# Manual Completo Pine Script - TradingView (v6)

## 📚 Índice

1. [Introdução ao Pine Script](#introdução-ao-pine-script)

2. [Estrutura Básica](#estrutura-básica)

3. [Novos Recursos v6](#novos-recursos-v6)

4. [Exemplos Práticos](#exemplos-práticos)

5. [Referência Rápida](#referência-rápida)

6. [Recursos Avançados](#recursos-avançados)

## Introdução ao Pine Script

### O que é Pine Script?

- Linguagem proprietária do TradingView

- Criada especificamente para análise técnica e automação

- Versão atual: v6

- Sintaxe similar a Python, com recursos específicos para trading

- Suporte a programação orientada a objetos

- Sistema de tipos mais robusto

### Principais Usos

- Indicadores técnicos customizados

- Estratégias de trading automatizadas

- Sistemas de alerta avançados

- Backtesting de estratégias

- Análise multi-timeframe

- Detecção de padrões (SMC, Wyckoff, etc.)

- Visualização de dados avançada

- Integração com APIs externas

## Estrutura Básica

### Template Básico

```pinescript

//@version=6

indicator("Nome do Indicador", overlay=true)

// Configurações e Variáveis

var float minhaVar = 0.0

// Classes e Objetos (Novo em v6)

class TradeSetup

float entry

float stop

float target

method getRisk() => math.abs(entry - stop)

method getReward() => math.abs(target - entry)

// Lógica Principal

var setup = TradeSetup.new()

setup.entry := close

setup.stop := low

setup.target := high

// Plotagem

plot(setup.getRisk())

```

### Componentes Essenciais

#### 1. Declaração de Versão e Tipo

```pinescript

//@version=6

indicator("Nome", overlay=true) // Para indicadores

// ou

strategy("Nome", overlay=true) // Para estratégias

// ou

library("Nome") // Para bibliotecas

```

#### 2. Inputs e Parâmetros (Melhorado em v6)

```pinescript

// Grupos para organização

var string G\_INPUTS = "Configurações"

// Diferentes tipos de input com validação

i\_periodo = input.int(14, "Período",

group=G\_INPUTS,

minval=1,

maxval=500,

tooltip="Período para cálculos")

i\_fonte = input.source(close, "Fonte de Preço",

group=G\_INPUTS)

i\_mostrar = input.bool(true, "Mostrar Indicador",

group=G\_INPUTS)

// Novo em v6: Input de cor com opacidade

i\_cor = input.color(color.blue, "Cor",

inline="cor",

group=G\_INPUTS)

i\_opacidade = input.int(50, "Opacidade",

inline="cor",

minval=0,

maxval=100,

group=G\_INPUTS)

```

#### 3. Funções e Métodos (Melhorado em v6)

```pinescript

// Função com tipos explícitos

detectarPadrao(float src, int len) =>

float alto = ta.highest(src, len)

float baixo = ta.lowest(src, len)

[alto, baixo]

// Classe com métodos

class Indicador

float[] valores

float media

method calcularMedia() =>

float soma = array.sum(this.valores)

int tamanho = array.size(this.valores)

this.media := soma / tamanho

method adicionarValor(float valor) =>

array.push(this.valores, valor)

this.calcularMedia()

// Uso de classes

var ind = Indicador.new()

ind.adicionarValor(close)

```

## Novos Recursos v6

### 1. Programação Orientada a Objetos

```pinescript

// Definição de classe

class Position

float entry

float size

string direction

method getRisk(float stop) =>

float risco = direction == "long" ?

entry - stop : stop - entry

math.abs(risco \* size)

method getProfit(float target) =>

float lucro = direction == "long" ?

target - entry : entry - target

math.abs(lucro \* size)

// Herança

class AdvancedPosition : Position

float[] targets

float[] stops

method getTotalProfit() =>

float total = 0.0

for target in targets

total += this.getProfit(target)

total

```

### 2. Matrizes e Operações Matriciais

```pinescript

// Criação de matriz

matrix<float> precos = matrix.new<float>(5, 5, 0.0)

// Operações matriciais

matrix.set(precos, 0, 0, close)

float media = matrix.avg(precos)

float determinante = matrix.det(precos)

```

### 3. Mapas (Dicionários)

```pinescript

// Criação de mapa

map<string, float> niveis = map.new<string, float>()

// Operações com mapas

map.put(niveis, "suporte", low)

map.put(niveis, "resistencia", high)

float suporte = map.get(niveis, "suporte")

