C 语言开卷总结资料

目录

ASCII 码表

1.范围：一共 128 个字符，从 0-127.
2.先出现 ABCDE，后出现 abcde.

运算优先级

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		
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		
	++	自增运算符	++变量名/变量名++		单目运算符
	--	自减运算符	--变量名/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

运算时的类型转换

1.算数表达式的类型转换（包括比大小）
有符号会自动变为无符号数

2.赋值运算的类型转换
右值会自动变为左值

内存分配状况

<table>
<tr><td colspan="3">程序代码区</td></tr>
<tr><td rowspan="2">数据区</td><td>静态存储区</td><td>外部变量和静态变量</td></tr>
<tr><td rowspan="2">动态存储区</td><td>堆区，用于动态内存分配</td></tr>
<tr><td>堆栈区</td><td>存放自动变量和函数的形参</td></tr>
</table>

常见算法

进制转换算法

将 N 进制表示为十进制数

```
#include<stdio.h>
#define N 8
int transN_10( char radix[] );
int transN_10( char radix[] )
{
int tens=0;
int i=0;

while(radix[i]!='\0')
tens + = tens * N + ( radix[i++] - '0' );
```

```
return tens;
}
```

将十进制表示为 N 进制数

```
#include<stdio.h>
#include<stdlib.h>
#define N 8
char* trans10_N( int );
char* trans10_N( int tens )
{
char radix[20];
    itoa(tens, radix, N);
    return radix;
}
```

数字逆序排列算法

```
#include<stdio.h>
int reverse_number(int);
int reverse_number(int a)
{
    int rev[20];
    int i, j, b, d;

    for(i = 0; a ; i ++)
    {
        rev[i] = a % 10;
        a /= 10;
    }
    for(j = i - 1, b = 0, d = 1; j >= 0; j --)
    {
        b += rev[j] * d;
        d *= 10;
    }
    return b;
}
```

选择排序法(113 页)

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

```
#define N 10
void func(int a[],int n)
{
    int i,j;
    int t;
    for(i=0;i<n-1;i++)
    {
    for(j=i+1;j<n;j++)
    {
        if(a[i]>a[j])
        {
            t=a[i];
            a[i]=a[j];
            a[j]=t;
        }
    }
    }
}
```

冒泡排序法

```
#include<stdio.h>
#define N 10
void func(int a[],int n)
{
    int i,j;
    for(i=0;i<n-1;i++)
    {
        for(j=0;i<n-1-I;j++)
        {
            if(a[j]>a[j+1])
            {
            t=a[j];
            a[j]=a[j+1];
            a[j+1]=t;
            }
        }
    }
}
```

分离整型数的各位数字

方法 1：用 itoa 函数，借助字符串转换数字。

```
#include<stdio.h>
#include<stdlib.h>
void divide1(int number, int* array)
{
    int i;
    char str[20];
    itoa(number,str,10);
    for(i=0;str[i]!='NULL';i++)
    {
        array[i]=str[i]-'0';
    }
}
```

方法 2：用计算分离各个数字

求最大公约数，最小公倍数

```
#include<stdio.h>
void main()    /*  辗转相除法求最大公约数  */
{
    int m, n, a, b, t, c;
    printf("Input two integer numbers:\n");
    scanf("%d%d", &a, &b);
    m=a;    n=b;
    while(b!=0)  /* 余数不为 0，继续相除，直到余数为 0 */
    {
        c=a%b;
        a=b;
        b=c;
    }
    printf("The largest common divisor:%d\n", a);
    printf("The least common multiple:%d\n", m*n/a);
}
```

生成随机数（73 页随机生成加减运算，报告 2 实验 7 随机猜数字）

方法一：

1.用 srand(time(NULL))进行播种。

2.用 rand()产生随机数。

3.用取余运算来约束随机数的范围。a%b 的结果在 0~b-1 之间。

4.每生成一个随机数，都要额外重新播种。

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

```
#include<stdlib.h>
#include<time.h>
#define N 10
#define M 5
int main()
{
    int a;
    srand(time(NULL));
    a=rand();//生成无范围的随机数
    srand(time(NULL));
    a=rand() % N;//利用取余运算，生成 0~N-1 的随机数
    srand(time(NULL));
    a=rand() % N +M;//利用取余运算和加法运算，生成 M~N+M-1 的随机数
    return 0;
}
```

