C语言

C语言基础

概念

1. **程序**：一系列指令序列，即人机对话的语言
2. **语言分类**
    - **低级语言**：机器语言、汇编语言
    - **高级语言**：C,C++等
3. 程序设计的三大基本结构
    1. **顺序结构**
    2. **选择结构**
    3. **循环结构**
4. 源程序：高级语言编写的程序
5. 目标程序：二进制代码表示的程序
6. 开发一个C程序的基本过程：
    - **编辑(.c 源程序)->编译(.obj 目标程序)->连接(.exe 可执行程序)->运行**
7. C代码编译成可执行程序经过4步
    - 预处理->编译>汇编->链接
8. C语言是一种**结构化的程序设计语言**,简明易懂，功能强大，适合于各种硬件平台。C语言**即具有高级语言特点，又具有低级语言的功能。**
9. 基本结构

```c
#include <stdio.h>
int main(){
    printf("Hello World\n");
    return 0;
}
```

10. **函数是程序的基本单位。一个程序有且只有一个main()函数，位置任意**程序执行**从main开始，到main结束**
11. **分号为语句的分隔符**
12. 大括号标识一个语句组，成对使用
13. 若**主函数为 void 可省略 主函数还可以是int**

```c
#include <stdio.h>
int main(){
    int a,b,c;
    a=4;
    b=10;
    c=a+b;
    printf("%d\n",c);
    return 0;
}
```

标识符

1. 标识符：在C语言中，有许多符号的命名如变量名、函数名、数组名等，都必须遵守一定**的规则**，按此规则命名的符号**称为标识符**
2. 标迟符的命名规则：
    - **字母、数字、下划线_构成的有限序列**
    - **以字母或者下划线开头**
    - 标识符不能包含除以外的任何特殊字符，如：%、#、逗号、空格等。
    - 标识符**必须以字母或（下划线）开头。**
    - 标识符不能包含空白字符（换行符、空格和制表符称为空白字符）
    - C语言中的某些词（例如int和float等）称为**保留字**，具有特殊意义，不**能用作标识符名。**
    - C语言**区分大小写**，因此标识符price.与标识符PRICE是两个不同的标识符。
3. 分类
    1. **关键字**(32个)：C语言规定了一批标识符，他们在C语言中代表着固定的含义，不能另做它用。

        ```
        auto break case char const continue default do double
        else enum extern float or goto if int long register return
        short signed sizeof static struct switch typedef union
        unsigned void volatile while
        ```

    2. **预定义标识符**：C语言**语法允许**用户把这类标识符另**做它用**，但是这些标识符将失去系统规定的原意。

        比如：printf define main scanf

    3. **用户标识符**：由用户根据需要定义的标识符称为用户标识符

注释

- ```
  //行注释
  /*注释内容不是程序代码，是给其他人员增强理解能力加上的文字说明当程序执行时注释内容视为空 */
  //块注释
  ```

常量

1. 常量是在程序中保持不变的量
2. 常量用于定义具有如下特点的数据
    - 在程序运行中**保持不变**
    - 在程序内部频繁使用
    - 用define关键字定义**符号常量**
    - 分类：**整型常量**、**实型常量**、**字符常量**、**字符串常量**、**符号常量**
      举例：1 2.5 'a' “abc”

