TALHA MUDASSAR L1F22BSCS0379

Compiler Construction Project Phase 1

Language Overview (Detailed Description)

Language Name: PUNJ++ (*Punjabi Programming Language Plus Plus*)
Theme: A natural-language-inspired programming language designed around Punjabi linguistic expressions and thought flow.

Purpose:
The main goal of **PUNJ++** is to simplify programming for Punjabi-speaking students by making it more intuitive and close to everyday communication. Where C++ uses foreign words like if, else, and return, PUNJ++ replaces them with Punjabi words like fher, nahiTa, and morjaa.

It bridges **human thought and machine logic** through natural language enabling first-time programmers to focus on **logic**, not **memorization of syntax**.
The structure and grammar are similar to C++, ensuring backward compatibility, but the keywords and semantics reflect Punjabi style and rhythm.

Features:

- Uses Roman Punjabi keywords for natural readability.
- Case-sensitive, like C++.
- Supports procedural programming.
- Retains standard operators and data types from C++.
- Uses likh and dass for input/output.
- Has clear structure blocks { } for logic and loops.

1. Regular Expressions Table

This table lists the main regex patterns used in the PUNJ++ lexical analyzer. Each pattern helps Flex recognize specific tokens.

TALHA MUDASSAR L1F22BSCS0379

TOKEN TYPE	**REGEX**	**Example**
DIGIT	[0-9]	0,1,4
LETTER	[A-Z a-z]	A,a,Z,b
ID_START	[A-Z a-z _]	x, _, myVar
ID_COUNT	[A-Z a-z 0-9 _]	_a1, name2
IDENTIFIER	{ID_START}{ID_CONT}*	Myvar,_count
INTEGER	{DIGIT}+	12,909
FLOAT	{INT}\.{DIGIT}+	3.14,2.900
EEPONENT	{INT}(\.{DIGIT)+)?[eE][+]?{INT}	1e10, 2.3E-4
STRING	`` `"([^\\n"]\.)*"` ``	“Hello”,”abc123”
CHARACTER	`` `'([^\\n'] \.)'` ``	‘a’,’\n’
COMMENT	[//(~\n)/(/*)(~*/*+~/)*(*/)]	//, /* */
INVALID ID	[@#\$][A-Za-z0-9_]*	@abc , #name
INVALID ID ERROR TOKEN	.	Anything else

2. Transition diagrams for Identifiers and Numbers

TALHA MUDASSAR L1F22BSCS0379

Numbers:

Regular expression

Integer [0-9]+

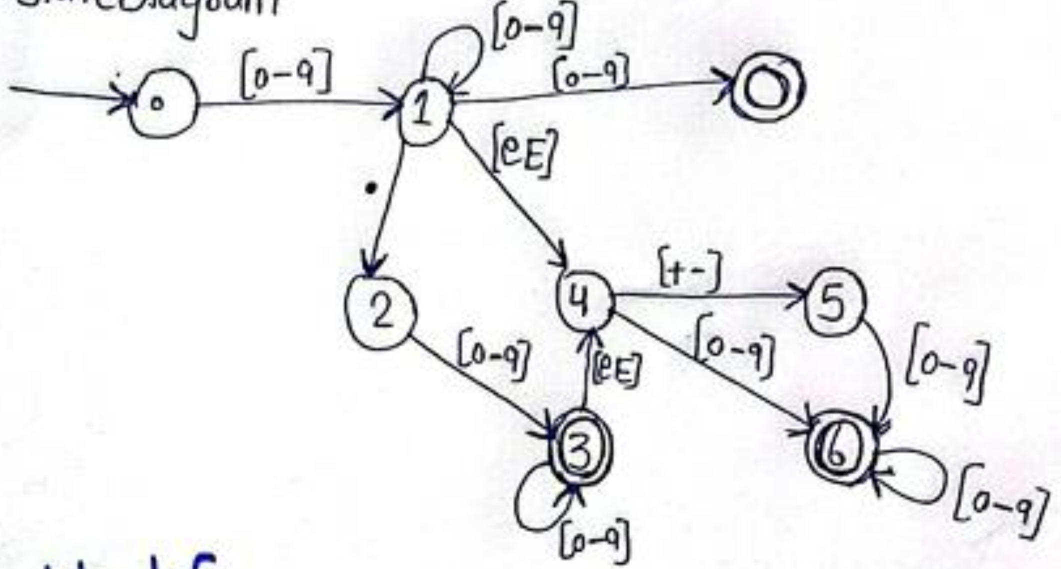
Float [0-9]+[.]+[0-9]+

Exponent [0-9]+ (. [0-9]+)? [eE] [+-] ? [0-9]+

AcceptingStrings = [0, 45, 908], 3.14, 0.5, 1e10, 3.5E, 2.3e-5]

