ECOTE - preliminary project

Date: 27-04-2021

Semester: 21L

Author and Group: Piotr Kochański (289372), group 103

Subject: Macro-generator with "key" parameters

I. General overview and assumptions

Macro-generator with "key" parameters is a universal general purpose macro processor. The macro call argument values are declared using name=value syntax, and macro body definitions reference parameters by their name. This allows to write macro call arguments in any order, decide to omit some or pass additional, not used in the body arguments.

1. Functional requirements

Macro generator is capable of transforming text with embedded macro definitions and macro calls to text with these macros defined in input text inserted and parameter substitution performed. Macro definitions reference parameters by name specified as arguments to the macro call in the form of name=value. The parameter specified by name in the macro body is substituted by the value.

Reading files or standard input containing special syntax and outputting it transformed according to rules specified in the input/output description section either to file or standard output with errors reported on the standard error stream. Huge input files as well as really big macro libraries should be handled efficiently. Issues such as errors with files, syntax errors are to be reported and execution should continue if possible.

II. Implementation

General architecture

The central part of the program is the function called the "switch". Its job is to read in text and decide what should be done as well as delegating that work to other parts of the program.

Upon encountering a discriminant character it should decide whether a macro call or macro definition was found and either instruct to add a macro to the macro library or try to find and apply macro that was previously defined. Otherwise it directs to output free text.

Macro library is global – that is non-hierarchical and non-local – new definitions of macros "hide" old definitions. Text substitution is non-hierarchical and only one text level is permitted.

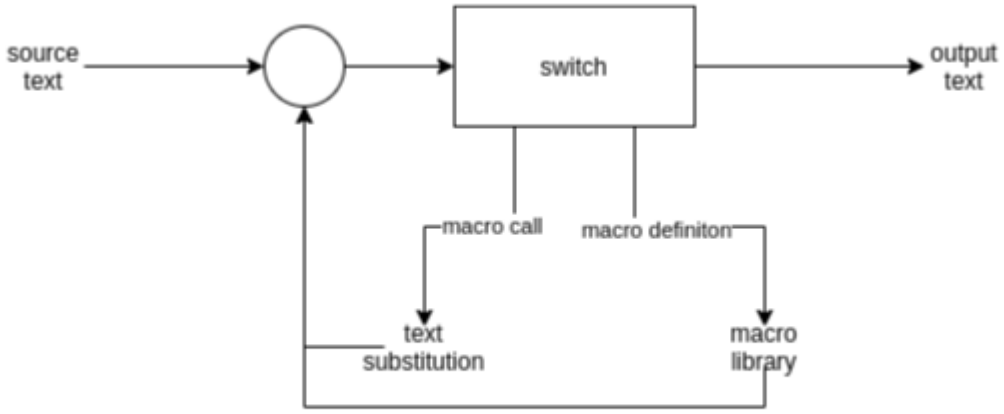
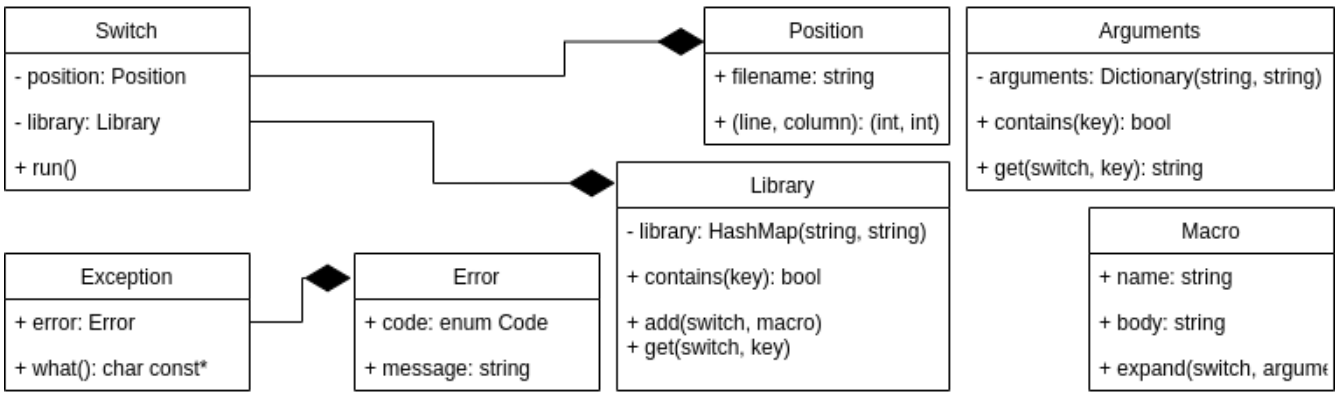


Figure 1: general architecture

Data structures

A macro is a pair of macro name and macro body. Both ought to be stored as strings. Because of restrictions imposed on macro library, namely it being non-hierarchical, and non-local, that is a flat list of global macros is to be stored, the same for all text levels, a simple associative container can be used. For constant time macro retrievals a hash map needs to be used with the key being the macro name. For macro calls a list of parameters is required, and these, bearing their nature of being key value pairs, are naturally predisposed to be stored in some sort of dictionary.