整型常量

1. 十进制表示：用一串连续的数字(0~9)表示十进制数。
   例如：345 3684 0 -23456 **只有十进制负数前面加-。**
2. 八进制表示：以数字**0开头**的一个连续数字序列，序列中只能有**0-7**这八个数字。
   例如：045 067451等。 而：019 423 -078都是非法的八进制数。（**无负数**）
3. 十六进制表示：以**0X或0x开头**的连续数字和字母序列，序列中只能有0-9、A-F和a-f这些数字和字母，字母a、b、c、d、e、f分别对应十进制数字10、11、12、13、14、15,大小写均可。**(0x里面x小写就用小写a-f 大写X就是A-F)（无负数）**
  ```

实型常量

1. 小数形式
    - 由数字和小数点组成，**必须有小数点。**(整数部分，小数部分为0可省略小数点不能省略)
    - 例如：3.14 0.15 .56 78. 0.0
2. 指数形式
    - 以**幂**的形式表示，以字母e或E后跟一个以10为底的幂数。
    - 形式：数字 字母+(e/E) 指数(正负)
    - 字母e或E的前后及数字之间不得有空格（实型变量也遵守这个规则)
    - 技巧记忆：**e前e后必有数，e后必须是整数(负数0整数)!**
    - 例：2.3e5 500e-2 .5e3 4.5e0，而e4 .5e3.6 .e5 e都不合法。

字符常量

- 定义：在程序中用**一对单引号把一个字符**括起来，作为字符常量
- 例'A','a','t','!','*','\n'等。Ascll A值65 a值97
- 字符常量只能**用单引号**括起来，**不能用双引号**； 如：“A”不是字符常量
- 字符常量**只能包含一个字符**；如：'abc',错误。
- **区分大小写**； 如：'A'和'a
- C语言约定采用 \ 开头的字符序列 如 \101 八进制 x开头十六进制 \xhh
- 注意：**\x十六进制就两位**因为3位超过ASCLL了 **\转义字符加8进制16进制需要注意越界**

转义字符	意义	ASCII码值（十进制）
\a	响铃(BEL)	007
\b	退格(BS)，将当前位置移到前一列	008
\f	换页(FF)，将当前位置移到下页开头	012
\n	换行(LF)，将当前位置移到下一行开头	010
\r	回车(CR)，将当前位置移到本行开头	013
\t	水平制表(HT)（跳到下一个TAB位置）	009
\v	垂直制表(VT)	011
\\	代表一个反斜线字符"\'	092
\'	代表一个单引号（撇号）字符	039
\"	代表一个双引号字符	034
\?	代表一个问号	063
\0	空字符(NUL)	000
\ddd	1到3位八进制数所代表的任意字符	三位八进制
\xhh	十六进制所代表的任意字符	十六进制

注意：

1. 区分，斜杠："/" 与 反斜杠："\",此处不可互换

2. \xhh 十六进制转义不限制字符个数 '\x000000000000F' == '\xF' [3]

- **字符**是**按**其代码 **ASCLL码(整数值)形式存储**，所有字符数据都作为整数来处理。因此，字符量可以**参与整数运算。**

字符串常量

1. 定义：在程序中用一**对双撇号把若干个字符**括起来 例：“123”，“boy”,“a”等。
2. 字符串常量只能用**双撇号**，不能用单引号括起来；如：‘A’不是字符串常量
3. 字符常量和字符串常量区别
    - 分界符不同**字符串常量双撇号**，**字符常量单撇号**
    - **字符串常量内容多个字符**，**字符常量一个字符**
    - **占用空间不同**字符串常量存储结尾**\0** (占用空间字节个数是**字符数加1**因为\0也占用一个字节)字**符常量占用1个字节**
    - 字符参与运算，**字符串不能参与运算**

符号常量

1. 用**编译预处理宏定义**一个符号名的方法来代表一个常量
2. 格式：**#define 宏名 宏值**

```
#include <stdio.h>
#define PI 3.14159
main(){
    float r;
    double s;
    r=5.0;
    s=PI*r*r;
    printf("s=%f\n",s);
}
```

常变量

1. 定义a为一个整型变量，指定其值为3,而且在变量存在期间其值不能改变
2. 常变量与常量的异同是
    - **常变量**具有变量的基本属性:有类型，占存储单元，只是**不允许改变其值。**
    - 常变量是有名字的不变量，而常量是没有名字的不变量。
    - **有名字**就便于在程序中被引用。

