***ANP (Programming Language Specification)***

**Group Members:**

Ameed Ghouri (Roll Number)

Neha Asif (Roll Number)

Poorab Gangwani (CS191092)

***Abstract:***

**ANP** is a general purpose language with numerical and string variables, arrays, loops and user-defined functions.

The motivation behind the development of the language was to actualize a programming language which could take standard programming constructs such as conditional statements, loops or functions and apply a user-friendly flavor and syntactical approach to it.

Some of its syntax was inspired from the more comprehensive syntax grammar in other programming languages.

**Lexical Aspects:**

1. **Identifier:**

It is a sequence of alphabets, digits and underscores that starts with an alphabet or an underscore.

1. **Constant:**

It is a sequence of one or more digits which will constitute one of the following:

**2.1.** **Integer**

Starts and ends with a digit and does not contain ‘.’ In between.

2.2 **Decimal**

Starts with a digit and ends with digit with a ‘.’ In between. **E.g. (12.134)**

**OR**

Starts with ‘.’ and is followed and completed by a sequence of digits. **E.g. (.15)**

1. **S\_Literal:**

It is a sequence of characters (excluding a quotation mark) and escape sequences surrounded by quotation marks (i.e. “”).

Each escape sequences are the following:

1. **\n (newline)**
2. **\r (Carriage Return)**
3. **\t (Horizontal Tab)**
4. **\b (Backspace)**
5. **\f (form feed)**
6. **\v (Vertical Tab)**
7. **\0 (Null)**
8. **\’ (single quote)**
9. **\” (double quote)**
10. **\\ (backslash)**
11. **C\_Literal:**

It is a printable character or escape sequence surrounded by single quotation marks.

1. **Operators:**

Operators are categorized as following:

1. **Boolean Operators ( and or )**
2. **Assignment Operator ( = )**
3. **Increment Operators ( ++ += )**
4. **Decrement Operators ( -- -= )**
5. **Comparison Operators (>= | <= | < | > | != |==)**
6. **Arithmetic Operators (+ / \* - % ~ @ ^ ? \*= /= %= ^= & &= |= |)**

Keywords are **char break const continue double default else float int str if long case func for until write read readLine return**

Separators are **; { } ( ) [ ] , :**

**Token and Expression Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Token | Description | **Example** | **Expression** |
| Keyword | Reserved words that perform a certain program functionality | **char break const continue double default else float int str if long case func for until write read readLine return** | The words in **Example** in linear sequential form.  e.g. break  (b)(r)(e)(a)(k) |
| Pre-processor Directive | Functionality for including libraries or files | **Use “iostream”** | (u)(s)(e) (“)(filename)(”) |
| Separator | Punctuation symbols | **; { } ( ) [ ] , :** | **; { } ( ) [ ] , :** |
| Identifier | Arbitrarily named variable used to contain numeric or text data. | **num1, arr** | (alphabets|\_)(alphanums|\_)\* |
| Constant | Numeric data | **0.12 , 12** | (digits)\*(.)(digit)+ | (digit)+ |
| S\_Literal | Content within quotation marks | **“This is a String \n”** | (“)(any sequence of characters or escape sequence excluding “)(“) |
| Operator | Used for arithmetic, Boolean, comparison and assignment operations | **+ / \* &**  **>= = ++**  **--** | e.g. (+)(+) or (&)  e.g. (>)(=)  e.g. (=)  e.g. (-)(-) |
| C\_Literal | A single character or escape sequence within single quotation marks. | **‘c’ , ‘\r’ , ‘;’** | (‘)(any character or escape sequence excluding ‘\’)(‘) |