

知识点1【深拷贝】0-1

总结：前提就是指针变量 作为 结构体的成员

浅拷贝：两个结构体变量 中的 指针成员 指向 同一块堆区空间。

深拷贝：两个结构体变量 中的 指针成员 指向 各自的堆区空间。

知识点2【文件】

1、文件的存取过程

2、磁盘文件的分类0-2

总结：（重要）

3、文件指针

注意：不要关系FILE的细节 只需要 会用FILE 定义指针变量就行：FILE *fp=NULL;

4、fopen打开一个文件

文件使用方式：

r:以只读的方式打开文件

w:以只写的方式打开文件

a:以追加的方式打开文件（往文件的末尾写入数据）

+: 以可读可写的方式打开

b:以二进制的方式打开文件

t:以文本的方式打开文件（省略）

打开方式的组合形式：1-1

关闭文件fclose

知识点3【文件的字节读写】

1、字节的读操作 fgetc函数

2、字节的写操作 fputc

练习：

知识点4【文件的字符串读写】1-2

1、使用fputs 往文件中写入一个字符串

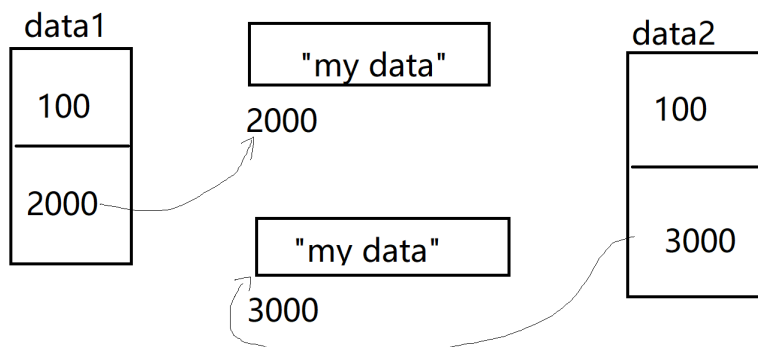
2、使用fgets从文件中获取字符串

知识点5【文件的块读写】fread fwrite

1、使用fwrite 将 数据块 写入到文件中

2、使用fread 从文件中 读取 数据块

知识点1【深拷贝】0-1




```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 typedef struct
5 {
6     int num;
7     char *name;
8 }DATA;
9 void test01()
10 {
11     DATA data1;
12     DATA data2;
13
14     data1.num = 100;
15     data1.name = (char *)calloc(1,12);
```

```

16  strcpy(data1.name, "my data");
17
18  data2.num = data1.num;
19  //为结构体变量 申请 独立的空间
20  data2.name = (char *)calloc(1,12);
21  strcpy(data2.name, data1.name);
22
23  printf("data1:num = %d, name=%s\n", data1.num, data1.name);
24  printf("data2:num = %d, name=%s\n", data2.num, data2.name);
25
26  if(data1.name != NULL)
27  {
28      free(data1.name);
29      data1.name = NULL;
30  }
31
32  if(data2.name != NULL)
33  {
34      free(data2.name);
35      data2.name = NULL;
36  }
37
38
39 }
40 int main(int argc, char *argv[])
41 {
42     test01();
43     return 0;
44 }

```

运行结果:

 "C:\WORK\C\CODE\DAY17\00_test\Debug\00_test.exe"

```

data1:num = 100, name=my data
data2:num = 100, name=my data
Press any key to continue.

