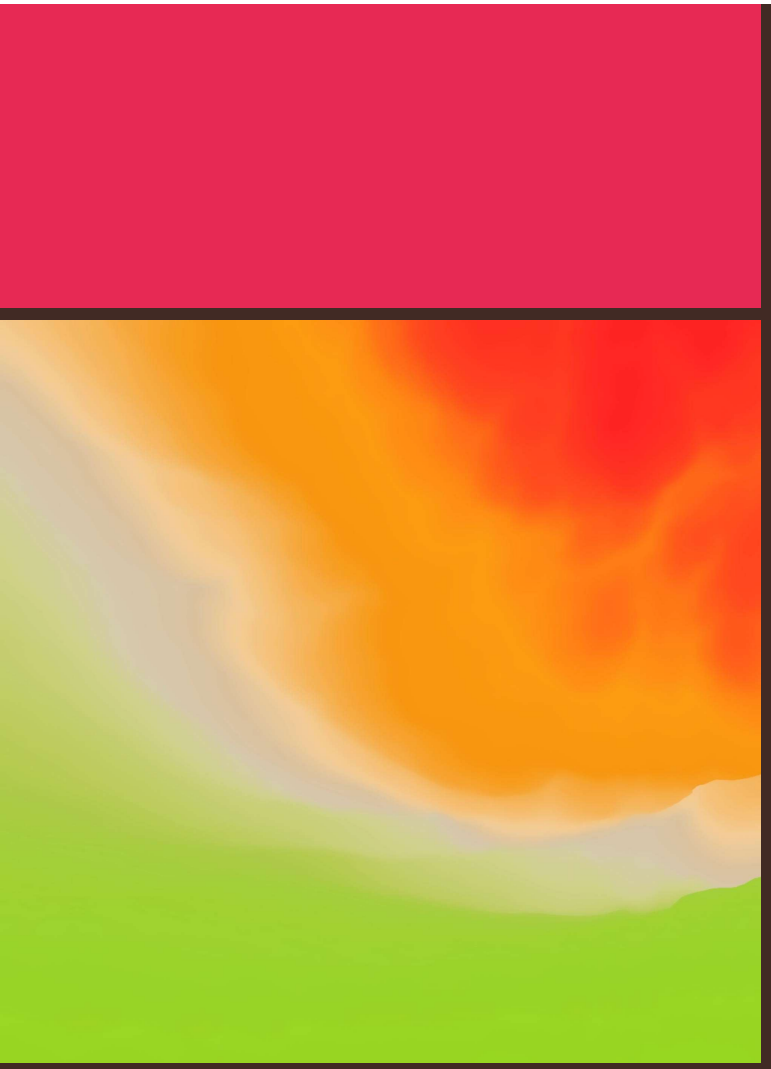

APS – LÓGICA DA COMPUTAÇÃO

Jonathan Sutton

Motivação

Com a missão de sempre tornar as novas tecnologias mais acessíveis, foi desenvolvida a 1a linguagem de programação da história com vocabulário em espanhol, visando aumentar o acesso e facilitar o aprendizado da população Latina na área de programação.

Golang

CARACTERÍSTICAS

A linguagem tem as mesmas características que Golang, mas com os tokens traduzidos para espanhol.

Por que Golang?

- **Simplicidade e Facilidade de Aprendizado**
- **Alto desempenho e eficiência**
- **Comunidade em crescimento**
- **Aplicações reais**

EBNF

```
PROGRAMA = { DECLARACION | FUNCION };

DECLARACION = ( λ | ASIGNACION | IMPRIMIR | CONTROL | COMENTARIO ), "¥n"
;

ASIGNACION = ["var"], IDENTIFICADOR, ["=", EXPRESION] ;

IMPRIMIR = "Imprimir", "(", EXPRESION, ")" ;

CONTROL = SI | BUCLE ;

SI = "si", "(", EXPRESION, ")", BLOQUE, [ "sino", BLOQUE ];

BUCLE = "mientras", "(", CONDICION_BUCLE, ")", BLOQUE ;

FUNCION = "func", IDENTIFICADOR, "(", [PARAMETROS], ")", [TIPO_RETORNO],
BLOQUE ;

PARAMETROS = IDENTIFICADOR, TIPO, { ",", IDENTIFICADOR, TIPO } ;

BLOQUE = "{", { DECLARACION }, "}" ;

EXPRESION = TERMINO, { ("+" | "-" | "&&" | "||" | "==" | "<" | ">" | "!"),
TERMINO } ;

TERMINO = FACTOR, { ("*" | "/" | "." | "!"), FACTOR } ;

FACTOR = (("+" | "-"), FACTOR) | NUMERO | "(", EXPRESION, ")" |
IDENTIFICADOR | CADENA ;

IDENTIFICADOR = LETRA, { LETRA | DIGITO | "_" } ;

NUMERO = DIGITO, { DIGITO } ;

CADENA = "¥"", { TODOS_EXCETO_COMILLA }, "¥"" ;

TIPO_RETORNO = "->", TIPO ;

TIPO = "ent" | "cadena" ;

CONDICION_BUCLE = EXPRESION, ";", EXPRESION, ";", EXPRESION ;

COMENTARIO = "//", { TODOS_EXCETO_SALTO_LINEA } ;

LETRA = ( a | ... | z | A | ... | Z ) ;

DIGITO = ( 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 ) ;
```

Exemplo de Código

.go

```go
func soma(x int, y int) int {
    return x + y
}

func read() int {
    return Scanln()
}

func concat(a string, b string) string {
    return a . b
}

func main() int {
    var x_1 int
    x_1 = soma(read()-1, 1)
    soma(2, 1)

    Println(x_1)

    if (x_1 > 1 && !!!(x_1 < 1)) || x_1 == 3 {
        x_1 = 2
    }


    var x int = 3+6/3   *  2 -+-  +  2*4/2 + 0/1 -((6+ ((
    var y_1 int = 3
    y_1 = soma(y_1, x_1)
    var z__ int
    z__ = soma(x, y_1)

    if x_1 == 2 {
        x_1 = 2
    }

    if x_1 == 3 {
        x_1 = 2
```

.goesp

```goesp

func suma(x ent, y ent) ent {
    devolver x + y
}

func leer() ent {
    devolver LeerLinea() //ler
}

func concat(a cadena, b cadena) cadena {
    devolver a . b
}

func main() ent {
    var x_1 ent
    x_1 = suma(leer()-1, 1)
    suma(2, 1)

    Imprimir(x_1)

    si (x_1 > 1 && !!!(x_1 < 1)) || x_1 == 3 {
        x_1 = 2
    }


    var x ent = 3+6/3   *  2 -+-  +  2*4/2 + 0/1 -((6+ ((
    var y_1 ent = 3
    y_1 = suma(y_1, x_1)
    var z__ ent
    z__ = suma(x, y_1)

    si x_1 == 2 {
        x_1 = 2
    }

    si x_1 == 3 {
        x_1 = 2
```