

# Projeto Final INF1022

Gustavo A B Sampaio - 1712045

Gustavo Rodrigues - 1811619

[Repositório no Github](#)

## Motivação:

O projeto final da disciplina consiste no desenvolvimento do *Provol-One*, uma linguagem imperativa que gera um objeto numa linguagem arbitrária fazendo o uso de ferramentas de geração de compiladores *Flex/Bison* ou *Lex/Yacc*.

## Desenvolvimento:

Para o nosso projeto escolhemos a linguagem C como a linguagem de output da *Provol-One*, e escolhemos as ferramentas *Lex/Yacc* para o desenvolvimento dos arquivos *lex* e *yacc* do nosso compilador.

Estendemos nossa linguagem para permitir o uso de operadores de:

- comparação (>, <, ==, !=, >=, <=)
- operadores aritméticos (+, -, \*, /)
- blocos condicionais (if-then e if-then-else)
- comando de repetição definida (FACA <cmds> X vezes).

O programa aceita múltiplas entradas e múltiplos retornos, todos números inteiros.

Também fizemos algumas mudanças na gramática da linguagem para um melhor funcionamento:

```
program → ENTRADA varlist SAIDA varlist cmds FIM
varlist → varlist id | id
ret → ret | ret id
cmds → cmd cmds | cmd
cmd → ENQUANTO cmp FACA cmds FIM
cmd → FACA operacao VEZES cmds FIM
cmd → SE cmp ENTAO FACA cmds FIMIF
cmd → SE cmp ENTAO FACA cmds SENAO FACA cmds FIMIF
cmd → ID IGUAL operacao
```

*cmd* → *INC ABREPAR id FECHAPAR*  
*cmd* → *DEC ABREPAR id FECHAPAR*  
*cmd* → *ZERA ABREPAR id FECHAPAR*  
*atomic* → *NUM | ID*  
*operacao* → *operacao MAIS atomic | operacao MENOS atomic*  
*operacao* → *operacao VEZES atomic | operacao DIVIDE atomic*  
*operacao* → *atomic*  
*cmp* → *atomic MAIOR atomic | atomic MENOR atomic*  
*cmp* → *atomic MAIOR atomic | atomic MENOR atomic*  
*cmp* → *atomic IGUALA atomic | atomic DIFERENTE atomic*  
*cmp* → *atomic*

Para resolver o conflito de *shift/reduce* do *yacc* no *if/else*, usamos a marcação de *FIMIF*, removendo a ambiguidade na interpretação.

## Artefatos:

O compilador consiste nos arquivos *parser.y* e *parser.l*, que podem ser executados a partir do *run.sh*, que recebe como argumento o nome do arquivo *Provol-One* a ser lido. Os arquivos *.provolone* podem ser utilizados para testar o compilador, que terá como saída o arquivo *result.c*.

Exemplo de execução:

```
~$ sh run.sh testewhileif.provolone
```

## Léxico:

```
ENTRADA      { yyldata.id=strdup(yytext); return (ENTRADA); }
SAIDA        { yyldata.id=strdup(yytext); return (SAIDA); }
ENQUANTO     { yyldata.id=strdup(yytext); return (ENQUANTO); }
FACA         { yyldata.id=strdup(yytext); return (FACA); }
VEZES        { yyldata.id=strdup(yytext); return (VEZES); }
INC          { yyldata.id=strdup(yytext); return (INC); }
DEC          { yyldata.id=strdup(yytext); return (DEC); }
ZERA         { yyldata.id=strdup(yytext); return (ZERA); }
FIM          { yyldata.id=strdup(yytext); return (FIM); }
FIMIF        { yyldata.id=strdup(yytext); return (FIMIF); }
SE           { yyldata.id=strdup(yytext); return (SE); }
ENTAO        { yyldata.id=strdup(yytext); return (ENTAO); }
SENAO        { yyldata.id=strdup(yytext); return (SENAO); }
" ("         { yyldata.id=strdup(yytext); return (ABREPAR); }
") "         { yyldata.id=strdup(yytext); return (FECHAPAR); }
"="          { yyldata.id=strdup(yytext); return (IGUAL); }
(>=)         { yyldata.id=strdup(yytext); return (MAIORI); }
(<=)         { yyldata.id=strdup(yytext); return (MENORI); }
(>)          { yyldata.id=strdup(yytext); return (MAIOR); }
(<)          { yyldata.id=strdup(yytext); return (MENOR); }
(==)         { yyldata.id=strdup(yytext); return (IGUALA); }
(!=)         { yyldata.id=strdup(yytext); return (DIFERENTE); }
[_a-zA-Z][_a-zA-Z0-9]* { yyldata.id=strdup(yytext); return (ID); }
[0-9]+       { yyldata.id=strdup(yytext); return (NUM); }
[\\n]        { err_line++; }
[ \\t]       {}
"+"         { yyldata.id=strdup(yytext); return (MAIS); }
"-"         { yyldata.id=strdup(yytext); return (MENOS); }
"*"         { yyldata.id=strdup(yytext); return (MULT); }
"/"         { yyldata.id=strdup(yytext); return (DIVIDE); }
```

## Exemplos:

### Teste 1: Repetições, Se então, Se então senão e comparações

*testewhileif.provolone*

```
1  ENTRADA X Y
2  SAIDA Z
3      Z=Y
4      ENQUANTO X FACA
5          SE X>Y ENTAO FACA
6              INC(Z)
7          FIMIF
8          SE X<=Y ENTAO FACA
9              SE X<=13 ENTAO FACA
10                 INC(Z)
11             FIMIF
12         SENAO FACA
13             DEC(Z)
14         FIMIF
15     FIM
16 FIM
```

*resultado.c*

```
1  int provol(int Y, int X)
2  {
3      int Z;
4      Z = Y;
5
6      while(X)
7      {
8          if(X > Y)
9          {
10             Z++;
11         }
12
13         if(X <= Y)
14         {
15             if(X <= 13)
16             {
17                 Z++;
18             }
19         }
20         else
21         {
22             Z--;
23         }
24     }
25
26     return Z;
27 }
```

## Teste 2: Repetição definida, operações aritméticas e retorno múltiplo

*aritmeticafor.provolone*

```
1  ENTRADA X Y
2  SAIDA Z W
3      Z=Y
4      Z=Z+1
5      Z=Z-1
6      Z=Z*X
7      Z=1 + 1/X+7+48
8      FACA Z+1*X/Y VEZES
9          INC(Z)
10         DEC(W)
11     FIM
12
13 FIM
```

*resultado.c*

```
1  int provol(int Y, int X)
2  {
3      int W, Z;
4      Z = Y;
5      Z = Z + 1;
6      Z = Z - 1;
7      Z = Z * X;
8      Z = 1 + 1 / X + 7 + 48;
9
10     for(int i = 0; i < Z + 1 * X / Y; i++)
11     {
12         Z++;
13         W--;
14     }
15
16     return W, Z;
17 }
```