```

总结：前提就是指针变量 作为 结构体的成员

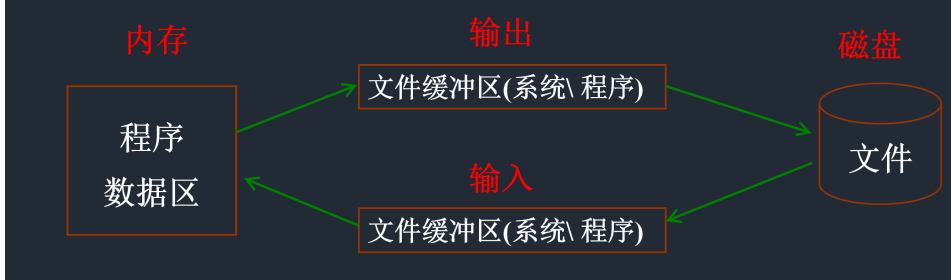
浅拷贝：两个结构体变量 中的 指针成员 指向 同一块堆区空间。

深拷贝：两个结构体变量 中的 指针成员 指向 各自的堆区空间。

知识点2【文件】

1、文件的存取过程

磁盘文件的存取过程



缓冲区的目的：提高存取效率 磁盘使用寿命

2、磁盘文件的分类0-2

物理上 所有的磁盘文件都是 二进制存储，以字节为单位 顺序存储。

逻辑上的文件分类：

文本文件：基于**字符编码**的文件

二进制文件：基于**值编码**的文件

文本文件：

1. 基于字符编码，常见编码有ASCII、UNICODE等，一般可以使用文本编辑器直接打开

例如：数5678的以ASCII存储形式为：

ASCII码：00110101 00110110 00110111 00111000

歌词文件(lrc):文本文件

二进制文件：

1. 基于值编码,自己根据具体应用,指定某个值是什么意思
2. 把内存中的数据按其内存中的存储形式原样输出到磁盘上
3. 一般需要自己判断或使用特定软件分析数据格式

例如：5678的存储形式为：

二进制码：00010110 00101110

音频文件(mp3):二进制文件

总结：（重要）

文本文件、二进制文件对比：

1. 译码：

文本文件编码基于字符定长，译码容易些；

二进制文件编码是变长的，译码难一些（不同的二进制文件格式，有不同的译码方式）。

2. 空间利用率：

二进制文件用一个比特来代表一个意思(位操作)；

而文本文件任何一个符号至少需要一个字节。

3. 可读性：

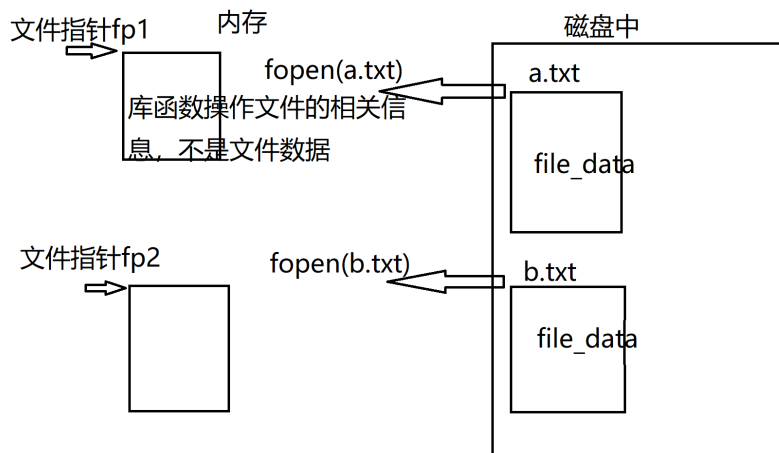
文本文件用通用的记事本工具就几乎可以浏览所有文本文件

二进制文件需要一个具体的文件解码器，比如读BMP文件，必须用读图软件



12

3、文件指针



FILE在stdio.h文件中的文件类型声明：

```
typedef struct
{
    short level;           //缓冲区“满”或“空”的程度
    unsigned flags;        //文件状态标志
    char fd;               //文件描述符
    unsigned charhold;     //如无缓冲区不读取字符
    short bsize;           //缓冲区的大小
    unsigned char *buffer; //数据缓冲区的位置
    unsigned ar*curp;       //指针，当前的指向
    unsigned istemp;        //临时文件，指示器
    short token;           //用于有效性检查
} FILE;
```

注意：不要关系FILE的细节 只需要 会用FILE 定义指针变量就行：FILE

***fp=NULL;**

(了解)

c语言中有三个特殊的文件指针无需定义、打开可直接使用：

1. **stdin**: 标准输入 默认为当前终端（键盘）

我们使用的scanf、getchar函数默认从此终端获得数据

2. **stdout**: 标准输出 默认为当前终端（屏幕）

我们使用的printf、puts函数默认输出信息到此终端

3. **stderr**: 标准出错 默认为当前终端（屏幕）

当我们程序出错或者使用:perror函数时信息打印在此终端

4、fopen打开一个文件

```
FILE * fp = NULL;
```

```
fp = fopen（文件名，文件使用方式）；
```

```
fp = fopen（文件名，文件使用方式）；
```

