****

**目录**

**第一章C语言基础**

[1.1 C语言概述](#_第—章C语言概述)····················································· 1

1. **基础 C语言**

2.1 [C语言的数据类型](#_2.1 C语言的数据类型)··············································

2.2 [赋值语句](#_2.2  赋值语句)·····················································

2.3 [输入输出语句](#_2.3 输入输出语句)·················································

2.4 [运算符与表达式](#_2.4  运算符与表达式)···············································

2.5 [循环语句](#_2.5  循环语句)·····················································

2.6 [条件判断语句](#_2.6 条件判断语句)················································

2.7 数组的定义及运用···············································

2.8 使用函数解决问题················································

2.9 使用指针解决问题················································

# 第—章C语言概述

本章要点：C语言的发展及其特点运行C程序的步骤与方法最简单的C语言程序程序设计的任务。C语言是国际上广泛流行的高级语言。C语言是在B语言的基础上发展起来的(BCPL)语言是1970年由美国贝尔实验室设计的,并用于编写了第一个UNIX操作系统,在PDP7上实现。优点:精练,接近硬件，缺点:过于简单,数据无类型。1973年贝尔实验室的D.M.Ritchie在B语言的基础上设计出了C语言，对B取长补短，并用之改写了原来用汇编编写的UNIX，(即UNIX第5版)，但仅在贝尔实验室使用。1975年UNIX第6版发布,C优点突出引起关注。1977年出现了《可移植C语言编译程序》，推动了UNIX在各种机器上实现，C语言也得到推广，其发展相辅相成.1978年年影响深远的名著《Programming Language》由Brian W.Kernighan和Dennis M.Ritchie合著;被称为标准C。之后，C语言先后移植到大、中、小、微型计算机上，已独立于unix和pdp，风靡世界，成为最广泛的几种计算机语言之一。美国国家标准化协会(ANSI)根据C语言各种版本对C的发展和扩充，制定了新的标准ANSI C，比标准C有了很大的发展.1988年年K&R按照和修改了他们的“C程序设计语言“。1989年年，ANSI公布了一个完整的c语言标准--ANSIC或C89.1990年年，国际标准化组织接受了C 89为ISOC的标准(IS098991990)。1995年，ISO又修订了C语言标准。1999年，ISO有对C语言标准进行修订在基本保留原来的C语言特征的基础上，针对应用的需要又增加一些功能。命名ISOIEC9899:1999 2001和2004年先后进行两次技术修正(TC1、TC2)ISOIEC9899:1999及其技术修正被称为C99。讲课依据C99标准。

## 第二章·基础C语言

#### 2.1 C语言的数据类型

**2.1.1 整形变量 int**

在C语言定义中int表示一个-32767~32767之间的整形变量，无小数部分，只保留整数部分，在Dev.c++编译器中默认占四个字节的储存空间，但只用其中的16位进行数字储存（其中15位保存大小，首位储存正负）。

**2.1.2 长整形变量 long**

在C语言定义中long表示一个运算方式与int相同但储存大小大于int，在Dev.c++编译器中默认使用四个字节进行储存，大小在 -2147483648～2147483647之间。

**2.1.3超长整形变量 long long**

在C语言定义中long long表示一个运算方式与int和long相同但储存大小大于int和long，在Dev.c++编译器中默认使用八个字节进行储存，大小在 -9223372036854775808~9223372036854775807之间。

**2.1.4单精度浮点数 float**

在C语言定义中float表示一个实数数位在小数点后8到9位之间，在Dev.c++编译器中占用4个字节进行储存，能表示32位数字，小数部分用二进制表示，float大小在是[-340282346638528859811704183484516925440,340282346638528859811704183484516925440]。

**2.1.5双精度实数型 double**

在C语言定义中float表示一个实数数位在小数点后8到9位之间，在Dev.c++编译器中使用 64 位（8字节） 来存储一个浮点数。 它可以表示十进制的15或16位有效数字，其可以表示的数字的绝对值范围大约是：2.23x10-308 ~ 1.79x10308。IEEE754为其定制标准。

**2.1.6字符类型 char**

在C语言定义中char表示一个字符或字符串，用于储存字符，例如’A’.”china”……。

**2.1.7无符号数 unsigned~**

例如：unsigned int 表示无符号整数型，储存范围在[0,2^16]之间，由于无符号数在储存时并没有使用符号位，将符号位的储存空间给了数位，使得无符号数的数字大小得以翻倍。但在负数赋给无符号数的过程中，执行反码操作。

