FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

Class	**T.E. Computer A**
Subject Name	**Systems Programming And Compiler Construction**
Subject Code	**CPC 601**

Practical No.	6
Title	Macro Processor
Date of Performance	15/04/2025
Date of Submission	23/04/2025
Roll No.	9914
Name	Vivian

Evaluation:

Sr. No	Rubric	Grade
1	Timeline(2)	
2	Output(3)	
3	Code Optimization(2)	
4	Postlab(3)	

Signature of the teacher:

CODE:

```
//MacroFirstPass.java

import java.io.IOException;
import java.nio.file.*;
import java.util.*;

class MNT {
  List<MNTLine> lines;

  public MNT() {
    this.lines = new ArrayList<>();
  }
}

class MNTLine {
  int indexOfMacro;
  int locationOfMacro;
  String nameOfMacro;

  public MNTLine(int indexOfMacro, int locationOfMacro, String nameOfMacro) {
    this.indexOfMacro = indexOfMacro;
    this.locationOfMacro = locationOfMacro;
    this.nameOfMacro = nameOfMacro;
  }
}

class Macro {
  List<Line> lines;
  HashMap<String, Integer> ALA;

  public Macro() {
    this.lines = new ArrayList<>();
    this.ALA = new HashMap<>();
  }

  public String parseMacroDefinition(String macroDefinition) {
    macroDefinition = macroDefinition.trim();
    String[] tokens = macroDefinition.split("\\s+", 2); // SWAP &X, &Y
    String macroName = tokens[0];
    if (tokens.length > 1) {
      parseArgs(tokens[1], 0);
    }
    return macroName;
  }

  private void parseArgs(String argString, int index) {
    String[] args = argString.split(",");
    for (int i = 0; i < args.length; i++) {
      String arg = args[i].trim();
      if (!arg.isEmpty()) {
```

```
        ALA.put(arg, i + index);
      }
    }
  }

  public void substituteArgsInBody() {
    for (int i = 0; i < lines.size(); i++) {
      String line = lines.get(i).line;
      for (Map.Entry<String, Integer> entry : ALA.entrySet()) {
        line = line.replace(entry.getKey(), "#{" + entry.getValue() + "}");
      }
      lines.set(i, new Line(line, lines.get(i).index));
    }
  }
}

class Line {
  String line;
  int index;

  public Line(String line, int index) {
    this.line = line;
    this.index = index;
  }
}

public class MacroFirstPass {

  public static List<String> readFile(String filename) {
    try {
      List<String> lines = Files.readAllLines(Paths.get(filename));
      System.out.println("Printing the lines of the " + filename + ":");
      for (String line : lines) {
        System.out.println(line);
      }
      System.out.println("========================================");
      return lines;
    } catch (IOException e) {
      e.printStackTrace();
      System.exit(2);
      return Collections.emptyList();
    }
  }

  public static List<Macro> getMacros(List<String> lines) {
    List<Macro> macros = new ArrayList<>();
    MNT mnt = new MNT();
    for (int i = 0; i < lines.size(); i++) {
      if (!lines.get(i).contains("MACRO"))
        continue;

      int j = i + 1;
```

```
      // Skip empty lines
      while (j < lines.size() && lines.get(j).trim().isEmpty()) {
        j++;
      }

      Macro macro = new Macro();
      String macroName = macro.parseMacroDefinition(lines.get(j));

      mnt.lines.add(new MNTLine(mnt.lines.size(), j, macroName));

      for (j = j; j < lines.size(); j++) {
        String line = lines.get(j).trim();
        macro.lines.add(new Line(line, j));
        if (line.contains("MEND")) {
          macro.substituteArgsInBody(); // substitute args with #{index}
          macros.add(macro);
          break;
        }
      }
      i = j;
    }

    if (macros.isEmpty()) {
      System.out.println("No macros definitions were found in the code");
    }

    printMNT(mnt);
    return macros;
  }

  public static void printMNT(MNT mnt) {
    System.out.println("\nPrinting MNT");
    System.out.println("Index\t|\tLocation\t|\tMacro Name");
    for (MNTLine line : mnt.lines) {
      System.out.println(line.indexOfMacro + "\t|\t" + line.locationOfMacro +
"\t\t|\t" + line.nameOfMacro);
    }
    System.out.println("=========================================");
  }

  public static void printMDT(List<Macro> MDT) {
    for (Macro macro : MDT) {
      System.out.println("\nNew macro");
      System.out.println("index\t|\tDefinition");
      for (Line line : macro.lines) {
        System.out.println(line.index + "\t|\t" + line.line);
      }
      System.out.println("=========================================");
      printALA(macro.ALA);
    }
  }
```

```
  public static void printALA(HashMap<String, Integer> ala) {
    System.out.println("ALA: " + ala);
    System.out.println("========================================");
  }

  public static void main(String[] args) {
    if (args.length < 1) {
      System.out.println("Usage: java MacroFirstPass <input_file>");
      System.exit(1);
    }

    List<String> lines = readFile(args[0]);
    List<Macro> MDT = getMacros(lines);
    printMDT(MDT);
  }
}
```

```
//MacroSecondPass.java

import java.io.IOException;
import java.nio.file.*;
import java.util.*;

public class MacroSecondPass {

  public static List<String> readFile(String filename) {
    try {
      return Files.readAllLines(Paths.get(filename));
    } catch (IOException e) {
      e.printStackTrace();
      System.exit(2);
      return Collections.emptyList();
    }
  }

  public static List<String> expandMacros(
      List<String> originalLines, List<Macro> MDT, MNT mnt) {
    List<String> expandedCode = new ArrayList<>();

    Set<Integer> macroDefLines = new HashSet<>();
    for (Macro macro : MDT) {
      for (Line line : macro.lines) {
        macroDefLines.add(line.index);
      }
    }

    for (int i = 0; i < originalLines.size(); i++) {
      String line = originalLines.get(i).trim();

      if (macroDefLines.contains(i) || line.equals("MACRO") ||
line.equals("MEND")) {
        continue; // Skip macro definition lines
```

```
      }

      String[] tokens = line.split("\\s+", 2);
      String macroName = tokens[0];
      Optional<MNTLine> mntLineOpt = mnt.lines.stream()
          .filter(m -> m.nameOfMacro.equals(macroName))
          .findFirst();

      if (mntLineOpt.isPresent()) {
        MNTLine mntLine = mntLineOpt.get();
        Macro macro = MDT.get(mntLine.indexOfMacro);

        HashMap<Integer, String> actualArgs = new HashMap<>();
        if (tokens.length > 1) {
          String[] args = tokens[1].split(",");
          for (int j = 0; j < args.length; j++) {
            actualArgs.put(j, args[j].trim());
          }
        }

        for (int k = 1; k < macro.lines.size(); k++) {
          Line macroLine = macro.lines.get(k);
          String expanded = macroLine.line;
          for (Map.Entry<Integer, String> arg : actualArgs.entrySet()) {
            expanded = expanded.replace("#{" + arg.getKey() + "}",
arg.getValue());
          }
          if (!expanded.equals("MEND")) {
            expandedCode.add(expanded);
          }
        }
      } else {
        expandedCode.add(originalLines.get(i));
      }
    }

    return expandedCode;
  }

  public static void writeToFile(List<String> lines, String filename) {
    try {
      Files.write(Paths.get(filename), lines);
      System.out.println("Macro-expanded code written to " + filename);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  public static void printToConsole(List<String> lines) {
    System.out.println("\nExpanded Code:");
    for (String line : lines) {
      System.out.println(line);
```

```
    }
  }

  public static void main(String[] args) {
    if (args.length < 1) {
      System.out.println("Usage: java MacroSecondPass <input_file>
[output_file]");
      System.exit(1);
    }

    String inputFile = args[0];
    String outputFile = args.length > 1 ? args[1] : null;

    // Step 1: Read input file
    List<String> lines = MacroFirstPass.readFile(inputFile);

    // Step 2: Extract macros (MDT, ALA) using first pass
    List<Macro> MDT = MacroFirstPass.getMacros(lines);

    // Step 3: Build MNT from MDT
    MNT mnt = new MNT();
    for (int i = 0; i < MDT.size(); i++) {
      String macroName = MDT.get(i).lines.get(0).line.split("\\s+")[0];
      mnt.lines.add(new MNTLine(i, MDT.get(i).lines.get(0).index, macroName));
    }

    // Step 4: Print MDT (including ALA) and MNT
    System.out.println("\n========== Macro Definition Table (MDT)
==========");
    MacroFirstPass.printMDT(MDT);

    System.out.println("\n========== Macro Name Table (MNT) ==========");
    MacroFirstPass.printMNT(mnt);

    // Step 5: Expand macros
    List<String> expandedCode = expandMacros(lines, MDT, mnt);

    // Step 6: Print final expanded code
    System.out.println("\n========== Final Expanded Code ==========");
    for (String line : expandedCode) {
      System.out.println(line);
    }
    System.out.println("========================================");

    // Step 7: Optionally write to output file
    if (outputFile != null) {
      writeToFile(expandedCode, outputFile);
    }
  }
}
```

