ECOTE – preliminary project

Semester: 20L

Author: Filip Kwiatkowski

Subject: Compilation Techniques

I. General overview and assumptions

The goal of this program is to serve as a macrogenerator, that can accept macro definitions without any parameters and with possibility of nested macros. The input has to be provided as a single file and all of the macro definitions (nested and not nested) have to be properly closed.

II. Functional requirements

The program will serve as a parser for the input text file. Based on the result of that parser, the macro calls will be removed from the result, macro calls will be properly replaced with free text, and free text will not be altered in any way.

III. Implementation

General architecture

The input will be transformed into lexems that will be then used to create an abstract syntax tree where each of the leaves will be a macro definition, macro call or the free text. When traversing this abstract syntax tree, if the algorithm will encounter a macro definition, the name of the macro will be pushed to the stack to indicate its beginning. When the closing symbol for the macro would be detected, the algorithm will close the last opened macro and its definition will be stored inside a dictionary with a parameter defining the context of the definition, that will allow for differentiating the global macros from the nested macros. If the algorithm encounters a macro call, it will look for it inside the dictionary and print the result of the macro to the output, and if the algorithm encounters a free text, it will just print it to the output.

Data structures

The data structures used in this project will be a lexems that will be generated during the process of analyzing the input file, an abstract syntax tree created using the lexems and the dictionary containing the macro definitions.

Module descriptions

Lexer – responsible for analyzing the input file and splitting it into one of the 3 categories (macro definition, macro call, free text).
Parser – responsible for analyzing the output of the lexer and performing appropriate actions based on the lexems.

Input/output description

The input and output will take form of 2 text files. The input file will contain 3 types of expressions: macro definition, macro call and some free text. The resulting output file will be based on the input,

but with removed macro definitions, properly replaced macro calls based on the macro definitions, and unchanged free text.

Others

IV. Functional test cases

Proper use test cases:

Input:
&a
Abc
&
$a
Output:
Abc

Input:
&a
Abc
&b
Cda
&
Xyz
$b
&
Output:
Abc
Xyz
Cda

Error detection test cases:

Input:
&a
Abc
$a
Output:
ERROR: DEFINITION OF MACRO HAS NO CLOSE STATEMENT

Input:
&a
Abc
$a
&
$a
Output:

Abc
ERROR: MACRO ‘a’ HAS NOT BEEN DEFINED

Input:
&a
Abc
&b
Cde
&
&
$a
$b
Output:
Abc
ERROR: MACRO ‘b’ HAS NOT BEEN DEFINED

Input:
&a
Abc
&b
Cde
&
Xyz
$b
$c
&
&
$a
Output:
Abc
Xyz
Cde
ERROR: MACRO ‘c’ HAS NOT BEEN DEFINED
ERROR: MACRO CLOSE STATEMENT WITHOUT OPENING FOUND