



知识点总结



- 在C语言中，**fgetc**、**fgets**、**fread**和**fscanf**是用于文件读取操作的函数，它们均被定义在**stdio.h**头文件里。

- **fgetc**函数：用于从指定文件中读取一个字符。

- 调用fgetc函数的语法格式：**fgetc(文件指针)**;
这里的“文件指针”指向关联着已打开文件的**FILE**结构体。

- 函数返回值：

若读取成功，则返回读取到的字符，以**int**类型返回读取到的字符。

若读取不成功（到达文件末尾或发生错误），则返回**EOF**（通常定义为-1）。

例：通过循环结构，利用fgetc函数逐个读取文件中的字符，直至抵达文件末尾或者出现读取错误。



```
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "r");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }

    // 定义一个int类型变量ch，用于储存每次读取到的字符
    int ch;
    // 使用while循环，不断调用fgetc函数逐个读取文件中的字符，直到返回EOF
    while ((ch = fgetc(fp)) != EOF) {
        // 将读取到的字符打印输出
        printf("%c", ch);
    }

    // 省略了对fclose函数的返回值进行检查
    fclose(fp);
    return 0;
}
```



注意：

- 1、fgetc函数的返回值是int类型，在处理读取结果时，要注意类型转换。
- 2、对于由多个字节表示的非ASCII字符（如中文），fgetc函数仍然可以正确读取其每个字节，但需要组合后解码，否则返回的结果可能不符合预期。



- **fgets函数**：用于从指定文件中**读取一行数据**，或者读取指定的**最大字符数 - 1** 个字符，以先满足的条件为准。

- 调用fgets函数的语法格式：**fgets(目标字符数组指针, 最大读取字符数, 文件指针);**
其中：

“目标字符数组指针”指向一个字符数组，该数组用于存放读取到的数据。

“最大读取字符数”是一个整数，表示最多可读取的字符个数（包含字符串结束符'**\0**'）。

“文件指针”指向关联着已打开文件的FILE结构体。

- 函数返回值：

若读取成功，则返回储存了**读取内容的字符数组的地址**，即调用函数时，传入的“目标字符数组指针”的值。

若读取失败（到达文件末尾或发生错误），则返回**NULL**。



例：通过循环结构，利用fgets函数逐行读取文件中的数据，直至抵达文件末尾或者出现读取错误。

```
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "r");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }

    // 定义一个字符数组str，用于储存每次读取的一行数据
    char str[100];
    // 使用while循环，不断调用fgets函数逐行读取文件内容，直到返回NULL
    // sizeof运算符返回的结果值是size_t类型，本质上，属于无符号整数类型
    while (fgets(str, sizeof(str), fp) != NULL) {
        // 将读取到的一行数据打印输出
        printf("%s", str);
    }

    // 省略了对fclose函数的返回值进行检查
    fclose(fp);
    return 0;
}
```



注意：

- 1、fgetc函数会读取包括换行符'\n'在内的字符。
- 2、fgetc函数完成读取后，会自动在读取到的内容末尾添加'\0'以构成完整字符串。



- **fscanf函数**：从指定的文件中按照指定的格式读取数据，并储存到相应的变量中。

调用fscanf函数的语法格式：

fscanf(文件指针, 格式字符串, 变量地址1, 变量地址2, ...);

其中：

“文件指针”指向关联着已打开文件的FILE结构体。

“格式字符串”用于指定读取数据的格式（与scanf函数中的格式字符串类似）。

“变量地址1, 变量地址2, ...”是与格式字符串中的格式符按顺序一一对应的变量地址。

- **函数返回值：**

若读取成功，则返回成功匹配和赋值的变量的数量。

若读取失败（到达文件末尾或发生错误），则返回EOF。



例：利用fscanf函数从文件中匹配并读取数据。

· 注意：

1、fscanf函数依赖**格式控制字符串**来解析文件中的数据，格式必须与文件中数据的实际格式**精确匹配**，否则可能导致读取错误。

2、与scanf函数类似，要确保提供的变量地址与格式符的类型和顺序一致。

```
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "r");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }

    // 定义一个整数变量num和一个字符数组str，用于存储从文件中读取的数据
    int num;
    char str[20];

    // 使用fscanf函数，按照格式字符串"%d %s"的规则从文件fp中读取数据
    // 将读取到的数据分别赋值给变量num和字符数组str
    // 并将函数返回值赋给result
    int result = fscanf(fp, "%d %s", &num, str);

    // 对fscanf函数的返回值result进行判断
    // 如果result为2，说明成功读取并赋值了两个变量
    if (result == 2) {
        printf("读取到的整数： %d，字符串： %s\n", num, str);
    }
    // 如果result为EOF，说明读取异常（已到达文件末尾或发生错误）
    else if (result == EOF) {
        printf("读取操作可能存在问题\n");
    }
    // 如果result既不为2也不为EOF，说明成功读取的参数数量既不是预期的两个，也没有发生读取错误
    // 而是在格式匹配过程中出现问题，导致读取的参数个数不符合预期
    else {
        printf("匹配失败： 成功读取%d个参数\n", result);
    }

    // 省略了对fclose函数的返回值进行检查
    fclose(fp);
    return 0;
}
```



