知识点总结

· 在C语言中，fputc、fputs、fprintf和fwrite是用于文件写入操作的函数，它们均被定义在stdio.h头文件里。

· fputc函数：用于将一个字符写入到指定的文件中。

· 调用fputc函数的语法格式：fputc(待写入字符, 文件指针);

其中：

“待写入字符”会以int类型传递。

“文件指针”指向关联着已打开文件的FILE结构体。

· 函数返回值：

若写入成功，以int类型返回写入的字符。

若发生错误，返回EOF（通常为-1）。

例：使用fputc函数，向以只写文本模式打开的example.txt文件写入一个字符。

```c
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "w");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }
    char ch = 'A';
    // 使用fputc函数写入字符，并获取返回值
    int result = fputc(ch, fp);
    if (result == EOF) {
        printf("写入失败\n");
        fclose(fp);
        return 1;
    }

    if (fclose(fp) != 0) {
        printf("关闭文件失败，数据可能未写入磁盘\n");
        return 1;
    }
    printf("写入成功\n");
    return 0;
}
```

· fputs函数：用于将一个字符串写入到指定的文件中，但不包括字符串结束符'\0'。

· 调用fputs函数的语法格式：fputs(待写入字符串, 文件指针);

其中：

“待写入字符串”本质上是一个指针，它既可以是字符串字面量，也可以是储存字符串的字符数组名。

“文件指针”指向关联着已打开文件的FILE结构体。

· 函数返回值：

若写入成功，返回一个非负整数。

若发生错误，返回EOF（通常为-1）。

例：使用fputs函数，向以只写文本模式打开的example.txt文件写入一个字符串。

```c
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "w");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }
    // 使用fputs函数写入字符串，并获取返回值
    int result = fputs("Hello, World!", fp);
    if (result == EOF) {
        printf("写入失败\n");
        fclose(fp);
        return 1;
    }

    if (fclose(fp) != 0) {
        printf("关闭文件失败，数据可能未写入磁盘\n");
        return 1;
    }
    printf("写入成功\n");
    return 0;
}
```

注意：fputs不会自动在写入的字符串末尾添加换行符'\n'，如果需要换行，需要手动在字符串中添加换行符'\n'。

· fprintf函数：用于按照指定的格式将数据写入到指定的文件中。

· 调用fprintf函数的语法格式：fprintf(文件指针, 格式字符串, 变量1, 变量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”用于指定写入数据的格式。

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

· 函数返回值：

若写入成功，返回实际写入文件的字符数。

若发生错误，返回一个负数。

例：使用fprintf函数，向以只写文本模式打开的example.txt文件写入格式化数据。

```c
#include <stdio.h>
int main() {
    FILE* fp = fopen("example.txt", "w");
    if (fp == NULL) {
        printf("无法打开文件\n");
        return 1;
    }
    int num = 42;
    char* str = "Answer";
    // 使用fprintf函数写入格式化数据，并获取返回值
    int result = fprintf(fp, "%s to the ultimate question is %d\n", str, num);
    if (result < 0) {
        printf("写入失败\n");
        fclose(fp);
        return 1;
    }

    if (fclose(fp) != 0) {
        printf("关闭文件失败，数据可能未写入磁盘\n");
        return 1;
    }
    printf("写入成功\n");
    return 0;
}
```

· fwrite函数：用于将指定数量的字节数据从特定内存区域写入到指定的文件中。常用于写入二进制文件，但也可用于文本文件。

· 调用fwrite函数的语法格式：fwrite(指针, 每个数据项的大小, 数据项个数, 文件指针);

其中：

“指针”是一个void类型指针，指向一块内存的首地址，该内存用于储存要写入文件中的数据。

“每个数据项的大小”为size_t类型，用于指定每个要写入的数据项的字节大小。

“数据项个数”为size_t类型，用于指定要写入的数据项的个数。

“文件指针”指向关联着已打开文件的FILE结构体。

· 函数返回值：

返回成功写入的数据项的数量，该返回值类型为size_t。

如果返回值小于我们想要写入的数据项数量，可能是因为发生错误或磁盘空间不足等原因。

例：使用fwrite函数，向以只写二进制模式打开的data.bin文件写入Data结构体类型的数据。

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

    Data data = {10, "Hello"};
    // 使用fwrite函数写入结构体变量data的数据，并获取返回值
    size_t result = fwrite(&data, sizeof(Data), 1, fp);
    if (result < 1) {
        printf("写入失败\n");
        fclose(fp);
        return 1;
    }