学生成绩统计

```
#include<stdio.h>
#include<stdlib.h>
#define N 10
struct student
{
    char name[20];
    char sex;
    char number[20];
    float score[4];
    float average;
};
int main()
{
    struct student stu[N];
    struct student* pstu[N+1];


    return 0;
}
```

输入成绩函数

```
void getch(struct student*,int);
void getch(struct student stu[],int n)
```

```
{
    int i,j;
    char str[20];
    for(i=0;i<n;i++)
    {
        printf("name:\n");
        gets(stu[i].name);
        printf("sex:\n");
        gets(str);
        stu[i].sex=str[0];
        printf("number:\n");
        gets(stu.number);
        for(j=0;j<4;j++)
        {
            printf("class %d:\n",j+1);
            gets(str);
            stu[i].score[j]=atof(str);
        }
    }
}
```

计算个人平均分函数

```
void aver(struct student[],int);
void aver(struct student stu[],int n)
{
    int i,j;
    for(i=0;i<n;i++)
        stu[i].average=0;
    for(i=0;i<n;i++)
    {
        for(j=0;j<4;j++)
            stu[i].average+=stu[i].score[j];
        stu[i].average/=4;
    }
}
```

计算单科平均分函数

```
void avers(struct student[],int);
void avers(struct student stu[],int n)
{
    static float aver_score[4]={0};
```

```
    int i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<n;j++)
            aver_score[i]+=stu[j].score[i];
        aver_score[i]/=n;
    }
}
```

数据

printf 格式

1.输出百分号反斜杠等字符，printf(“%%　\\　\’　\”　”);
2.
%u，输出**无符号**十进制数
%o，输出**无符号**八进制数
%x，%X，输出**无符号**十六进制数。x 的大小写标明 a~f 的大小写。
%e，%E，输出指数形式的浮点数，e 的大小写对应指数中的大小写。
3.
printf(“%md”,a) 输出**至少** m 位大小的整型数。若少，则**左**补空格。若多，则全部输出。
printf(“%-md”,a) 输出**至少** m 位大小的整型数。若少，则**右**补空格。若多，则全部输出。
printf(“%m.nf”,a) 输出小数点后 n 位的小数，同时整个浮点数占 m 位。若少，则**左**补空格。若多，则整数部分输出。
printf(“%-m.nf”,a) 输出小数点后 n 位的小数，同时整个浮点数占 m 位。若少，则**右**补空格。若多，则整数部分输出。
printf(“%m.ns”,a) **输出一共占 m 列，只输出左端 n 个字符**，其余不再输出，空余部分左补空格。如果 n 比 m 还大，m 自动变为 n。
printf(“%-m.ns”,a) **输出一共占 m 列，只输出左端 n 个字符（不变）**，其余不再输出，空余部分右补空格。如果 n 比 m 还大，m 自动变为 n。

scanf 格式

1.scanf(“%md”,&a) 只输出 m 列数字，其余的不要了。
2.scanf(“%*md”) 读入 m 个数字，但不传给任何变量。

整型数（常量）

有符号范围：-32768~32767
无符号范围：0~65535

%x 十六进制数　0x12f　0x012f(x 后面可多放一个 0)
%o **无符号**八进制数。
后缀 l/L u/U
123456789l 长整型数
123u 无符号整型数

浮点数（常量）

两种表达形式：十进制小数，指数形式。
小数形式：
1.必须有小数点
例如 0.0　3.14　300.**(小数点后可以不加数字)**　11**(小数点前可以不加数字)**

指数形式：
1.可以不加小数点，必须有 e/E
2.e 后不加正负号则默认正号
3.阶码不能省去

整数部分.小数部分 e/E±n(阶码)后缀
例如：
1.23456e+4
1.23456e+04**(阶码可以前置 0，不代表八进制**)
.234e+12(可以不要整数部分)
1.e+3(可以不要小数部分)
25e5(可以不要小数点，**可以不要阶码的正负号**)