①文件名：

要操作的文件的名字、可以包含路径信息

②文件使用方式：

“读”、“写”、“文本”或“二进制”等

③fp文件指针：

指向被打开的文件，失败返回空，成功返回相应指针

文件使用方式：

r:以只读的方式打开文件

r: 以只读方式打开文件

1、文件不存在返回NULL；

2、文件存在返回文件指针，进行后续的读操作

w:以只写的方式打开文件

w: 以只写方式打开文件

① 文件不存在，以指定文件名创建此文件；

② 若文件存在，清空文件内容，进行写操作；

③ 如果文件打不开（比如文件只读），返回NULL

a:以追加的方式打开文件（往文件的末尾写入数据）

a: 以追加方式打开文件

① 文件不存在，以指定文件名创建此文件（同w）

② 若文件存在，从文件的结尾处进行写操作

+: 以**可读可写**的方式打开

b:以**二进制**的方式打开文件

t:以**文本**的方式打开文件（省略）

打开方式的组合形式：1-1

模 式	功 能
r或rb	以只读方式打开一个文本文件（不创建文件）
w或wb	以写方式打开文件（使文件长度截断为0字节，创建一个文件）
a或ab	以添加方式打开文件，即在末尾添加内容，当文件不存在时，创建文件用于写
r+或rb+	以可读、可写的方式打开文件(不创建新文件)
w+或wb+	以可读、可写的方式打开文件 (使文件长度为0字节，创建一个文件)
a+或ab+	以添加方式打开文件，打开文件并在末尾更改文件（如果文件不存在，则创建文件）

关闭文件fclose

➤关闭文件

1. 调用的一般形式是：

fclose（文件指针）；

文件指针：指向要关闭的文件

2. 返回值：

- ① 关闭文件成功，返回值为0.
- ② 关闭文件失败，返回值非零.

知识点3 【文件的字节读写】

1、字节的读操作 fgetc函数

字节的读写

ch = fgetc (fp) ; //读一个字节

说明:从指定文件读一个字节赋给ch（以“读”或“读写”方式打开）

文本文件： 读到文件结尾返回EOF

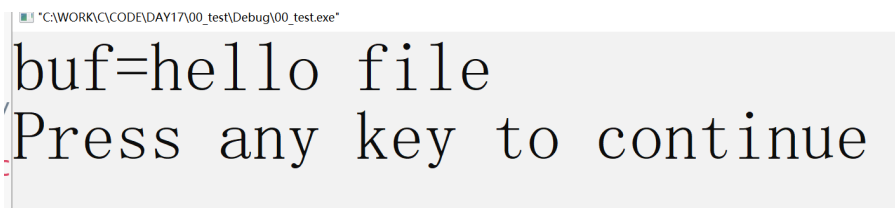
二进制文件：读到文件结尾，使用feof(后面会讲)判断结尾

```

1 void test03()
2 {
3     char buf[128]="";
4     int i=0;
5     FILE *fp = NULL;
6     //1、使用fopen打开一个文件 获得文件指针
7     fp = fopen("a.txt", "r");
8     if(fp == NULL)
9     {
10        perror("fopen");
11        return;
12    }
13
14    //2、对文件的操作 fgetc
15    while(1)
16    {
17        //fgetc调用一次 读取到一个字节
18        buf[i] = fgetc(fp);
19        if(buf[i] == EOF)//EOF表已经对到文件末尾
20        {
21            break;
22        }
23        i++;
24    }
25    printf("buf=%s\n", buf);
26
27    //3、关闭文件
28    fclose(fp);
29 }

```

运行结果：事先 本地 创建a.txt 文件内容 为 hello file



```

C:\WORK\C\CODE\DAY17\00_test\Debug\00_test.exe
buf=hello file
Press any key to continue

```

2、字节的写操作 fputc

`fputc(ch, fp);` //写一个字符

说明:把一个ch变量中的值(1个字节)写到指定的文件

如果输出成功, 则返回输出的字节;

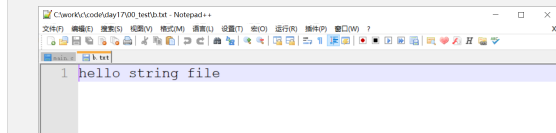
如果输出失败, 则返回一个EOF。

EOF是在stdio.h文件中定义的符号常量, 值为-1

```
1 void test04()
2 {
3     char buf[128]="";
4     int i=0;
5     FILE * fp =NULL;
6     fp = fopen("b.txt", "w");
7     if(fp == NULL)
8     {
9         perror("fopen");
10        return;
11    }
12
13    //使用fputc进行文件的数据写入
14    printf("请输入要写入文件的字符串:");
15    fgets(buf, sizeof(buf), stdin); //会获取换行符
16    buf[strlen(buf)-1] = 0; //去掉键盘输入的换行符
17
18    //将字符串buf中的元素 逐个写入文件中
19    while(buf[i] != '\0')
20    {
21        fputc(buf[i], fp);
22        i++;
23    }
24
25    fclose(fp);
26 }
```

