

§ . C语言的格式化输入与输出

1. 格式化输出函数

形式: printf(格式控制, 输出表列);

格式控制的内容:

格式说明: 以%开始+格式字符, 表示按格式输出

普通字符(含转义符): 原样输出

输出表列:

要输出的数据 (常量、变量、表达式、函数)

```
#include <stdio.h>
int main()
{
    int a=10, b=5;
    printf("a=%d, b=%d\n", a, b);

    printf("Hello, Welcome!\n");
    return 0;
}
```

纯C语言的包含头文件
也可换为C++形式:
`#include <iostream>`
`using namespace std;`

第1个%d对应变量的a
第2个%d对应变量的b

无%d, 全部
原样输出即可

对应的C++形式:

```
cout << "a=" << a << ", b=" << b << endl;
```

```
cout << "Hello, Welcome!" << endl;
```

§. C语言的格式化输入与输出

1. 格式化输出函数

形式: printf(格式控制, 输出表列);

格式控制的内容:

格式说明: 以%开始+格式字符, 表示按格式输出

普通字符(含转义符): 原样输出

输出表列:

要输出的数据 (常量、变量、表达式、函数)

```
#include <stdio.h>

int main()
{
    int student_num = 87;
    printf("There are %d students in the classroom\n", student_num);
    //输出: There are 87 students in the classroom
    return 0;
}
```

%d对应student_num

对应的C++形式

```
cout << "There are " << student_num << "in the classroom\n";
```

§ . C语言的格式化输入与输出

1. 格式化输出函数

形式: printf(格式控制, 输出表列);

例: 转义符表示字符

```
printf("Hello, Welcome\x21");
```

Hello, Welcome! (\x21为!的ASCII码)

格式字符的种类: 参考C语言类书籍

printf所用的格式字符的种类:

d, i	带符号的十进制形式整数(正数不带+)
o	八进制无符号形式输出整数(不带前导0)
x, X	十六进制无符号形式输出整数(不带前导0x)
u	十进制无符号形式输出整数
c	以字符形式输出(一个字符)
s	输出字符串
f	以小数形式输出浮点数
e, E	以指数形式输出浮点数
g, G	从f, e中选择宽度较短的形式输出浮点数

printf所用的附加格式字符的种类:

字母l	表示长整型整数, 用于d, o, x, u前
字母h	表示短整型整数, 用于d, o, x, u前
正整数m	表示输出数据的宽度
正整数.n	对浮点数, 表示n位小数 对字符串, 表示前n个字符
-	输出左对齐

§. C语言的格式化输入与输出

1. 格式化输出函数

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

```
int main()  
{
```

```
    long a=70000;
```

```
    printf("a=%ld*\n", a);
```

```
    printf("a=%10ld*\n", a);
```

```
    printf("a=%-10ld*\n", a);
```

```
    return 0;
```

```
}
```

%ld : long int 方式输出

%10ld : long int, 总长10, 右对齐

%-10ld: long int, 总长10, 左对齐

a=70000*

a= 70000*

a=70000 *

最后加*, 仅为了看清楚
空格的数量, 无其它意义

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

```
int main()  
{
```

```
    float f=123.456f;
```

```
    char *s="Student"; //字符串
```

```
    printf("f=%f*\n", f);
```

```
    printf("f=%10.2f*\n", f);
```

```
    printf("f=%.2f*\n", f);
```

```
    printf("s=%.4s*\n", s);
```

```
    return 0;
```

```
}
```