- **fread函数**：从指定的文件中读取指定数量的字节数据，并储存到指定的内存区域。常用于读取二进制文件，但也可用于文本文件。

- 调用fread的语法格式：

fread(指针变量, 每个数据项的大小, 数据项个数, 文件指针);

其中：

“指针变量”是一个void类型指针，指向一块内存的首地址，该内存用于储存读取到数据。

“每个数据项的大小”为size_t类型，用于指定每个数据项的字节大小。

“数据项个数”为size_t类型，用于指定要读取的数据项的数量。

“文件指针”指向关联着已打开文件的FILE结构体。

- 函数返回值：返回成功读取的数据项的数量，该返回值类型为size_t。

如果返回值小于我们想要读取的数据项数量，可能是因为到达文件末尾或发生错误。



例：利用fread函数从二进制文件中读取指定数量的结构体数据到结构体数组中。

· 注意：fread函数按指定的字节数和数据项数量进行读取，适合读取固定格式的数据，如结构体数组等。

```
#include <stdio.h>
typedef struct Data {
    int num;
    char str[20];
} Data;
int main() {
    FILE* fp = fopen("data.bin", "rb");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }
```

```
    // 定义一个Data类型的结构体数组data，用于储存从文件中读取的10个结构体数据
    Data data[10];
```

```
    // 使用fread函数从文件fp中读取数据，并将返回的成功读取的数据项数量，储存在result变量中
```

```
    size_t result = fread(data, sizeof(Data), 10, fp);
    if (result < 10) {
        printf("实际读取的数据项数量小于期望数量，读取操作可能存在问题\n");
    }
```

```
    // 省略了对fclose函数的返回值进行检查
    fclose(fp);
    return 0;
```

```
}
```



补充：

1、在同一次文件打开期间，当执行读文件操作时，文件指针指向的FILE结构体中的文件位置指示器会发生变动。它会跳过已读取的内容，移动到下一个未读取数据的位置。

例如，假设某文件内容为"abc"，通过 `FILE* fp = fopen("文件名", "r");` 这条语句使文件指针fp指向该文件。

那么在执行 `int ch = fgetc(fp);` 这条语句后，ch会得到字符'a'，同时文件指针fp就移动到了字符'b'的位置，下次再调用fgetc(fp);就会读取到字符'b'。

2、可以通过检查feof和ferror这两个函数的返回值，来判断是到达文件末尾还是发生错误。（feof和ferror函数均被定义在stdio里）



- **feof函数**：用于检测文件指针是否已到达文件末尾。

- 调用feof函数的语法格式：**feof(文件指针);**

这里的“文件指针”指向与正在被读取的文件相关联的FILE结构体。

- 函数返回值：

如果位置指示器已到达文件末尾，则返回非0值（表示真）；

如果位置指示器未到达文件末尾，则返回0（表示假）。

- **ferror函数**：用于检测文件在操作过程中是否发生错误。

- 调用ferror函数的语法格式：**ferror(文件指针);**

这里的“文件指针”指向与正在被操作的文件相关联的FILE结构体。

- 函数返回值：

若在最近一次操作中发生了错误，返回非0值（表示真）；

若操作正常，未发生错误，返回0（表示假）。

例：基于fread函数的读取结果，利用feof和ferror函数来判断是到达文件末尾还是发生错误。

```
#include <stdio.h>
typedef struct Data {
    int num;
    char str[20];
} Data;
int main() {
    FILE* fp = fopen("data.bin", "rb");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }

    Data data[10];
    size_t result = fread(data, sizeof(Data), 10, fp);

    if (result < 10) {
        // 使用feof函数检查是否到达文件末尾
        if (feof(fp)) {
            printf("到达文件末尾\n");
        }
        // 使用ferror函数检查是否在读取过程中发生错误
        else if (ferror(fp)) {
            printf("读取文件时发生错误\n");
        }
    }

    // 省略了对fclose函数的返回值进行检查
    fclose(fp);
    return 0;
}
```





fgets(目标字符数组指针, 最大读取字符数, 文件指针);

fgets函数会在读取到的内容末尾自动添加'\0', 需要留一个位置给空字符, 因此限制了读取长度不能超过n - 1个



题目1、标准库函数fgets(s, n, f)的功能是 (A)

- A. 从文件f中读取长度不超过n-1的字符串存入指针s所指的内存
- B. 从文件f中读取长度为n的字符串存入指针s所指的内存
- C. 从文件f中读取n个字符串存入指针s所指的内存
- D. 从文件f中读取n-1个字符串存入指针s所指的内存



fread(指针变量, 每个数据项的大小, 数据项个数, 文件指针);



读入数据存放的地方



题目2、读取二进制文件的函数调用形式为

“fread(buffer, size, count, fp);”, 其中buffer代表的是 (A)

- A. 一个内存块的首地址, 代表读入数据存放的地址
- B. 一个整型变量, 代表待读取的数据的字节数
- C. 一个文件指针, 指向待读取的文件
- D. 一个内存块的字节数



feof函数是用来判断文件是否读到尾的



题目3、设fp为指向某二进制文件的指针，且已读到此文件末尾，
则函数feof(fp)的返回值为 (A)

A. 非0值

B. '\0'

C. 0

D. NULL

 fscanf可以用于读文件，而非写文件。



题目4、把文件指针传给fscanf函数，就可以向文本文件中写入任意的字符。（✗）