```
const int a=3
```

变量

1. 定义：在**程序运行过程中，值可以改变**的量
2. 注意：
    - 每个**变量**有一个**名**字作为标识，属于用户标识符。
    - **变量**必须**先定义后使用**（定义后还得赋值才能用）
    - 变量代表了内存中的若干个存储单元，**变量名**实际上是以一个名字**代表**的一个**存储空间**（存储地址）
    - **变量名**和**变量值**是两个不同的概念
        - 定义变量时指定该变量的名字和类型
        - 从变量中取值，实际上是通过变量名找到相应的内存地址，从该存储单元中读取数据。
3. 格式：**数据类型 变量名**； 如：int i;
4. 变量初始化

- **边定义变初始化： 数据类型 变量名=值**；（注意区分变量名和变量值）
- **先定义后初始化：数据类型变量名**； 变量名=值；

5. 变量类型
   - **int(4个字节) char(1个字节) float(单精度4字节) double(双精度4字节)**（长度和机器有关）
6.

类型/编译器	16位编译器	32位编译器	64位编译器
void	0	0	0
char	1	1	1
char *	2	4	8
short int	2	2	2
int	2	4	4
float	4	4	4
double	8	8	8
long	4	4	8
long double	8	12	16
long long	8	8	8

运算符

基本运算符

1. 基本的算术运算符

表 3-4 最常用的算术运算符

运算符	含义	举例	结果	优先级
++	自增运算符	++a	a 的值加 1	14
--	自减运算符	--a	a 的值减 1	
*	乘法运算符	a*b	a 和 b 的乘积	13
/	除法运算符	a/b	a 除以 b 的商	
%	求余运算符	a%b	a 除以 b 的余数	
+	加法运算符	a+b	a 和 b 的和	12
-	减法运算符	a-b	a 和 b 的差	

   - 另外，对于+和-俩个运算符，还具有取原值和取负值的含义 如：a=+a;a=-a;
2. 算术运算符

- / % 双目/二元 从左向右 + - 双目 从左向右 若以上5个运算符混合 *** / % 高于+ -**，但是**小括号()改变优先级**
- **%求余要求两个操作数都为整数** 如果**两个数是负数看分子 分子为负结果为负数 例子：5%3=2 2%5 =2 -5%7=-5 7%-5=2**

3. 类型转换(自动类型转换
   - char char->int
   - int char->int
   - int int->int
   - char/int float/double--->double
   - float float/double--->double
   - double double--->double
   - 注意：商 取整 向0 取整（下取整）
   - **如果参与运算的操作数包括float或double，运算结果都是double**
   - **如果参与运算的操作数只包括int或char型，运算结果是int型**
4. ++ -- 自增 自减 单目 需要一个操作数**（整型）**
   - **前缀 ++i --i 先变后用新值 先自增1在用新值 i=i+1在用新i 减同理**
   - **后缀 i++ --i 先用原值后变 先用i i在自增1 先用 i 原值 i=i+1**
   - **注意：前缀和后缀只有在表达式才有区别若作为单独语句功能相同没有区别**
5. 强制类型转换：（类型名）(表达式)
   - **强制类型转换只会改变当前表达式的类型**（一个变量定义后在生命周期中类型不会改变）
   - 如：3.5%2错误 （int）3.5%2正确
6. **运算数类型不一致，**系统自动进行类型转换（**由低向高**转换)
   例：**3/2的结果就是 1 3/2.0的结果就是1.5**

赋值运算符

1. 值运算符和赋值表达式：变量名=表达式
2. **优先级倒数第二，结合方向：左<——右**
3. **是一种赋予的关系而不是等价的关系**
4. = **左侧只能是变量**，不能是表达式
5. 如果**赋值运算符两侧的类型不一致**，但都是**基本类型**时，在赋值时要进行类型转换。类型转换是由**系统自动进行的**，转换的规则是
   - 将**浮点型数据**（包括单、双精度）赋给**整型变量**时，先对浮点数取整，即**舍弃**小数部分，然后赋予整型变量。**int i=3.65; printf("%d",i);//输出3**
   - 将**整型数据赋给单、双精度变量时 数值不变** 但**以浮点数形式存储到变量**中
   - 将一个**double型**数据赋给**float**变量时，先将双精度数转换为单精度，即**只取6~7位有效数字**，存储到float型变量的**4个字节**中。应**注意**双精度数值的大小不能超出float型变量的**数值范围**；将一个**float**型数据赋给**double**型变量时，**数值不变**，在内存中以**8个字节存储，有效位数扩展到15位。**
   - **字符型数据**赋给**整型变量时**，将字符的**ASCII码赋给整型变量。**
   - 将一**个占字节多的整型数据**赋给一**个占字节少的整型**变量或**字符变量**时，只将其**低字节原封不动地送到被赋值的变量**（即发生"**截断**")。

复合赋值表达式

1. 复合赋值运算符
   +=、-=、*=、/=、%=、<=、>=等（两个运算符之间不能有空格)
2. **借助复合的赋值运算符**将形如
   - 变量名=变量名+表达式"的表达式
   - 简化成："**变量名+=表达式**"的形式

- 说明：凡是有赋值运算符参加的运算都是**从右往左算**例
  - a+=3 等价于a=a+3
  - x * =y+8 等价于x=x*(y+8)
  - x%=3等价于x=x%3

3. 嵌套赋值
   - 结合方向：自右向左
   - 每一个赋值符号出现时，计算一下右边表达式的值
   - 如果给变量赋值，记录变量的变化

> 已知 `int a,b=5,c=4;`计算表达式`a+=a-=a=b+c`的值。　　得0

逗号运算符和逗号表达式

1. 定义：用逗号运算符将表达式连接起来的式子
2. 一般形式：表达式1，表达式2，表达式3，...，表达式n
3. 求解过程
   - **从左到右**一个一个求解，**最后一个表达式的值**就是整个、逗号表达式的值。
   - 结合方向左--右
   - 优先级最低
   - 例如
     - a=3　a=3,a+3结果：表达式的值为6
     - b=a+3,a-3 结果：表达式的值为0
     - a=(2,3,4)a的值是多少？ a=4
     - a=2,3,4 a的值是多少？　a=2 后面不保留

C语言中的逻辑值

1. 逻辑值只有两个，分别用“真”和“假”表示。
2. **任何基本类型的值都可作为逻辑值**使用。
   - C语言没有专门的逻辑值，**所有非0**的值被都被当作“真使用，而0值被当作“假”使用。

关系运算符

1. 系运算实际上是“比较运算”
2. C语言的关系运算符共6种

操作符	含义
==	等于
!=	不等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于

3. **优先级**：后四种优先级高于前两种 **结合方法**：从左向右
4. 关系运算符的操作数可以是**变量、常量或表达式。**

5. **计算结果=逻辑值（真或假）** 在C语言中，**“0”表示“假”，“非0”表示“真”**

逻辑运算符

1.

运算符	含义	举例	说明
!	(逻辑非)	!-1 结果0	非取反真变假 假变真
&&	(逻辑与)	a&&b	逻辑与同真为真其他为假
||	(逻辑或)	a||b	逻辑或同假为假其他未真

2. 双目 || && 单目 !
3. ! → && →|| (**!为三者中最高**)
4. 短路：**&&与遇见0就短路 后面不算**结果0 **||或遇见1就 不算结果1**

