Descrição da Linguagem

CleBeN é uma linguagem de programação projetada para ser facilmente compreendida por humanos, utilizando uma sintaxe similar ao inglês falado. Seu principal propósito é facilitar o aprendizado de programação para iniciantes, tornando o código mais intuitivo e menos intimidador. Além disso, CleBeN é suficientemente poderosa para implementar algoritmos básicos e realizar tarefas comuns de programação.

Índice

Introdução

CleBeN é uma linguagem de programação destinada a pessoas que estão dando seus primeiros passos na programação. Ela utiliza uma sintaxe simples e direta, semelhante ao inglês falado, para tornar o processo de escrita e leitura de código o mais intuitivo possível.

Atribuições e Declarações

A atribuição de valores a variáveis em CleBeN é feita utilizando a palavra-chave `assign`. Valores podem ser números, textos, nulos, listas ou intervalos.

```
assign number 12 to x
assign text "Hello, world" to greeting
assign null to y, z
```

```
assign list [] to l
assign range 1..20 to m
```

Operações Aritméticas e Lógicas

Operações aritméticas e lógicas podem ser realizadas e atribuídas a variáveis usando a palavra-chave `assign`.

```
assign x + y to z  # z = x + y
```

Estruturas de Controle

CleBeN suporta loops `for` e `while`.

- **Loop For**: Itera sobre intervalos ou listas.

```
for every i in 0..10 do
    print i
endfor

for every i in l do
    print i
endfor
```

- **Loop While**: Executa enquanto a condição for verdadeira.

```
while x eq y do
    print x
    decr y
endwhile
```

Funções

Funções são definidas usando a palavra-chave `function` e finalizadas com `endfunction`.

```
function add (number a, number b) does
    assign a + b to result
    return result
endfunction
```

Condicionais

Condicionais são usadas para tomar decisões no código, utilizando `if`, `then`, `elseif`, `else` e `endif`.

```
if x eq y then
    print "x is equal to y"
elseif y neq z then
    print "y is not equal to z"
else
    print "none of the above"
endif
```

Funções Builtin

CleBeN inclui várias funções builtin úteis para manipulação de dados e implementação de algoritmos simples.

- **lengthof(lista)**: Retorna o tamanho da lista.

```
assign lengthof(l) to len
```

- **append(item, lista)**: Adiciona um item ao final da lista.

```
append(5, l)
```

- **remove(item, lista)**: Remove o item da lista, se presente.

```
remove(5, l)
```

- **indexof(item, lista)**: Retorna o índice do item na lista, ou -1 se o item não estiver presente.

```
assign indexof(5, l) to idx
```

- **contains(item, lista)**: Retorna verdadeiro se a lista contiver o item.

```
if contains(5, l) then
    print "List contains 5"
endif
```

- **copy(list):** retorna uma cópia da lista

```
assign copy(l) to list_copy
print list_copy
```

- **print(value)**: Imprime o valor no console.

```
print "Hello, world"
print x
```

Sintaxe e Semântica

- **Operadores de Comparação**:
  - `eq` para `==`
  - `neq` para `≠`
  - `lt` para `<`
  - `gt` para `>`
  - `le` para `≤`
  - `ge` para `≥`
- **Incremento e Decremento**:
  - `incr` para incrementar
  - `decr` para decrementar
- **Indexação em Listas**: Similar ao Python, usando colchetes `[]`.

```
assign l[0] to first_element
```

Exemplo Completo

Função Bubble Sort em CleBeN usando For Loops

```
function bubble_sort (list l) does
    assign lengthof(l) to n

    for every i in 0..(n - 1) do
        for every j in 0..(n - i - 2) do
            if l[j] gt l[j + 1] then
                assign l[j] to temp
                assign l[j + 1] to l[j]
                assign temp to l[j + 1]
            endif
        endfor
    endfor

    return l
endfunction
```

Explicação da Função Bubble Sort

1. **Declaração da Função**:
   - A função `bubble_sort` recebe uma lista `l` como parâmetro.
2. **Inicialização de Variáveis**:
   - `n` é a variável que armazena o comprimento da lista `l`.
3. **Loop Externo**:
   - O loop `for every i in 0..(n - 1)` controla o número de passadas pelo array.
4. **Loop Interno**:
   - O loop `for every j in 0..(n - i - 2)` percorre a lista comparando e trocando elementos adjacentes se necessário.
   - A condição `j in 0..(n - i - 2)` garante que elementos já ordenados nas últimas posições não sejam reconsiderados.
5. **Troca de Elementos**:
   - Se `l[j] gt l[j + 1]`, os elementos são trocados usando uma variável temporária `temp`.
6. **Retorno da Lista Ordenada**:
   - A função retorna a lista `l` após o término da ordenação.

Exemplo de Uso

Aqui está um exemplo de como usar a função `bubble_sort` para ordenar uma lista em CleBeN:

```
assign [64, 34, 25, 12, 22, 11, 90] to unsorted_list
assign bubble_sort(unsorted_list) to sorted_list
print sorted_list
```

Este exemplo atribui uma lista não ordenada a `unsorted_list`, chama a função `bubble_sort` para ordenar a lista e armazena o resultado em `sorted_list`. Finalmente, imprime a lista ordenada.

CleBeN oferece uma abordagem amigável e intuitiva para a programação, ideal para iniciantes e para educação, mantendo a simplicidade sem sacrificar a funcionalidade básica necessária para a implementação de algoritmos.