**2.1.7 转义字符**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 转义字符 | “\n” | “\t” | “\”” | “\a” | “\b” | “\f” | “\r” |
| 效果 | 换行 | 跳跃到下一个tab处 | 代表一个字符” | 响铃 | 退格，将当前位置移到前一列 | 换页，将当前位置移到下页开头 | 将当前位置移到本行开头。 |
| 转义字符 | “\\” | “\’” | “\？” | “\0” | …… | …… | …… |
| 效果 | 代表一个字符\ | 代表一个字符’ | 代表一个字符? | 代表一个空字符 | \ | \ | \ |

#### 2.2 赋值语句

**2.2.1 整形赋值** 例：int x;x=10;<==>int x=10;

**2.2.2 浮点数赋值** 例：float x,x=10.0;<==>float x=10.0;

**2.2.3 字符型赋值** 字符赋值 例：char x;x=’A’;<==>char x=’A’;

字符串赋值例：char x;x=”China”;<==>char x=”China”;

**2.2.4 表达式赋值** 例：a=x+y;b=x\*y; 但不能写成x+y=3;(表达式不能被赋值)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Long~** |  | **Long~** |
|  |  | **Unsigned~** |  | **Unsigned~** |
|  |  | **double** | **<----** | **float** |
|  |  | **Long~** |  |  |
|  |  | **Unsigned~** |  |  |
| **short** | **---->** | **int** | **<----** | **char** |

**2.2.5 不同类型数的相互赋值**

不同类型的相互转化遵循以下优先级

**2.2.6 强制转换符**

例1：

Float a=1.5;

Printf(“%d”,(int)a);//输出的值为1

Int a=2;

Printf(“%6.2f”,(float)a);//输出的格式为”2.00”

表2.2.1

#### 2.3 输入输出语句

**2.3.1 printf( )语句**

Printf( )语句结合转义字符和变量或直接输入的常量可以进行C语言交互输出。

例1：

Printf(“This is a c program.\n”);

本语句在运行时无条件输出一条“This is a c program.”，并且光标转入下一行。

例2：

Printf(“x+y=%d”,x+y);

本语句在运行时输出x+y的值及“x+y”的字符

基本格式：Printf(“\_1\_“, \_2\_);

1处录入字符、转义字符及数据类型，2处录入数据类型对应的实参或指针对应的参数。

**注：若转义字符内的类型和后面对象的类型不相同则可能进行强制转换。**

**例1：**

**原因**：A为int类型，当int类型对象转换为char型变量时转换过程遵循assic码表

Printf(“%c”,a)；（其中int a=65）

**结果为：**“A”

例2：

**原因**：a为int类型，当int类型对象转换为float对象时遵循表2.1

Printf(“%f”,a); （其中int a=10）

**结果为：**“10.000000”

**2.3.2 scanf( )语句**

Scanf( )语句在程序运行时要求输入实参，并赋给对应的变量。

例1：

Scanf(“%d %d”,&a,&b);

本语句要求在运行程序时输入两个数字且格式应为“数字”\_\_“数字”，并赋给a和b。

例2：

Scanf(“%c%s”,&a,&b);

本语句要求在程序运行时录入单个字符给a，并将其他字符录入给b。

基本格式：scanf(“\_\_1\_\_”,&\_\_,&\_\_……)；

1处要求有字符类型，2处按照一处的类型顺序将程序运行时的数据录入到2处对应的变量中，且若赋值对象不为指针或数组（单个数组元素除外）就需要在前面加上地址符。

**2.3.3 getchar();语句**

Getchar( );语句表示在程序执行的过程中对录入的一个字符进行处理，若括号内为空时，就消费一个字符，若括号内为一个字符型变量则将得到的一个字符赋给这个变量。