```

### 4. Melhorias em Arrays

```pinescript

// Arrays tipados

array<float> precos = array.new<float>()

array.push(precos, close)

// Novos métodos

array.median(precos)

array.mode(precos)

array.percentile\_linear\_interpolation(precos, 90)

```

### 5. Funções de Data e Hora Melhoradas

```pinescript

// Manipulação de datas

timestamp hora = timestamp(2024, 1, 1, 0, 0)

int ano = year(hora)

int mes = month(hora)

bool ehFimDeSemana = dayofweek(hora) >= 6

```

## Exemplos Práticos

### 1. Indicador Multi-Timeframe com SMC (v6)

```pinescript

//@version=6

indicator("Análise SMC Multi-Timeframe", overlay=true)

// Classes para organização

class SwingPoint

float price

string type

int bar\_index

method isValid() => not na(price)

method age() => bar\_index - bar\_index

class MarketStructure

SwingPoint[] swings

int max\_swings

method addSwing(float price, string type) =>

var swing = SwingPoint.new()

swing.price := price

swing.type := type

swing.bar\_index := bar\_index

if array.size(this.swings) >= this.max\_swings

array.shift(this.swings)

array.push(this.swings, swing)

// Parâmetros

var string G\_CONFIG = "Configurações"

i\_tf = input.timeframe("240", "Timeframe Superior", group=G\_CONFIG)

// Inicialização

var ms = MarketStructure.new()

ms.max\_swings := 10

// Funções SMC modernizadas

detectarBOS(MarketStructure ms) =>

var lastSwings = array.slice(ms.swings, -2)

if array.size(lastSwings) < 2

[false, false]

else

bullish = lastSwings[1].price > lastSwings[0].price and lastSwings[1].type == "low"

bearish = lastSwings[1].price < lastSwings[0].price and lastSwings[1].type == "high"

[bullish, bearish]

// Dados do timeframe superior

[htf\_high, htf\_low] = request.security(syminfo.tickerid, i\_tf, [high, low], cache\_size=20)

// Detecção e plotagem

[bullishBOS, bearishBOS] = detectarBOS(ms)

if bullishBOS

label.new(bar\_index, low, "BOS\nBULL",

style=label.style\_label\_up,

color=color.green,

textcolor=color.white)

if bearishBOS

label.new(bar\_index, high, "BOS\nBEAR",

style=label.style\_label\_down,

color=color.red,

textcolor=color.white)

```

### 2. Sistema de Alertas Avançado com Webhooks (v6)

```pinescript

//@version=6

indicator("Sistema de Alertas v6", overlay=true)

// Classe para gerenciar sinais

class Signal

float price

string type

string message

method format() =>

str.format(

'{{"price": {0}, "type": "{1}", "message": "{2}"}}',

str.tostring(this.price),

this.type,

this.message)

// Configurações

var signal = Signal.new()

// Condições usando médias móveis otimizadas

fastMA = ta.ema(close, 20)

slowMA = ta.ema(close, 50)

crossOver = ta.crossover(fastMA, slowMA)

crossUnder = ta.crossunder(fastMA, slowMA)

// Geração de sinais

if crossOver

signal.price := close

signal.type := "BUY"

signal.message := "Cruzamento de Média Móvel para cima"

alert(signal.format(), alert.freq\_once\_per\_bar\_close)

if crossUnder

signal.price := close

signal.type := "SELL"

signal.message := "Cruzamento de Média Móvel para baixo"

alert(signal.format(), alert.freq\_once\_per\_bar\_close)

// Interface Visual Moderna

var table infoTable = table.new(position.top\_right, 3, 3,

bgcolor=color.new(color.black, 70),

border\_width=1)

if barstate.islast

table.cell(infoTable, 0, 0, "Status",

bgcolor=color.new(color.gray, 70),

text\_size=size.small)

table.cell(infoTable, 1, 0,

crossOver ? "COMPRA" : crossUnder ? "VENDA" : "NEUTRO",

bgcolor=crossOver ? color.new(color.green, 70) :

crossUnder ? color.new(color.red, 70) :

color.new(color.gray, 90),

text\_size=size.small)

