



# COMPILER PROGRAM FOR CUSTOM PROGRAMMING LANGUAGE

## USER MANUAL

### Purpose of the document

This document provides comprehensive guidance and information to users on how to effectively use the compiler for a specific programming language. The user manual serves as a reference document that helps users understand the compiler's functionality, features, and usage.

This document is the property of computer science students and faculty of the University of the Philippines, Mindanao  
It may not be reproduced or communicated without the author's prior agreement.

Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



## TABLE OF CONTENTS

<b>TABLE OF CONTENTS.....</b>	<b>1</b>
<b>1. ICONS USED IN THE MANUAL.....</b>	<b>2</b>
<b>2. PRESENTATION OF THE SOLUTION.....</b>	<b>3</b>
2.1. Brief Program Description.....	3
2.2. Program Players and Task Distribution.....	3
2.3. Support.....	4
2.4. Referenced Documentation.....	4
2.5. Program History.....	4
<b>3. CONNECTING TO THE PROGRAM.....</b>	<b>6</b>
3.1. Set-Up and Access Considerations.....	6
<b>4. STRUCTURING OF THE PROGRAM.....</b>	<b>6</b>
4.1. Programming Language Specification.....	6
4.2. Program User Interface.....	8
4.4. Other Functions.....	9
<b>5. PROGRAM CONTROL FLOW.....</b>	<b>10</b>




Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



## 1. ICONS USED IN THE MANUAL

---

1.1. Throughout this document, the pictograms below are used to underline points or important notions.

	Important information
	Good to know: tricks
	Mandatory action

Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



## 2. PRESENTATION OF THE SOLUTION

### 2.1. Brief Program Description

This program was developed using the Python programming language and serves as a non-recursive predictive parser for conducting syntax and semantic analysis on a custom programming language referred to as IOL (Integer Oriented Language). The application provides functionality for reading, displaying, and editing the contents of uploaded files with a (.iol) extension within an integrated editor. It facilitates the generation and export of tokenized code to an external file with a (.tkn) extension. The graphical user interface (GUI) includes features such as status updates during compilation and parsing processes, offering a comprehensive environment for IOL programming. The main buttons in the GUI are categorized for file handling (new, open, save, save as, and close) and compilation processes (compile and show tokenization), enhancing user interaction and ease of use.

### 2.2. Program Players and Task Distribution

This program was made by a group of fourth-year BS Computer Science students at the University of the Philippines, Mindanao. The following are the players and the task distribution in building the said compiler for a custom programming language program.

#### Camyl Magdalyn Tanjay

- Program User Interface and Status
- Program Display/Menu
- Process of reading and display of input and output files
- Program error trapping and execution

#### Rod Jhon Cagay

- Program User Interface and Status
- Program Display/Menu
- Process of reading and display of input and output files
- Program error trapping and execution

#### Cedrick Jared Durias


- Program Compiler Process

Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	1.0	Principles of Compiler Design



### 2.3. Support

If you encounter any issues, have questions, or need assistance regarding the program, please refer to the support resources provided below.

Email address: [rmcagay@up.ph](mailto:rmcagay@up.ph)  
 [cetanjay@up.edu.ph](mailto:cetanjay@up.edu.ph)  
[ccdurias@up.edu.ph](mailto:ccdurias@up.edu.ph)

### 2.4. Referenced Documentation



The documentation file of this program (Cagay\_Durias\_Tanjay\_Programing\_Project) was also made, which can help understand the functions of this program.

### 2.5. Program History

The following is a series of programs made by the students under the Principles of Compiler Design subject under Prof. Jon Henly Santillan that help in the successful development of the compiler for custom programming language programs:

#### Program 1: Strings and DFA

A Java program that recognizes if the given input strings are valid or invalid based on deterministic finite automata. It can read and display the contents of the uploaded (.in) file for strings and the (.dfa) file into a transition table, including displaying the output and status during the process and exporting it to an external file (.out).

#### Program 2: Lexical Analysis

A Python program is a simple lexical analyzer that performs lexical analysis on the code and outputs the necessary information related to the result of the analysis. It can read, display, and edit the contents of the uploaded (.iol) file in the editor, including the tokenized code and status during the compilation process, and export it to an external file (.tkn).

Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



### Program 3: Non-Recursive Predictive Parsing

A Python program is a non-recursive predictive parser that checks if a given sequence of tokens is valid based on the given syntactic rules. It can read and display the contents of the uploaded production (.prod) and parse table (.ptbl) file in the GUI, display its parsing status, and export it to an external file (.prsd).

### Program 4: Syntax Analysis

A program that was developed using the Python programming language and serves as a non-recursive predictive parser for conducting syntax and semantic analysis on a custom programming language referred to as IOL (Integer Oriented Language). The application provides functionality for reading, displaying, and editing the contents of uploaded files with a (.iol) extension within an integrated editor. It facilitates the generation and export of tokenized code to an external file with a (.tkn) extension. The graphical user interface (GUI) includes features such as status updates during compilation and parsing processes, offering a comprehensive environment for IOL programming. The main buttons in the GUI are categorized for file handling (new, open, save, save as, and close) and compilation processes (compile and show tokenization), enhancing user interaction and ease of use.

## 3. CONNECTING TO THE PROGRAM

### 3.1. Set-Up and Access Considerations

The user must have a desktop or laptop with Microsoft Windows or Mac OS for use and navigate the features of the program with the following program file or software:



Program python (.py) file or Java Archive (.jar) file



IDE (Integrated Development Environment) software for Python programs. Ex: Visual Studio

Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	1.0	Principles of Compiler Design



You can download an IDE here: [Download Visual Studio Tools - Install Free for Windows, Mac, Linux \(microsoft.com\)](https://visualstudio.microsoft.com/)



The Python software version must be 3.12 and above.



You can download here: [Download Python | Python.org](https://python.org/)

## 4. STRUCTURING OF THE PROGRAM

### 4.1. Programming Language Specification

The following is the programming language specification that will be used by the compiler program.

**Language Name:** Integer-Oriented Language (IOL)

#### Programming Language Overview:

IOL is a simplified custom programming language designed exclusively for integer-type values, numerical operations, and expressions. The language follows specific rules and conventions outlined in the following specifications.

#### 1. Formatting:

- **Whitespace:** Spaces serve as delimiters, separating tokens of the language. Extra spaces (horizontal or vertical) are inconsequential.
- **Case-sensitivity:** The language is case-sensitive.
- **File extension:** Source code files must have the ".iol" extension (e.g., test.iol).

#### 2. Coding:

- A valid IOL code begins with the keyword "**IOL**," followed by program statements and concludes with the keyword "**LOI**."
- Before "**IOL**," only whitespace or the start of the file is allowed.
- After "**LOI**," only whitespace or the end of the file is permitted.
- Between "**IOL**" and "**LOI**" are the program statements, encompassing expressions and operations.

#### 3. Data Types:

Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



- Integer Type (INT): The only numeric type in IOL, represented by integer literals defined as per the given EBNF.
- String Type (STR): No literal exists for this type; the variable value can only be user input.

#### 4. Variable

- - Naming: Variables follow the format "var\_name," adhering to specific rules for letters and digits.
- Definition: Variables must be defined before use, either as "data\_type var\_name" or "data\_type var\_name IS value."
- Default Values: Variables without an initial value assume default values (0 for numeric type, empty string for string type).

#### 5. Assignment, Input, and Output:

- Assignment: Uses the syntax "INTO var\_name IS expr," ensuring the expression is valid for the target variable's data type.
- Input: Utilizes the syntax "BEG var\_name," storing valid user-input values to the target variable.
- Output: Employs the syntax "PRINT expr," where expr can be a literal, variable, or numerical expression.

#### 6. Numerical Expressions:

- Prefix Notation: IOL uses prefix notation for numerical expressions.
- Evaluation: Follows C's precedence and associativity rules.
- Simple Expression: Consists of a numeric literal or variable.
- Complex Expression: Arises from combining two expressions using numerical operations.

Operation	Complex Expression	Equivalent in C
Addition	ADD expr1 expr2	expr1 + expr2
Subtraction	SUB expr1 expr2	expr1 - expr2
Multiplication	MULT expr1 expr2	expr1 * expr2
Division	DIV expr1 expr2	expr1 / expr2
Modulus	MOD expr1 expr2	expr1 % expr2

#### 7. Built-in Commands:

- **NEWLN**: Appends a new line command (\n in C/C++/Java) at the end of the current output console line.

Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design

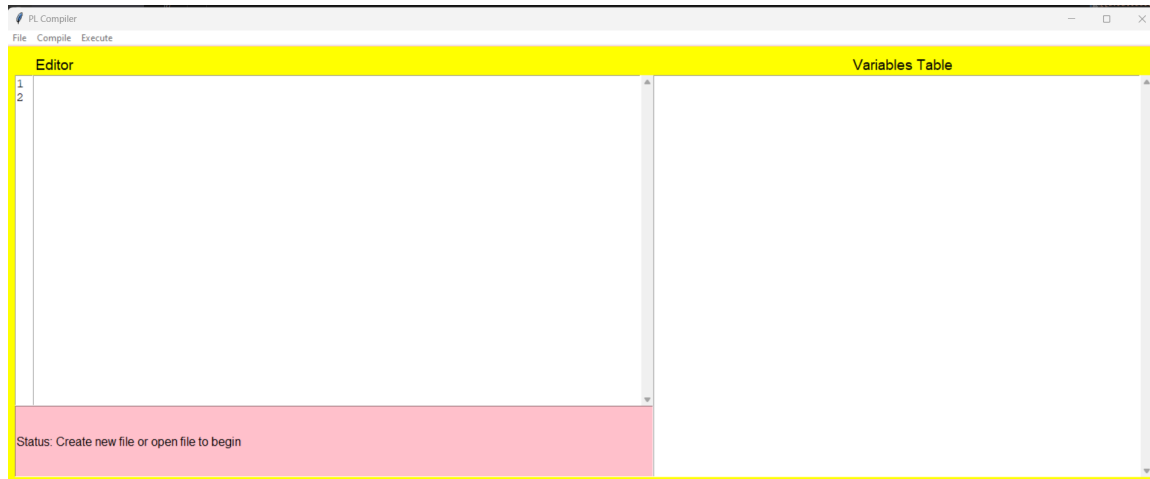




## 4.2. Program User Interface

As the program executes from the IDE, the program will lead to the program user shown in Figure 1.

- You may see the menu bar (white), code editor (yellow), variable table placeholder (yellow), and status bar/console (pink).



**Figure 1.** Program User Interface

## 4.3. Program Menus and Functionalities

The following are the buttons under the menu bar, which are the main buttons for the file-handling and compilation processes:

**4.3.1. File Button:** a drop-down button that is used in the file-handling process and consists of the following:

- New:** button that creates a new file with default content and updates the editor.
- Open:** a button that opens a PL (programming language) file, reads its content, and displays it in the editor.
- Save:** a button that saves the content of the editor to the current file or prompts for a filename if not saved before.
- Save as:** a button that prompts the user for a filename and saves the content of the editor to the specified file.
- Close:** a button that closes the current file, discarding any unsaved changes.

**4.3.2. Compile:** a drop-down button that is used in the compilation process and consists of the following::

- Compilecode:** this button compiles the code by performing lexical analysis, displaying errors, tokenizing the code, and saving the compiled code.

Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



- **Show tokenized code:** this button shows the tokenized code in a new window if the code has been successfully compiled.

