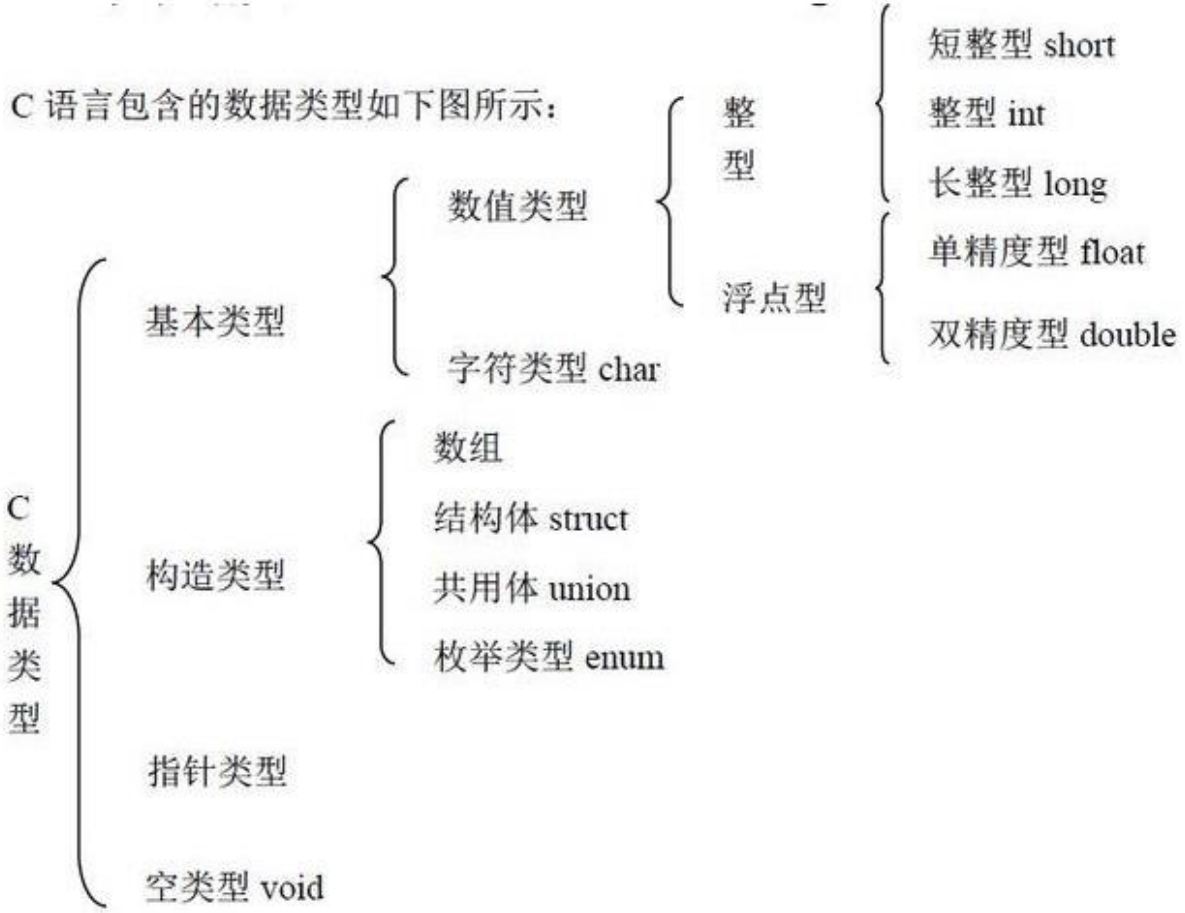
习题课

C语言程序设计

数据类型

C 语言包含的数据类型如下图所示：



short 占据的内存大小是2 个byte；
int占据的内存大小是4 个byte；
long占据的内存大小是4 个byte；
float占据的内存大小是4 个byte；
double占据的内存大小是8 个byte；
char占据的内存大小是1 个byte。

C语言优先级

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	[]	数组下标	数组名[常量表达式]	左到右	
	()	圆括号	（表达式）/函数名(形参表)		
	.	成员选择（对象）	对象.成员名		
	->	成员选择（指针）	对象指针->成员名		
	++	后置自增运算符	++变量名		单目运算符
	--	后置自减运算符	--变量名		单目运算符
2	-	负号运算符	-表达式	右到左	单目运算符
	(类型)	强制类型转换	(数据类型)表达式		
	++	前置自增运算符	变量名++		单目运算符
	--	前置自减运算符	变量名--		单目运算符
	*	取值运算符	*指针变量		单目运算符
	&	取地址运算符	&变量名		单目运算符
	!	逻辑非运算符	!表达式		单目运算符
	~	按位取反运算符	~表达式		单目运算符
	sizeof	长度运算符	sizeof(表达式)		
3	/	除	表达式/表达式	左到右	双目运算符
	*	乘	表达式*表达式		双目运算符
	%	余数（取模）	整型表达式/整型表达式		双目运算符
4	+	加	表达式+表达式	左到右	双目运算符
	-	减	表达式-表达式		双目运算符

