

C语言与程序设计

The C Programming Language



第3章 基本的标准输入与输出

华中科技大学计算机学院

毛伏兵



第3章 基本的标准输入与输出

主要内容

- C语言提供了一些标准的输入/出函数——系统函数

putchar、puts、printf

getchar、**gets**、scanf

注意： 用这些函数时,须用预编译指令:

#include <stdio.h>



3.1 字符输出与输入

3.1.1 单字符输出/入

- putchar : 字符输出函数
- getchar : 字符输入函数



putchar

函数原型: `int putchar(int);`

功能: 向标准输出设备(显示器)输出一个字符, 函数正确执行时返回该字符码, 否则返回EOF。

调用方式: `putchar (c);`

↑
char 或 int

举例: 欲输出字符A:

`putchar ('A');`

`putchar (65);`

`putchar ('\x41');`

getchar

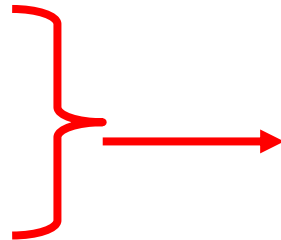
函数原型: `int getchar(void);`

功能: 从标准输入设备(键盘)输入一个字符,
并将该字符作为函数的值。

调用方式: `getchar();`

举例:

```
char c;  
c=getchar();  
putchar(c);
```



嵌套调用



```
putchar(c=getchar());  
或  
putchar(getchar());
```



3 2 字符串输出/入

- puts : 字符串输出函数
- gets : 字符串输入函数



puts

函数原型: `int puts (const char *s) ;`

功能: 将s指向的字符串输出到标准输出设备，并换行。

正确执行: 返回一个非负整数值

出错: 返回**EOF**

调用方式: `puts(s);`

↑
`char [] 或 char *`

```
char name[]="Rita";  
puts (name) ;
```



gets

函数原型: `char *gets(char *s);`

功能: 读取一行字符存放到s指定的内存缓冲区

正确执行: 返回该内存缓冲区的首地址

出错: 返回空指针**NULL**

调用方式: `gets(s);`

↑
`char [] 或 char *`

```
char name[15];  
gets(name);  
puts (name) ;
```




3.3 格式化输出/入

- printf : 格式化输出函数
- scanf : 格式化输入函数



printf

是C语言中使用得最多的一种输出函数，
它可一次按格式输出多个不同类型的数据。

例：

```
printf ( “This is the first program.\n” );
```

```
x=10; y=10.512;
```

```
printf ( “x=%d, y=%.1f ” , x, y);
```

```
This is the first program.  
x=10, y=10.5
```



printf 的调用方式

printf (“ . . . ” , 参数1, 参数2, . . .) ;

格式控制串

1. 普通字符：原样输出
2. 格式说明：由%开始，转换字符结尾，
如：%f, %d 等

◆ 格式说明的基本组成

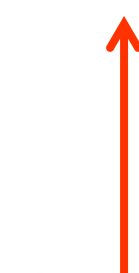
% [特征符] [域宽] [长度修饰符] 转换字符

↑
表3.2: 前5行



设置显示的最小宽度及小数位数

表3.2: m . n *



指出输出参数的类型
表3.2: h / L

↑
输出数据类型和格式
表3.1

```
long x=300000;  
printf("x=%-10ld ", x);
```

x=300000



◆ 长度修饰符

h : 加在 d、o、x、u 之前, 表示输出 short

l : 加在 d、o、x、u 之前, 表示输出 long

L : 加在 f、e、g 之前, 表示输出 long double

short a;

long b;

long double y;

printf("a=%hd,b=%ld,y=%Lf\n",a,b,y);



scanf

在标准输入设备上按指定格式输入各种类型的数据到内存中。

例：

```
int x, y;
```

```
printf("Input two integers:\n")
```

```
scanf("%d %d", &x, &y);
```

Input two integers:

10 20 ↵



scanf 的调用方式

scanf (“... ” , 地址参数1, 地址参数2, ...);

格式控制串



1. 格式说明

2. 空白字符: 空格, \n, \t

3. 非空白字符: 最好不要用

◆ 格式说明的基本组成

% **[*]** **[m]** **[h/l/L]** **转换字符**

↑ 输入数据类型和格式，表3.3

↑ 长度修饰符 (*h*, *l*, *L*), 表3.4

↑ 域宽说明符 (最大宽度*m*), 表3.4

↑ 赋值抑制符(*), 表3.4



◆ 长度修饰符

```
long a;
```

```
double x;
```

```
scanf( “ %ld %lf ” , &a , &x) ;
```

转换说明与输入参数类型不匹配时可能导致的后果：

读入的数据值不正确或程序非正常终止；



输入数据的形式（即分隔数据项的方法）

自学

1. 用隐含的分隔符，如空格、回车键
(只分隔 整数、浮点数、字符串，字符不需分隔)
2. 根据转换字符的含义从输入流中取得数据
3. 根据转换项中指定的域宽分隔出数据项
4. 使用显示的分隔符

实例\源程序\实例.c

```
#include<stdio.h>
void main()
{
    int a,b;
    char op;
    printf("Input two integers : \n");
    scanf("%d%d", &a, &b);
    printf("Input an operator:\n");
    scanf("%c", &op);
    printf("%d%c%d\n ", a, op, b);
}
```

Input two integers :

2 3 ✓

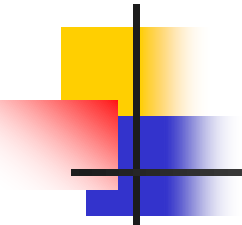
Input an operator:

2

3

用隐含的分隔符分隔整数、浮点数、字符串

字符不需分隔, 根据转换字符的含义自动分隔



```
#include<stdio.h>
void main()
{
    int a,b;
    char op;
    printf("Input two integers :\n ");
    scanf("%d%d", &a, &b);
    printf("Input an operator:\n");
    scanf("%*c%c", &op);
    printf("%d%c%d\n ", a, op, b);
}
```

Input two integers :

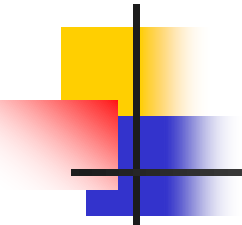
2 3 ✓

Input an operator:

+ ✓

2 + 3

*** : 表示跳过相应的数据，不赋值**



```
#include<stdio.h>
void main()
{
    int a,b;
    char op;
    printf("Input two integers :\n ");
    scanf("%d%d", &a, &b);
    printf("Input an operator:\n");
    scanf("%1s", &op);
    printf("%d %c %d \n ", a, op, b);
}
```

Input two integers :

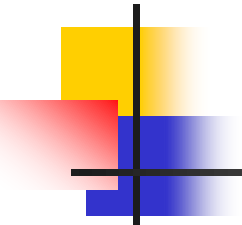
2 3 ✓

Input an operator:

+ ✓

2 + 3

根据指定的域宽分隔出数据项



```
#include<stdio.h>
void main()
{
    int a,b;
    char op;
    printf("Input two integers : ");
    scanf("%d , %d", &a, &b);
    printf("Input an operator:");
    scanf("%1s", &op);
    printf("%d %c %d \n ", a, op, b);
}
```

程序执行时，如果输入：

Input two integers :

2 , 3 ✓

Input an operator:

+

2 + 3

使用显示的分隔符，即用户定义的分隔符（要有提示）
如按点分十进制输入IP地址