```
&&断路规则：
int a=0,b=0;
计算下列表达式执行后a和b的值以及表达式的值
1）a++&&b++ a=1,b=0,result=0
2）a++&&++b a=1,b=0,result=0
3）++a&&b++ a=1,b=1,result=0
4）++a&&++b a=1,b=l,result=1
int a=1,b=1;
计算下列表达式执行后a和b的值以及表达式的值
1）a--&&b-- a=0,b=0,result=1
2）a--&&--b a=0,b=0,result=0
3）--a&&b-- a=0,b=1,result=0
4）--a&&--b a=0,b=1,result=0
||断路规则
int a=0,b=0;
计算下列表达式计算后a,b的值以及表达式的值
1）a++||b++  a=1,b=1 result =0
2）a++||++b   a=1,b=1 result =1
3) ++a||b++  a=1,b=0 result =1
4) ++a||++b  a=1,b=0,result =1
int a=1,b=1:
计算下列表达式计算后a,b的值以及表达式的值
1）a--||b-- a=0,b=1 result =1
2)a--||--b  a=0,b=1,result =1
3)--a||b-- a=0,b=0,result =1
4)--a||b-- a=0,b=0,result =0
```

5. 运算符的优先次序：！—>算数运算符—>关系运算符—>&&—>||—>赋值运算符
6. 逻辑运算符与逻辑表达式
   - **表示逻辑运算结果时：1“真”，0“假”**
   - **判断一个量是否为“真”时：0“假”，非0“真”** 注意：**将一个非零的数值认作为“真”**

条件运算符

1. 条件表达式的一般形式为

