Universidade Federal de Alagoas
Instituto de Computação
Curso de Ciência da Computação

Softy
Especificação da Linguagem

Thiago Tenório Cavalcante Costa
Guilherme Volney Mota Amaral

Maceió 2019.2

Introdução

A linguagem de programação Softy foi criada com inspiração nas linguagens de programação C e Python, tendo como meta de utilização o ensino instrutivo da programação.

Softy não é orientada a objetos e é *case-sensitive* - diferencia letras maiúsculas de minúsculas -. Partindo para legibilidade, Softy não faz coerção (não admite conversões implícitas de tipo) e possui palavras reservadas, as quais estão em inglês e foram escolhidas para que fique o mais claro possível o que o código está fazendo.

Estrutura geral do programa

- ## Ponto de início de execução

  O ponto inicial de execução do programa será identificado por uma construção específica da linguagem, sempre será a ***função main()*** com o tipo de retorno obrigatório ***int***.

```
fun int main() {
      .
      .
      .
      return 0;
}
```

- ## Definições de procedimentos / funções

  Estes poderão existir se forem declaradas fora do corpo de uma outra função ou procedimento. Estas podem ser acessadas se tiverem sua implementação ou assinatura declaradas antes da função chamadora. Softy não aceita passagem de subprogramas como parâmetros.

  A declaração de função é sempre iniciado pela palavra reservada ***fun***, seguido pelo seu tipo de retorno obrigatório (***int***, ***float***, ***bool***, ***string***, ***char***), seu ***identificador*** único e uma lista de parâmetros (os parâmetros acompanharão de seu tipo e seu identificador) delimitada por abre e fecha parênteses ***( )***. A abertura e fechamento do bloco é feita por abre e fecha chaves ***{ }***.

A declaração de um procedimento seguirá mesmo padrão, com exceção de que não terá o tipo de retorno e será sempre iniciada com a palavra reservada ***proc***.

A passagem de parâmetro para os procedimentos e funções se dá por meio do modo de entrada e saída, para todos os tipos e estruturas.

Ex.:

```
fun tipoDoRetorno id ( listaDeParametros ) {
      .
      .
      .
      return x;
}

proc id ( listaDeParametros ) {
      .
      .
      .
}
```

A declaração de assinaturas seguirá o padrão de função ou procedimento, com exceção de que finaliza com um **ponto e vírgula** ao invés de abre e fecha chaves.

Ex.:

```
fun tipoDoRetorno id ( listaDeParametros );

proc id ( listaDeParametros );
```

- ## Definições de instruções

As instruções podem ser declaradas somente dentro do bloco de funções ou procedimentos, não podendo ser escritas instruções dentro do corpo dos parâmetros de uma função, tal como estruturas de iteração e estruturas de seleção.

Conjuntos de tipos de dados e nomes

- ## Identificador

  Possuirão sensibilidade à caixa, limite de tamanho de 31 caracteres e seu formato será definido a partir da seguinte expressão regular:

  *(‘letter’ | ‘_’)(‘digit’ | ‘letter’ | ‘_’)**

  Exemplos de nomes aceitos pela linguagem: _var1, var2, var, _, _a;
  Exemplos de nomes não aceitos pela linguagem: .abc, ab@c, 1ac.

- ## Palavras reservadas

  As palavras reservadas são sempre escritas em inglês, buscando objetividade e simplicidade, são listadas a seguir:

  const, char, int, float, bool, string, bool, fun, proc, read, print, if, ceif (check else if)*, else, false, true, for, while, return, length.*

- ## Definição de variáveis

  Softy possui escopo global, logo as variáveis podem ser definidas fora do escopo e acessadas globalmente ou dentro do escopo como variáveis locais. São declaradas iniciando com o ***tipoDaVariável*** *(**int**, **float**, **bool**, **string**, **char**)* e seguidas pelo seu **identificador único**. Caso haja múltiplas variáveis em uma declaração de tipo único, as variáveis serão separadas por ***vírgula***. A declaração termina com um ***ponto e vírgula***.

  Podem também ser acompanhadas de uma atribuição na própria declaração. Para fazer isso, precisa colocar o símbolo de ***igualdade*** seguido do ***valor***.

  Ex.:

  ```
  int b = 2;
  float c = 3.4, d;
  char e;
  bool g, h = true;
  string i = “hello”;
  ```

- **Definição de constantes**

    Constantes possuem o mesmo padrão de declaração e definição de variáveis, com exceção que iniciam sempre com a palavra reservada ***const*** e que a atribuição de valor é obrigatória na declaração.

    Ex.:

```
const int i = 0, j = 1;
const bool debug = true;
```

Tipos e Estrutura de Dados

Todos os tipos e estruturas possuirão compatibilidade por nome.

- **Inteiro**

    É a identificação da variável como sendo do tipo inteiro de 32 bits, identificado pela palavra reservada ***int***. Seus literais são expressos como uma sequência de números decimais.