%f : 小数形式, 缺省6位小数

%10.2f : 小数形式, 总长10位, 小数点后2位

%.2f : 小数形式, 小数点后两位, 总长缺省

%.4s : 输出字符串的前4个字符

f=123.456001*

f= 123.46*

f=123.46*

f=Stud*

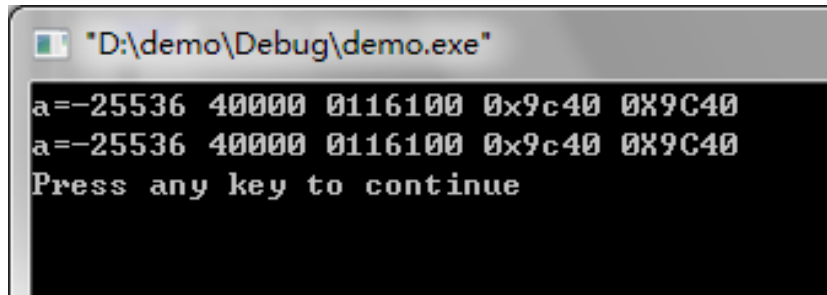
最后加*, 仅为了看清楚
空格的数量, 无其它意义

§. C语言的格式化输入与输出

1. 格式化输出函数

例：同一种输出形式分别用C和C++实现

```
#include <iostream> //因为有cout, 不能换<stdio.h>
#include <iomanip>
using namespace std;
int main()
{
    short a=(short)40000;
    printf("a=%hd %hu 0%ho 0x%hx 0X%hX\n", a, a, a, a, a);
    cout << "a=" << a
        << " " << (unsigned short)a
        << " 0" << oct << a
        << " 0x" << hex << a
        << hex<<setiosflags(ios::uppercase)<<" 0X"<< a
        << endl;
    return 0;
}
```



```
"D:\demo\Debug\demo.exe"
a=-25536 40000 0116100 0x9c40 0X9C40
a=-25536 40000 0116100 0x9c40 0X9C40
Press any key to continue
```

§. C语言的格式化输入与输出

1. 格式化输出函数

使用:

★ 同一个变量, 因指定格式的不同而输出不同

```
short a=-1;
printf("%hd %hu %ho %hx", a, a, a, a);
      -1 65535 177777 ffff
```

```
unsigned short a=65535;
printf("%hd %hu %ho %hx", a, a, a, a);
      -1 65535 177777 ffff
```

```
long a=70000;
printf("%ld %hd", a, a);
      70000 4464
```

★ 实型数据可能有误差

★ 对实型数据, 总长度包含小数点

```
printf("%10.2f", 123.456); 123.46
```

★ 对于指定长度的, 若实际长度大于指定长度, 按实际长度输出

```
printf("%5.2f", 123.456); 123.46
```

```
printf("%5s", "student"); student
```

```
printf("%.5s", "student"); stude !!!
```

注意%5和%.5对于字符串
输出形式的区别

§. C语言的格式化输入与输出

1. 格式化输出函数

使用:

★ 其他字符(含转义符)原样输出

```
printf("a=%d\x41!", 12);    a=12A!
```

★ 仅输出字符的基本值, 不包括识别标记

```
printf("a=%o", 65);          a=101
```

```
printf("a=%x", 65);          a=41
```

```
printf("ch=%c", 65);          ch=A
```

```
printf("s=%s", "Student");    s=Student
```

```
printf("a=0%o", 65);          a=0101
```

```
printf("a=0x%x", 65);          a=0x41
```

```
printf("ch='\ %c\ '", 65);      ch=' A'
```

```
printf("s=\"\ %s\\"", "Student"); s="Stduent"
```

若需要前导字符、单双引号等, 需要自己加

★ 输出%要用%%

```
printf("r=%d%%", 87);    r=87%
```

§. C语言的格式化输入与输出

1. 格式化输出函数

思考:

(1) 指定的输出宽度比实际宽度小时, C语言如何处理?

例: `printf("%3d*\n", 123456);`
`printf("%3s*\n", "Hello");`

(2) 字符串的指定输出字符数量小于字符串长度宽时, C语言如何处理?

例: `printf("%.7s*\n", "Hello");`

(3) 负数的符号是否计入指定输出的总宽度?

例: `printf("%10.3f*\n", -123.45);` //前面有几个空格?
`printf("%-10.3f*\n", -123.45);` //前面有几个空格?