```
表达式1?表达式2:表达式3
```

2. 条件运算符的执行顺序

- 求解表达式1
- 真：求解表达式2，值为整个条件表达式的值
- 假：求解表达式3，值是整个条件表达式的值
- **注：唯一的三目运算符 记忆： 真前假后**

3. **算术>关系>逻辑>条件>赋值>逗号**
4. 结合性：“自右至左”
5. 以下为合法的使用方法

```
a>b?(max=a):(max=b);
a>b?printf("%d",a):printf"%d",b)
```

长度测试运算符

1. c语言提供了测试数据长度运算符sizeof
2. 其功能是给出exp所占用的内存字节数。

```
sizeof(exp)
//其中，exp可以是类型关键字、变量或表达式。
//例如
sizeof(double),sizeof(x)
```

流程控制结构

顺序结构

语句

1. 语句：C语言中描述计算过程的最基本单位。 由分号；结束。
2. 顺序结构：按语句在程序中出现的顺序逐条执行，没有分支、没有转移。

复合语句和空语句

1. 复合语句
   - 定义：用一对**花括号**把若干语句括起来构成一个语句组。
   - 注意：
     - 花括号内语句的数目不限
     - 在**花括号外面不能加分号**
2. 空语句

```
main(){
 ;
}
```

数据输出

1. 基本概念
   - 输出：把数据从计算机**内部**送到计算机**外部设备**上的操作称为"输出”
2. 注意：C语言本身不提供输入和输出语句，但是有输入和输出函数。
3. 在使用输入输出函数时，要在程序文件的开头用预编译指令
   - 按指定路径查找文件
   - 源程序文件所在目录

- C编译系统指定的include目录

```
#include "c:\cpp\linclude\lmyfile.h"
#include <stdio.h> /根目录
#include "stdio.h" /用户目录
```

printf()函数

1. 作用：在终端设备上按指定格式输出
2. 语句：**printf (格式控制，输出项表)**;
3.

格式字符	说　明
d,i	以带符号的十进制形式输出整数（正数不输出符号）
o	**以八进制无符号形式输出整数（不输出前导符 0）**
x,X	**以十六进制无符号形式输出整数（不输出前导符0x），用x则输出十六进制数的a～f时以小写形式输出，用X时，则以大写字母输出**
u	以无符号十进制形式输出整数
c	以字符形式输出，只输出一个字符 'a'
s	输出字符串"a" "ab"
f	**以小数形式输出单、双精度数，隐含输出6位小数 %f double 15-16位，%f 6位 float 6-7位 1.0f/3=0.333333**
e,E	以指数形式输出实数，用e时指数以"e"表示(如1.2e+02)，用E时指数以"E"表示(如1.2E+02)=1.2*10的2次幂
g,G	选用%f或%e格式中输出宽度较短的一种格式，**不输出无意义的0**。用G时，若以指数形式输出，则指数以大写表示 64.000000 %g 64

附加字符	说　明
l	长整型整数，可加在格式符 d 、 o 、 x 、 u 前面）
m (代表一个正整数)	数据最小宽度
n (代表一个正整数)	对实数，表示输出 n 位小数；对字符串，表示截取的字符个数
-	输出的数字或字符在域内向左靠

4. 常用格式字符
   - d格式符：用来**输出一个有符号的十进制整数** 输出int型数据
   - 指定列宽：**%md** 右对齐
     - **实际宽度>设定宽度m时，按实际宽度输出；**
     - **实际宽度<设定宽度m时，数据右对齐，左边用空格补位。**
   - **%-md** 左对齐
     - **实际宽度>设定宽度m时，按实际宽度输出；**
     - **实际宽度<设定宽度m时，数据左对齐，右边用空格补位。**
   - c格式符：用来输出一个字符

