ECOTE - PRELIMINARY PROJECT

SEMESTER: 2020L

AUTHOR: MICHAŁ KRZYSZTOF SKARZYŃSKI 293054

SUBJECT: 30 MACROG3

GENERAL OVERVIEW AND ASSUMPTIONS

Write a macro generator with nesting of macro definitions. & is a definition denominator.

&CC - beginning of macro definition of name CC

&CC

definition

&

Call: $name_of_macro

The task of this program is to process a string of characters defining a set of macros and it's calls, and as a result of processing print out the free text generated by the macro. If the number of text levels doesn't match or when a definition of a given macro call is not found an appropriate error message will be printed, showing the possible source of the error. In such a case the macro execution will try to continue and reach the end of the file.

FUNCTIONAL REQUIREMENTS

Macro definitions will begin with a '&' character, followed by a name of macro. In next line/lines will be provided body macro definition closed with another '&' character.

character.

In case of a macro call a '$' character will mark its beginning, after space character arguments will be provided separated by spaces, and the end of call will be indicated by a new line character.

Special characters: &, $, '\n'

SYNTAX

MACRO DEFINITION

In task basic definition of macro might look like this:

```
&CC
A
B
C
&
```

As stated in task description macro definitions can be nested:

```
&A
text
&B
text 2
&
text
&
```

In task there is no explicitly said that macro definition can have macro call inside, but I assume that this situation might happen.

Therefore this syntax will also be proper for this macro generator.

```
&A
text
&B
text 2
&
$B
&
```

Of course macro can have multiple calls and definition, with any combination of them, inside its definition.

ADDITIONAL ARGUMENTS

The task says nothing about additional arguments, and after quick analysis of other projects, I see that this particular task focuses on nested definitions rather than passing arguments. As in other tasks that use arguments (namely Task 28. MacroG1, Task 29. MacroG2, Task 31. MacroG4), there is a clear mention of them and about their syntax, while in this task there is no such information, **therefore I assume that the macro can not have additional arguments.**

GENERAL MACRO DEFINITION SYNTAX

&name_of_macro

(free text)

(definitions of other macros)

(calls of other macros)

&

MACRO CALL

As mentioned before macro will not have additional arguments, therefore call of a macro is just special character '$' then name of a macro, and then '\n' character or EOF

GENERAL MACRO CALL SYNTAX

$name_of_macro

MACRO SCOPE

Since every macro can be defined in another macro there exist tree like structure of execution.

In particular each macro is only valid in portion of the source code called its scope.

(This will be very similar to scope of classes / structures in C++ program.)

Each macro can have multiple children macros defined inside it, then outside macro will be called their parent macro.

There exist global scope which is a parent of every macro defined by it self in input file.

In each scope can be defined macros which have the same name but different depth in tree like structure, then effect called variable shadowing will occur, or in this case macro definition shadowing.

Each macro call will search for macro definition using bottom up search, in tree like structure.

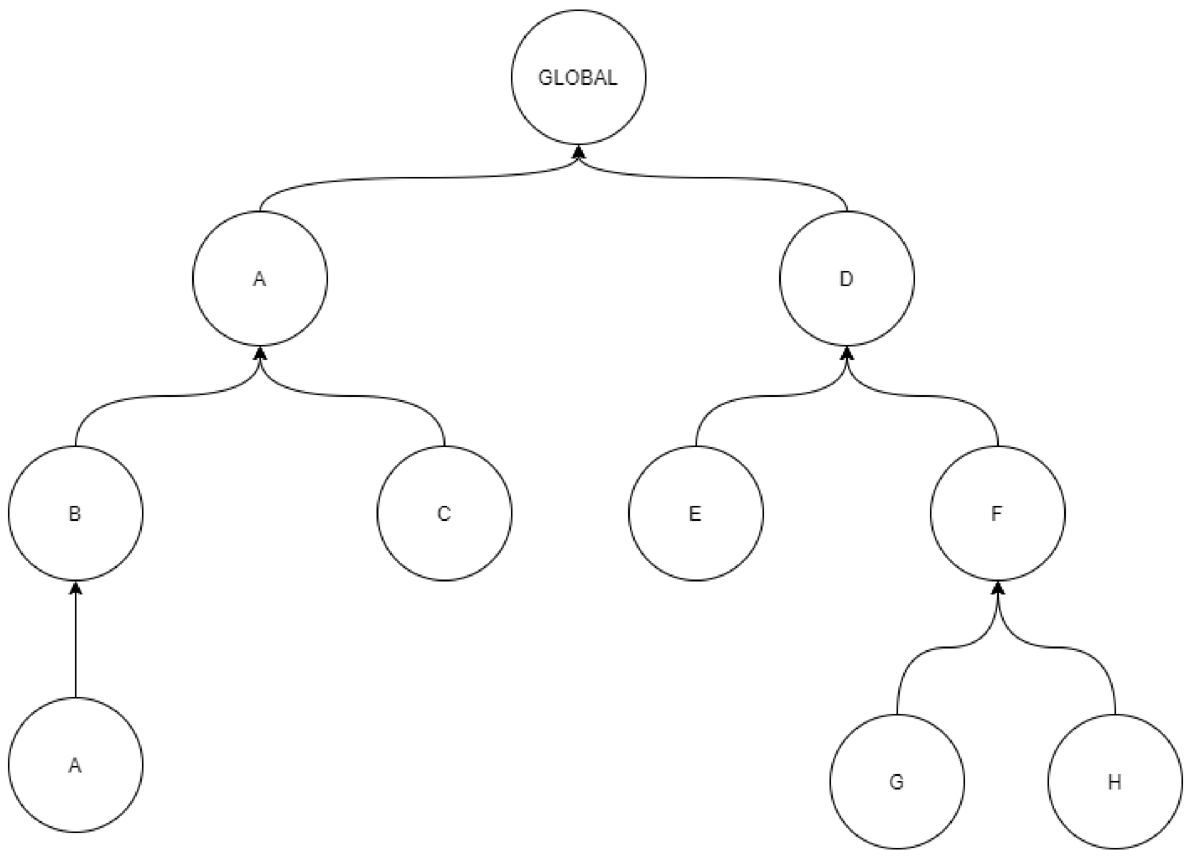
Each portion of execution have indicator of level in tree like structure when this execution is happening, therefore it's possible to maintain information about available at this level macro definitions.