运行结果:

请输入要写入文件的字符串:hello string file
Press any key to continue



练习:

练习

从一个文件(文本文件)中读取所有信息，写入另一个文件中

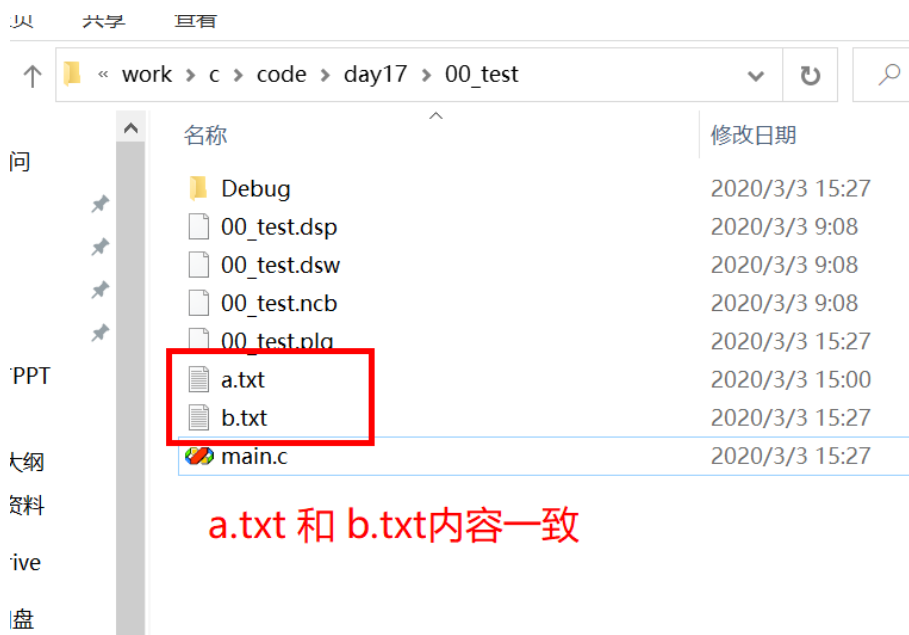
```
1 void test05()
2 {
3     //需求: 从a.txt读取文件内容 写入到b.txt
4     FILE *fp1 =NULL;
5     FILE *fp2 =NULL;
6
7     //以只读的方式打开a.txt
8     fp1 = fopen("a.txt","r");
9     if(fp1 == NULL)
10    {
11        perror("fopen");
12        return;
13    }
14
15    //以只写的方式打开b.txt
16    fp2 = fopen("b.txt","w");
17    if(fp2 == NULL)
18    {
19        perror("fopen");
20        return;
21    }
22
23
24    //从fp1中 每读取一个 字节 写入到fp2中
25    while(1)
26    {
27        char ch;
28        //读
29        ch = fgetc(fp1);
30        if(ch == EOF)//已经读到文件末尾
```

```

31  break;
32
33  //写
34  fputc(ch,fp2);
35
36  }
37  fclose(fp1);
38  fclose(fp2);
39  return;
40  }

```

运行结果:



知识点4【文件的字符串读写】1-2

1、使用fputs 往文件中写入一个字符串

➤ 字符串的读写

```
fputs("china",fp); //写一个字符串
```

说明:

1. 向指定的文件写一个字符串
2. 第一个参数可以是字符串常量、字符数组名或字符指针
3. 字符串末尾的'\0' 不会写到文件中

案例:

```

1 void test06()
2 {
3  //指针数组

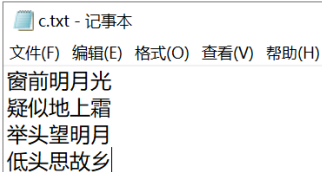
```

```

4 char *buf[]={ "窗前明月光\n", "疑似地上霜\n", "举头望明月\n", "低头思故乡"};
5 int n = sizeof(buf)/sizeof(buf[0]);
6 FILE *fp = NULL;
7 int i=0;
8
9 fp = fopen("c.txt", "w");
10 if(fp == NULL)
11 {
12     perror("fopen");
13     return;
14 }
15
16 for(i=0; i<n; i++)
17 {
18     fputs(buf[i], fp);
19 }
20
21 fclose(fp);
22 }