**4.3.3. Execute:** button for executing the code

**4.3.4. Code Editor :** a placeholder that displays and edits the contents of uploaded files.

**4.3.5. Variable Table:** a placeholder that displays the variables and their data types

**4.3.6. Status Bar/ Console:** a placeholder that updates the status bar with the provided message

#### **4.4. Other Functions**

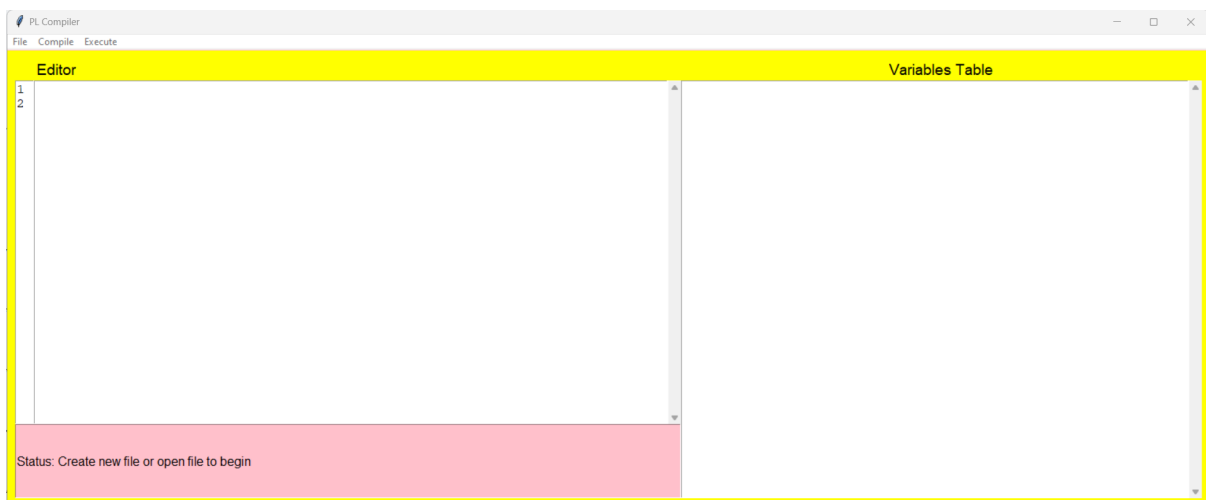
**4.4.1. Line Numbers:** Updates the line numbers in the line number section based on the current editor content.

Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design

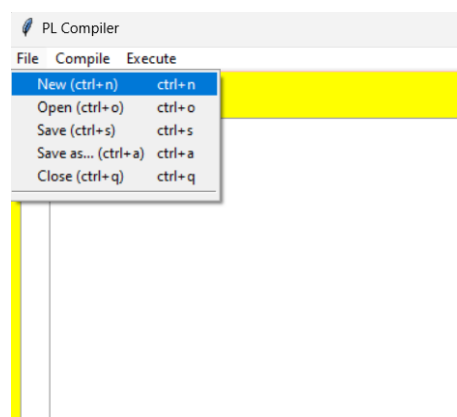


## 5. PROGRAM CONTROL FLOW

- 5.1.** Once the program runs, the Tkinter GUI window is created and its components set up.



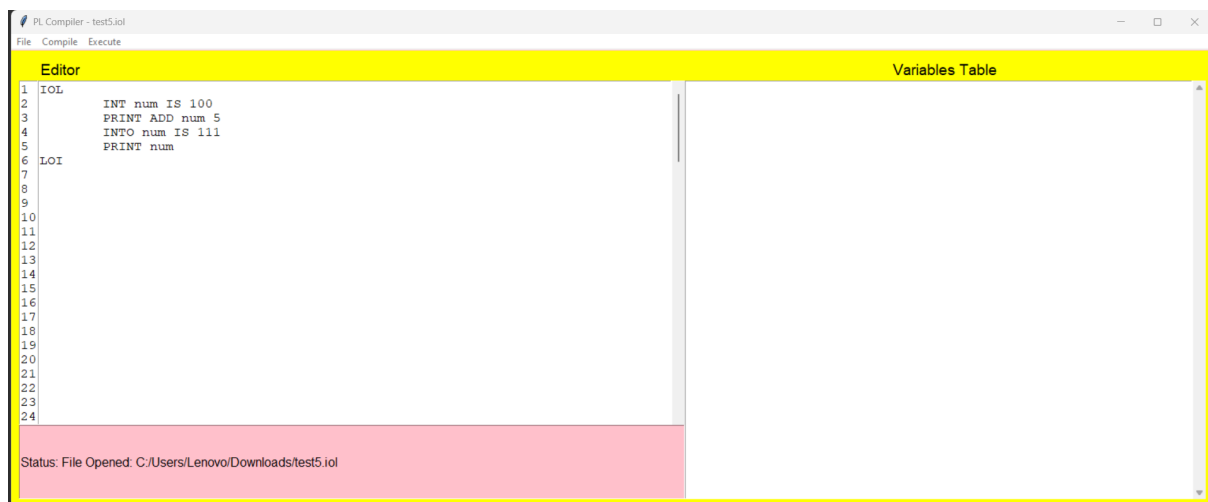
- 5.2.** The user can perform various file operations, such as "New," "Open," "Save," or "Save as..." under the File Menu.