```

### 3. Biblioteca de Funções SMC (v6)

```pinescript

//@version=6

library("SMCLibrary", overlay=true)

// Tipos personalizados

type SwingPoint

float price

int index

string direction

type Structure

SwingPoint[] points

string trend

float[] levels

// Funções exportadas

export detectSwing(float[] highs, float[] lows, int length) =>

SwingPoint result = na

if ta.pivothigh(highs, length, length)

result := SwingPoint.new(

high[length],

bar\_index[length],

"high")

if ta.pivotlow(lows, length, length)

result := SwingPoint.new(

low[length],

bar\_index[length],

"low")

result

export detectStructure(SwingPoint[] points) =>

var structure = Structure.new()

if array.size(points) >= 2

last = array.get(points, -1)

prev = array.get(points, -2)

if last.price > prev.price

structure.trend := "up"

else

structure.trend := "down"

structure

// Funções auxiliares internas

method SwingPoint.isValid() =>

not na(this.price) and this.index > 0

method Structure.addLevel(float level) =>

if not array.includes(this.levels, level)

array.push(this.levels, level)

// Constantes

export var SWING\_MIN\_LENGTH = 5

export var SWING\_MAX\_LENGTH = 50

```

## Referência Rápida

### 🎨 Funções de Visualização (v6)

```pinescript

// Plotagem Avançada

chart.bg(color, opacity) // Fundo personalizado

chart.overlay(true|false) // Sobrepor ao gráfico

chart.format("{0:.2f}") // Formatação de números

// Elementos Visuais

label.new(

x=bar\_index,

y=high,

text="Topo",

style=label.style\_label\_down,

textcolor=color.white)

box.new(

left=bar\_index[10],

top=high[10],

right=bar\_index,

bottom=low,

bgcolor=color.new(color.blue, 90))

// Tabelas Modernas

table.new(

position=position.top\_right,

columns=2,

rows=2,

bgcolor=color.new(color.black, 90),

border\_width=1)

```

### 📊 Funções Técnicas Otimizadas (v6)

```pinescript

// Análise Técnica

ta.variance(source, length) // Variância

ta.correlation(x, y, length) // Correlação

ta.percentile\_nearest\_rank(source, length, percent) // Percentil

// Manipulação de Dados

array.slice(arr, start, end) // Cortar array

matrix.submatrix(m, row, col, rows, cols) // Submatriz

map.values(map) // Valores do mapa

// Estatísticas

math.sum(array) // Soma de array

math.abs(value) // Valor absoluto

math.round\_to\_mintick(value) // Arredondar para tick

```

### ⚙️ Funções de Controle e Otimização (v6)

```pinescript

// Variáveis Tipadas

var float myVar = 0.0

var int[] myArray = array.new<int>()

var map<string, float> myMap = map.new<string, float>()

// Estruturas de Controle

for i = 0 to array.size(arr) - 1

if array.get(arr, i) > 0

// código

switch str

"A" => // código

"B" => // código

=> // default

// Otimização

@function // Cache de função

method cached() => // Cache de método

```

## Recursos Avançados

### 1. Programação Orientada a Objetos em Pine Script v6

```pinescript

// Definição de classe base

class Indicator

float[] values

float current

string name

method calculate() =>

this.current := array.avg(this.values)

method addValue(float value) =>

array.push(this.values, value)

this.calculate()

// Herança

class RSI : Indicator

int period

float overbought

float oversold

method isOverbought() =>

this.current > this.overbought

method isOversold() =>

this.current < this.oversold

// Uso das classes

var rsi = RSI.new()

rsi.period := 14

rsi.overbought := 70

rsi.oversold := 30

rsi.addValue(close)

```

### 2. Manipulação Avançada de Dados

```pinescript

// Matrizes multidimensionais

matrix<float> precos = matrix.new<float>(10, 5)

for i = 0 to 9

matrix.set(precos, i, 0, open[i])

matrix.set(precos, i, 1, high[i])

matrix.set(precos, i, 2, low[i])

matrix.set(precos, i, 3, close[i])

matrix.set(precos, i, 4, volume[i])

// Operações matriciais

matrix<float> correlacao = matrix.correlation(precos)

float determinante = matrix.det(correlacao)

// Mapas para cache de dados

var map<int, float> cache = map.new<int, float>()

map.put(cache, bar\_index, close)

float valorAnterior = map.get(cache, bar\_index - 1)