    if (fclose(fp) != 0) {
        printf("关闭文件失败，数据可能未写入磁盘\n");
        return 1;
    }
    printf("写入成功\n");
    return 0;
}
```

注意：fwrite按照指定的字节数和数据项数量进行写入操作，适合写入固定格式的数据，如结构体数组等。

· 注意：

1、要确保文件以可写模式（如 "w"、"wb"、"a"、"ab" 等）打开，否则写入操作会失败。

2、在同一次文件打开期间，当执行写文件操作时，文件指针所指向的FILE结构体中的文件位置指示器会自动移动到写入的内容之后的位置。

3、当文件以追加模式（如："a"、"ab" 等）打开时，写入操作会强制在当前文件末尾执行。

题目1、有以下程序：

```
#include <stdio.h>
main()
{
    FILE *f;
    f = fopen("filea.txt", "w");
    fprintf(f, "abc");
    fclose(f);
}
```

若文本文件filea.txt中原有内容为：hello，则运行以上程序后，文件filea.txt中的内容为（C）

A. Helloabc

B. abclo

C. abc

D. abchello

"w"模式打开文件时，如果文件已存在，会清空文件里的原有内容，然后才开始写入新内容

题目2、有以下程序：

"w"模式打开文件时，如果文件已存在，会清空文件里的原有内容，然后才开始写入新内容

```
#include <stdio.h>
main()
{
    int i;
    FILE* fp;
    for (i = 0; i < 3; i++)
    {
        fp = fopen("res.txt", "w");
        fputc('K' + i, fp);
        fclose(fp);
    }
}
```

程序运行后，在当前目录下会生成一个res.txt文件，其内容是（ A ）

A. M

B. EOF

C. KLM

D. L

题目3、有以下程序:

```
#include <stdio.h>
main()
{
    FILE *fp;
    int a[10] = {1, 2, 3}, i, n;
    fp = fopen("d1.dat", "w");
    for(i = 0; i < 3; i++) fprintf(fp,"%d", a[i]);
    fprintf(fp, "\n");
    fclose(fp);
    fp = fopen("d1.dat", "r");
    fscanf(fp, "%d", &n);
    fclose(fp);
    printf("%d\n", n);
}
```

程序的运行结果是（B）

A. 12300

B. 123

C. 1

D. 321

题目4：有以下程序：

```
#include <stdio.h>
main()
{
    FILE *fp;
    int a[10] = {1, 2, 3, 0, 0}, i;
    fp = fopen("d2.dat", "wb");
    fwrite(a, sizeof(int), 5, fp);
    fwrite(a, sizeof(int), 5, fp);
    fclose(fp);
    fp = fopen("d2.dat", "rb");
    fread(a, sizeof(int), 10, fp);
    fclose(fp);
    for(i = 0; i < 10; i++) printf("%d,", a[i]);
}
```

程序的运行结果是（D）

A. 1,2,3,0,0,0,0,0,0,0,

B. 1,2,3,1,2,3,0,0,0,0,

C. 123,0,0,0,0,123,0,0,0,0,

D. 1,2,3,0,0,1,2,3,0,0,

题目5、请根据以下各小题的要求设计C应用程序（包括界面和代码）。

请补充main函数，该函数的功能是：先以只写方式打开文件“out52.dat”，再把字符串str中的字符保存到这个磁盘文件中。

注意：部分源程序给出如下。

请勿改动主函数main和其他函数中的任何内容，仅在main函数的横线上填入所编写的若干表达式或语句。

```
#include <stdio.h>
#define N 80
main()
{
    FILE* fp;
    int i = 0;
    char ch;
    char str[N] = "I'm a student!";
    if((fp = fopen(① "out52.dat", "w" )) == NULL)
    {
        printf("cannot open out52.dat\n");
        exit(0);
    }
    while(str[i])
    {
        ch = str[i];
        ② fputc(ch, fp) ;
        putchar(ch);
        i++;
    }
    ③ fclose(fp) ;
}
```