

Tutorial de P.I.C.A.T

Paulo Victor de Aguiar

Claudio Cesar de Sá e Outros

Universidade do Estado de Santa Catarina – UDESC

Departamento de Ciência da Computação – DCC

Joinville – SC

16 de maio de 2016

Índice

1 Introdução

2 Tipo de dados

- Variável
- Âtomo
- Número
- Termo Composto
 - Listas
 - Strings
 - Estruturas
 - Vetores
 - Mapas
 - Conjuntos

3 Exemplos Práticos

4 Formulário

O que é P.I.C.A.T?

Pattern-matching: Utiliza o conceito de casamento de padrão.

Intuitive: O Picat oferece atribuições e laços de repetição (*loops*) para a programação dos dias de hoje.

Constraints: Picat suporta a programação por restrições.

Actors: Atores são chamadas orientadas a eventos.

Tabling: É possível guardar o resultado de certas operações na memória.





Histórico

O P.I.C.A.T é uma linguagem multiparadigma projetada para aplicações gerais de programação. Foi criada em 2013 por Neng-Fa Zhou e Jonathan Fruhman utilizando o B-Prolog como base na implementação. Ambas as linguagens utilizam regras lógicas na programação, porém o P.I.C.A.T possui mais funcionalidades.

Comparação com outras linguagens

	C	Haskell	Java	Prolog	P.I.C.A.T
Paradigma(s)	procedu- ral	funcional	orientado à objetos	lógico	multi- paradigma
Tipagem	fraca	forte	forte	fraca	fraca
Verificação de tipos	estático	estático	estático	dinâmico	dinâmico
Possui segu- rança?	não	sim	sim	sim	sim
Passagem de parâmetros	valor	-	valor	valor	casamento
Legibilidade	baixa	média	média	média	boa

PICAT

DownloadModulesProjectsResources

Download

Version 1.8 ([Get Started With Picat](#))

picat18_win.zip	MS Windows
picat18_cygwin64.tar.gz	Cygwin 64-bit
picat18_linux64.tar.gz	Linux 64-bit
picat18_macx.tar.gz	MacOS X (64-bit)
picat18_src.tar.gz	C source code

Figura: <http://picat-lang.org/download.html>

Tipo de dados

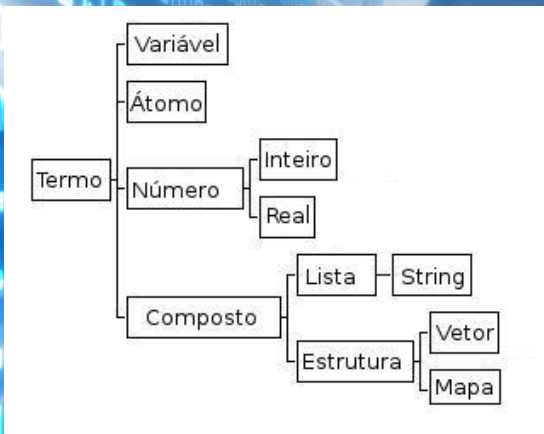


Figura: Hierarquia dos Tipos de dados

Operadores básicos

Precedência	Símbolo	Significado
<i>Alta</i>	. @	teste padrão
	<i>div, mod, rem</i>	divisão, modulo, resto da divisão
	**	potenciação
	» «	conversores binários
	^	disjunção exclusiva (<i>Xor</i>)
	..	enumerador de conjunto
	++	concatenação
	<i>not, once</i>	negação, único
	&& ,	conjunção (<i>And</i>)
<i>Baixa</i>	// ;	disjunção (<i>Or</i>)

Descrição

As variáveis em Picat são similares as variáveis das matemática, pois ambas guardam valores. Quando uma variável ainda não foi instanciada com um valor, ela fica em um estado **livre**. Uma vez quando for **instanciada** com um valor, ela terá a mesma identidade como se fosse um valor até que ela seja liberada de novo. O nome de uma variável é identificado por iniciar com uma letra maiúscula ou com *underline*, como demonstrado nos seguintes exemplos:

X, Y, Xa, Y1, a, bc

Funções

- `var(Termo)`: verifica se a variável é livre, se for retorna `true`.
- `nonvar(Termo)`: verifica se a variável não é livre, se for retorna `true`.
- `attr_var(Termo)`: verifica se a variável está instanciada, se for retorna `true`.
- `dvar(Termo)`: verifica se a variável instancia está dentro do domínio, se for retorna `true`.

Descrição

Um átomo é uma constante simbólica e seu nome pode ser representado tanto com aspas simples ou sem. Como por exemplo:

`x`, `x_1`, `'a'`, `'b1'`.

Funções

- `atom(Termo)`: verifica se o termo é um átomo.
- `atom_chars(Termo)`: retorna uma *string* contendo os caracteres do átomo, irá ocorrer um erro se a função não for um átomo.
- `atom_codes(Termo)`: retorna uma lista de códigos dos caracteres do átomo, irá ocorrer um erro se a função não for um átomo.
- `char(Termo)`: verifica se o átomo possui um único carácter, se for retorna `True`.
- `digit(Termo)`: verifica se o átomo possui um único dígito, se for retorna `True`.
- `len(Termo)`: retorna o número de caracteres de um átomo.

Número

Descrição

Um número é um átomo inteiro ou real. Um número inteiro pode ser representado na forma decimal, binária, octal ou hexadecimal. Já o número real usa o ponto no lugar da virgula para separar os valores depois de zero como: 3.1415.