EXAMPLE OF SCOPE

Lets assume this kind of input file. (Tabs are used for clarity)

```
	&A
		&B
			&A
			&
		&
		&C
		&
	&
	&D
		&E
		&
		&F
			&G
			&
			&H
			&
		&
	&
```

Then tree like structure will look like this:



Macros visible at execution level = All direct siblings of execution level + all direct children of execution level + all ancestors from GLOBAL to this execution level.

Execution level	Visible macros
GLOBAL	A, D
A	B, C, D
A’	B, A
B	C, A’, A
C	B, A
D	A, E, F
E	F, D
F	E, G, H, D
G	H, F, D
H	G, F, D

Order of search macro definition:

1. Child
2. Sibling
3. Parent

Therefore if in this structure inside macro B will be macro call $A,

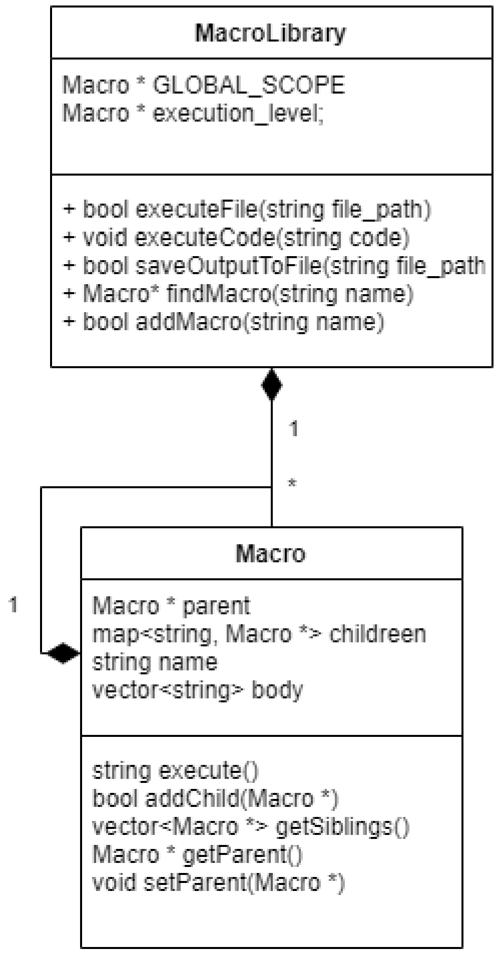
then macro A'(children of B) will be called instead of macro A (parent of B).

IMPLEMENTATION

GENERAL ARCHITECTURE

Since every macro definition can be defined on different level, and therefore have different scope of referenceable objects, therefore I will use tree-like structures to optimize this feature.

DATA STRUCTURES


MacroLibrary
Macro * GLOBAL_SCOPE
Macro * execution_level;
+ bool executeFile(string file_path)
+ void executeCode(string code)
+ bool saveOutputToFile(string file_path
+ Macro* findMacro(string name)
+ bool addMacro(string name)
1
*
1
Macro
Macro * parent
map<string, Macro *> childreen
string name
vector<string> body
string execute()
bool addChild(Macro *)
vector<Macro *> getSiblings()
Macro * getParent()
void setParent(Macro *)

MODULE DESCRIPTIONS

Since this is recursive algorithm its quite hard to write It in pseudocode here.