后缀 l/L

字符（常量）

字符实际是对应 ASCII 码的整型数

用多种方法表示字符'A'
1.'A'(一般字符表示)
2.65(十进制 ASCII 码)
3.0101(八进制 ASCII 码)
4.0x41(十六进制 ASCII 码)
5.'\101'(转义字符八进制)
6.'\x41'(转义字符十六进制)

整型变量

范围
有符号数：-32768-32767
无符号数：0-65535

取余运算，%后面不能是 0。除法运算更不能了。

位运算

1.所有位运算符的运算量都必须是整型或字符串型数据。

按位与&

两者同时为 1 时才为 1

按位或|

两者有一个为 1 就为 1

按位异或^

两者不同即为 1，不同则为 0

二进制左移<<

a<<b 将二进制的 a 左移 b 位

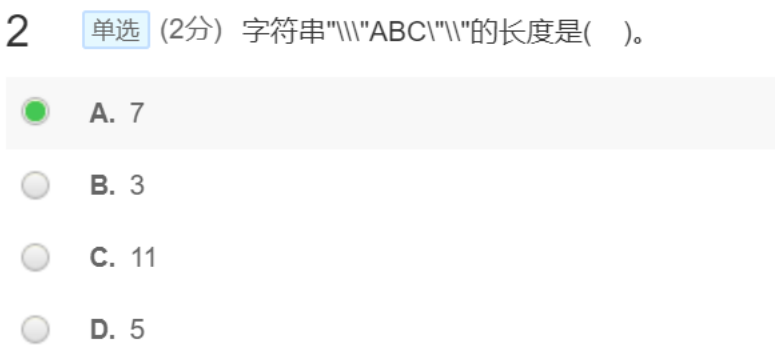
二进制右移>>

a>>b 将二进制的 a 右移 b 位

按位取反~

1 变成 0，0 变成 1

数字符串长度时不包括‘\0’



整型数的二进制表示

正数就是其原码
负数用其绝对值的补码形式表示。
补码：原码按位取反，然后加 1.

函数

函数的原型声明

1.函数声明可以在函数内部，也可以在函数外部。

2.函数原型声明的格式：函数数据类型 函数名 （参数类型...） **不包括存储类型！**

函数的定义

1.函数的定义由函数头和函数体两部分组成。

```
<存储类型><数据类型>函数名(<形式参数及说明>)
{
    说明语句;
    执行语句;
}
```

2.存储类型只有 extern 和 static 两种。extern 可以被任何源文件调用。缺省时默认为 extern。
3.当函数返回值为 int 型时，数据类型可缺省。不需要返回值时，为 void。

4.不允许在函数体内定义另一个函数。（包括 main 函数体内）
5.形如 int printf(char* format,...)意为参数数目可变。

变量的存储类型

储存类型	在哪定义？	在哪储存？	寿命？	使用范围	注意
auto	函数内或复合语句内	内存动态储存区	从定义开始，到函数运行结束终止。	本文件，局部变量	auto 可缺省
register	函数内或复合语句内	CPU 通用寄存器	从定义开始，到函数运行结束终止。	本文件，局部变量	1.运行速度快，但也要省着用 2。不能取地址
extern	函数外	内存静态储存区	从定义开始，到程序运行结束终止。	本文件以及其他文件，全局变量	定义一般全局变量不需要加 extern，引用时才需要加 extern。另外，如果一般全局变量的定义不在程序开头，则该变量只能在本文件使用。初始化默认为 0
static	函数外，函数内或复合语句内	内存静态储存区	从定义开始到程序运行结束终止。	本文件，局部变量或者全局变量	可局部可全局，但仅限于本文件，默认初始化为 0.

