Listing 1: Implementation of macro processor

```
/*
 * 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.
 * The Text Editor: VIM - Vi IMproved 9.0
 * The Compiler: Apple clang version 13.1.6 (clang-1316.0.21.2.3)
 * The Machine: MacBook Air M1 (Running ARM64 architecture)
 */

#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>

class deftab {
public:
  char lab[10];
  char opc[10];
  char oper[10];
} d[10];

int main() {
  char label[10], opcode[10], operand[10];
  char macroname[10];
  int i = 0, lines = 0;
  FILE *f1, *f2, *f3;

  f1 = fopen("macin.dat", "r");
  f2 = fopen("macout.dat", "w");
  f3 = fopen("deftab.dat", "w");

  fscanf(f1, "%s%s%s", label, opcode, operand);

  while (strcmp(opcode, "END") != 0) {
    if (strcmp(opcode, "MACRO") == 0) {
      strcpy(macroname, label);
      fscanf(f1, "%s%s%s", label, opcode, operand);
      lines = 0;

      while (strcmp(opcode, "MEND") != 0) {
        fprintf(f3, "%s\t%s\t%s\n", label, opcode, operand);
        strcpy(d[lines].lab, label);
        strcpy(d[lines].opc, opcode);
        strcpy(d[lines].oper, operand);

        fscanf(f1, "%s%s%s", label, opcode, operand);
        lines++;
      }
    }
    else if (strcmp(opcode, macroname) == 0) {
      printf("Lines = %d\n", lines);

      for (i = 0; i < lines; i++) {
        fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);
        printf("DLAB = %s \nDOPC = %s \nDOPER = %s \n", d[i].lab, d[i].opc,
```

```
            d[i].oper);
      }
    }
    else {
      fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);
    }

    fscanf(f1, "%s%s%s", label, opcode, operand);
  }
  fprintf(f2, "%s\t%s\t%s\n", d[i].lab, d[i].opc, d[i].oper);

  fclose(f1);
  fclose(f2);
  fclose(f3);

  printf("FINISHED\n");
  return 0;
}
```

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

```
CALC    START   1000
SUM     MACRO   **
**      LDA     #5
**      ADD     #10
**      STA     2000
**      MEND    **
**      LDA     LENGTH
**      COMP    ZERO
**      JEQ     LOOP
**      SUM     **
LENGTH  WORD    0
LOOP    SUM     **
**      END     **
```

(commends for linux or MacOS or Unix-Like OS)

```
(Note: store main.cpp and macin.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
```

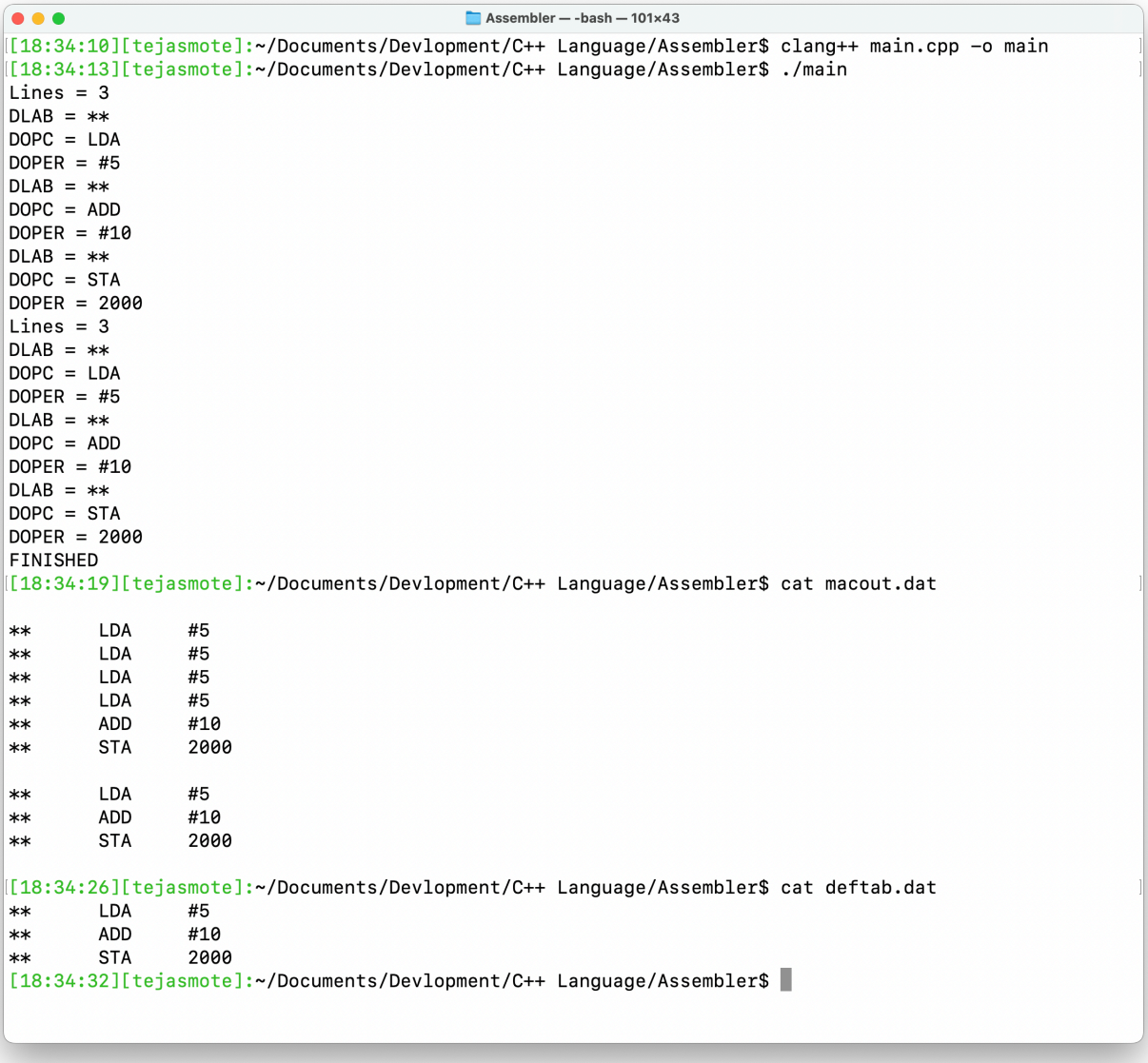

Figure 1: The output terminal snap 1