Compilation of C program — **Frequently Asked Questions (FAQ)**

by Jaser Li (HDU) Oct. 2023

* EOF（End of File）

EOF是一个表示文件结尾的标志符号。在C语言中，它用于表示从输入流（例如标准输入）读取数据时遇到文件结尾的情况。

EOF 实际上是一个特殊的整数常量，在 stdio.h 头文件中定义为 -1。当从输入流读取数据时，一旦达到文件结尾，就会返回 EOF。

EOF 的使用通常与输入函数（如 scanf、getchar 和 fgets 等）配合使用，以判断是否已经读取到文件的末尾。例如，我们可以使用 scanf 函数读取整数，直到读取到文件结尾：

int num;

while(**scanf**("%d",&*num*)!=**EOF**) // 处理读取到的整数

**printf**("Read number:%d\n",num);

在上述代码中，scanf函数每次尝试从输入流中读取一个整数，并将其保存在num变量中。如果成功读取到整数，则返回读取到的项目数（在这里应该为1），否则返回EOF（-1）。

循环会一直执行，直到遇到文件结尾。当输入流到达文件结尾时，scanf函数无法再继续读取数据，此时返回EOF，循环结束。

需要注意的是，在不同的平台和操作系统上，触发文件结尾的条件可能会有所不同。例如，在Windows系统中，可以使用Ctrl+Z（在一行的开头按下Ctrl键同时按下字母Z）作为文件结尾的标志符；而在UNIX和Linux系统中，可以使用Ctrl+D来表示文件结尾。

除了scanf函数，还可以在其他输入函数（如getchar、fgets等）中使用EOF值来判断文件结尾的情况。每个输入函数都有自己的方式来指示文件结尾。详细的用法可以参考相应的函数文档和示例代码。

* %3.f的含义

%3.f 是格式化输出中的一种格式控制符，用于指定浮点数的输出格式。解释如下：

%：格式化输出控制符的起始符号。

3：表示字段宽度，即输出的最小字符数。在这里，字段宽度为3。

.：小数点是可选的，用于分隔字段宽度和精度。

f：表示输出一个浮点数。

因此，%3.f 表示将浮点数以宽度为3的格式进行输出。如果浮点数的位数少于3，则在左边用空格进行填充。例如：

double num = 123.456;

printf("%3.f", num);

输出结果为：123

注意，由于字段宽度为 3，所以输出结果仅保留整数部分，小数部分会被舍弃。如果要输出更多位的小数，可以调整字段宽度和精度，例如 %5.2f 表示字段宽度为5，保留两位小数。

* int类型的强制类型转换

在C语言中，**int类型的强制类型转换**可以通过使用类型转换运算符(int)或者使用函数int()来实现。对于正数，强制类型转换的结果通常是向下取整；而对于负数，强制类型转换的结果则是向上取整。

例如，以下代码演示了正数和负数的强制类型转换：

double d = 3.5; int i = (int)d;*//* **i 的值为 3**

double e = -3.5;int j = (int)e;*//* **j 的值为 -3**

double f = -3.2;int k = (int)f;*//* **k 的值为 -3**

double g = -3.8;int l = (int)g;*//* **l 的值为 -4**

需要注意的是，强制类型转换可能会导致精度损失或者数值溢出等问题，需要特别注意。

* 头文件 <ctype.h> 的用法

<ctype.h> 是C语言标准库中的头文件，提供了一组用于字符处理的函数。

以下是 <ctype.h> 头文件中常用的函数和它们的作用：

|  |  |
| --- | --- |
| isalpha(int c) | 判断给定的字符是否为字母（a-z、A-Z） |
| isdigit(int c) | 判断给定的字符是否为数字（0-9） |
| isalnum(int c) | 判断给定的字符是否为字母或数字 |
| islower(int c) | 判断给定的字符是否为小写字母（a-z） |
| isupper(int c) | 判断给定的字符是否为大写字母（A-Z） |
| isspace(int c) | 判断给定的字符是否为空白字符（空格、制表符、换行符等） |
| tolower(int c) | 将给定的字符转换为小写字母 |
| toupper(int c) | 将给定的字符转换为大写字母 |

这些函数可以用于对字符进行简单的判断、分类和转换操作，从而简化字符处理的任务。它们通常以字符的ASCII值作为参数，并返回一个非零值（真）或零值（假），表示字符是否满足特定的条件。

* fgets的用法

fgets()是C语言标准库中的函数，用于从指定的流（文件、键盘等）读取一行字符数据。

fgets()函数的声明如下：

char \***fgets**(char \**s*, int *size*, FILE \**stream*);

其中，参数s是用于存储读取数据的字符数组地址；参数size是最多读取的字符数（包括终止符'\0'）；参数stream是要读取的流对象（文件、键盘等）。该函数返回值是指向字符串s的指针，或者是一个空指针（如果读取失败或者到达文件结尾）。

在上述代码中，我们在一个while循环中使用fgets( )函数逐行读取输入数据。具体而言，我们使用以下代码读取一行输入数据并将其存储在字符数组line中：

**fgets**(line, sizeof(line), stdin)

其中，line是用于存储读取数据的字符数组地址，sizeof(line)表示最多读取的字符数，即字符数组line的大小，而stdin表示从标准输入流（也就是键盘）中读取数据。

当读取操作成功完成时，fgets()函数将在字符数组line的末尾自动添加一个终止符'\0'，以便我们可以将其作为C语言中的字符串进行处理。如果读取失败或者到达文件结尾，fgets()函数将返回一个空指针，此时我们可以通过判断是否为NULL来处理这种情况。

* 位于格式字符串'%c'前的空格有何作用？

跳过之前可能还在**输入缓冲区**中的空白字符（空格、制表符、换行等）并读取下一个**非空白字符**。

当输入字符时，我们通常会敲击键盘，每次敲击都会在输入缓冲区留下一个字符。如果在读取字符之前有一个空格字符（或其他空白字符）存在于输入缓冲区中，那么%c将会读取该空白字符，而非我们期望的字符。通过在格式字符串中加上空格字符，可以告诉输入函数跳过之前可能存在的空白字符，直到遇到第一个非空白字符为止。这样，我们就可以确保只读取到我们需要的有效字符。

这种方法通常用于与其他格式控制字符（例如%d、%f等）混合使用，以避免由于输入缓冲区中的残留空白字符导致的意外行为。但请注意，如果您确实需要读取空白字符，例如读取一行字符串（包括空白字符），则不需要在%c之前添加空格。