```
char ch='a';
printf("%c",ch); //输出字符：a
printf("%5c",ch); //输出字符：空4个a
```

   - s格式符：用来输出一个字符串

```
printf("%s","CHNA") //输出字符串：CHINA
```

     - %m.ns:指定输出字符串数据的宽度为m,
       - 若n<m,**截取字符串中左端n个字符，输出靠右**，左端空格补齐。
       - 若n>m,输出字符串中截取的前n位字符字符。
   - f格式符：用来输出实数，以小数形式输出
     - %m.nf对于float、double数据可以用"m.n"形式

- **m指定数据总宽度，n指定小数部分位数**，即精度
- **%-m.nf输出数据左对齐；**
- **%+m.nf输出数据带正负号；**
- **%0m.nf不够位数用0补位；**
- ```
  //用%输出实数，只能得到6位小数。
  double a=1.0;
  printf("%f\n",a/3);
  0.333333
  ```
- ```
  //指定数据宽度和小数位数：用%m.nf(右对齐)  m列宽 n小数位数
  printf("%20.15f\n",1.0/3);
  //空4位 0.333333333333333 15位小数
  printf("%.0f\n",10000/3.0):
  //3333
  float n=9.478689;
  printf("%f",n);
  //输出结果：9.47869
  //默认情况下精确到六位小数 上面是7位四舍五入
  //m宽度，表示所有的数字和小数点所占的位数，不够20位右对齐。如果m比较小比如1输出的数位数大于m就没有作用
  //n精度（精确到小数点后多少位）
  ```

- **e(E)**格式符。指定以**指数形式输出**实数
  - %e,VC++给出小数位数为6
  - 小数点前必须有而且只有1位非零数字

    ```
    printf("%e",123.456;
    //输出：1.234560 e+002  002表示10的2次方
    ```

  - %m.ne 设置列宽和小数位数 m列宽 n小数位数

    ```
    printf("%13.2e",123.456)
    //输出：     1.32 e+001 前面4个空格
    ```

5. 注意
   - 遇到%号字符，按后面输出列表变量的值代替
   - 遇到 \ 转义符体现功能
   - 其他一般字符原样输出
   - 字符型用%d输出是字符码的值
6. 注意事项
   - 格式控制中应包含与输出项**一一对应**的输出格式说明，类型必须匹配；
   - 若格式说明的**个数少于输出项个数**，则多余的输出项不予输出

     ```
     如：int x=12,y=28;printf('%d',x,y);
     ```

   - 若格式**说明的个数多于输出项个数**，输出乱码。

     ```
     int x=12,y=28;
     printf("%d%d%d",x,y);
     ```

- **同一变量，不同形式**，出现在同一条输出函数调用中。