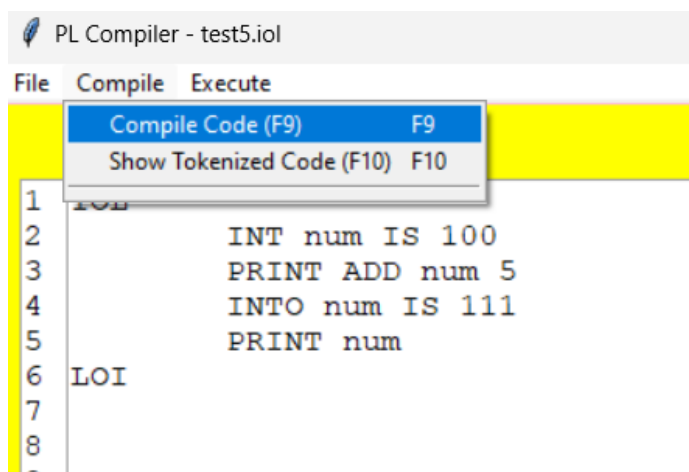
Compiler for Custom Programming Language User Manual				
Last modification	07 January 2023 at 20:00:00			
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



- 5.3. If the input (.iol) files are loaded through an Open Button, the contents of the file will be loaded in the editor frame in the GUI.
- 5.4. If "Save" or "Save as..." is chosen, a file dialog opens, and the user specifies the file name and location.
- 5.5. The user interacts with the code editor and writes or modifies code in the editor.
- 5.6. Scroll through the code using the scrollbar in the editor.



- 5.7. The user can compile the code through "Compile Code" under compile in the menu bar. Lexical, semantic, and non-recursive predictive parsing is performed.



Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



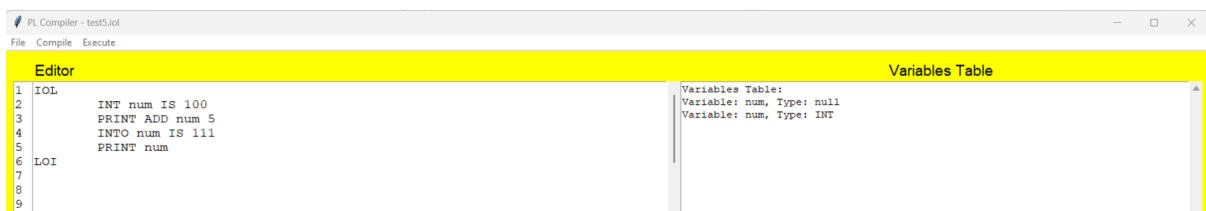
**5.8.** For lexical analysis;

5.8.1. If no lexical errors are found, the status is updated with "Code tokenized successfully."

5.8.2. If lexical errors are found, the status is updated with an error message.

**5.9.** For Non-Recursive Predictive Parsing;

5.9.1. If the given string is valid or invalid, the parsing status will be updated. For variable table display, the user can view the table of variables and their types:



**5.10.** The user can view the tokenized version of the code through "Show Tokenized Code" under "Compile" in the menu bar

