**COMPILER CONSTRUCTION LAB**



**PROJECT PROPORSAL**

**CAPITA LANGUAGE**

**GROUP MEMBERS:**

BASIL ASLAM (02-134191-002)

ALI FAWAD (02-134191-094)

**INRODUCTION**

Capita is a user friendly language in which the programmer will be able to solve complex problems without any difficulty. In capita language there will be some keywords, identifiers, punctuators and operators which will help the user to solve as much complex algorithms in an easy way. This language of computer programming will transform the coding world with its promise of more sophisticated virtual data types and objects. In this language, programmers will typically saw a codebase as composed of individual command line instructions. The identification of objects with data and functions built in led to a new way of packaging and automating code work.

Capita will prove as a hybrid that contains the functionality of the C++ programming language. This means that you have all the features that are available in C++. The user can experience universally usable modular programs, efficient and close to the machine programming and portable programs for various platforms. The large quantities of existing Capita source code can also be used in C++ programs. This language will also support the concepts of object-oriented programming. Various language elements are added to Capita, such as references, templates, and exception handling. Even though these elements of the language are important for efficient program implementation.

Further the language specifications are given below:

**Language Specification**

1. **Targeted audience**

The targeted audience of the Language are all the learners and coders who want to study parser and for the individuals who want to see the backend working of a compiler. Through this individual can easily see how capita treats the input given by the user and how the tokens are identified by the lexical analyzer and then given to parser. This document will help individuals to know the Language that is valid for the Parser.

1. **Paradigm of language**

Paradigm can also be termed as a method to solve some problems or do some tasks. A programming paradigm is an approach to solve the problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach. There are lots of programming languages that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms. Apart from varieties of programming languages, there are lots of paradigms to fulfill each and every demand.

# Capita will support four main programming paradigms: imperative, functional, procedural, and object-oriented. Whether you agree that they are valid or even useful, it will strive to make all four available and working.

# Components of Language

# The components of the Language Capita will include numbers from 0 to 9, alphabets from A to Z either capital or in lowercase. There will be some function definitions for print, some mathematical functions and some other conditional and loop functions that will help in the most efficient creation.

# Syntax of language components

Capita language will be completely object oriented, and not "statically typed". You do not need to declare variables before using them, or declare their type. Every variable in Capita will be an object.

The integer declaration syntax will be as simple as assigning a number with any of the variable or name of the identifier. Similarly, the declaration of string will be also similar to integer declaration we have to assign the value of string only by providing the variable name and the value of string with it. The data types for declaration will be

There will be statements for print to display the output on the screen, the keywords for display and user input will be.

There will be conditioning statements in which we will be able to compare a single or more than one thing. In many eras we have more than one condition to compare as per the situations available at that specific time.

For evaluating more than one statements, conditions and functions there will be loops in which we can easily evaluate more than one conditions, statements and function. By the implementation of loops, it will be easier for the user to evaluate all conditions and statements without any repetition of code in an efficient manner by using the loops. There will be also the possibility of using nested loops so that there would be no limit of evaluating the statements and conditions.

The keywords for loops, condition stat and data types will be:

|  |  |
| --- | --- |
| Number (numeric values) | Input |
| Alpha (character values) | output |
| Sentence (String Values) | Choice and options |
| Decimal (Floating point values) | Do and whereas |
| If and else |  |
| Loop |  |
| interrupt |  |
| resume |  |

# Lexical Components (Patterns)

# In the language the lexical components include keywords, identifiers, constants, punctuators and operators.

In keywords there will be some special words that will be reserved for some special functions. There will be words for printing anything on the display screen, there would be conditional keywords and loops as discussed above. And there will be many other reserved words that will provide different functionality as per required.

In identifiers there will be the variable associated with different data types which will be selected according to the condition. The data types include for integer numbers, for letters and character and for floating point numbers. These data types will be used in the code and the user will select it according to given situation.

There will be some constants in the language, which will also be associated with some data types and these constants will remain the same throughout the program. Some constants are declared in global which will remain same throughout the program and some constants are declared in the main function which remains same only within the main function. We can have number, letter, sentence and float constants as well.

In the language there will be some punctuators that will be used as for string and character constant whenever we are assigning a string and character constants to the variables the punctuators will be used to identify the constants. For separating two sentences or variables and statements as well there will be some word breakers and at some places brackets will also be used for some special purposes.

There will also be some operators like in mathematics we have operators of addition, multiplication, subtraction and division. As in mathematics these operators are used according to their precedence similarly in this language operators will also be used according to their precedence.

# Semantic Components

**Assignment Operations:**

Assigning values to the variables will be as simple as C++ language. We can assign any value to a variable by “=” sign and a value with it and in some eras we have some other operators associated with it, assignment operators are =, +=, \*=, /=.

**Arithmetic Operations:**

In this language there would be some arithmetic operators which will be exactly same as operations we perform in mathematics. These operators are addition (+), subtraction (-), multiplication (\*) and division (/).

**Logical Operations:**

In this language there would be two logical operators to compare the conditions or statements. These two operators will be AND (&&) and OR (||) operators. The OR operator will compare two or more conditions and if anyone it gets satisfied it will return true otherwise false. On the other side the AND operator will also compare two or more conditions and will return true if all the conditions are satisfied otherwise it will return false.

**Relational Operations:**

Another type of operators which will be included in the language will be relational operators. These relational operators will be very helpful in decision making operations. The relational operators include <, <=, >, >= and = =. The user will perform these as per his requirement for making decisions in various situations.