```

运行结果：



c.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

窗前明月光
疑似地上霜
举头望明月
低头思故乡

2、使用fgets从文件中获取字符串

➤ 字符串的读写

```
fgets(str, n, fp); //读一个字符串
```

说明：从fp指向的文件读入n-1个字符，在读入n-1个字符之前遇到换行符或EOF，读入提前结束，并读取换行符，在最后加一个' \0'，str为存放数据的首地址

返回值：

成功:返回读到字符串的首元素地址

失败：返回NULL

获取文件一行的数据：

```
1 void test07()
```

```

2 {
3   char buf[128]="";
4   FILE *fp = NULL;
5   char *path = "c.txt";
6   fp = fopen(path, "r");
7   if(fp == NULL)
8   {
9     perror("fopen");
10    return;
11  }
12
13  //err 打开一个文件名叫"path" 而不是path指向的文件名"c.txt"
14  //fp = fopen("path", "r");
15  //fp = fopen("c.txt", "r");
16
17  while(fgets(buf, sizeof(buf), fp))
18  {
19    printf("%s\n", buf);
20  }
21
22  fclose(fp);
23 }

```

运行结果：

```

窗前明月光
疑似地上霜
举头望明月
低头思故乡
Press any key to continue

```

知识点5 【文件的块读写】 fread fwrite

1、使用fwrite 将 数据块 写入到文件中

`fwrite(buffer, size, count, fp);`

说明:

参数:

buffer: 指向存储数据空间的首地址的指针

size: 一次读写的数据块大小

count: 要读写的数据块个数

fp: 指向要进行写操作的文件指针

返回值:

实际读写的数据块数（不是总数据大小，重要）

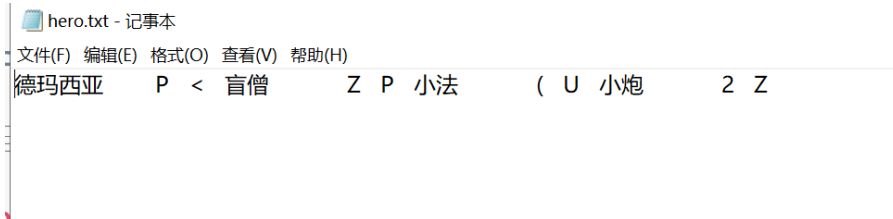
```
1 typedef struct
2 {
3     char name[16]; // 姓名
4     int deff; // 防御
5     int atk; // 攻击
6 } HERO;
7
8 void test08()
9 {
10     HERO hero[] = {
11         {"德玛西亚", 80, 60},
12         {"盲僧", 90, 80},
13         {"小法", 40, 85},
14         {"小炮", 50, 90}
15     };
16     int n = sizeof(hero) / sizeof(hero[0]);
17
18     FILE *fp = NULL;
19     fp = fopen("hero.txt", "w");
20     if(fp == NULL)
21     {
22         perror("fopen");
23         return;
24     }
```

```

25
26 //fwrite 将内存的数据原样的 输出到 文件中
27 //写入文件的数据 不便于 用户查看 但是 不会影响 程序的读
28 fwrite(hero, sizeof(HERO), n, fp);
29
30 fclose(fp);
31 }

```

运行结果：



2、使用fread 从文件中 读取 数据块

```
fread(buffer, size, count, fp);
```

说明：

参数：

buffer: 指向存储数据空间的首地址的指针
size: 一次读写的数据块大小
count: 要读写的数据块个数
fp: 指向要进行写操作的文件指针

返回值：

实际读写的数据块数（不是总数据大小，重要）

案例：

```

1 void test09()
2 {
3     HERO hero[4];
4     int i=0;
5     FILE *fp = NULL;
6     fp = fopen("hero.txt", "r");
7     if(fp == NULL)
8     {
9         perror("fopen");
10    return;
11    }
12

```

```
13  fread(hero,sizeof(HERO), 4, fp);
14
15  for(i=0;i<4;i++)
16  {
17  //printf("英雄姓名:《%s》,防御:《%d》,伤害:《%d》\n", \
18  hero[i].name,hero[i].deff,hero[i].atk);
19  printf("英雄姓名:《%s》,防御:《%d》,伤害:《%d》\n", \
20  (hero+i)->name,(hero+i)->deff,(hero+i)->atk);
21  }
22
23
24  fclose(fp);
25 }
```

运行结果:

```
"C:\WORK\C\CODE\DAY17\00_test\Debug\00_test.exe"
英雄姓名:《德玛西亚》,防御:《80》,伤害:《60》
英雄姓名:《盲僧》,防御:《90》,伤害:《80》
英雄姓名:《小法》,防御:《40》,伤害:《85》
英雄姓名:《小炮》,防御:《50》,伤害:《90》
Press any key to continue_
```