    Ex.: `int i;`

- **Ponto Flutuante**

    É a identificação da variável como sendo do tipo ponto flutuante de 64 bits, identificado pela palavra reservada ***float***. Seus literais são expressos como uma sequência de números decimais, seguido de um ***ponto*** e os demais dígitos.

    Ex.: `float f;`

- **Caractere**

    É a identificação da variável como sendo do tipo caractere de 8 bits, identificado pela palavra reservada ***char***. Guarda um código referente ao seu símbolo na tabela ASCII. A constante literal do caractere é delimitada por ***apóstrofo***.

    Ex.: `char c;`

- ## Cadeia de Caracteres

    É a identificação da variável como sendo do tipo cadeia de caracteres, identificado pela palavra reservada ***string***. Seus literais são expressos como um conjunto de caracteres, mínimo de 0 caracteres e de tamanho máximo ilimitado. A constante literal é delimitada por ***aspas***.

    Ex.: `string s;`

- ## Booleano

    É a identificação da variável como sendo do tipo booleana, identificado pela palavra reservada ***bool***. Possui dois valores possíveis: ***true***, ***false***.

    Ex.: `bool b;`

- ## Arranjos unidimensionais

    Os arranjos unidimensionais são compostos pelos tipos determinados acima. Sua declaração iniciará com o ***tipo*** seguido do ***identificador único*** e, delimitado por abre e fecha parênteses ***( )***, o ***tamanho*** do arranjo.

    Há uma função que dirá o tamanho do arranjo, a função retorna um inteiro. *Exemplo: int s = length(arr);*.

    Ex.:

```
int arr(9);
float arr(1024);
bool arr(2);
```

- ## Operações suportadas

Operador	Tipos que realizam a operação
‘!’	bool
‘^ ’ ‘*’ ‘/’ ‘+’ ‘-’	int, float
‘<’ ‘<=’ ‘>’ ‘>=’	int, float, char
‘==’ ‘!=’ ‘=’	int, float, bool, char, string
‘&&’ ‘||’	bool

‘::’	string, char

- **Coerção**

  Softy é estaticamente tipada, não aceitando coerção entre variáveis de tipos diferentes. Dessa forma, todas as verificações de compatibilidade de tipo serão feitas estaticamente.

- **Valores padrão**

Tipo	Valor de inicialização
int	0
float	0.0
char	‘ ’ (caracter vazio)
string	“ ” (string vazia)
bool	false

Conjunto de Operadores

- **Aritméticos**

```
- (unário negativo)
* (multiplicação)
/ (divisão)
% (resto)
+ (soma)
- (subtração)
```

- **Relacionais**

```
<  (menor que)
<= (menor ou igual que)
>  (maior que)
```

```
>= (maior ou igual que)
== (igual)
!= (diferente)
```

Ao utilizar a operações relacionais de igualdade ou desigualdade entre caracteres ou entre cadeia de caracteres, primeiro é verificado se possuem o mesmo tamanho (1 se for um caractere). Depois checa os caracteres um a um até achar um diferente ou o final da cadeia. Os demais operadores relacionais comparam a ordem lexicográfica dos caracteres um a um.

- **Lógicos**

```
!  (negação unária)
&& (conjunção)
||  (disjunção)
```

- **Concatenação de cadeia de caracteres**

O operador de concatenação é dado por ***::***, pode ser aplicada a strings e caracteres, resultando sempre em uma string.

- **Precedência e Associatividade**

Operadores (Precedência)	**Associatividade**
‘!’ (Not)	Direita para esquerda
‘-’ (Unário Negativo)	Direita para esquerda
‘^’ (Exponenciação)	Direita para esquerda
‘*’ (Multiplicação) e ‘/’ (Divisão)	Esquerda para direita
‘+’ (Soma) e ‘-’ (Subtração)	Esquerda para direita
‘<’, ‘<=’, ‘>’ e ‘>=’	Não associativo
‘==’ e ‘!=’	Não associativo
‘&&’	Esquerda para direita
‘||’	Esquerda para direita

‘::’ (Concatenação)	Esquerda para direita
‘=’ (Atribuição)	Direita para esquerda

Instruções

- ## Atribuição

  É definida pelo símbolo ‘=’, sendo o lado esquerdo o ***identificador*** único e o lado direito o valor ou expressão. Todos os operandos devem possuir o mesmo tipo da variável alvo.

- ## Estrutura condicional

```
if ( expressaoLogica ) {
      .
      .
} ceif ( expressaoLogica ) {
      .
      .
} else {
      .
      .
}
```

- ## Estrutura iterativa com controle lógico

```
while ( expressaoLogica ) {
      .
      .
}
```

- ## Estrutura iterativa controlada por contador

```
for (int i : (a, b, c)) {
      .
      .
}
```

Ambos os loops farão um pré-teste para verificar se a repetição ainda ocorrerá ou não. Os três valores (a, b, c) representam expressões aritméticas. O valor de a será o valor inicial do contador, b será o valor final e c será o valor de incremento.

