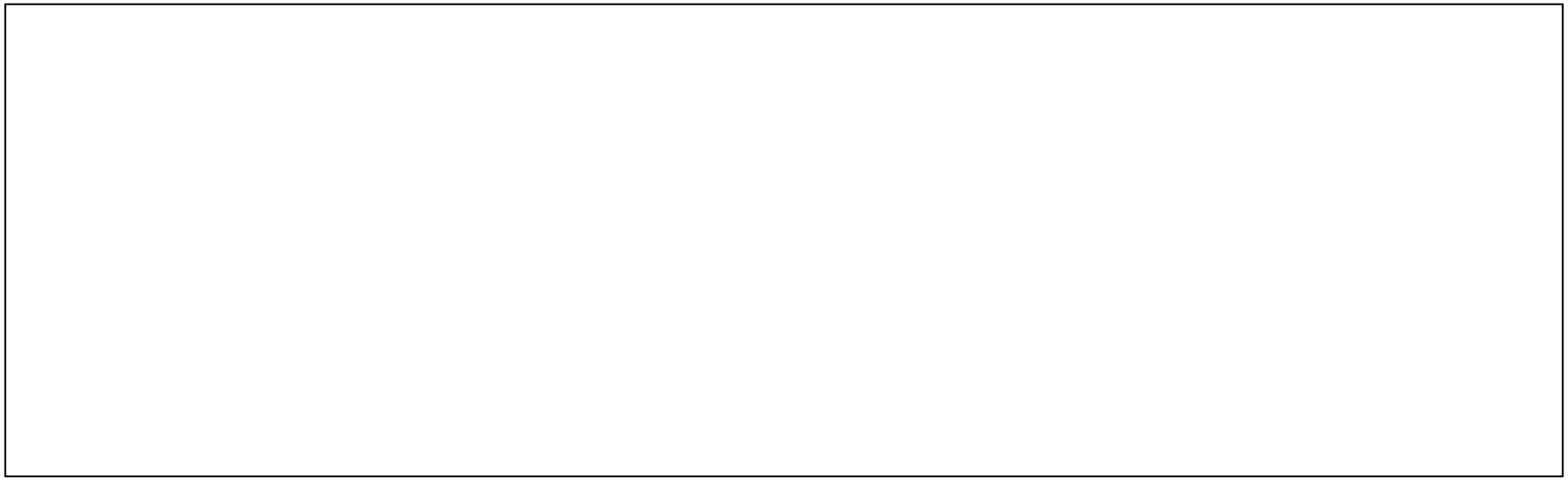
（2）这个地址存的是整型的变量地址；

（3）*地址是整型的内存名；

（4）地址代表所控制的内存长度为4字节。

示例4.6.3-1 编写一个函数，用来申请一个内存，存储两个数据后，将地址返回，打印这两个数。

```
#include <malloc.h>
#include <stdio.h>
int* creatmem(int n){
    int i, *tmp=(int *)malloc(4*n);  //堆内存申请
    for(i=0;i<n;i++)
    scanf("%d",tmp+i);
    return tmp;
}
int main(){
    int i;
    int *a=creatmem(2); //a存的是tmp的值
    for(i=0;i<2;i++){
      printf("%d\n",a[i]);
```



4.6.4指针数组和指针的指针

1.指针数组

指针不仅可用于指向一个数组，还可作为指针数组。由若干类型相同的指针所构成的数组，称为指针数组。由定义可知，指针数组的每个元素都是一个指针，且这些指针指向相同数据类型的变量。指针数组的定义一般形式为：

类型关键字 *数组名[常量N];

在解释下面的变量定义语句时，说明符[]的优先级高于*，即先解释[ ]，再解释*。

char *pStr[5]; //pStr是由5个元素的数组，每个元素都是指向一个char型数据的指针

char *suit[4]={ "Hearts"，"Diamonds"，"Clubs"，"Spades"};

suit是一个拥有4个元素的字符指针数组，数组的每个元素的类型都是一个“指向字符的指针”。存放在数组中的4个元素分别是"Hearts"，"Diamonds"，"Clubs"，"Spades"。

示例4.6.4-1 数组指针应用

```
#include <stdio.h>
int main(int argc,char **argv){
int *A[5];
int a=1,b=2,c=3;
A[0]=&a;
A[1]=&b;
A[2]=&c;
printf("%d,%d,%d\n",*A[0],*A[1],*A[2]);
}
```