```

### 3. Otimização de Performance

```pinescript

// Cache de função com decorador

@function

calcularIndicador(float[] dados) =>

var resultado = 0.0

if barstate.isfirst

resultado := array.avg(dados)

resultado

// Vetorização de operações

method Indicator.calculateBatch(float[] valores) =>

array.concat(this.values, valores)

this.current := array.avg(this.values)

// Uso eficiente de memória

var int MAX\_SIZE = 100

if array.size(valores) > MAX\_SIZE

array.slice(valores, array.size(valores) - MAX\_SIZE)

```

### 4. Integração com APIs Externas

```pinescript

// Formatação de dados para webhook

formatarSinal(float preco, string tipo) =>

str.format(

'{{"timestamp": {0}, "price": {1}, "type": "{2}", "symbol": "{3}"}}',

str.tostring(timestamp("UTC")),

str.tostring(preco),

tipo,

syminfo.ticker

)

// Sistema de alertas avançado

var sinais = array.new<string>()

if conditionBuy

var sinal = formatarSinal(close, "BUY")

array.push(sinais, sinal)

alert(sinal, alert.freq\_once\_per\_bar\_close)

```

### 5. Visualização de Dados Moderna

```pinescript

// Tabelas interativas

var table dadosTable = table.new(

position=position.top\_right,

columns=3,

rows=5,

bgcolor=color.new(color.black, 90),

border\_width=1

)

// Atualização dinâmica

if barstate.islast

table.cell(dadosTable, 0, 0, "Indicador",

bgcolor=color.new(color.gray, 70))

table.cell(dadosTable, 1, 0, str.tostring(close),

bgcolor=color.new(color.blue, 90))

// Elementos visuais modernos

box.new(

left=bar\_index[10],

top=high[10],

right=bar\_index,

bottom=low,

bgcolor=color.new(color.blue, 90),

border\_color=color.blue,

border\_width=2

)

```

### 6. Boas Práticas de Desenvolvimento

1. \*\*Organização do Código\*\*

- Use classes para agrupar funcionalidades relacionadas

- Mantenha métodos pequenos e focados

- Documente funções e classes importantes

2. \*\*Gestão de Recursos\*\*

- Limite o uso de arrays e matrizes grandes

- Use cache para cálculos repetitivos

- Limpe dados antigos periodicamente

3. \*\*Tratamento de Erros\*\*

- Verifique valores nulos

- Valide parâmetros de entrada

- Use valores padrão seguros

4. \*\*Otimização\*\*

- Prefira operações vetorizadas

- Use cache quando apropriado

- Minimize alocações de memória

5. \*\*Manutenibilidade\*\*

- Mantenha documentação atualizada

- Use nomes descritivos

- Siga padrões de codificação

## Links Úteis

- [Documentação Pine Script v6](https://www.tradingview.com/pine-script-docs/en/v6/)

- [Referência de Funções](https://www.tradingview.com/pine-script-reference/v6/)

- [Comunidade TradingView](https://www.tradingview.com/chat/#pine)

- [Blog Pine Script](https://www.tradingview.com/blog/en/category/pine-script/)

- [Exemplos de Estratégias Pine Script v6](https://pineify.app/resources/blog/pine-script-v6-strategy-examples)

- [Guia do Editor Pine Script](https://pineify.app/resources/blog/tradingview-pine-script-editor-guide)

- [Tudo sobre Pine Script v6](https://pineify.app/resources/blog/pine-script-v6-everything-you-need-to-know)

## Recursos Adicionais

### Ferramentas de Desenvolvimento

- \*\*Pineify\*\*: Plataforma para geração de código Pine Script sem programação

- \*\*Editor TradingView\*\*: Editor oficial com suporte a syntax highlighting

- \*\*Comunidade de Desenvolvedores\*\*: Fórum para troca de experiências

### Documentação Complementar

- Manuais em PDF

- Exemplos práticos de código

- Tutoriais passo a passo

- Guias de migração de versões anteriores

### Recursos de Aprendizado

- Cursos online

- Webinars

- Workshops

- Material de referência

## Estratégias Práticas em Pine Script v6

### 1. Estratégia de Cruzamento de Médias Móveis

```pinescript

//@version=6

strategy("[Estratégia] Cruzamento de Médias Móveis", overlay=true)

// Parâmetros

fastLength = input(9, "Período MA Rápida")

slowLength = input(21, "Período MA Lenta")

// Cálculo das médias

fastMA = ta.sma(close, fastLength)

slowMA = ta.sma(close, slowLength)

// Plotagem

plot(fastMA, "MA Rápida", color=color.green)

plot(slowMA, "MA Lenta", color=color.red)

// Sinais de entrada

if (ta.crossover(fastMA, slowMA))

strategy.entry("Compra", strategy.long)

if (ta.crossunder(fastMA, slowMA))

strategy.close("Compra")