指针

void 指针（无类型指针）

如:
int a=1;
int* pint;
void* pvoid;
pint=&a;
pvoid=pint;//无类型指针可以任意指向其他数据
pint=(int*)pvoid;//无类型指针中的数据必须经过类型转换才能赋给有类型的指针

定义的注意事项

char* a,b; 则 a 是指针，b 是一般字符变量。
char a,* b; 则 a 是一般字符变量，b 是指针。

指针的运算

1.指针与指针的加法无意义，指针与指针的减法得到数据的距离。&a[1]-&a[5]=-4
2.*和自增自减的运算优先级相同，结合律从右至左。
3.指针与整数的加减，移动的都是对应的数据大小的倍数。
4.关系运算 &a[1]<&a[8]

多级指针

1.多级指针，c 语言不能将 int**类型的值分配到 int*的实体。
2.**二级指针可以直接等于指针数组 int* []**，但不能直接等于数组指针 int(*)[]。
3.二级指针也可以用 pp[i][j]来表示其中的数据。
4.int a[];**二级指针 p 不能等于&a**

指向函数的指针

1.定义
形如：

```
float func(int,int);
main()
{
    float (*pfun)(int,int)=func;
}
```

<存储类型> <函数返回值的数据类型> (*指针名)(参数 1,参数 2...)=<函数名>
2.使用
以下形式均可：
1.(*pfun)(1,2); *是取内容操作
2.pfun(1,2);
3.func(1,2);
以上效果等价。函数名实际就是函数定义的地址，所以 pfun(1,2)跟 func(1,2)实际是一回事。
3. 对函数指针使用++等是无意义的.

字符串

1.char str[20]=”hello”;是合法的。

其他

1.后置的自增在逗号或分号后开始自增
例如：a=1,b=1;　(a++,a+b)的结果是 3

2.
a=a++;
这是一个未定义行为。结果是 a 不会自增。

3.switch 中 default 情况

4.int x=1,y=x;是合法的
int x=y=1;不合法

5.putchar 不自动输入一个回车。

6.
有符号数-1 变为无符号数 2^16-1
有符号数-19 变为无符号数 2^16-19

7.
char i;
for(i=0;i<n;i++)
{...}
这是正确的。**不过要注意 n 不能超过 256**

8.
注意自定义函数是传递了指针还是数字。如果只是传进去了数字，不可能对 main 函数造成影响。

9.所有的递归函数都是可以写成循环的.

10. strstr(s1,s2)函数找到 s1 中第一次出现 s2 的地方，如果 s2 是空指针，那么就返回 s1，**如果没有出现 s2，则返回 NULL**

11.二维数组也是“一维的”！**a[2][3],那么 a[0][2]地址后面紧跟着的就是 a[1][0]!**

12.sizeof 运算

1)sizeof(int)，sizeof(a)//a 为某变量,则得到的结果是这类变量的字节数。

char，unsigned char 是一字节，其他字节数不同编译器不同。

2)sizeof(数组)，则得到数组的总长度，如果是字符数组，注意是否有额外的'\0'

表 2-2　BC31 中数据类型的长度和值域

类 型 名	说　明	数的范围	字 节 数
int	整型	−32 768～32 767	2
signed int	有符号整型	−32 768～32 767	2
unsigned int	无符号整型	0～65 535	2
short int	短整型	−32 768～32 767	2
signed short int	有符号短整型	−32 768～32 767	2
unsigned short int	无符号短整型	0～65 535	2
long int	长整型	−2 147 483 648～2 147 483 647	4
signed long int	有符号长整型	−2 147 483 648～2 147 483 647	4
unsigned long int	无符号长整型	0～4 294 967 295	4
float	单精度浮点型约	$\pm\|3.4\times10^{-38}\|$～$\pm\|3.4\times10^{38}\|$ 有效数位 7 位	4
double	双精度浮点型	约 $\pm\|1.7\times10^{-308}\|$～$\pm\|1.7\times10^{308}\|$ 有效数位 15 位	8
long double	长双精度浮点型	约 $\pm\|3.4\times10^{-4932}\|$～$\pm\|3.4\times10^{4932}\|$ 有效数位 18 位	10
char	字符型	−128～127	1
signed char	有符号字符型	−128～127	1
unsigned char	无符号字符型	0～255	1