2.指针的指针

指针还可以指向一个指针，即指针中存储的是指针的地址，这样的指针称为二级指针。

使用二级指针可以间接修改一级指针的指向，也可以修改一级指针指向的变量的值。

二级指针定义格式如下：

变量类型 **变量名;

变量类型就是该指针变量指向的指针变量所指变量的数据类型，两个符号“*”表明这个变量是个二级指针变量。

04

预处理指令

4.7.1 宏定义

宏定义是最常用的预处理功能之一，它用于将一个标识符定义为一个字符串。这样，在源程序被编译器处理之前，预处理器会将标识符替换成所定义的字符串。根据是否带参数，可以将宏定义分为不带参数宏定义和带参数的宏定义。

格式：

#define 宏名 [数据]

注意：后面没有分号

1.宏的基本用法

(1)在多处使用3.14时，如果使用宏，方便改动

```
#define PI 3.14      //宏定义
int main(){
   int r=2;
   double s=r*r*PI;  //本质，在编译时，将PI替换为3.14
   }
```

(2)将数字用来代表具体函义的值，此时使用宏。

```
#define SUN  0
#define MON  1
int main(){
    int a;
    scanf("%d",&a);
    switch(a){
    case SUN:printf("周日") break;
    case MON: printf("周一") break;
    ...
     }
}
```

2.带参数的宏

格式：#define 宏(参数列表) 表达式

```
#define MUL(x,y) x*y
int main(){
int a=MUL(12,2);    //a=12*2=24
int b=MUL(12+1,2);  // b=12+1*2=14
}
```

注意事项：写这类的宏，一定要在后面表达式的参数上加（）。

```
#define MUL(x,y) (x)*(y)
int main(){
int a=MUL(12,2);    //a=12*2=24
int b=MUL(12+1,2);  //b=(12+1)*(2)=26
}
```

4.7.2 #include

除宏定义外，文件包含也是一种预处理语句，它的作用是将一个源程序文件包含到另外一个源程序文件中。

文件包含其实就是一种头文件引入，它使用#include实现，格式如下：

#include <文件名>

#include "文件名“

第一种格式：编译系统在系统指定的路径下搜索尖括号中的文件。

第二种格式：系统首先会在用户当前工作的目录中搜索双引号（""）中的文件，如果找不到，再按系统指定的路径进行搜索。

编写C语言程序时，一般使用第一种格式包含C语言标准库文件，使用第二种格式包含自定义的文件。

4.7.3条件编译

一般情况下，C语言程序中的所有代码都要参与编译，但有时出于程序代码优化的考虑，希望源代码中一部分内容只在指定条件下进行编译。这种根据制定条件决定代码是否需要编译的称为条件编译。

（1）#ifndef .... #endif 判断是否定义了宏，如果没定义则编译，通常用于头文件中。

（2）#ifdef .... #endif 判断是否定义了宏，如果定义了则编译，通常用于系统移植。

示例4.7.3-1 条件编译应用1

```
#include<stdio.h>
#define LINUX  //定义宏LINUX
int main(){
    double s=12;
    #ifdef WIN
    int r=2;
    s=r*r*3.14;
    #endif
    printf("%lf\n",s);
}
```

（3）#if…#else…#endif

```
#if 值
        代码
 #elif
        代码
  ....
#else
        代码
 #endif
```

作业3：

1.2022年是虎年，问29年后是哪一年?
年份的顺序是：

鼠	牛	虎	兔	龙	蛇	马	羊	猴	鸡	狗	猪
0	1	2	3	4	5	6	7	8	9	10	11

2.求1+1/2+2/3+3/5+5/8+8/13+....的值。

3. 求 s=a+aa+aaa+aaaa+.... 的值，其中 a 是一个数字。例如 2+22+222+2222+ 22222(此时共有5个数相加)

4.编写程序统计输入的一串字符中大写字母和小写字母的个数。

5.将输入英文语句中每个单词的第一个字母改写成大写，然后输出该语句。

重庆理工大学/电气学院

CHONGQING UNIVERSITY OF TECHNOLOGY

嵌入式Linux系统开发教程

—基于ARM处理器通用平台（arm9-arm11-contexA系列）