5	<<	左移	变量<<表达式	左到右	双目运算符
	>>	右移	变量>>表达式		双目运算符
6	>	大于	表达式>表达式	左到右	双目运算符
	>=	大于等于	表达式>=表达式		双目运算符
	<	小于	表达式<表达式		双目运算符
	<=	小于等于	表达式<=表达式		双目运算符
7	==	等于	表达式==表达式	左到右	双目运算符
	!=	不等于	表达式!= 表达式		双目运算符
8	&	按位与	表达式&表达式	左到右	双目运算符
9	^	按位异或	表达式^表达式	左到右	双目运算符
10	|	按位或	表达式|表达式	左到右	双目运算符
11	&&	逻辑与	表达式&&表达式	左到右	双目运算符
12	||	逻辑或	表达式||表达式	左到右	双目运算符
13	?:	条件运算符	表达式1? 表达式2: 表达式3	右到左	三目运算符
14	=	赋值运算符	变量=表达式	右到左	
	/=	除后赋值	变量/=表达式		
	*=	乘后赋值	变量*=表达式		
	%=	取模后赋值	变量%=表达式		
	+=	加后赋值	变量+=表达式		
	-=	减后赋值	变量-=表达式		
	<<=	左移后赋值	变量<<=表达式		
	>>=	右移后赋值	变量>>=表达式		
	&=	按位与后赋值	变量&=表达式		
	^=	按位异或后赋值	变量^=表达式		

	|=	按位或后赋值	变量|=表达式		
15	,	逗号运算符	表达式,表达式,...	左到右	从左向右顺序运算

说明：

同一优先级的运算符，运算次序由结合方向所决定。

简单记就是：！ > 算术运算符 > 关系运算符 > && > || > 赋值运算符 [4]

```
short x=1,y=2,z=0x0043;
char c1=4,c2=16;
int i=0,j=1;
```

计算:

(1)j++ ? ++j : i++ 3

(2)++c1|c2<<2 69

(3)~x<<1&0x0f|y 14

(4)c1<<++i+i 16

(5)c2=z 67 或 0x43

```
char a=4,b=6,c;
short x=0xe0ff,y=0xff23;
#define A a+b*a+b
```

计算:

(1)a^b<<2　　28

(2)++a|b<<2　　30

(3)c=x>>8　　0xe0 或-32

(4)a*A+b　　52

(5)a>>2 ? a+b ? y&0x004f : b/2 : a-b ? y&0x003e : a*2　　3

a>>2 ?(a+b ? y&0x004f : b/2): (a-b ? y&0x003e : a*2)

```
char s1[]="abcdefg";
char s2[]="hijklmn";
struct T{
    char *s;
    float x;
    int y[3];
}a[]={{s1,102.3,{-1,1,2}},{s2,78.2,{0,2,1}}},*p=a;
```

计算:

(1)(p+1)->s[5]　　m

(2)*(++p)->y+2　　2

(3)++ *p->y　　0

(4)p[0].y[0]+*((a+1)->y+1)　　1

(5)(p++)->s[0]='k',p->s[a[1].y[1]-a[0].y[2]]　　h

```
char s[]="abcdefg";
char t[]="hijkl";
int x[4]={-1,0,3,2},y[3]={5,6,7},z[5]={0,10,3,4,1};
struct T{
    char s;
    char *t;
    int *x;
} a[]={{'s',s,x},{'t',"mnop",y},{'x',t,z}},*p=a;
```

计算:

(1)*(p->t ++) a

(2)(++p)->x[z[4]] 6

(3)++ *p->x 0

(4)(*p).t[(p+1)->x[1]] g

```
char s[]="abcdefg";
char t[]="hijkl";
int x[4]={-1,0,3,2},y[3]={5,6,7},z[5]={0,10,3,4,1};
struct T{
    char s;
    char *t;
    int *x;
} a[]={{'s',s,x},{'t',"mnop",y},{'x',t,z}},*p=a;
```

计算：

(5)*(p->x+2)+ *(p++ ->x ++)　　2

(p++ ->x ++)的值为-1，(p->x+2)的值为3

(6)p->t[*(p->x+2)+ *(p++ ->x ++)]　　c

p->t[2]是c，右边的p++不会影响最左边的p->t[]

(7)*(p++)->t=p->t[3],*p->t - *a->t　　9

*(p++)->t=p->t[3]使得a[0].t的值为“dbcdefg”,p指向第二个元素，*a->t的值为d,*p->t的值为m，‘m’-‘d’=9

程序改错：

下面的程序计算并输出1!,2!,3!,4!,5!的值。

```
#include<stdio.h>
int factorial (int n){
    int k=1;
    k*=n;
    return k;
}
int main(void){
    int i;
    for(i=1;i<6;i++){
        printf( “%d\n",factorial(i)) ;
    }
    return 0;
}
```

k乘上n倍

factorial(i)返回 i！ 的值

答案：将k声明为static int k =1;

写出程序的运行结果

```
# include<stdio.h>
void fun (unsigned long *n){
    unsigned long x=0, i;
    int t;
    i=1;
    while(* n){
        t=*n%10;
        if(t%2!=0){
            x=x+t*i;
            i=i * 10;
        }
        *n=*n/10;
    }
   * n=x;
}
```

```
int main(){
    unsigned long n= 2356789;
    fun(&n) ;
    printf ("\nThe result is:
        %ld\n",n);
     return 0;
}
```

答案：The result is:3579

需要写出程序的输出结果

完善程序：

本程序的功能是:输人一个字符串存放到字符数组s中，接着将s中连续的多个空格压缩成一个空格,并输出压缩空格后的字符串。

```
#include<stdio.h>
int main() {
    char s[100];
    int flag=_____0_____, i=0,j=0;
    gets(s) ;
    while(_____s[i]!='\0'_____){        //循环结束条件
        if (s[i]!=' ') { s[ j++]=s[i]; flag=0; }
        else if (_____flag==0_____) { s[ j++]=s[i]; flag=1; }
        i++;
    }
    s[ j]='\0';puts(s);return 0;
}
```

flag=0，代表s[j]存放的是非空格字符；flag=1，代表f[j]存放了空格字符，再次遇到空格的时候需要省略s[i]的赋值，当遇到非空格符时，重新置为0。