

Listing 1: Implementation of macro processor

```

1  /*
2  * Practical 2: Design suitable data structures and implement Pass-I and
    * Pass-II of a two-pass macro- processor. The output of Pass-I (MNT, MDT
    * and intermediate code file without any macro definitions) should be
    * input for Pass-II.
3  * The Text Editor: VIM - Vi IMproved 9.0
4  * The Compiler: Apple clang version 13.1.6 (clang-1316.0.21.2.3)
5  * The Machine: MacBook Air M1 (Running ARM64 architecture)
6  */
7
8  #include <iostream>
9  #include <cstdio>
10 #include <cstring>
11 #include <cstdlib>
12
13 class deftab {
14 public:
15     char lab[10];
16     char opc[10];
17     char oper[10];
18 } d[10];
19
20 int main() {
21     char label[10], opcode[10], operand[10];
22     char macroname[10];
23     int i = 0, lines = 0;
24     FILE *f1, *f2, *f3;
25
26     f1 = fopen("macin.dat", "r");
27     f2 = fopen("macout.dat", "w");
28     f3 = fopen("deftab.dat", "w");
29
30     fscanf(f1, "%s%s%s", label, opcode, operand);
31
32     while (strcmp(opcode, "END") != 0) {
33         if (strcmp(opcode, "MACRO") == 0) {
34             strcpy(macroname, label);
35             fscanf(f1, "%s%s%s", label, opcode, operand);
36             lines = 0;
37
38             while (strcmp(opcode, "MEND") != 0) {
39                 fprintf(f3, "%s\t%s\t%s\n", label, opcode, operand);
40                 strcpy(d[lines].lab, label);
41                 strcpy(d[lines].opc, opcode);
42                 strcpy(d[lines].oper, operand);
43
44                 fscanf(f1, "%s%s%s", label, opcode, operand);
45                 lines++;
46             }
47         }
48         else if (strcmp(opcode, macroname) == 0) {
49             printf("Lines = %d\n", lines);
50
51             for (i = 0; i < lines; i++) {
52                 fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);
53                 printf("DLAB = %s \nDOPC = %s \nDOPER = %s \n", d[i].lab, d[i].opc,

```

```

        d[i].oper);
54     }
55 }
56 else {
57     fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);
58 }
59
60     fscanf(f1, "%s%s%s", label, opcode, operand);
61 }
62     fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);
63
64     fclose(f1);
65     fclose(f2);
66     fclose(f3);
67
68     printf("FINISHED\n");
69     return 0;
70 }

```

Listing 2: The input File name (macin.dat)

```

1  CALC      START    1000
2  SUM       MACRO    **
3  **        LDA      #5
4  **        ADD      #10
5  **        STA      2000
6  **        MEND     **
7  **        LDA      LENGTH
8  **        COMP     ZERO
9  **        JEQ      LOOP
10 **        SUM      **
11 LENGTH    WORD     0
12 LOOP     SUM      **
13 **       END      **

```

\_\_\_\_\_ (commends for linux or MacOS or Unix-Like OS) \_\_\_\_\_  
(Note: store main.cpp and main.dat are in same folder)

#### Linux OS

```
$ touch main.cpp           // For creating file  
(open that file in any text editor you want and write above code)  
  
$ g++ main.cpp -o main     // For compiling the file into executable  
  
$ ./main                   // For running that output file
```

#### MacOSX

```
$ touch main.cpp           // For creating file  
(open that file in any text editor you want and write above code)  
  
$ clang++ main.cpp -o main // For compiling the file into executable  
  
$ ./main                   // For running that output file
```

```
Assembler -- -bash -- 101x43
[[18:34:10][tejasmote]:~/Documents/Development/C++ Language/Assembler$ clang++ main.cpp -o main
[[18:34:13][tejasmote]:~/Documents/Development/C++ Language/Assembler$ ./main
Lines = 3
DLAB = **
DOPC = LDA
DOPE = #5
DLAB = **
DOPC = ADD
DOPE = #10
DLAB = **
DOPC = STA
DOPE = 2000
Lines = 3
DLAB = **
DOPC = LDA
DOPE = #5
DLAB = **
DOPC = ADD
DOPE = #10
DLAB = **
DOPC = STA
DOPE = 2000
FINISHED
[[18:34:19][tejasmote]:~/Documents/Development/C++ Language/Assembler$ cat macout.dat

**      LDA      #5
**      LDA      #5
**      LDA      #5
**      LDA      #5
**      ADD      #10
**      STA      2000

**      LDA      #5
**      ADD      #10
**      STA      2000

[[18:34:26][tejasmote]:~/Documents/Development/C++ Language/Assembler$ cat deftab.dat
**      LDA      #5
**      ADD      #10
**      STA      2000
[[18:34:32][tejasmote]:~/Documents/Development/C++ Language/Assembler$ █
```

Figure 1: The output terminal snap 1