Gramática da linguagem P:

Programa → Funcao FuncaoSeq
FuncaoSeq → Funcao FuncaoSeq | ε
Funcao → **fn** NomeFuncao **(** ListaParams **)** TipoRetornoFuncao Bloco
NomeFuncao → **ID** | **MAIN**
ListaParams → **ID :** Type ListaParams2 | ε
ListaParams2 → **,** **ID :** Type ListaParams2 | ε
TipoRetornoFuncao → **->** Type | ε

Bloco → **{** Sequencia **}**
Sequencia → Declaracao Sequencia | Comando Sequencia | ε
Declaracao → **let** VarList **:** Type **;**
VarList → **ID** VarList2
VarList2 → **,** **ID** VarList2 | ε
Type → **int** | **float** | **char**

Comando → **ID** AtribuicaoOuChamada |
ComandoSe |
while Expr Bloco |
println(FMT_STRING, ListaArgs **) ;** |
return Expr **;**

AtribuicaoOuChamada → **=** Expr **;** | **(** ListaArgs **)** **;**
ComandoSe → **if** Expr Bloco ComandoSenao | Bloco
ComandoSenao → **else** ComandoSe | ε

Expr → Rel ExprOpc
ExprOpc → OpIgual Rel ExprOpc | ε
OpIgual → **==** | **!=**
Rel → Adicao RelOpc
RelOpc → OpRel Adicao RelOpc | ε
OpRel → **<** | **<=** | **>** | **>=**
Adicao → Termo AdicaoOpc
AdicaoOpc → OpAdicao Termo AdicaoOpc | ε
OpAdicao → **+** | **-**
Termo → Fator TermoOpc
TermoOpc → OpMult Fator TermoOpc | ε
OpMult → ***** | **/**

Fator → **ID** ChamadaFuncao |
INT_CONST |
FLOAT_CONST |
CHAR_LITERAL |
(Expr **)**

ChamadaFuncao → **(** ListaArgs **)** | ε
ListaArgs → Arg ListaArgs2 | ε
ListaArgs2 → **,** Arg ListaArgs2 | ε
Arg → **ID** ChamadaFuncao | **INT_CONST** | **FLOAT_CONST** | **CHAR_LITERAL**

Observações sobre a linguagem:

- Os terminais, exceto a cadeia vazia, estão destacados em negrito na gramática acima.
- **ID** é o token para os identificadores. Um identificador, ou seja, o nome de uma variável ou função, pode começar com letra e depois ser seguido por letra, número e underline sendo representados pela expressão regular [a-zA-Z]([a-zA-Z0-9_])*.
- **MAIN** é o token para a palavra reservada “main”.
- **INT_CONST** é o token que representa os números inteiros ([0-9]([0-9])*) e **FLOAT_CONST** é o token para os números de ponto flutuante (representados pela expressão regular [0-9]([0-9])*.[0-9]([0-9])*). Não são aceitos números na notação científica.
- O operador = é o operador de atribuição.
- **CHAR_LITERAL** é o token para um caractere literal que é definido entre aspas simples (‘’). Exemplo: ‘a’, ‘+’, ‘_’, etc; Entre as aspas simples é aceito qualquer caractere válido na linguagem P.
- **FMT_STRING** é o token para uma string de formatação. A string de formatação sempre inicia e termina com aspas duplas (“…”) e pode conter qualquer combinação de símbolos válidos na linguagem P juntamente com o marcador especial **{}**. Esse marcador é usado para marcar a posição onde será impresso valores inteiros, de ponto flutuante ou do tipo char.

Características da linguagem P:

- Linguagem de tipagem forte: todas as variáveis e parâmetros de funções devem ter um tipo.
- Todas as variáveis devem ser declaradas antes do uso.
- Regras restritas para a compatibilidade dos tipos.
- Regras de compatibilidade de tipos:
  - int é compatível somente com int;
  - float é compatível somente com float;
  - char é compatível somente com char;
- Todas as funções começam com a palavra reservada **fn**.
- A função principal é definida como **fn main** () { ... }.
- Tabela de precedência de operadores (da menor para a maior):

Precedência
== !=
< <= > >=
+ -
* /

Obs: Os operadores com maior precedência são * e /. Operadores que estão na mesma linha possuem a mesma precedência. A precedência pode ser alterada pelos parêntesis.

Palavras reservadas da linguagem: fn, main, let, int, float, char, if, else, while, println e return.

Sinais de pontuação e estruturação: ( ) -> : , { } ;

Operadores: =, ==, !=, >, >=, <, <=, +, -, *, /