Rejected = [.5, 5., 3.4.5, e10, 12e, +45, 1e+]

StateDiagram

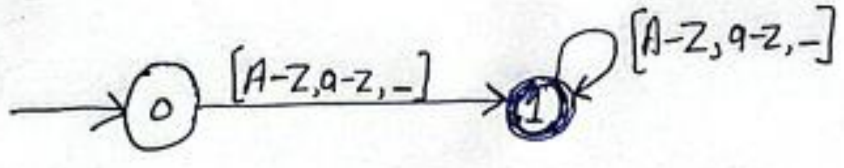


Identifiers

Regular Expression : [A-Z a-z _][A-Z a-z 0-9 _]*

AcceptingStrings = [x, _a, count, myVar, _count2, Name_, loop123]

Rejected Strings = [1abc, @name, #temp, $id, -count]



3. Explanation of chosen 15 keywords + operators + punctuations

Keywords

These keywords are Punjabi-inspired replacements for C++ words.

KEYWORDS	MEANING	EQUIVALENT C++
Fher	Conditional check	If
Nahi Ta	Other wise	Else
Jad Tak	Loop Until fail	While
Likh	Take input	Cin
Dass	Print output	Cout
Morjaa	Return Value	Return
Kaam	Loop	For
Chakkar	Repeat Loop	Do/while
Rok	Break	Break
Jaari	Continuous	Continue
Nava	New	New
Class	Class decleration	Class
Dekh	Switch-Like structure	Switch
Halat	Case Condition	Case
Mukao	Default case	Default

Reasoning: These words are chosen because they are easy for Punjabi speakers to understand and make programming feel natural and intuitive.

Operators:

Operators	Meaning
<+>	Add and assign
<->	Equal comparsion
<!>	Not equal
++>	Increment

Punctuations:

Symbol	Description
:::	Start of custom block
:::;	End of custom block
~>	Custom end marker
<>	Custom separator

Explanation:

Keywords

- fher: This keyword is used to start a decision-making block. It works the same as if in C++ and is used when we want to check a condition, for example fher (x > 0).
- nahiTa: Used when the condition in fher is not true. It behaves like the else part in an if-else statement and means “otherwise”.
- jadTak: Works like the while loop. It keeps running the block of code as long as the given condition is true.
- likh: This keyword is used to take input from the user. It is similar to cin in C++.
- dass: Used to show or print output on the screen, just like cout.
- morjaa: Works like the return keyword in C++. It is used to send a value back from a function.
- kaam: Represents a for loop and is used when a task needs to be repeated a fixed number of times.
- chakkar: This keyword is used for a do-while type loop, meaning the loop runs at least once before checking the condition.
- rok: Used to stop a loop immediately, the same as break.
- jaari: Used to skip the current loop cycle and move to the next one, just like continue.

- nava: Means “new” and is used when creating new objects or variables, similar to the new keyword.
- class: Defines a class in the program. This keyword is kept the same as in C++ for better understanding.
- dekh: Works like a switch statement to check multiple possible values or conditions.
- halat: Used inside a dekh block and represents a case.
- mukao: Acts like the default part in a switch statement and runs when no other case matches.

These Punjabi-based keywords make programming easier to understand for native speakers while keeping the same logical flow as C++.

Operators:

The operators in PUNJ++ are made to look simple yet different from regular C++ ones. The <+> operator adds a value and assigns it to a variable, just like +=. The <-> operator checks if two values are equal, working like ==, and the arrows on both sides clearly show comparison. The <!> operator means “not equal” and does the same job as !=. The ++> operator increases the value of a variable by one, just like the increment ++ in C++. These symbols were chosen to make code easier to read and to give the language a unique style while still behaving the same way as standard C++ operators.

Punctuations:

PUNJ++ uses a few new punctuation marks to make the structure of the code more clear and special. The ::: symbol is used to start a custom block, and :::; is used to end that block. This makes it easy to see where a class or function begins and ends. The ~> symbol acts as a custom end marker for a statement, giving the code a clean look instead of just using ;. The <> symbol works as a separator between different elements or parameters. These punctuations not only make the syntax different from regular C++ but also help in keeping the code neat and readable.