Exemplos:

```
for(int i : (0, 5, 1)) {
     //código
}
```

```
while(var < 10) {
     //código
}
```

```
for(int i : (var, var + 5, 1)) {
     //código
}
```

Constantes Literais e Suas Expressões

As expressões regulares das constantes literais são denotadas da seguinte maneira:

1. Constantes de inteiros: *((‘digit’)+)*;
2. Constantes de floats: *((‘digit’)+)(‘\.’)((‘digit’)+)*;
3. Constantes de char: *(‘\’’)(‘letter’ | ‘symbol’ | ‘digit’)(‘\’’)*;
4. Constantes de bool: *(‘true’ | ‘false’)*;
5. Constantes de string: *(‘\”’)((‘letter’| ‘symbol’ | ‘digit’)*)(‘\”’)*;

Desvios Incondicionais

Softy não aceita nenhum tipo de desvio incondicional.

Entrada e Saída

- Entrada: É realizada através da função read().
    - O comportamento da função read dependerá da variável a qual se está sendo lida. No caso de ser um inteiro, read atribuirá um único inteiro ao valor designado ao ***id***, assim como floats, char, string,

bool. Caso a variável seja um arranjo, read irá ler ***length(arr)***. A função read também aceita vários parâmetros, atribuindo, assim, cada um dos valores recebidos como entrada para o código será atribuído às respectivas variáveis. Caso um dos valores seja um arranjo e a função read tenha recebido mais de um valor como parâmetro, ela irá esperar receber ***length(arr)*** valores quando encontrar o identificador relacionado ao arranjo na ordem de entrada e em seguida os valores relacionados às sucessivas variáveis.

- Saída: É realizada através da função print().
    - O comportamento da função print dependerá do tipo da variável que será passada como parâmetro. No caso de ser um inteiro, print irá mostrar na tela um único inteiro: o valor designado ao ***id***, assim como para floats, char, string, bool. Caso a variável seja um arranjo, print irá mostrar na tela ***length(arr)*** valores, cada um relacionado a cada valor nas posições do arranjo, separados por vírgula. Caso o valor seja relacionado a uma constante, print irá imprimir diretamente o valor dessa constante.
    - A saída poderá ser formatada, os identificadores dos tipos de formatação são: %d inteiro, %c caractere, %f float, %s string, %b bool. Caso a saída se encontre formatada, no código, deverá ser especificado quais são as variáveis específicas a cada um dos identificadores de formatação.

      Ex: print("%s%d%c%b%f", str, integer, ch, boolean, flo);

Errata

Durante o desenvolvimento do analisador sintático, algumas mudanças ocorreram em relação a estruturação da linguagem:

1. Operador módulo: foi removido da linguagem;
2. Declaração de um arranjo: a declaração de um arranjo sofreu mudanças, anteriormente declarado como "(***tamanhoDoArranjo***)", agora possui a estrutura "[***tamanhoDoArranjo***]";
3. Print: Softy não aceita mais prints que não possuam constantes literais de string em seu parâmetro. Por exemplo: anteriormente era aceito

print(***listaDeInteiros***) tais como de floats, char, string e bool. Agora somente será aceito print("***constanteLiteralString***");

4. For: agora o terceiro parâmetro do for (a passada da variável declarada anteriormente) poderá ser omitido, de forma que, ao ser omitido, o passo de incremento do for será de 1;

Códigos Exemplo

- Hello World:

```
fun int main() {
  print("Hello World");
}
```

- Shell Sort:

```
int n = 1000;

proc shellSort(int arr(n)) {
	int gap = n / 2;
	while (gap > 0) {
		for (int i :(gap, n, 1)) {
			int temp = arr[i];
			int j = i;
			while (j >= gap && (arr[j - gap] > temp)) {
				arr[j] = arr[j - gap];
				j = j - gap;
			}
			arr[j] = temp;
		}
		gap = gap / 2;
	}
}

fun int main() {
	int arr(n);
	read(arr);
	for (int i : (0, n, 1)) {
		print("%d ", arr[i]);
	}
```

```
        shellSort(arr);
        for (int i : (0, n, 1)) {
                print("%d ", arr[i]);
        }
        print("\n");
        return 0;
}
```

- Série de Fibonacci

```
proc fib(int n) {
        int n1 = 0, n2 = 1, n3;
        if (n == 0) {
                print("%d\n", n);
        } ceif (n == 1) {
                print("0, %d\n", n);
        } else {
                string separator = "";
                print("0, 1, ");
                while (true) {
                        n3 = n1 + n2;
                        print("%s%d", separator, n3);
                        if (n3 >= n) {
                                print("\n");
                                return;
                        }
                        separator = ", "
                        n1 = n2;
                        n2 = n3;
                }
        }
}

fun int main() {
        int n;
        read(n);
        fib(n);
        return 0;
}
```