```
//如：
int k=21;
printf("%d,%d",k,++k);
//输出22,22
```

  - 原因：printf函数其参数**从右往左进行处理**，先计算++k。显示值时，从左往右。其实k是先执行的但是进入栈了++k后进栈 但是栈后进先出

putchar函数

1. 用putchari函数既可以输出**可显示字符**，也可以输出**控制字符**和**转义字符**
2. putchar(c)中的**c**可以是**字符常量、整型常量、字符变量或整型变量**（其值在字符的ASCII-代码范围内)。

```
putchar('A');//输出大写字母A
putchar(x);//输出字符变量x的值
putchar('\101');//也是输出字符A
putchar('65');//也是输出字符A
putchar('\n');//换行
//对控制字符则执行控制功能，不在屏幕上显示。
```

数据输入

1. 输入：从计算机**外部设备**将数据送入**计算机内部**的操作称为"输入"。

scanf函数

1. 作用：
2. 在终端设备上输入数据
3. 语句：scanf(格式控制，地址表列)；
   - **格式控制**含义同printf函数
   - **地址表列**可以是变量的地址，或字符串的首地址
4. 注意：格式控制必须与你对应的变量的类型相等，否则会出现意想不到的数据。
   - 记忆：**第一部分格式控制的形式在终端输入数据**　一模一样！
5. scanf函数中的格式声明

格式控制符	说明
%c	读取一个单一的字符
%hd、%d、%ld	读取一个十进制整数，并分别赋值给 short、int、long 类型
%ho、%o、%lo	读取一个八进制整数（可带前缀也可不带），并分别赋值给 short、int、long 类型
%hx、%x、%lx	读取一个十六进制整数（可带前缀也可不带），并分别赋值给 short、int、long 类型
%hu、%u、%lu	读取一个无符号整数，并分别赋值给 unsigned short、unsigned int、unsigned long 类型
%f、%lf	读取一个十进制形式的小数，并分别赋值给 float、double 类型
%e、%le	读取一个指数形式的小数，并分别赋值给 float、double 类型
%s	读取一个字符串（以空白符为结束）
%g、%lg	既可以读取一个十进制形式的小数，也可以读取一个指数形式的小数，并分别赋值给 float、double 类型
- %+格式字符，中间可以插入附加的字符，**附加字符必须原样输入。**

```
scanf("a=%f,b=%f,c=%f",&a,&b,&c);
//a=是附加字符输入的时候必须输入a=...
```

- scanf函数中**没有精度控制。**
- scanf中要求给出**变量地址**，如给出变量名则会出错。
- 在输入多个数值数据时，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。**C编译系统在遇到**空格、TAB、回车或非法数据**（如对“%d"输入“12A”时，“A”即为非法数据)**时即认为该数据结束。**
- **在用“%c”格式声明输入字符时，空格字符和“转义字符”中的字符都作为有效字符输入。**
- 如输入的数据与格式指示字符不一致时，虽然编译能够通过，但结果将不正确。

6. 使用scanf函数时应注意的问题

```
scanf("%f%f%f",a,b,c);//错
scanf("%f%f%f",&a,&b,&c);//对
scanf("a=%f,b=%f,c=%f",&a,&b,&c);
//132错
//a=1,b=3,c=2对
//a=1b=3c=2错
scanf("%c%c%c",&c1,&c2,&c3);
//abc对
//a b c 错
对于scanf("%d%c%f",&a,&b,&c);
//若输入1234a123o.262
 a=1234 b=a c=123 剩下不要但不是不存在
//若输入1234 a 123o.262
 a=1234 b=空格 c=空的没有录入
```

getchar函数

1. 函数**没有参数。**
2. 函数的值就是从输入设备得到的字符，
3. 只能接收**一个**字符。
4. 如果想输入多个字符就要用**多个函数。**
5. 不仅可以从输入设备获得一个**可显示的字符**，而且可以获得**控制字符。**
6. 用getchar函数得到的字符可以赋给一个字符变量或整型变量，也可以作为表达式的一部分。如，putchar(**getchar()**);将接收到的字符输出。

选择结构

1. 选择结构：根据条件进行判断真假，执行不同的操作。

if语句

1. 简单if语句的一般形式为

