# Project

## Theory Of Computation

### Make own Language

It provides a deep understanding of the fundamental principles that underlie computer science, such as algorithms, data structures, and computational complexity.


**Group Members:**

- Zainab (f2020266124)
- Qainat Naeem (f2021266---)
- Maheen Aslam (f2021266378)

**Submitted By :**

Sir Rana Marwat

Theory of computational

# Introduction:

In my project, I'm working on making my very own programming language. I'm setting up specific rules that help generate things like 'if-else' conditions and class structures, all using C++ as a base. The cool part is, I'm not just deciding how it looks, but also how it works – defining how these special conditions and classes behave in my language. The main goal is to create a unique way of building classes that goes beyond the usual methods in programming languages.

## 1. Conditional Statement

My own conditional statement works the same as an if – else statement.

Formal definition:

- **Start symbol**: S
- **Nonterminal symbol**: {A, COND, ELSE, THEN, TRUECOND}
- **Terminal Symbol**: {unhealthy child, good child eats healthy food, eat apple,! }
- **Production Rule**: {S ->! A!,  A -> IF X2, X2 -> COND ELSE, COND -> TRUECOND THEN, ELSE -> unhealthy child, THEN -> good child eats healthy food, TRUECOND -> eat apple}

### Derivation parser tree:

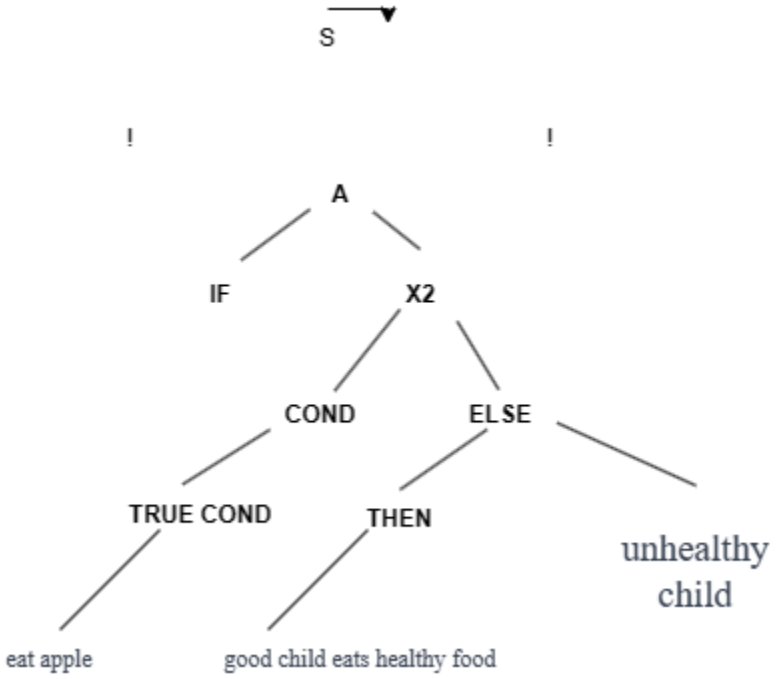
! If (eat apple)

then

Good child

Else

Unhealthy child !




Online tool  parser link : [if else parser tree]

## 2. Define class structure

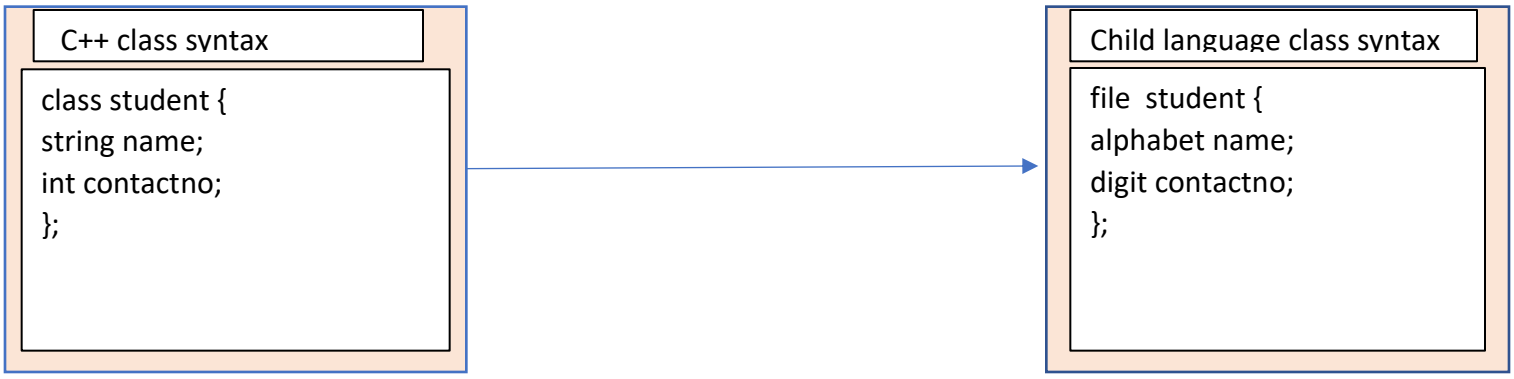
Formal Definition:

- Start symbol: **S**
- Terminal symbol: {file, student, teacher, home, employee, newline, alphabet, digit, ; , name, contact no, {,}}
- Non – Terminal symbol / Variable: **{A, B, C Class Keyword}**
- Production Rule: {**S** —> Class Keyword A, Class Keyword -> file**A** -> student A| teacher A |home A |employee A| {A} |newline B| newline, **B** -> alphabet C; |; newline B | newline, C-> name B |contact no B}

Used for class keyword replace My own keyword “file”.

Use datatypes string replace with “alphabet” datatype and int data type replace with “digit data type”.

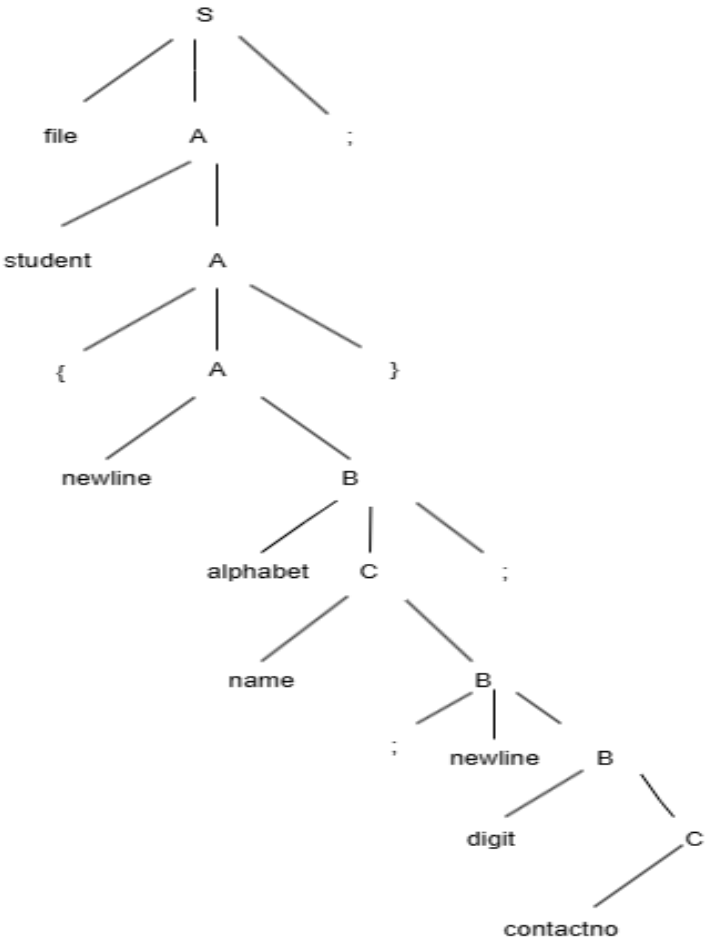
My generic syntax of class :




## Derivation Tree (Parse tree)

Parse trees play a crucial role in comprehending the organization of a parsed program or sentence. They assist in debugging parsers and offer a visual depiction of how the input adheres to grammar rules. This significance is particularly pronounced in the realm of compiler construction, where parsing stands as a pivotal stage in transforming source code into a format suitable for subsequent processing or execution.

Easily understand what type of data store in variable. Alphabet in characters and Digit contains the number. Everyone knows this data type and what type of data is stored.




Online tool : [online parser tree]

## Motivation Point :

**User-Friendly Syntax:**

- Your effort to create a syntax that is not only unique but also user-friendly contributes to the overall appeal of your language.

**Class Structure Derivation Tree:**

  - A clear derivation tree illustrates how your class structures are formed, emphasizing the syntax and organization of class-related elements.

**Datatype Replacement:**

- Your project replaces conventional datatypes with more descriptive terms, such as replacing 'string' with 'alphabet' and 'int' with 'digit data type,' enhancing readability and user understanding.