```

### 2. Estratégia RSI com Filtro de Tendência

```pinescript

//@version=6

strategy("[Estratégia] RSI com Filtro de Tendência", overlay=true)

// Configurações

rsiPeriod = input(14, "Período RSI")

rsiOverbought = input(70, "Nível Sobrecomprado")

rsiOversold = input(30, "Nível Sobrevendido")

trendPeriod = input(200, "Período Tendência")

// Indicadores

rsi = ta.rsi(close, rsiPeriod)

trendMA = ta.sma(close, trendPeriod)

// Condições

longCondition = rsi < rsiOversold and close > trendMA

shortCondition = rsi > rsiOverbought and close < trendMA

// Execução

if (longCondition)

strategy.entry("Compra RSI", strategy.long)

if (shortCondition)

strategy.entry("Venda RSI", strategy.short)

```

### 3. Estratégia de Volume com Smart Money Concepts

```pinescript

//@version=6

strategy("[Estratégia] Volume SMC", overlay=true)

// Classes para análise

class VolumeProfile

float[] volumes

float[] prices

float poc // Point of Control

method addBar(float price, float vol) =>

array.push(this.prices, price)

array.push(this.volumes, vol)

this.updatePOC()

method updatePOC() =>

int maxIndex = array.indexof(this.volumes, array.max(this.volumes))

this.poc := array.get(this.prices, maxIndex)

// Configurações

var profile = VolumeProfile.new()

lookback = input(20, "Período de Análise")

// Atualização do perfil

if barstate.islast

profile.addBar(close, volume)

// Lógica de trading

float volumeMA = ta.sma(volume, lookback)

bool highVolume = volume > volumeMA \* 1.5

bool priceAbovePOC = close > profile.poc

if (highVolume and priceAbovePOC)

strategy.entry("Compra Volume", strategy.long)

if (highVolume and not priceAbovePOC)

strategy.close("Compra Volume")

```

### 4. Estratégia Multi-Timeframe com Gestão de Risco

```pinescript

//@version=6

strategy("[Estratégia] Multi-Timeframe", overlay=true)

// Classes

class RiskManager

float maxRisk

float accountSize

method getPositionSize(float entryPrice, float stopPrice) =>

float risk = math.abs(entryPrice - stopPrice)

float maxLoss = this.accountSize \* (this.maxRisk / 100)

math.floor(maxLoss / risk)

// Configurações

var rm = RiskManager.new()

rm.maxRisk := input(1.0, "Risco Máximo %")

rm.accountSize := input(10000.0, "Tamanho da Conta")

htfPeriod = input.timeframe("240", "Timeframe Superior")

ltfPeriod = input.timeframe("15", "Timeframe Inferior")

// Análise Multi-timeframe

[htf\_high, htf\_low] = request.security(syminfo.tickerid, htfPeriod, [high, low])

[ltf\_close, ltf\_volume] = request.security(syminfo.tickerid, ltfPeriod, [close, volume])

// Lógica de entrada

float stopLoss = low[1]

float entryPrice = close

float positionSize = rm.getPositionSize(entryPrice, stopLoss)

if (ltf\_close > htf\_high and ltf\_volume > ta.sma(volume, 20))

strategy.entry("Compra MTF", strategy.long, qty=positionSize)

strategy.exit("Saída", "Compra MTF", stop=stopLoss)

```

### 5. Dicas de Otimização de Estratégias

1. \*\*Gestão de Memória\*\*

- Use `var` para variáveis que não mudam a cada barra

- Limite o tamanho de arrays e matrizes

- Limpe dados históricos desnecessários

2. \*\*Performance\*\*

- Prefira operações vetorizadas

- Use cache para cálculos repetitivos

- Minimize o uso de loops

3. \*\*Robustez\*\*

- Teste em diferentes timeframes

- Valide em diferentes mercados

- Implemente gestão de risco

4. \*\*Boas Práticas\*\*

- Documente seu código

- Use nomes descritivos

- Mantenha o código organizado

- Implemente tratamento de erros

---

*\*Este manual é atualizado regularmente com as últimas funcionalidades do Pine Script v6 e mantém compatibilidade com as melhores práticas da comunidade.\**