Funções

- `number(Termo)`: verifica se o termo é um número.
- `float(Termo)`: verifica se o termo é um número real.
- `int(Termo)`: verifica se o termo é um número inteiro.
- `max(X, Y)`: compara dois termos e retorna o maior deles.
- `min(X, Y)`: compara dois termos e retorna o menor deles.

Operadores Aritméticos

Tabela: Operadores Aritméticos

Formula	Operação
$X + Y$	<i>Adição</i>
$X - Y$	<i>Subtração</i>
$X * Y$	<i>Multiplicação</i>
X / Y	<i>Divisão</i>
$X // Y$	<i>Divisão Truncada</i>
$X \bmod Y$	<i>Resto da Divisão</i>


```
Terminal - pv@manjaro:~/Downloads/Picat
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> A = 5, B = 7, number(A), number(B), max(A, B) = Maximo, min(A, B) = Mini
mo.
A = 5
B = 7
Maximo = 7
Minimo = 5
yes
Picat> 
```

Figura: Exemplo de execução

Termo Composto

Descrição

Um termo composto se divide entre listas, *strings*, estruturas e outros tipos compostos derivado destes são: vetores, mapas e conjuntos. Entretanto, ambos tem seus elementos acessados via casamento de padrões de fatos, predicados e funções.

Lista

Descrição

A forma de uma lista reúne um conjunto de termos e os coloca dentro de colchetes: $[t_1, t_2, \dots, t_n]$.

Listas

```
Terminal - pv@manjaro:~/Downloads/Picat/
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> A=[1,2,3], list(A), length(A)=L_A, B= [4,5,6], list(B), length(B) = L_B,
A ++ B = C, list(C), length(C) = L_C.
A = [1,2,3]
L_A = 3
B = [4,5,6]
L_B = 3
C = [1,2,3,4,5,6]
L_C = 6
yes
Picat> 
```

Figura: Exemplo de execução

Descrição

Uma *string* pode ser representada em forma de lista, ou seja, cada caractere de uma *string* é um termo de uma lista. Por exemplo: a palavra "carro" pode ser expressa da forma [c,a,r,r,o]..

Strings

```
Terminal - pv@manjaro:~/Downloads/Picat
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> X = "Isto eh uma", string(X), to_uppercase(X) = Y.
X = ['I','s','t','o',' ','e','h',' ','u','m','a']
Y = ['I','S','T','O',' ','E','H',' ','U','M','A']
yes
Picat> 
```

Figura: Exemplo de execução

Descrição

A forma de uma estrutura é definida como $s(t_1, t_2, \dots, t_n)$, onde s é um átomo e $\$$ é usado para diferenciar uma função. Seus principais elementos são o nome da estrutura que é o átomo que fica na frente e a aridade (número de argumentos do predicado).


```
Terminal - pv@manjaro:~/Downloads/Picat
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> N = $(1,2,3,4,5), struct(N), arity(N) = Aridade, to_list(N) = Lista.
N = (1,2,3,4,5)
Aridade = 2
Lista = [1,(2,3,4,5)]
yes

Picat> []
```

Figura: Exemplo de execução

Descrição

Um vetor ou um *array* tem o formato de $\{t_1, t_2, \dots, t_n\}$, onde t_i é um termo desta estrutura de aridade n .

```
Terminal - pv@manjaro:~/Downloads/Picat  
File Edit View Terminal Tabs Help  
[pv@manjaro ~]$ cd Downloads/Picat/  
[pv@manjaro Picat]$ ./picat  
Picat 1.6, (C) picat-lang.org, 2013-2015.  
Picat> A = {a, b, c}, array(A), length(A) = Comprimento.  
A = {a,b,c}  
Comprimento = 3  
yes  
  
Picat> █
```

Figura: Exemplo de execução

Mapas

Descrição

Os mapas tem a mesma forma de uma estrutura, porém eles possuem um valor especial para ser usado como chave.

Mapas

```
Terminal - pv@manjaro:~/Downloads/Picat
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> new_map(3)=M, put(M,a,3), put(M,b,2), put(M,c,1), Valor = get(M,b).
M = (map)[a = 3,b = 2,c = 1]
Valor = 2
yes
Picat> █
```

Figura: Exemplo de execução

Conjuntos

Descrição

Um conjunto é um mapa onde todos os elementos da estrutura estão associados à uma chave de valor não-numérico. Todas as funções existentes para os mapas podem ser também utilizadas. Para criá-lo é necessário o comando `new_set`.

Conjuntos

```
Terminal - pv@manjaro:~/Downloads/Picat
File Edit View Terminal Tabs Help
[pv@manjaro ~]$ cd Downloads/Picat/
[pv@manjaro Picat]$ ./picat
Picat 1.6, (C) picat-lang.org, 2013-2015.
Picat> new_set(2) = N, put(N,a), put(N,b), put(N,a), size(N)=Cardinalidade.
N = (map)[a,b]
Cardinalidade = 2
yes
Picat> 
```

Figura: Exemplo de execução

Exemplos Práticos

Exemplos podem ser vistos em:

- <https://github.com/claudiosa/CCS/tree/master/picat>

Formulário

- ① Qual característica do P.I.C.A.T é mais chamativa?
- ② Em quais aplicações você usaria P.I.C.A.T?
- ③ Quais são os pontos positivos e negativos do P.I.C.A.T que você identifica?
- ④ Se pudesse melhorar algo no P.I.C.A.T, o que melhoraria?
- ⑤ O P.I.C.A.T pode substituir alguma linguagem?
- ⑥ Você usaria o P.I.C.A.T no lugar de alguma linguagem como C ou Java?