Pascal 语言中，复合语句是用 begin……end 表示，条件语句的格式是 if ……then……else……表示，其中，else 子句可以出现也可以不出现。编一函数，检查 Pascal 源文件中的 begin 和 end 是否配对，if、then、else 是否配对。

【解】Pascal 的主程序或函数体和过程体本身就是一个由 begin……end 括起来的复合语句。如果有一个函数 skipComplexSentence 可以检查一个复合语句的正确性，那么主程序的实现就非常简单。主程序要求用户输入需要检查的源文件的名字，并打开源文件，从中找出第一个关键词（begin，end，if，then， else）。如果第一个关键词不是 begin，则报错，否则调用 skipComplexSentence 完成复合语句的检查。

skipComplexSentence 函数不断地读文件，在得到与 begin 对应的 end 后，skipComplexSentence 函数结束。此时，应该正好读完文件。如果文件中还有其他没有读完的内容，这些内容是多余的内容，报错。否则检查完毕，程序终止。

skipComplexSentence 函数的原型为

int skipComplexSentence(ifstream &infile);

其中，参数是输入文件对应的文件流对象，返回值是函数中是否发现错误的标志。没有错误，返回 0。发现错误，返回 1。错误信息在函数内输出。

主函数如代码清单 3-13 所示。

代码清单 3-13 检查 begin 和 end 是否配对，if、then、else 是否配对的主程序

```
1.  int main()
2.  {
3.      char filename[80];
4.       char txt[80];
5.       ifstream infile;                    //源文件对应的流对象
6.
7.       //输入并打开源文件
8.      cout << "请输入文件名：";
9.      cin >> filename;
10.     infile.open(filename);
11.      if (!infile) {
12.            cerr << "文件不存在";
13.            return 1;
14.      }
15.
16.     //寻找函数体的起点
17.      while (infile >> txt && strcmp(txt, "begin") != 0)  {
18.      if ((strcmp(txt, "end") == 0)  | | (strcmp(txt, "if") == 0)
19.        | | (strcmp(txt, "then") == 0) | | (strcmp(txt, "else") == 0)){
20.              cerr << "函数没有以 begin 开始" << endl;
21.                infile.close();
22.              return 1;
23.          }
```

```
24.         }
25.
26.         if (infile.eof()) {                         //没有找到 begin
27.               cerr << "没有函数体" << endl;
28.               infile.close();
29.               return 1;
30.         }
31.
32.         if (skipComplexSentence(infile)) {          //复合语句中发现错误
33.               infile.close();
34.               return 1;
35.         }
36.
37.         if (infile >> txt)                          // end 后是否还有语句
38.            cerr << "函数体外有多余的语句";
39.          else cerr << "程序正确";
40.
41.         infile.close();
42.         return 0;
43.  }
```

skipComplexSentence 不断地读文件，直到读到匹配的 end。期间可能遇到各种其他的语句。我们忽略其他语句的正确性，只检查 if 语句。于是对读到的词语检查是否为 if、then 或 else。如果是 then 或 else，表示这个子句缺少 if，报错。如果读到的是 if，开始检查 if 语句的正确性。If 语句的正确性用一个函数 skipIfSentence 实现 。skipComplexSentence 的实现如代码清单 3-14 所示。

代码清单 3-14 检查一个复合语句是否正确

```
1.    int skipComplexSentence(ifstream &infile)
2.    {
3.          char txt[80];
4.          while (infile >> txt && strcmp(txt, "end") != 0) { //读文件，直到发现 end
5.            if (strcmp(txt, "then") == 0 || strcmp(txt, "else") == 0){// then 和 else
6.                     cerr << "then 或 else 子句缺少 if" << endl;
7.                     return 1;
8.               }
9.          if (strcmp(txt, "if") == 0) {                    // if 的处理
10.                    if (skipIfSentence(infile)) return 1;
11.              }
12.         }
13.         if (infile.eof()) {                    //直到文件结束都没有读到 end
14.               cerr << "缺少 end" <<endl;
15.               return 1;
16.         }
```

```
17.
18.         return 0;
19.  }
```

比较复杂的是 if 语句的处理，这是由函数 skipIfSentence 完成的。该函数不断地读文件，试图找到 then。如果在发现 then 之前读到了 else，这个 if 语句肯定出错。If 语句可以没有 else 子句但不能没有 then 子句。如果在发现 then 之前读到了 begin 或 end，这也是个错误。因为 if 和 then 之间应该是一个逻辑表达式，不应该出现 begin、end。当正确读到 then 后，函数开始处理 then 子句。then 后面应该是一个语句，可以是一个简单的语句，也可以是一个复合语句。我们设计了一个函数 skipSentence，用以跳过 then 子句。如果 then 子句是正确的，那么后面应该是一个 else 子句或另一个语句，于是继续读文件。如果读到的是 else，则跳过 else 子句。如果不是 else，则是另一个语句。将读入的词语放回文件流。处理结束。这个过程如代码清单 3-15 所示。

代码清单 3-15 skipIfSentence 的处理

```
1.    int skipIfSentence(ifstream &infile)
2.    {
3.          char txt[80];
4.          while (infile >> txt && strcmp(txt, "then") != 0) { //读文件，直到 then
5.                if (strcmp(txt, "else") == 0) {                    // 读到 else
6.                      cerr << "else 无相应的 if……then" << endl;
7.                      return 1;
8.                }
9.                if (strcmp(txt, "begin") == 0 || strcmp(txt, "end")==0) {//begin、end
10.                     cerr << "if 的条件格式错" << endl;
11.                     return 1;
12.               }
13.         }
14.      if  (infile.eof()) {                          // 没有读到 then
15.               cerr << "if 语句缺少 then 子句" << endl;
16.               return 1;
17.         }
18.
19.         if (skipSentence(infile)) return 1;                //检查 then 子句
20.         infile >> txt;                                     //继续读入
21.         if  (strcmp(txt, "else") == 0) {                   //else 子句的处理
22.               if (skipSentence(infile)) return 1;
23.               }
24.           else back(infile, txt);                // 无 else 子句，放回输入流
25.         return 0;
26.   }
```

skipIfSentence 函数中用到了两个函数 skipSentence 和 back。前者用于跳过一个语句，后者用于将刚读入的关键词放回输入流。它们的实现分别见代码清单 3-16 和 3-17。函

数 skipSentence 首先读入一个单词。如果读入的是 begin，表示 then 子句是一个复合语句，调用 skipComplexSentence 跳过这个复合语句。否则 then 子句由一个简单语句组成。PASCAL 语言的语句都是以分号结束的，于是不断读文件直到读到分号。

代码清单 3-16 skipSentence 的实现

```
1.  int skipSentence(ifstream &infile)
2.  {
3.      char txt[80];
4.
5.      infile >> txt;                          // 读入一个单词
6.      if (strcmp(txt, "begin") == 0) {        // 复合语句处理
7.          if (skipComplexSentence(infile)) return 1;
8.      }
9.      else {                          // 简单语句处理，语句以分号结束
10.         while (!infile.eof() && txt[strlen(txt)-1] != ';') infile >> txt;
11.         if (infile.eof()) {
12.             cerr << "语句没有结束" << endl;
13.             return 1;
14.         }
15.     }
16.
17.     return 0;
18. }
```

代码清单 3-16 back 函数的实现

```
19. void back(ifstream &infile, char *txt)
20. {
21.     for (int i = strlen(txt)-1; i>=0; --i)
22.         infile.putback(txt[i]);
23. }
```