Universidade Federal de Alagoas

Instituto de Computação
Curso de Ciência da Computação

Linguagem D

Especificação da Linguagem

Arthur Sávio Bernardo de Melo
João Messias Lima Pereira

Maceió 2020.1

Introdução

A linguagem de programação D foi criada com inspiração nas linguagens de programação C e Python, tendo como meta de utilização o ensino instrutivo da programação.
D não é orientada a objetos e é case-sensitive - diferencia letras maiúsculas de minúsculas -.

Partindo para legibilidade, D 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.

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

● Definições de procedimentos / funções

Só poderão ser declaradas fora do corpo de uma outra função ou procedimento e só podem ser acessadas se tiverem sua implementação ou assinatura declaradas antes da função chamadora. A linguagem também não aceita passagem de subprogramas como parâmetros.

A declaração de função é sempre iniciado 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 é feita seguindo o padrão de função, porém o tipo de retorno deve ser identificado como void.
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.:

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

```
tipoDoRetorno id ( listaDeParametros );
void id ( listaDeParametros );
```

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 e são listadas a seguir:

const, char, int, float, bool, string, bool, fun, void, scan, print, if, else, False, True, for, while, return.

● Definição de variáveis

D 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.

Múltiplas variáveis podem ser declaradas em uma declaração de tipo único sendo separadas por vírgula. Cada declaração termina com um ponto e vírgula e podem ser acompanhadas de uma atribuição desde que seja acompanhada de um símbolo de igualdade seguido do valor da atribuição. Esses valores de atribuição devem seguir o tipo da declaração.

Ex.:
int a = 1;
float b = 1.5;
float c = b;
char e;
chr f = 'y'
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 teste = 0;

Tipos e Estrutura de Dados

● Inteiro

É a identificação da variável como sendo do tipo inteiro de 32 bits, identificado pela palavra reservada int. A variável é expressa como 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. A variável é expressa 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.
Ex.:
int arr(2);
float arr(10);
bool arr(15);

● 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);

● Coerção

D não aceita coerção entre variáveis de
tipos diferentes. Dessa forma, todas as verificações de compatibilidade de tipo serão feitas estaticamente.

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)

● 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.

Instruções

● Atribuição

É definida pelo símbolo ‘=’, sendo o lado esquerdo o identificador único e o lado direito o valor ou expressão.

● Estrutura condicional

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

● Estrutura iterativa com controle lógico

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

● **Estrutura iterativa controlada por contador**

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

Os 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)) { while(var < 10) {
//código //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

D não aceita nenhum tipo de desvio incondicional.

Entrada e Saída

● **Entrada:** É realizada através da função scan().

O comportamento da função scan dependerá da variável a qual se está sendo lida. No caso de ser um inteiro, scan atribuirá um único

inteiro ao valor designado ao identificador, assim como floats, char, string,bool. A função scan também aceita vários parâmetros, atribuindo, assim, cada um dos valores recebidos como entrada para o código será atribuído às respectivas 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 identificador, assim como para floats, char, string, bool. Caso o valor seja relacionado a uma constante, print irá imprimir diretamente o valor dessa constante.

Códigos exemplo

- Hello world

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

- Fibonacci

```
int fib(int n)
{
	int fib1 = 1, fib2 = 1, soma;

	for (int i = 3; i <= n; i = i + 1)
	{
		soma = fib1 + fib2;
		fib1 = fib2;
		fib2 = soma;
	}

	return fib2;
}

int main()
{
	int n, fib_result;
	scan(n);

	fib_result = fib(n);

	print(fib_result);
	return 0;
}
```

- Shell Sort

```
void shellSort(int vet[], int size)
```

```
{
    int i, j, value;

    int h = 1;

    while (h < size) {
        h = 3 * h + 1;
    }

    while (h > 0) {
        for(i = h; i < size; i = i + 1) {
            value = vet[i];
            j = i;
            while (j > h-1 && value <= vet[j - h]) {
                vet[j] = vet[j - h];
                j = j - h;
            }
            vet[j] = value;
        }
        h = h/3;
    }
}

int main()
{
    int vet[] = { 3, 1, 2, 4 };

    shellSort(vet, 4);

    print(vet);

    return 0;
}
```