```
if(表达式){
    <语句块>
}
else{
    <语句块>
}
```

   - if语句中的"表达式"**可以是关系表达式、逻辑表达式、甚至是数值表达式。**其中最直观、最容易理解的是关系表达式。
   - **else子句为可选的**，即可以有，也可以没有。
   - 如果条件**为真**，语句**执行一个语句或一组语句**；
   - 如果条件**为假**，则**执行if语句后面(else)的语句**（如果有)。
   - **闰年判定**：年份能被4整除**但**(&&)不能100整除，**或者**(||)年份能被400整除

```
//编写程序，输入一个年份，判断是否为闰年
#include <stdio.h>
int main(){
    int y;
    scanf("%d",&y);
    if((y%4==0&&y%100!=0)||y%400==0)
       printf("闰年")
    else printf("平年")
}
```

2. 嵌套if
   - 匹配规则
     - **有else必有if**
     - 匹配：**else与它前面的、紧挨着的、未被匹配的相匹配。**

```
#include "stdio.h"
main(){
int a=1,b=2,c=3,d=4;
if(a=b)
 {b++;c+=2;}
else
 b+=2;c+=3;
d=a+b+c;
printf("%d,%d,%d,%d",a,b,c,d);
}
//答案  a=2 b=3 c=8 d=13
```

switch语句

1. switch语句用来实现多分支选择结构
2. switch语句的作用**是根据表达式的值，使流程跳转到不同的语句。**
3. switch语句的一般形式

```
switch(表达式){  //整型/字符型/枚举类型的表达式
 case 常量表达式1:语句1;
 case 常量表达式2:语句2;
     ...
 case 常量表达式n:语句n; //不能相同
 default :语句n+1;  //位置任意
}
```

4. 在使用switch结构时应注意以下几点
   - 括号内的"表达式"，**其值的类型应为整数类型（包括字符型）。**
   - 在case后的**各常量表达式的值不能相同**，否则会出现错误；
   - case和常量表达式之间要**有空格**，如：case 10
   - 在case后，可以**省略语句**；也可以有多个语句，多个语句不用{}括起来；
   - 每个case语句后**都应该有一个break语句**；
   - 各case和default子句的先后顺序可以变动；
   - **多个case标号可以共用一组执行语句。**
   - default子句可以省 位置任意 **如果没有匹配的就执行default语句然后退出**
   - **break用于结束跳出本switch**
   - 注：**没有break的时候，只要有一个case匹配，剩下的语句都执行，直至结束switch**
5. 比较多重if和switch结构
   - 多重if结构用来实现两路、三路分支比较方便，而switch:结构实现三路以上分支比较方便。
   - 在使用switch结构时，应注意分支条件要求是简单数据类型(int、char)表达式，而且case语句后面**必须是常量表达式。**
   - 有些问题只能使用多重if结构来实现，例如要判断一个值是否处在某个**区间**的情况。
6. 例子
   - ```
     //1.输入一个整数，判断是正数、负数还是零
     #include "stdio.h"
     main(){
     int a;
     scanf("%d",&a);
         if(a>0)printf("正数");
          else if(a==0) printf("0");
         else printf("负数");
     }
     ```
     ```

```
//【例5-3】给出一百分制成绩，要求输出成绩等级'A'、B'、'C'、D'、E'。90分以上为'A',80~89分为'B',70~79分为C',60~69分为'D',60分以下为E'。
#include <stdio.h
main(){
    float score;
    char grade;
    scanf("%f",&score);
    switch((int)(score/10)){
    case 10:
    case 9:grade='A';break;
    case 8:grade='B';break;
    case 7:grade-'C';break;
    case 6:grade='D';break;
    default:grade='E';
    }
printf("成绩是%5.1f,相应的等级是%cn",score,grade);
}
```

循环结构

- 需要多次重复执行一个或多个任务的问题一般用循环解决

while循环