Language	Files	Lines	Code	Comments	Blanks
Java	2	298	242	8	48
Plain Text	1	13	0	12	1
Total	3	311	242	20	49

INPUT:

```
//input.txt
```

```
MACRO
SWAP &X, &Y
LD TEMP, &X
LD &X, &Y
LD &Y, TEMP

MEND
start
ADD A
Load B
SWAP z1, z2
ADD C
end
```

OUTPUT:

```
college/spcc/macro-expansion (main)  v23
javac *.java && java MacroSecondPass input.txt
Picked up _JAVA_OPTIONS: -Djava.util.prefs.userRoot=/home/shadow/.config/ja
va
Picked up _JAVA_OPTIONS: -Djava.util.prefs.userRoot=/home/shadow/.config/ja
va
Printing the lines of the input.txt:
MACRO
SWAP &X, &Y
LD TEMP, &X
LD &X, &Y
LD &Y, TEMP

MEND
start
ADD A
Load B
SWAP z1, z2
ADD C
end
=========================================

Printing MNT
Index   |       Location        |       Macro Name
0       |       1               |       SWAP
=========================================

========== Macro Definition Table (MDT) ==========

New macro
index   |       Definition
1       |       SWAP #{0}, #{1}
2       |       LD TEMP, #{0}
3       |       LD #{0}, #{1}
4       |       LD #{1}, TEMP
```

```
========== Macro Definition Table (MDT) ==========

New macro
index   |       Definition
1       |       SWAP #{0}, #{1}
2       |       LD TEMP, #{0}
3       |       LD #{0}, #{1}
4       |       LD #{1}, TEMP
5       |
6       |       MEND
=========================================
ALA: {&X=0, &Y=1}
=========================================

========== Macro Name Table (MNT) ==========

Printing MNT
Index   |       Location        |       Macro Name
0       |       1               |       SWAP
=========================================

========== Final Expanded Code ==========
start
ADD A
Load B
LD TEMP, z1
LD z1, z2
LD z2, TEMP

ADD C
end
=========================================
```