# Especificações da Implementação da Lista de Inteiros ao Pi Framework para Disciplina de Compiladores

**Aluno: Vitor Costa Hardoim**

**Professor: Christiano Braga**

## Especificação das novas equações adicionadas ao Pi Framework que dão semântica às novas construções em Pi IR que dão suporte à criação e manipulação de listas de inteiros.

Edições relativas ao arquivo pi.py.

Foram adicionadas classes para dar suporte à criação e manipulação das listas, sendo elas:

- IntegerArray
- ArrayProjection
- ArrayConcat
- ArrayAppend
- ArrayLength
- ArrayAssign

Novas KeyWords relativas às expressões anteriores também foram adicionadas à classe de KeyWords para expressões:

- `IDX = "#IDX"`
- `CONCAT = "#CONCAT"`
- `APPEND = "#APPEND"`
- `LENGTH = "#LENGTH"`
- `ARRAYASSIGN = "#ARRAYASSIGN"`

Além disso foram adicionadas as funções que efetivamente fazem o tratamento das informação da lista, as avaliando e inserindo à pilha de valores.

- `__evalIntegerArray`
- `__evalIntegerArrayKW`
- `__evalArrayProjection`
- `__evalArrayProjectionKW`
- `__evalArrayConcat`
- `__evalArrayConcatKW`
- `__evalArrayAppend`
- `__evalArrayAppendKW`
- `__evalArrayLength`
- `__evalArrayLengthKW`

- `__evalArrayAssign`
- `__evalArrayAssignKW`

Funções essas que serão utilizadas posteriormente pelo impiler. Além da adição de um condicionais em Bind e em Print para o reconhecimento de objetos do tipo IntegerArray, assim aceitando essa estrutura.

## Especificação da extensão da gramática de Imp para listas de inteiros.

Edições relativas ao arquivo imp2.ebnf.

Para implementação de listas na bnf da linguagem foram adicionadas as seguintes regras:

```
integer_array = '[' e:exp {','e:exp}* ']' ;

array_projection = idn:identifier '[' e:exp ']' ;

array_append = a1:exp '++' a2:exp | a1:integer_array '++'
a2:integer_array | a1:exp '++' a2:integer_array | a1:integer_array
'++' a2:exp ;

array_assign = idn:identifier '[' idx:exp ']' op:':=' e:exp;
```

Assim reconhecemos o formato de um array, a busca de um dado em certa posição no array, a opção de *append* no array, possilibando a assim o anexo de expressões e outras listas à uma lista, além da possibilidade de associar um novo valor à uma posição da lista.

Além disso algumas mudanças foram necessárias nas regras já existentes.

```
exp = integer_array | array_projection | array_append | paren_exp |
bin_exp | un_exp | @:atom ;

un_exp = op:"\#" e:exp | op:"not" e:exp ;

binop = "\+" | "and"| "or" | "==" | "<=" | ">=" | "<" | ">"  | "+" |
"-" | "*" | "/"  ;
```

O reconhecimento das operações em lista como expressões, a adição de uma operação unária para retornar o tamanho da lista(\#), além da adição de uma operação binária para a ação de concatenação.

## Especificação das Pi denotações de Imp à Pi IR relativa às extensões realizadas.

Edições relativas ao arquivo impiler.py.

Foi adicionado um condicional de expressões unárias para o tratamento da função “length” retornando o tamanho da lista, utilizando a classe criada anteriormente em pi.py “ArrayLength”. E um condicional para a expressão binária de concatenação, chamando a classe com esse nome em pi, “ArrayConcat”.

Além disso, foram criadas 4 novas funções, com o objetivo de utilizar as funções criadas em pi.py, sendo elas:

- integer_array
- array_projection
- array_append
- array_assign

Propostas para tratar das regras de produção com mesmo nome declaradas em imp2.bnf.

integer_array: Chama a estrura de IntegerArray em pi para tratar estruturas de array.

array_projection: Chama a estrutura de ArrayProjection em pi para tratar situações em que é dado o identificador do array e uma posição no mesmo, retornando o valor nessa posição.

array_append: Chama a estrutura de ArrayAppend em pi para situações em que se recebe 2 expressões ou arrays, retornando um novo array composto por esses 2 parâmetros.

array_assign: Chama a estrutura de ArrayAssign em pi para situações iguais a projeções seguidas de ":=" e uma expressão, assim associando um novo valor àquela posição do array.