Context-Free Grammars

→ Chomsky (Linguist) described four classes of generative devices or grammars that defines four classes of languages.

→ Two of these grammars classes

  Context-Free Regular

→ These grammars turned out to be useful for describing the syntax of Programming Language.

  • Regular Grammar Describes Tokens of P.L.
  • Context-Free Grammar Describe Syntax

Basically, Context-Free Grammar is a Language Generator which describes Syntax of Natural Language.
In 1960 John Backus and Peter Naur introduced a formal notation method for describing syntax of programming languages which is known as Backus-Naur Form, or simply BNF.

BNF was basically designed for ALGOL 60. BNF and context-free grammars were nearly identical.

BNF is a metalanguage for programming languages. Metalanguage is a language that is used to describe another language.
BNF: Symbols to describe a Syntax

:::= means 'is defined as'
<> means 'can be described as'
| means 'or'

BNF uses "abstractions" for syntactic structures. For example, Java assignment statement:

<assign> → <var> = <expression>

L:H:S
↓
(Abstraction)
↓
It is being Defined
Non Terminal

R:H:S
( Tokens, lexemes, References)
Terminal.

Computer Science Lectures By ER. Deepak Garg
**Terminals:** Symbols that can appear in the output of a language because of its rules, but cannot be changed by rules themselves.

**Non-Terminals:** Syntactic entities that defines a part of the grammar.

Non-Terminals can be replaced and they are string, composition of Terminals and Non-Terminals.

**Example of Terminals**

```
+ = $\{ \star \} + \ l$
if while do , int
do double float A, B
```

**Example of Non-Terminals:**

- Sentences
- Words
- Terms
- Expressions
- Statements
- List of Numbers
- Programs
- Programs

---

**Computer Science Lectures By ER. Deepak Garg**
How To write Rules in BNF

L.H.S

\[ \langle NT \rangle \]

---

Start symbol (Non-Terminals)

\[ \langle NT \rangle \]

R.H.S

\[ \Rightarrow \{ T \} \quad \{ \langle NT \rangle \} \]

---

or

\[ \Rightarrow \{ T \} \quad \{ \langle NT \rangle \} \]

---

Rules

---

One or more Terminals

\[ :: = \{ T \} \quad \{ \langle NT \rangle \} \]

---

One or more Nonterminal

---

Example: if loop

```
Structure

if (condition)
{

body

}
```

---

Subscribe to our

YouTube Channel

---

Computer Science Lectures By ER. Deepak Garg
BNF of Structure ①

<if-loop> ::= <if-pact>

<if-pact> ::= if (<condition>) { <body> }

or

<if-loop> ::= if (<condition>) { <body> }

Structure ②

if (condition)
  { body
  } else
  { body
  }

BNF of Structure ②

① <if-loop> ::= <if-pact> <else-pact>

<if-pact> ::= if (<condition>) { <body> }

<else-pact> ::= else { <body> }

or

Computer Science Lectures By ER. Deepak Garg
2. \( \langle \text{if-loop} \rangle :: \langle \text{if (condition)} \rangle \langle \text{body} \rangle \text{ else } \langle \text{body} \rangle \)

Example: Recursion in BNF

```
Begin
  a = b + e;
  b = a + d;
  c = c + 1;
  d = e + f;
End
```

\( \langle \text{program} \rangle :: \langle \text{Begin} \langle \text{Body} \rangle \text{ End} \rangle \)

\( \langle \text{Body} \rangle :: \langle \text{Stmt} \rangle \mid \langle \text{Stmt} \rangle \langle \text{Body} \rangle \)