**2.3.4 putchar();语句**

Putchar( );语句将括号内的字符（可以是整形变量和转义字符）输出。

2.3.5 输出格式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字符 | “%%” | “%md” | “m.nf” | “mld”(“%i”) | “m.nlf” | “%ms” |
| 代表变量 | 输出/输入一个字符“%” | 输出/输入一个整数若长度超过m，则输出原值，否则右对齐输出。 | 输出/输入一个浮点数若长度超过m，则输出原值，否则右对齐输出，并保留n位小数。 | 输出/输入一个长整数若长度超过m，则输出原值，否则右对齐输出。 | 输出/输入一个长浮点数若长度超过m，则输出原值，否则右对齐输出，并保留n位小数。 | 输入/输出一个字符串，当长度超过m时，输出原值，否则右对齐输出。 |
| 字符 | “%c” | “%x”(“%X”) | “%o” | “%g”(“%G”) | “%m.nu” |  |
| 代表变量 | 输入/输出一个字符。 | 输入输出一个十六进制数。 | 输入/输出一个八进制数。 | 输入/输出一个浮点数且，去除无效的零。 | 输入输出一个无符号数，若长度超过m，则输出原值，否则同 |  |

表2.3.1

#### 2.4 运算符与表达式

**2.4.1 运算符**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 运算符 | “ + ” | “ - ” | “ \* ” | “ / ” |
| 效果 | 将两个数进行加法。 | 将两个数进行减法。 | 将两个数进行乘法。 | 将两个数进行除法。 |
| 例子 | Int a=1,b=2,c;  c=a+b;  结果c为3. | Int a=2,b=1,c;  c=a-b;  结果c为1. | Int a=2,b=3,c;  C=a\*b;  结果c为6. | Int a=5,b=2,c;  c=a/2;  注意结果取整c=2. |
| 运算符 | “%” | “^” | “&&” | “||” |
| 效果 | 将一个对象取另一个对象的余数。 | 取x的y次，即阶乘 | 与逻辑运算符 | 或逻辑运算符 |
| 例子 | Int a=5,b=3,c;  c=a%b;  结果c为2. | Int a=3,b=4,c;  c=a^b;  结果c为81. | A=真，B=假  (A&&B)的值为假。 | A=真，B=假  （A||B）的值为真 |
| 运算符 | “==” | “>=” | “<=” | “<” |
| 效果 | 判断符号两边是否相等，若相等输出一个不为零的数，若不等，则输出一个零。 | 判断符号左边是否大于等于右边，若成立输出一个不为零的数，若不成立，则输出一个值为零。 | 判断符号右边是否大于等于左边，若成立输出一个不为零的数，若不成立，则输出一个值为零。 | 判断符号右边是否大于左边，若成立输出一个不为零的数，若不成立，则输出一个值为零。 |
| 例子 | A=1,B=1  (A==B)的逻辑值为真 | A=1,B=2  (A>=B)的值为假 | A=1,B=2  (A<=B)的值为真 | / |
| 运算符 | “>” | “->” | “++” | “--” |
| 效果 | 判断符号左边是否大于右边，若成立输出一个不为零的数，若不成立，则输出一个值为零。 | 将一个指针指向一个结构体中的成员 | 对一个变量进行自加处理，且该符号在前，先进行自加，在后，自加优先级降低 | 对一个变量进行自减处理，且该符号在前，先进行自减，在后，自减优先级降低 |
| 例子 | / | （\*p）->name; | Int i=1;  A=i++;//A=1  i=1;  A=++i;//A=2 | Int i=1;  A=i--;//A=1  i=1;  A=--i;//A=0 |

#### 2.5 循环语句

自左向右分别是初始表达式//条件表达式//循环变量

**2.5.1 基本循环**

Do

{

S+=i;

i++;

} while(i<=100);

For(i=0;i<=100;i++)

{

S+=i;

Sum+=S;

花括号内为循环体

}

**2.5.2 for循环**

For(i=0;i<101;)

{

i+=1;s+=i;

}

例。for(i=0;i<=100;i++)

{ s=s+i;

等价于

等价于

}

对For进行解析 for（初始化语句 ;条件表达式 ; 循环变量语句）

**2.5.3 do whlie(……)循环**

例。int i=0,s=0

do

{

s=s+i;

i=i+1 整个花括号里面的语句为循环体，循环体一般为复合语句

} whlie (i<=100) ; i为循环变量，一个循环体中必然要有一个改变循环判断的语句

While(……) // 条件判断语句

{

复合语句

}

**2.5.4使用break / continue语句提前终止循环**

break语句：提前终止整个循环

Do

{

If(i==4)

{

Break;//提前终止整个循环

}

Printf(“i=%d”,i) ; i++;

} while(i<=5);//end i=4.