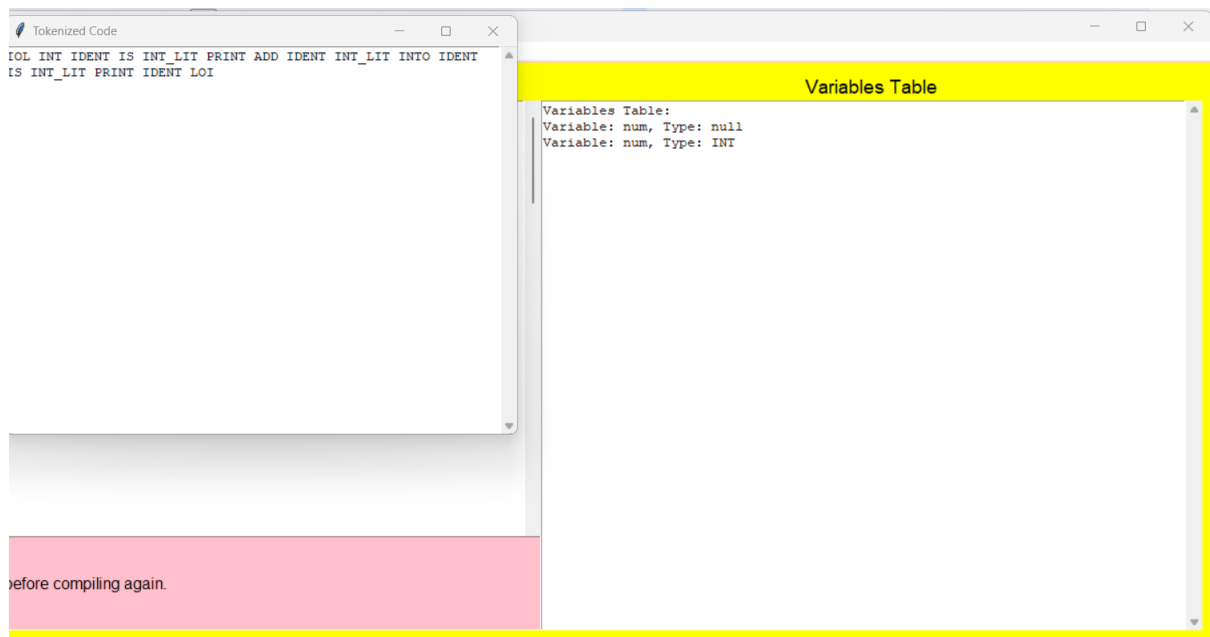
**5.11.** A new window displays the tokenized code.



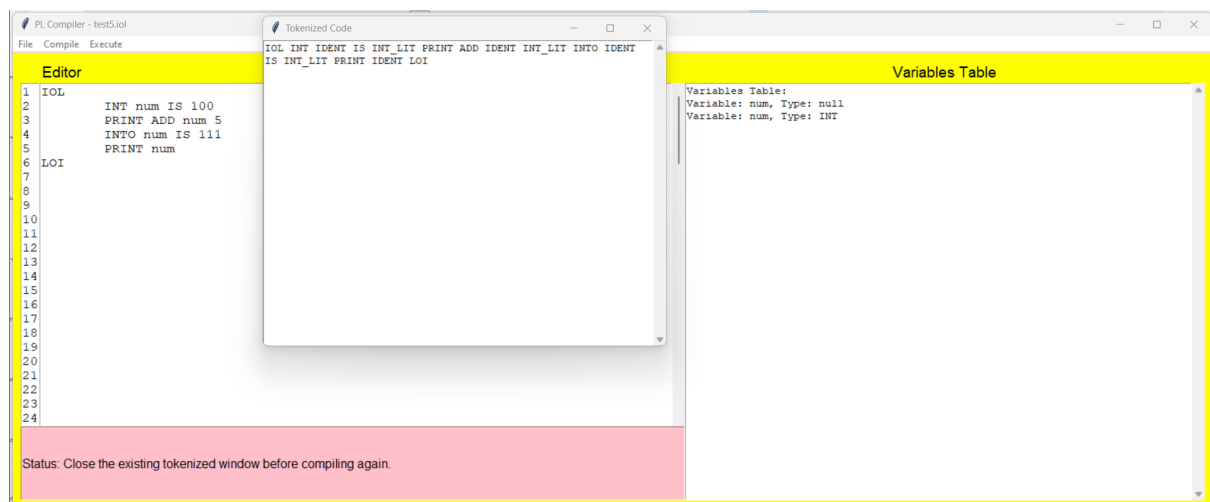
**5.12.** For variable table display, the user can view the table of variables and their types:

**5.13.** After successful compilation, the variables table is updated with variable information.

Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	<b>1.0</b>	Principles of Compiler Design



- 5.14. The user can execute the code through the "Execute" button in the menu bar.
- 5.15. Throughout the process, the user will give status updates and
- 5.16. Messages are updated during file operations, compilation, parsing, and execution.
- 5.17. The program terminates "Close" in the "File" menu.



Compiler for Custom Programming Language User Manual				
Last modification		07 January 2023 at 20:00:00		
Author : Cagay, R., Durias, C., Tanjay, C.	4BSCS	Version :	1.0	Principles of Compiler Design