Barış Eren Gezici - 210104004041

to make it work
>sbcl
*** (load "convert.lisp")**
*** (main "input.c" "output.lisp")**

1. line-type

- Purpose: Determines the type of a line of code using regular expressions.
- Algorithm:

1. Uses cl-ppcre: scan to match the line against various patterns.
2. Checks for patterns in a specific order to identify the type of line (e.g., return statement, function declaration).
3. Returns a symbol representing the type of the line.

2. convert-arithmetic-operation

- Purpose: Converts an arithmetic operation to a Lisp format.
- Algorithm:

1. Splits the expression into operands and operator using cl-ppcre: split.
2. Extracts the first operand, operator, and second operand.
3. Formats the extracted parts into a Lisp expression.

3. convert-logical-operation

- Purpose: Converts a logical operation to a Lisp format.
- Algorithm:

1. Splits the expression into operands and operator using cl-ppcre: split.
2. Extracts the first operand, operator, and second operand.
3. Formats the extracted parts into a Lisp expression.

4. convert-closebrace

- Purpose: Converts a closing brace } to a closing parenthesis)
- Algorithm: Simply returns a closing parenthesis

5. convert-while

- Purpose: Converts a C-style while loop to a Lisp Loop while construct.
- Algorithm:

1. Extracts the condition part of the while loop using c1-ppcre: split
2. Trims the extracted condition.
3. Converts the condition using convert-logical-operation.
4. Formats the converted condition into a Lisp Loop while construct.

6. convert-for

- Purpose: Converts a C-style for loop to a Lisp loop for construct.
- Algorithm:

1. Splits the loop into initialization, condition, and increment parts using cl-ppcre: split.
2. Extracts and trims the initialization and condition parts.
3. Splits the initialization part to extract the variable and its initial value.

4. Extracts the limit from the condition part.
5. Formats the extracted parts into a Lisp loop for construct.

7. convert-function-call

- Purpose: Converts a function call to a Lisp format.
- Algorithm:

1. Splits the line into function name and arguments using cl-ppcre:split.
2. Trims the arguments.
3. Checks if the function is print and handles it specially.
4. Formats the function call into a Lisp expression.

8. convert-variable-definition

- Purpose: Converts a variable definition to a Lisp format.
- Algorithm:

1. Splits the line into variable part and value part using cl-ppore: split
2. Extracts and trims the variable name.
3. Determines the type of the value (function call, arithmetic, logical) and converts it accordingly.
4. Formats the variable definition into a Lisp expression.

9. convert-assignment

- Purpose: Converts an assignment statement to a Lisp set statement.
- Algorithm:

1. Splits the line into variable and value parts using cl-ppcre:split.
2. Extracts and trims the variable name.
3. Determines the type of the value (function call, arithmetic, logical) and converts it accordingly.
4. Formats the assignment into a Lisp setf expression.

10. convert-types

- Purpose: Converts C types to Lisp types.
- Algorithm:

1. Converts the type to lowercase.
2. Maps the lowercase type to its corresponding Lisp type.

11. parse-types

- Purpose: Parses and extracts types from function parameters.
- Algorithm:

1. Initializes an empty list for types.
2. Iterates over each parameter, trims it, and extracts the type.
3. Adds the extracted type to the list.
4. Returns the list of types.

11. parse-types

- Purpose: Parses and extracts types from function parameters.
- Algorithm:

1. Initializes an empty list for types.
2. Iterates over each parameter, trims it, and extracts the type.

3. Adds the extracted type to the list.
4. Returns the list of types.

12. parse-params-as-list-from-func

- Purpose: Parses function parameters and returns them as a list.
- Algorithm:

1. Extracts the parameters part from the function signature using cl-ppcre:split.
2. Splits the parameters into individual parameters.
3. Trims each parameter and adds it to a list.
4. Returns the list of cleaned parameters.

13. convert-function-declaration

- Purpose: Converts a function declaration to a Lisp declaim statement.
- Algorithm:

1. Splits the line into return type and function part using cl-ppcre:split
2. Converts the return type to its Lisp equivalent.
3. Extracts the function name and parameters.
4. Parses and converts the parameter types.
5. Formats the function declaration into a Lisp declaim statement.

14. convert-function-definition

- Purpose: Converts a function definition to a Lisp defun statement.
- Algorithm:

1. Splits the line into function part using cl-ppcre:split.
2. Extracts the function name and parameters.
3. Parses the parameter names.
4. Formats the function definition into a Lisp defun statement.

15. convert-if

- Purpose: Converts an if statement to a Lisp if construct.
- Algorithm:

1. Extracts the condition part of the if statement using cl-ppcre: split
2. Trims the extracted condition.
3. Converts the condition using convert-logical-operation.
4. Formats the converted condition into a Lisp if construct.

16. convert-return

- Purpose: Converts a return statement to a Lisp format.
- Algorithm:

1. Extracts the return expression using cl-ppere: split.
2. Determines the type of the return value (function call, arithmetic, logical) and converts it accordingly.
3. Formats the return value into a Lisp expression.

17. convert-other

- Purpose: Handles unknown line types.
- Algorithm: Returns a formatted string indicating the unknown type

18. conversion-foo

- Purpose: Maps line types to their corresponding conversion functions.
- Algorithm: Uses a cond statement to return the appropriate conversion function based on the line type.

19. convert

- Purpose: Applies the appropriate conversion function to a line of code.
- Algorithm: Calls the conversion function with the line as an argument.

20. read_file

- Purpose: Reads a file and returns its lines as a list.
- Algorithm:

1. Opens the file for reading.
2. Reads each line and collects them into a list.
3. Closes the file and returns the list of lines.

21. write_file

- Purpose: Writes a list of lines to a file.
- Algorithm:

1. Opens the file for output.
2. Writes each line to the file.
3. Closes the file.

22. clean-line

- Purpose: Trims whitespace from a line.
- Algorithm: Uses string-trim to remove leading and trailing whitespace.

23. recursive_convert

- Purpose: Recursively converts a list of lines using the appropriate conversion functions.
- Algorithm:

1. Checks if the list of lines is empty; if so, returns an empty list.
2. Cleans the first line.
3. If the line is empty, adds an empty string to the result and recurses on the rest of the lines.
4. Determines the type of the line and gets the corresponding conversion function.
5. Converts the line using the conversion function.
6. Adds the converted line to the result and recurses on the rest of the lines.

24. main

- Purpose: Main function that reads an input file, converts its lines, and writes the converted lines to an output file.
- Algorithm:

1. Reads the input file and gets its lines.
2. Converts the lines using recursive_convert.
3. Writes the converted lines to the output file.

4. Returns "success" upon completion.