For(i=0;i<=5;i++)

{

If(i==4)

{

Break;//提前终止整个循环

}

Printf(“i=%d”,i);

} printf(“end i=%d”,i);(此处读取循环语句终止)

If(i==4)

Printf(“循环进行正常”)；（用于判断循环运行是否正常，该语句常用于调试语言）

continue语句

提前终止当前次循环

例：

For(i=0;i<=5;i++)

{

If(i==2)

{

Continue;//终止本轮循环。（在for语句内仍执行表达式3（循环变量），不会死循环）

}

Printf(“i=%d\t”,i);

} // end i=6

Do

{

If(i==2)

{

i++; //常见错误。(若失去本句可能会导致死循环)

continue;

}

} while (i<=5);

**特殊情况**

For( ; ; ) { } <=>while( 1 ) { } (永远执行，除非内有break)

For(赋值语句1，赋值语句2…… ;判断语句;循环变量1，循环变量2……)

#### 2.6 条件判断语句

**2.6.1 If语句**

If（条件判断语句）

条件判断语句中的值为真则运行其下花括号的语句

注：若输入条件判断语句内为赋值语句，赋值对象的值若为0则逻辑值为假，若为非0则逻辑值为真。

{ }

Else If（条件判断语句1）

{ }

Else if（条件判断语句2）

{ }

Else if（条件判断语句3）

{ }

else if（条件判断语句4）

{ }

……

else

{ }

**2.6.2 Swich语句**

Swich（x）(x的要求为是整数型的变量/表达式/若输入为字符则转化为Ascii码运算)

{

Case a:（语句块1）（a,b,c中为整型变量或整型表达式）（取商或取余后可表示范围）

Case b:（语句块2）（a,b,c中为字符可以进行判断）

复合语句（不需要输入限制范围的花括号）

Case c: case d: case e:（语句块3）几个情况同时运行同一个语句

可以使用scanf（" ", ）语句

......

Case z:（）；break;//可以使用break语句提前结束分支结构

……

Default :（）结束退出判断语句

}

无break语句将继续执行接下来的所有语句直到结束（可以使用break语句直接中断输出）

2.6.3 逻辑值运算讲解

2.7 定义数组及其运用

2.7.1 定义数组a[ ]

Int为数组的类型，其中a为数组的数组名，“[ ]”内为数组元素个数

Int a[10];

2.8 自定义函数编写及其运用

2.8.1 基本自定义函数的结构

自右向右分别是函数返回值类型，函数名，函数调用形参

Int max(int x,int y)

{

//\*语句块\*//

此语句为函数的返回值，在定义函数值类型的函数内必须具备

Return 0;

}

2.8.2 空函数的作用

Void print(int x)

void表示该函数的类型为空函数，本身没有赋值，但可以执行内部的语句。一般用于输出字符或结合指针对数组进行修改。

{

Int i;

For(i=0;i<x;i++)

{

Printf(“\*\*\*\*\*\n”);

}

\*return; （该语句不必要打，但若打出则一定不能在”;”之前输入表达式）

}

**常见错误:**

1. 当计算过程中的商为无理数时，除数为整型，得不到精确结果。 即赋值对象为浮点数时，赋值语句中出现数字应加".0"后缀。 例 float a,b;

a=10.0\*b;

1. 化简公式可能会出现精度错误。double和float的精度计算,计算顺序不一样会导致计算结果的不同。

例： float a,b,c; double x;

Printf(“%f”,pow(a,3)\*log10(b)\*log10(c)/x);

Printf(“%f”,log10(b)\*log10(c)\*pow(a,3)/x);

1. 字符参与整数运算是assic码参加运算，当两个数字字符参加运算时输出数字之和的assic码值
2. **Scanf(“ ”,&a)** 语句中若赋值变量不是**指针（\*p）**或**数组元素（a[9]）**则需要地址符。
3. 条件判断语句若输入格式类似于“100<=a<=200”则执行**自左向右**结合

例： int a=80;(80<=100)则第一个条件语句值为真（1），于是执行“1”<=200,结果也为真。

显然，判断结果是错误的。原式应写为（a>=100 && a<=200）。

1. If语句、for语句、while语句、switch语句的break语句要看花括号的位置。