

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light mint green. They are positioned diagonally, with the blue one partially covering the green one.

# Numscript

Carlos Dip



# Motivação

- Criar uma linguagem de programação
  - Muitas linguagens apresentam conceitos e nomes de funções numa língua específica (como inglês)
  - Número reduzido de caracteres, sacrificando o mínimo possível da legibilidade



# Proposta

- Linguagem somente com números e símbolos
  - Idioma universal ( símbolos da matemática )
  - Nenhuma letra



# Características

- Funcionalidades básicas de programação, variáveis, funções, loops, condicionais
- Toda variável e palavra reservada é precedida por um cifrão '\$'
- Sintaxe semelhante a C
- Todo programa deve conter um bloco na raiz, caracterizado pela abertura e fechamento de chaves '{ ... }'
- Somente variáveis do tipo inteiro, portanto não faz sentido declarar uma variável

## Palavras Reservadas:

Symbol	Python Equivalent
:-----:	:-----:
\$ 436 0	if
\$ 436 1	else
\$ 436 2	while
\$ 436 3	print
\$ 436 4	return
\$ _436	def



# Exemplos


Programa para somar dois números:

**C:**

```
int a, b;  
a = 10;  
b = 5;  
c = a + b;
```

**Numscript:**

```
{  
    $0 = 10;  
    $1 = 5;  
    $2 = $0 + $1;  
}
```




Programa que imprime 1 se uma variável for verdadeira (diferente de zero), ou zero caso contrário:

**C:**

```
int a;  
a = 1;  
if(a){  
    print(1);  
} else {  
    print(0);  
}
```

**Numscript:**

```
{  
    $0 = 1;  
    $_436_0($0){  
        $_436_3(1);  
    } $_436_1 {  
        $_436_3(0);  
    }  
}
```




Programa demonstrando escopo de variáveis, imprime variáveis globais e locais com mesmo nome de dentro de função:

**C:**

```
int a, b
a = 0;
b = 3;
void func() {
    int a;
    a = 1;
    printf(a);
    printf(b);
}
printf(a);
printf(b);
func();
```

**Numscript:**

```
{
    $1 = 0;
    $0 = 3;
    $_436 $101() {
        $0 = 1;
        $_436_3($0);
        $_436_3($1);
    }
    $_436_3($0);
    $_436_3($1);
    $101();
}
```



Função que multiplica suas duas entradas e retorna o resultado:


**C:**

```
void prod(int x, int y){  
    return x * y;  
}  
  
int a;  
a = prod(2, 10);  
printf(a);
```

**Numscript:**

```
{  
    $_436 $1008($0, $2){  
        $_436_4 $0 * $2 ;  
    }  
    $0 = $1008(2, 10);  
    $_436_3($0);  
}
```





Função recursiva que imprime todos os números naturais, começando da entrada, até 0, depois até a entrada, decrementando e depois incrementando de 1 em 1:

**C:**

```
void rec_func(int x){  
    if(x > 0){  
        printf(x);  
        rec_func(x - 1);  
    }  
    printf(x);  
}  
rec_func(10);
```

**Numscript:**

```
{  
    $_436 $1008($0){  
        $_436_0($0 > 0){  
            $_436_3($0);  
            $1008($0 - 1);  
        }  
        $_436_3($0);  
    }  
    $1008(10);  
}
```



# Obrigado

Código-Fonte:

<https://github.com/CEDipEngineering/Numscript>