We can state main assumptions of such algorithm:

Algorithm of reading input, will add new macro in recursive way after encountering new '&' character followed by macro name, and then increase execution_level.

It will decrease execution_level when '&' encountered.

After encountering '$' character followed by macro name it will search current execution_level scope and will try to execute such macro.

At any stage of algorithm error monitoring will be conducted, and possible errors or inaccuracies will be reported.

INPUT GENERALIZATION

Input can have multiple macro definitions, with multiple nested definitions and calls, and multiple macro calls. Input will be read top to bottom line by line, and executed immediately.

EXAMPLE STRUCTURE OF INPUT FILE

```
&name_of_macro

(free text)

(definitions of other macros)

(calls of other macros)

&

$name_of_macro

&name_of_macro2

(free text)

(definitions of other macros)

(calls of other macros)

&

$name_of_macro

$name_of_macro2
```

OUTPUT

The program will print out the free text that is encountered during the macro set execution as well as errors in case when a definition is not found, a loop might occur or there is some syntax error.

In case of a macro execution the following information will be printed:

Macro name: <MACRO_NAME>

Macro body: <MACRO_BODY>

Output: <OUTPUT>

In case of errors an appropriate error message followed by the output will be printed:

Macro name: <MACRO_NAME>

Macro body: <MACRO_BODY>

Error!: Line:<LINE>/Pos:<POS> <ERROR_DESCRIPTION>

Output: <OUTPUT>

ERROR MESSAGES

Possible <ERROR_DESCRIPTION> values:

ERROR_DESCRIPTION message	Possible cause
Incorrect macro call: <NAME>	Use of special characters in macro call
Incorrect macro name: <NAME>	Use of special characters in macro name
Incorrect text level! Missing '&'	Incorrect text level
Missing definition symbol '&'	No definition symbol '&'
Possible infinite loop for macro: <NAME>	Infinite loop condition
Macro: <NAME> not found in current scope	Macro not found

OTHERS

Whole program will be written in C++ language.

Repository of code can be found at: https://github.com/MisterSawyer/ECOTE

FUNCTIONAL TEST CASES

Free text

INPUT	OUTPUT
Free text	Output: Free text

Simple macro

INPUT	OUTPUT
&A **&** **$A**	Macro name: A Macro body: Output:

Simple macro with free text

INPUT	OUTPUT
&A **Free text** **&** **$A**	Macro name: A Macro body: Free text Output: Free text

‘&’ missing

INPUT	OUTPUT
&A **Free text**	Macro name: A Macro body: Free text Error!: Line: 3 Missing definition symbol '&'

Too much ‘&’

INPUT	OUTPUT
&&A **Free text** **&**	Macro name: &A Macro body: Free text Error!: Line: 1 Incorrect macro name: &A

Too much ‘$’

INPUT	OUTPUT
&A	Error!: Line 4 Incorrect macro call: $A
Free text	
&	
$$A	

Simple nested macro

INPUT	OUTPUT
&A	Macro name: A
Free text A	Macro body:
&B	&B
Free text B	Free text A
&	$B
$B	
&	Macro name: B
$A	Macro body:
	Free text B
	Output: Free text

Triple nested macro

INPUT	OUTPUT
&A	Macro name: A
Free text A	Macro body:
&B	&B
Free text B	Free text A
&C	$B
Free text C	
&	Macro name: B
$C	Macro body:
&	&C
$B	Free text B
&	$C

$A	Macro name: C Macro body: Free text C Output: Free text A Free text B Free text C

Macro does not exist

INPUT	OUTPUT
&A **Free text** **&** **$B**	Error!: Line 4 Macro does not exist in current scope

Macro not found in current scope

INPUT	OUTPUT
&A **&B** **Free text** **&** **&** **$B**	Error!: Line 6 Macro does not exist in current scope

Advanced nesting

INPUT	OUTPUT
&A **Free text A** **&B** **Free text B** **&** **&C** **Free text C** **&D**	Macro name: A Macro body: Free text A &B &C $B $C

Free text D	Macro name: B
&	Macro body:
$D	Free text B
&	
$B	Macro name: B
$C	Macro body:
&	Free text C
$A	&D
	$D
	Macro name: D
	Macro body:
	Free text D
	Output:
	Free text A
	Free text B
	Free text C
	Free text D

Use of siblings macro

INPUT	OUTPUT
&A	Macro name: A
&B	Macro body:
Free text B	&B
&	&C
&C	$C
$B	
&	Macro name: C
$C	Macro body:
&	$B
$A	
	Macro name: B
	Macro body:
	Free text B
	Output:
	Free text B

Possible infinite loop

INPUT	OUTPUT
&A	Macro name: A
$A	Macro body:
&	$A
$A	Error!: Line 2 Possible infinite loop for macro: A