§. C语言的格式化输入与输出

2. 格式化输入函数

形式: scanf(格式控制, 地址表列);

格式控制的内容:

格式说明: 以%开始+格式字符, 表示按格式输入

普通字符(含转义符): 原样输入

地址表列:

&表示取地址

&变量名: 取该变量的内存地址

★ &不能跟表达式/常量(理由与=、++、--等相同)

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

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
    return 0;
}
```

假设键盘输入为:
10 15✓
则输出为:
a=10, b=15

特别说明:

VS2017认为scanf函数是不安全的输入, 因此缺省禁止使用(编译报error), 如果想继续使用, 必须在源程序一开始加定义

#define _CRT_SECURE_NO_WARNINGS
为了和其它编译器兼容, 以及方便后续课程的学习, 我们仍然会继续使用scanf

另: 加 _CRT_SECURE_NO_WARNINGS 的程序
在其它编译器中可正常使用

注: VS2017中C语言用于安全输入的函数是
scanf_s, 使用方法同scanf, 考虑到
兼容性, 不建议大家使用scanf_s,
有兴趣可以自行查阅有关资料

§ . C语言的格式化输入与输出

2. 格式化输入函数

形式：scanf (格式控制，地址表列)；

格式字符的种类：参考C语言类书籍

scanf所用的格式字符的种类：

d, i	输入带符号的十进制形式整数
o	输入八进制无符号形式整数(不带前导0)
x, X	输入十六进制无符号形式整数(不带前导0x)
u	输入十进制无符号形式整数
c	输入单个字符
s	输入字符串
f	输入小数/指数形式的浮点数
e, E, g, G	同f

scanf所用的附加格式字符的种类：

字母l	输入长整型数，用于d, o, x, u前 输入double型数，用于f, e, g前
h	输入短整型数，用于d, o, x, u前
正整数n	指定输入数据所占的宽度
*	本输入项不赋给相应的变量

§ . C语言的格式化输入与输出

2. 格式化输入函数

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a;
    long b;
    short c;
    scanf("%d %ld %hd", &a, &b, &c);
    printf("a=%d, b=%ld, c=%hd\n", a, b, c);
    return 0;
}
```

假设键盘输入为:

10 15 30000 ✓

则输出为:

a=10, b=15, c=30000

假设键盘输入为:

10 15 40000 ✓

则输出为:

a=10, b=15, c=-25536

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

%3d : 表示取3位整型数字

/*2d: 表示跳过2位整型数字

```
int main()
{
    int a, b;
    scanf("%3d /*2d %3d", &a, &b);
    printf("a=%d b=%d\n", a, b);
    return 0;
}
```

假设键盘输入为:

12345678 ✓

则输出为:

a=123 b=678

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ 必须是变量的地址, 不能是常量/表达式等

```
int a;  
char ch;  
scanf("%d %c", &a, &ch); //必须有&符号  
输入: 10 'a'  
则: a=10 ch='a'
```

★ 其他字符原样输入

```
int a, b;
```

```
scanf("%d,%d", &a, &b);
```

若希望a=12, b=34, 则键盘输入为:

12,34 (要原样输入逗号)

```
scanf("a=%d b=%d", &a, &b);
```

若希望a=12, b=34, 则键盘输入为:

a=12 b=34 (要原样输入a= b=等其他字符)

```
int a;  
char ch;  
char s[80];  
scanf("%d %c %s", &a, &ch, s);  
scanf("%c", &s[0]);
```

char s[80]: 表示定义数组s, 在C/C++中,
数组名表示数组的首地址,
因此不需要&

&s[0] : s数组的第0个元素的地址

若不理解, 可等到第5章学完再看

§ . C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ 输入终止条件(回车、空格、宽度到、非法输入)

<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { int a; scanf("%d", &a); printf("%d\n", a); return 0; }</pre>	输入: 123✓
	输出: 123
	输入: 123 123\~**✓
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { int a; scanf("%x", &a); printf("%d\n", a); return 0; }</pre>	输出: 123
	输入: 123a**✓
	输出: 123
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { int a; scanf("%x", &a); printf("%d\n", a); return 0; }</pre>	输入: 123✓
	输出: 291 (0x123)
	输入: 123 123\~**✓
<pre>#define _CRT_SECURE_NO_WARNINGS #include <stdio.h> int main() { int a; scanf("%at", &a); printf("%d\n", a); return 0; }</pre>	输出: 291 (0x123)
	输入: 123at**✓
	输出: 4666 (0x123a)

