Universidade Federal de Santa Catarina

INE5426 – Construção de Compiladores 2016/1

Projeto 1 – Linguagem: Allen

A linguagem de programação Allen é uma linguagem procedural, estaticamente tipada, fortemente tipada e com escopos estáticos. Ela é inspirada em outras linguagens como Pascal e Pico. Ela tem múltiplas versões, cada qual com características e mecanismos adicionais.

Os grupos irão desenvolver um compilador para Allen que gera a árvore de sintaxe para um dado código e consegue imprimir informações relevantes sobre o código posteriormente. Diferentemente dos compiladores construídos na disciplina até o momento, o compilador para Allen não servirá como um interpretador da linguagem (não irá computar o código). O projeto ainda poderá ser estendido para incluir a geração de código para diferentes versões.

O nome da linguagem de programação deste projeto vem de Frances Elizabeth Allen, uma pioneira na área de otimização de compiladores. Suas contribuições na área a levaram a ser a primeira mulher a receber um Turing Award, a premiação máxima da computação.

Índice

Versão 1.0 – Linguagem base

A versão base da linguagem tem as seguintes características:

1. Três tipos:
    a. Inteiros (int) com valores representados por dígitos.
    b. Ponto flutuante de precisão dupla (real) com valores representados por dígitos e um ponto.
    c. Booleanos (bool) com valores TRUE e FALSE.
2. Quatro operadores binários para inteiros e ponto flutuante: adição (+), subtração (-), multiplicação (*) e divisão (/).
3. Um operador unário para inteiros e ponto flutuante: menos (-).
4. Seis operadores relacionais: igual (=), diferente (~=), maior (>), menor (<), maior ou igual (>=), menor ou igual (<=)
5. Dois operadores binários para booleanos: e (AND) e ou (OR).
6. Um operador unário para booleanos: não (~).
7. Coerção de valores inteiros para valores em ponto flutuante.
8. Atribuições (:=).
9. Parênteses (()).
10. Símbolo de fim de expressão/linha (;).

A precedência dos operadores para valores inteiros e de ponto flutuante é a padrão. A precedência para expressões booleanas é: [maior] operadores relacionais, operador unário, operadores binários.

É importante ressaltar que variáveis não podem ser usadas antes de serem declaradas nem antes de serem atribuídas.

Exemplo de código

```
int: i, j;
real: a;
bool: comp;
a := 10.0 + .57 - 2.;
i := -2;
j := i * ( 3 / 5 );
comp:= a > j;
comp:= TRUE;
```

Saída do compilador

```
Declaracao de variavel inteira: i, j
Declaracao de variavel real: a
Declaracao de variavel booleana: comp
Atribuicao de valor para variavel real a: valor real 10.0 (soma
real) valor real .57 (subtracao real) valor real 2.
Atribuicao de valor para variavel inteira i: (menos unario inteiro)
valor inteiro 2
Atribuicao de valor para variavel inteira j: variavel inteira i
(vezes inteiro) (abre parenteses) valor inteiro 3 (divisao inteira)
valor inteiro 5 (fecha parenteses)
Atribuicao de valor para variavel booleana comp: variavel real a
(maior real) variavel inteira j para real
Atribuicao de valor para variavel booleana comp: valor booleano TRUE
```

Versão 1.1 – Arranjos

A versão 1.1 inclui o uso de arranjos unidimensionais. Em Allen, os arranjos usam índices inteiros que começam em 1.

Exemplo de código

```
int[10]: ar;
int: i, j;
i:= 2;
j:= 5;
ar[1]:= i ;
ar[i]:=3;
ar [j-i]:= 0;
```

Saída do compilador

```
Declaracao de arranjo inteiro de tamanho 10: ar
Declaracao de variavel inteira: i, j
Atribuicao de valor para variavel inteira i: valor inteiro 2
Atribuicao de valor para variavel inteira j: valor inteiro 5
Atribuicao de valor para arranjo inteiro ar:
+indice: valor inteiro 1
+valor: variavel inteira i
Atribuicao de valor para arranjo inteiro ar:
+indice: variavel inteira i
+valor: valor inteiro 3
Atribuicao de valor para arranjo inteiro ar:
+indice: variavel inteira j (subtracao inteira) variavel inteira i
+valor: valor inteiro 0
```

Versão 1.2 – Declaração, definição e uso de funções

A versão 1.2 inclui o uso de funções. Algumas regras importantes sobre funções são as seguintes:

1. Uma função não pode ser usada antes de sua declaração.
2. Erros devem ser gerados caso funções acabem não sendo definidas no código.
3. Todos os parâmetros de uma função devem ser passados em seu uso.
4. Funções só podem ser declaradas e definidas no escopo global.
5. A definição de uma função pode usar variáveis globais.