Input/output description

Input is taken either from the standard input stream or from a file. Conversely output is to either standard output or to a file. In place editing is prohibited. Errors are reported on the standard error. The input should be either normal free text or special forms indicated by a discriminant character that is macro definitions or macro calls.

The normal free text is put directly to output without any modifications.

Discriminant

The discriminant character for macro definitions and macro calls is #. Once that character is encountered it must be followed without any characters between by MDEF, MEND, MCALL or another #. If it is followed by anything else or not followed by anything at all error is reported and # nor optional following word is not inserted.

Macro definition

The #MDEF directive designates a macro definition and is specified using following syntax:

#MDEF name
body
#MEND

First word after the #MDEF directive is the macro name that can be later used to reference that macro. Macro name is any number of non-whitespace characters. It is separated from the #MDEF directive and the macro body by whitespace, the macro body is ended with newline.

The body of the macro is anything that follows a newline after macro name and is before the #MEND directive that closes off the macro definition. A macro body might contain names of parameters to be substituted in a macro call. These are designated by and are of a form $name. The body might be empty – is optional.

The #MEND directive must be only used to end started macro definitions – each #MDEF must be ended with #MEND and there must not be any #MENDs not belonging to #MDEF. If any of #MDEF, name or #MEND is missing, error is reported and macro is not inserted into the library.

Parameters

Parameters in the macro body are of the form: $name - that is dollar sign followed directly without whitespace by the name of the parameter. In order to output literal dollar sign $$ must be used in the macro body. In order for the parameter to be recognized there must be whitespace after the end of the name – the name is any number of characters until whitespace is encountered. Parameter names starting with double underscore are reserved.

Macro call

The #MCALL directive is used to insert the macro body of corresponding macro with appropriate parameter substitutions performed.

#MCALL name list-of-parameters

The name is the name of the macro to be used. If the macro name is missing or there is no macro by that name error is reported and no call happens.

The list of parameters are any number of parameters specified in the form name=value until newline is reached – the newline terminates the list of parameters, and everything up until the

newline is treated as belonging to the list of parameters. Each parameter is separated from other parameters by whitespace. That means that there must not be any whitespace in the parameter name, parameter value or between parameter name, value, and the equals sign. The form name=value also prohibits equals sign from either name or value. The parameters specified in the parameter list are substituted in the macro body, and the macro body after this transformation is inserted into output. If any of the parameters does not follow this syntax error is reported and the call follows as if that parameter was not specified.

If a parameter in the macro body was not supplied to the macro call it is reported as an error and empty string is used as substitution.

Literal #

In order to insert literal hash symbol one must use ##.

Errors

Errors are printed to the standard error stream and are in the form:

filename:line:column error message

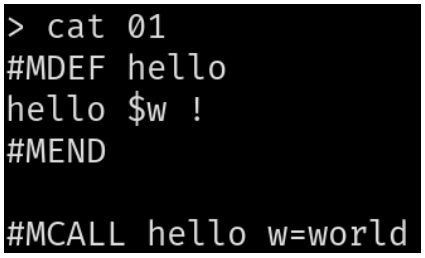
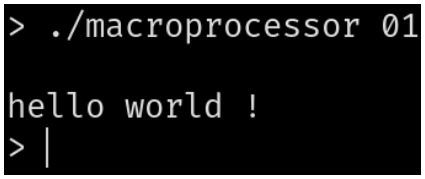
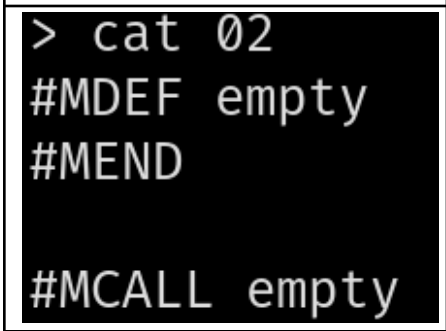
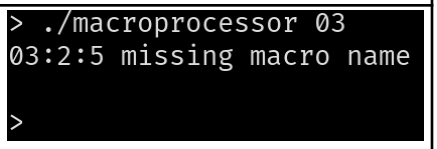
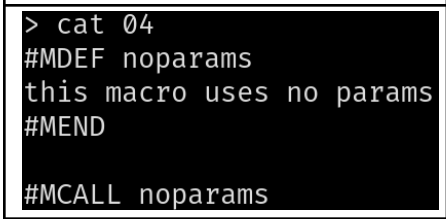
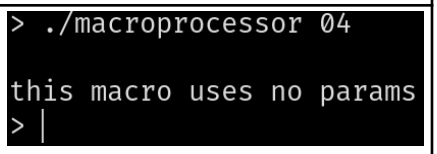
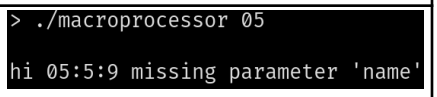
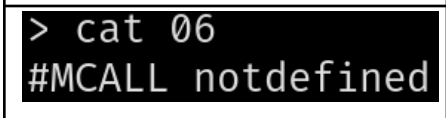
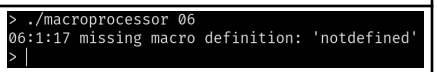
Errors are reported whenever an issue occurs:

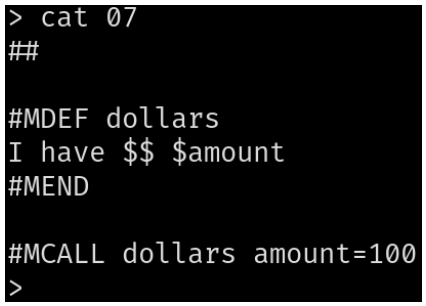
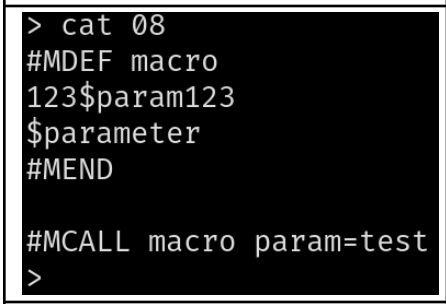
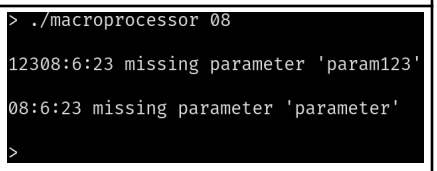
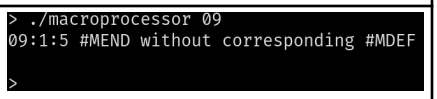
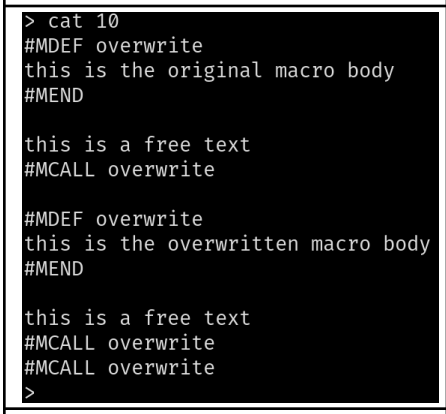
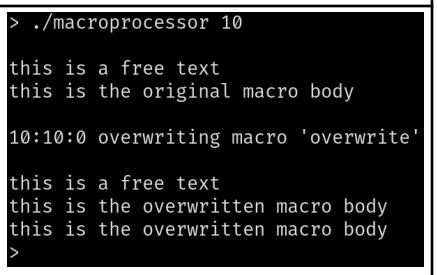
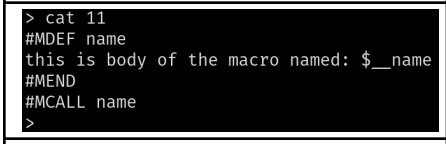
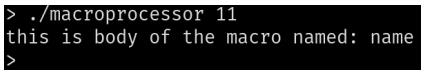
- there is problem with the input – file do not exist, is not possible to open it for reading
- there is a problem with the output – file does not exist, it is not possible to open it for writing
- input file and output file are the same – inplace editing is not supported
- # discriminant is not properly escaped or followed by something different than MDEF, MEND, MCALL or #
- macro name in the macro definition is missing
- #MDEF is not closed by corresponding #MEND
- #MEND without corresponding #MDEF
- arguments to macro call are misformed
- parameter used in the macro body was not supplied to the macro call
- macro name is missing in the macro call

Others

I intend to implement this project in c++.

III. Functional test cases

Case	Description	Output
	Correct usage $w is substituted for world	
	Empty body macro with no body	
	Missing name	
	No parameters macro that uses no parameters	
	Missing parameters parameters used in macro body not provided	
	Not defined macro macro is not in macro library	

	Escaping discriminant ## is used for literal # outside macro body $$ used for literal $ inside macro body	
	No space around parameter	
	#MEND without #MDEF	
	overwriting macro in library	
	using special $__name parameter containing macro name	
	empty file