

说明 - 作业四

题目要求

本次作业需要大家实现一个日记管理系统，包括四个命令行程序，分别是：

- pdadd
- pdlist []
- pdshow
- pdremove

这四个程序分别实现添加日记，列出指定日期范围内的日记，展示某篇日记的全文和删除某篇日记的功能。具体要求见pta。

在这里做一下补充解释：

1. 四个程序都要从指定的数据文件中读取日记数据，并把处理后的数据写回该文件。
2. 虽然要求四个程序是相互独立的，但事实上日记的数据结构是可以在四个程序之间共用的。对共用的数据结构尽量采用代码复用的方式实现，而不是在项目之间复制代码。

关于重定向

什么是重定向？

我们在调试控制台程序时，往往需要通过stdin, stdout与程序进行交互。当输入输出数据量较小时，直接用键盘输入即可，但当数据量较大或者需要反复多次输入时，键盘输入耗时较长，且容易出错，效率较低。

如何解决这个问题呢？可以采用读取和写入文件的方式。相信基本的文件IO大家应该不陌生。这里推荐另外一种方法叫做重定向。

重定向是指将原本从标准输入流中读取数据的方式改为从指定的文件（或其他流）中读取，或者将原本输出到标准输出流的数据改为指定的文件（或其他流）。使用重定向方法可以实现不修改代码或者仅修改少量代码，即可像操作标准输入输出流一样操作文件。

如何实现

实现重定向的方法有两类：

1. 通过命令行调用
2. 通过代码实现

命令行调用方式

例如下面一段程序：

```
#include <iostream>
#include <cmath>

int main(int argc, char* argv[]) {
    std::cout << "input n: ";
    int n;
    std::cin >> n;
    int* data = new int[n];
    int i = 0;
    std::cout << "input data: \n";
    while(i < n) {
        std::cin >> data[i++];
    }
    i = 0;
    std::cout << "output data: \n";
    while(i < n) {
        std::cout << pow(data[i++], 2) << ' ';
    }
    std::cout << "\n";
    delete[] data;
    return 0;
}
```

这段代码实现的功能是读取一些整数，输出他们的平方。编译并执行这段代码

```
g++ main.cpp -o main
./main
```

输出为：

```
(base) → hw g++ main.cpp -o main
(base) → hw ./main
input n: 5
input data:
1
2
3
4
5
output data:
1 4 9 16 25
```

采用重定向方法可以这样做:

首先创建我们需要数据文件data.txt:

```
5
1
2
3
4
5
```

同样的代码不变，我们在执行程序的时候采用如下命令:

```
./main < data.txt
```

结果如下:

```
(base) → hw ./main < data.txt
input n: input data:
output data:
1 4 9 16 25
```

我们得到了相同的答案但并没有通过键盘输入。
这里的

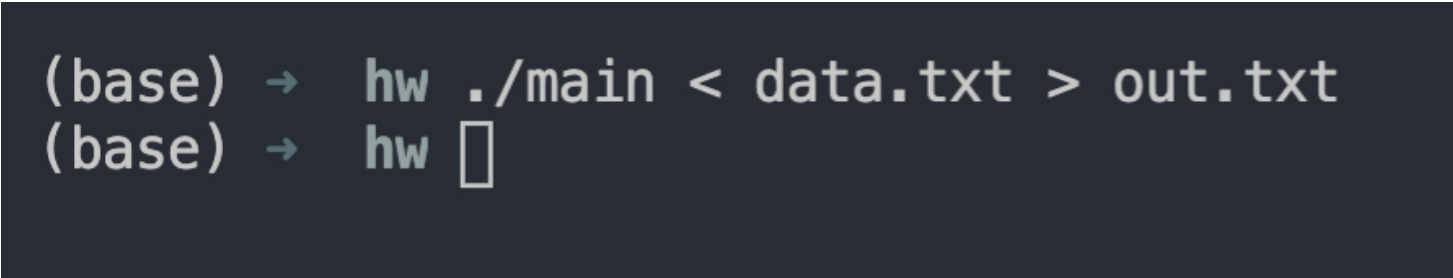
```
< data.txt
```

可以理解为将标准输入流重定向为文件data.txt。

同理，我们可以将输出也进行重定向：

```
./main < data.txt > out.txt
```

运行结果如下：



```
(base) → hw ./main < data.txt > out.txt  
(base) → hw □
```

貌似什么都没有发生，但是我们的程序目录下多出了一个out.txt，里面的内容是：

```
input n: input data:  
output data:  
1 4 9 16 25
```

out.txt里记录了程序的标准输出内容，实现了重定向。

更多的细节可以参考一下[这篇博客](#)

在Windows 上如何实现上述调用呢？

一种方法呢是通过Windows 控制台，使用上述调用程序的命令，可以实现同样的功能。另一种方法是在VS的项目设置中，增加命令参数
可以参考[这篇博客](#)

通过代码实现

通过代码实现文件重定向也有多种方式：

第一种方法是通过freopen函数

```
freopen("data.txt", "r", stdin);  
freopen("out.txt", "w", stdout);
```

之后std::cout(或scanf函数)和std::cout(或printf函数) 就会分别对data.txt和out.txt进行操作了。

第二种方法是通过C++的流对象缓冲区指针实现，主要用到ios::rdbuf()方法，具体可以参考[这篇博客](#)

作业对重定向的要求

对于使用到stdin, stdout的程序，需要大家使用重定向方法，将输入输出流重定向到文件，用来读取测试用例。注意这里说的重定向到文件，并不是保存日记数据的文件，而是测试用例的文件。也就是说大家在提交作业时，除了代码和工程文件外还要提交至少两个数据文件，一个是用来保存日记数据的文件，其他的一个或者多个是测试用例文件。我们在测试打分时候使用重定向方法读取和输出测试用例文件会节省我们测试的时间。

我推荐大家采用通过代码实现重定向的方法，如果使用的是通过命令行调用实现重定向，请务必同时提交一个说明文档，告诉我们应该如何调用！！

事实上，作业中的程序间可以通过管道的方式进行重定向，实现在程序间交互。例如可以将pdlist的输出作为pdremove的输入，可以直接删除指定范围内的所有日记。**这个功能不作为评分依据，但鼓励学有余力的同学尝试实现这一功能。**