Exemplo de código

```
def fun bool: true() return TRUE; end def
int: xyz;
decl fun int: soma(int: a, int: b);
def fun int: soma(int: a, int: b)
  return a + b;
end def
xyz:= soma(1,1);
```

Saída do compilador

```
Definicao de funcao booleana: true
+parametros:
+corpo:
Retorno de funcao: valor booleano TRUE
Fim definicao
Declaracao de variavel inteira: xyz
Declaracao de funcao inteira: soma
+parametros:
Parametro inteiro: a
Parametro inteiro: b
Fim declaracao
Definicao de funcao inteira: soma
+parametros:
Parametro inteiro: a
Parametro inteiro: b
+corpo:
Retorno de funcao: variavel inteira a (soma inteira) variavel
inteira b
Fim definicao
Atribuicao de valor para variavel inteira xyz: chamada de funcao
inteira soma {+parametros: valor inteiro 1, valor inteiro 1}
```

Versão 1.3 – Expressões condicionais

A versão 1.3 inclui o uso de expressões condicionais no estilo if-then e if-then-else. O teste da expressão condicional recebe um valor ou variável booleana.

Exemplo de código

```
int: i;
bool: true;
if 10.0 > -2.0
then
i:= 5;
else i:= 3;
end if
true:= TRUE;
if true then i:= 0; end if
```

Saída do compilador

```
Declaracao de variavel inteira: i
Declaracao de variavel booleana: true
Expressao condicional
+se: valor real 10.0 (maior real) (menos unario real) valor real 2.0
+entao:
Atribuicao de valor para variavel inteira i: valor inteiro 5
+senao:
Atribuicao de valor para variavel inteira i: valor inteiro 3
Fim expressao condicional
Atribuicao de valor para variavel booleana true: valor booleano TRUE
Expressao condicional
+se: valor booleano true
+entao:
Atribuicao de valor para variavel inteira i: valor inteiro 0
Fim expressão condicional
```

Versão 1.4 – Laços

A versão 1.4 de Allen inclui laços do tipo while.

Exemplo de código

```
int: i;
real[10]: ar;
i:= 1
while i <= 10 do ar[i]:= 0.0; i:= i + 1; end while
```

Saída do compilador

```
Declaracao de variavel inteira: i
Declaracao de arranjo real de tamanho 10: ar
Atribuicao de valor para variavel inteira i: valor inteiro 1
Laco
+enquanto: variavel inteira i (menor ou igual inteiro) valor inteiro
10
+faca:
Atribuicao de valor para arranjo real ar:
+indice: variavel inteira i
+valor: valor real 0.0
Atribuicao de valor para variavel inteira i: variavel inteira i
(soma inteira) valor inteiro 1
Fim laco
```

Versão 1.5 – Definição de tipos compostos

A versão 1.5 da linguagem permite definir tipos compostos para serem usados em variáveis, funções, arranjos e afins.

Exemplo de código

```
def type: complex
real: r;
real: i;
end def

complex[2]: vals;
vals[1].r := 1.0;
vals[1].i := 2.0;
```

Saída do compilador

```
Definicao tipo: complex
+componentes:
Componente real: r
Componente real: i
Fim definicao
Declaracao de arranjo complex de tamanho 2: vals
Atribuicao de valor para arranjo complex vals componente real r:
+indice: valor inteiro 1
+valor: valor real 1.0
Atribuicao de valor para arranjo complex vals componente real i:
+indice: valor inteiro 1
+valor: valor real 2.0
```

Termos para mensagens de saída do compilador

Versão 1.0

nome: [a-z]+
int: [0-9]+
real: [0-9]+.[0-9]* | [0-9]*.[0-9]+
tipo masculino: {inteiro | real | booleano | tipo definido no código}
tipo feminino: {inteira | real | booleana | tipo definido no código}
Declaracao de variavel *tipo feminino*: *nome*{, *nome*}*
Declaracao de arranjo *tipo masculino*: *nome*{, *nome*}*
Atribuicao de valor para variavel *tipo feminino nome*:
({abre | fecha} parenteses)
({soma | subtracao | multiplicacao | divisao} {inteira | real})
(menos unario {inteiro | real})
({igual | diferente | maior | maior ou igual | menor | menor ou igual} {inteiro | real})
({e | ou | nao} booleano)
variavel *tipo feminino*
variavel *tipo feminino* para real
valor *tipo masculino*

Versão 1.1

Declaracao de arranjo *tipo masculino* de tamanho *int*: *nome*
Atribuicao de valor para arranjo *tipo masculino nome*:
+indice: expressão na linha
+valor: expressão na linha

Versão 1.2

Definicao de funcao *tipo feminino*: *nome*
+parametros: em novas linhas
Parametro *tipo masculino*: nome
+corpo: em novas linhas
Retorno de funcao:
Fim definicao
Declaracao de funcao *tipo feminino*: *nome*
+parametros: em novas linhas
Fim declaracao
chamada de funcao *tipo feminino nome* {+parametros: }

Versão 1.3

Expressao condicional
+se: expressão na linha
+entao: em novas linhas
+senao: em novas linhas
Fim expressao condicional

Versão 1.4

```
Laco
+enquanto: expressão na linha
+faca: em novas linhas
Fim laco
```

Versão 1.5

```
Definicao tipo: nome
+componentes: em novas linhas
Componente tipo masculino: nome
Fim definicao
Atribuicao de valor para arranjo tipo masculino nome {componente
tipo masculino nome}+:
Atribuicao de valor para variavel tipo feminino nome {componente
tipo masculino nome}+: expressão na linha
```