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

**Exemplos:**
**Teste 1: Repetições, Se então, Se então senão e comparações**

*testewhileif.provolone*

```
ENTRADA X Y
SAIDA Z
    Z=Y
    ENQUANTO X FACA
        SE X>Y ENTAO FACA
            INC(Z)
        FIMIF
        SE X<=Y ENTAO FACA
            SE X<=13 ENTAO FACA
                INC(Z)
            FIMIF
        SENAO FACA
            DEC(Z)
        FIMIF
    FIM
FIM
```

*resultado.c*

```c
int provol(int Y, int X)
{
    int Z;
    Z = Y;

    while(X)
    {
        if(X > Y)
    {
        Z++;
    }

    if(X <= Y)
    {
        if(X <= 13)
    {
        Z++;
    }
    }
    else
    {
        Z--;
    }
    }

    return Z;
}
```

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

*aritmeticafor.provolone*

```
ENTRADA X Y
SAIDA Z W
    Z=Y
    Z=Z+1
    Z=Z-1
    Z=Z*X
    Z=1  + 1/X+7+48
    FACA Z+1*X/Y VEZES
        INC(Z)
        DEC(W)
    FIM

FIM
```

*resultado.c*

```c
int provol(int Y, int X)
{
    int W, Z;
    Z = Y;
    Z = Z + 1;
    Z = Z - 1;
    Z = Z * X;
    Z = 1 + 1 / X + 7 + 48;

    for(int i = 0; i < Z + 1 * X / Y; i++)
    {
        Z++;
    W--;
    }

    return W, Z;
}
```