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ 输入终止条件(回车、空格、宽度到、非法输入)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    short a;
    scanf("%hd", &a); // %hd 用于短整数
    printf("%d\n", a);
    return 0;
}
```

输入: 70000 ✓
输出: 4464

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a;
    scanf("%3d", &a);
    printf("%d\n", a);
    return 0;
}
```

输入: 1234567 ✓
输出: 123

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```

输入: 123 ✓
456 ✓
输出: 123 456

输入: 123 456 ✓
输出: 123 456

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ 输入终止条件(回车、空格、宽度到、非法输入)

```
int a, b;  
scanf("%3d%3d", &a, &b);
```

运行6次, 依次输入:

12✓ 345✓	a=12 b=345	
12✓ 3456✓	a=12 b=345	
123✓ 456✓	a=123 b=456	
1234✓ 5678✓	a=123 b=4	特别容易错!!!
123456✓	a=123 b=456	
12345678✓	a=123 b=456	

```
int a, b;  
scanf("%3d%*2d%3d", &a, &b);
```

运行4次, 依次输入:

123456✓	a=123 b=6	
12345678✓	a=123 b=678	
123456789✓	a=123 b=678	
123✓ 45✓ 678✓	a=123 b=678	

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ %c,%s输入时, 单/双引号及转义符、空格均作为单字符输入

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char ch, s[80];
    scanf("%c%s", &ch, s);
    printf("ch=%c\n", ch);
    printf("s=%s\n", s);
    return 0;
}
```

输入: abcd✓
输出: ch=a
s=bcd

输入: 'A' ✓
输出: ch='
s=A'

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char ch;
    scanf("%c", &ch);
    printf("%c\n", ch);
    return 0;
}
```

输入: A ✓
输出: A

输入: 'A' ✓
输出: '

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char a, b, c;
    scanf("%c%c%c", &a, &b, &c);
    printf("a=%d\n", a);
    printf("b=%d\n", b);
    printf("c=%d\n", c);
    return 0;
}
```

输入: a\bcd ✓
输出: a=97
b=32
c=98

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ %c,%s输入时, 单/双引号及转义符、空格均作为单字符输入

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char s[80];
    scanf("%s", s);
    printf("%s\n", s);
    return 0;
}
```

输入: "\r\n\tabc" ✓
输出: "\r\n\tabc"
注意: 输出中并未体现出
回车换行tab等特性
即字符串实际为:
"\\r\\n\\tabc\\"

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char s[80], t[80];
    scanf("%s,%s", s,t);
    printf("s=%s\n", s);
    printf("t=%s\n", t);
    return 0;
}
```

输入: abc, def ✓
输出: s=abc, def
t=乱码

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ %s不能输入含空格的字符串（空格是输入结束）

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    char s[80];
    scanf("%s", s);
    printf("s=%s\n", s);
    return 0;
}
```

输入: abc↵def↵
输出: s=abc

含空格的字符串可用gets_s输入(第5章会讲, 此处可先忽略)

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ %c输入时不能用int/short/long型地址

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    int a;

    scanf("%c", &a);
    printf("%d\n", a);
    printf("%c\n", a);

    return 0;
} //有一个warning
```

输入: A✓

输出: 非65的数字

A

具体原因:

a占4字节, 值不确定, %c输入A后, 仅1个字节(最低8bit)有确定值(65), 其它3个字节(高24bit)的值仍不确定

%d: 输出4字节, 故不确定值

%c: 输出A, 因为进行了4字节整型赋值1字节整型的操作, 丢弃了高24bit, 故确定是A

§. C语言的格式化输入与输出

2. 格式化输入函数

使用:

★ 输入实型数据时, 可指定宽度但不能指定精度

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    float f;
    scanf("%7.2f", &f); //指定了精度(.2)
    printf("%f\n", f);
    return 0;
} //有warning
```

输入: 1234.56✓

输出: 不确定值

输入: 12.3456✓

输出: 不确定值

输入: 123✓

输出: 不确定值

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    float f;
    scanf("%7f", &f); //指定宽度但不指定精度
    printf("%f\n", f);
    return 0;
}
```

输入: 1234.56✓

输出: 1234.5600**

**：可能的误差

输入: 12.3456✓

输出: 12.345600

输入: 12345678✓

输出: 1234567.000000