# SIENGE

## Cobrança Escritural - Fórmulas

Manual de referência para a criação de fórmulas na cobrança escritural

# Sumário


Introdução ..... 3
Regras e limitações ..... 3
Exemplo de fórmula básica ..... 3
API ..... 3
O objeto cobranca ..... 3
Cobrança ..... 3
Cedente ..... 5
Cliente ..... 6
Avalista ..... 6
Os parâmetros de conta ..... 7
parametrosConta ..... 7
Funcionamento ..... 7
Utilização ..... 8
Chamando fórmulas ..... 8
chamarFormula ..... 8
Funcionamento ..... 8
Utilização ..... 8
Funções especiais ..... 8
funcoes ..... 8
texto ..... 9
cortar ..... 9
truncar ..... 9
cortarInicioPosicao ..... 9
cortarInicioTamanho ..... 9
preencherInicio ..... 10
cortarOuPreencherInicio ..... 10
preencher ..... 10
tratarNulo ..... 11
numero ..... 11
formatar ..... 11
formatarDinheiro ..... 11
arredondar ..... 11
arredondarABNT ..... 12

somenteDigitos ..... 12
removerZerosIniciais ..... 12
numeroAleatorio ..... 12
data ..... 12
formatarData ..... 12
dataAtual ..... 13
somarDias ..... 13
subtrairDias ..... 13
igualOuDepois ..... 14
calcularFatorVencimento ..... 14
moduloDez ..... 14
moduloOnze ..... 15
calcularModuloOnzeInvertido ..... 15
calcularModuloOnzeZeroOuResto ..... 15
calcularModuloOnzeZeroOuRestoComSequencia ..... 15
calcularModuloOnzeInvertidoZeroOuRestoComSequencia ..... 16
calcularModuloOnzeVariavel ..... 16
calcularModuloCem ..... 16

# Introdução

A geração de arquivo de remessa e boletos do Sienge utiliza três formatos de preenchimento de campos: **fixo**, **parâmetro de conta** e **fórmula**. Este documento trata do último.

As fórmulas são utilizadas para preencher campos que precisam de algum processamento ou cálculo. Elas são programadas na linguagem *javascript*, e possuem uma API própria além dos recursos padrão da linguagem.

## Regras e limitações

- Uma nova fórmula é executada a cada campo, e os dados não são compartilhados entre cada execução;
- Todas as fórmulas devem ser funções puras, ou seja, devem sempre retornar o mesmo valor se os mesmos parâmetros forem informados;
- Não são todos recursos da linguagem *javascript* que estão disponíveis na API. Estruturas muito complexas, lambdas e ES6, por exemplo, devem ser evitados.
- Toda fórmula deve obrigatoriamente conter uma função chamada *formula* que não recebe nenhum argumento.
- Toda fórmula deve retornar um valor.

## Exemplo de fórmula básica

```
function formula() {
    return parametrosConta.texto('enviarEmail') == 'S'
            ? cobranca.cliente.email
            : '';
}
```

# API

A *api* de fórmulas é composta por 4 objetos:

- **cobranca**: objeto que contém os dados da geração (título, parcela, cliente, cálculos);
- **parametrosConta**: parâmetros da conta corrente que foram configurados pelo usuário;
- **chamarFormula**: permite executar outras fórmulas e obter o resultado;
- **funcoes**: conjunto de funções úteis para realizar tarefas comuns como recorte de dados, conversão de moeda e cálculo de dígito verificador.

# O objeto cobranca

## Cobrança

Acesso: cobranca

O objeto *cobranca* possui todos os dados necessários para geração de uma linha do arquivo de remessa ou um boleto. Os dados são coletados de módulos do sistema (como informações de título, parcela, cliente, etc), das configurações da conta corrente utilizada na geração e dos cálculos realizados de acordo com as opções selecionadas no momento de gerar a cobrança.

| Valor | Tipo | Descrição | Origem |
|---|---|---|---|
| **cedente** | Cedente | Informações do cedente. | Empresa do módulo Apoio |
| **cliente** | Cliente | Informações do cliente. | Cliente do módulo Apoio |
| **avalista** | Avalista | Informações do avalista. | Avalista do módulo Comercial |
| **rateios** | List<Rateio> | Rateios de crédito | Configuração de Rateios |
| **mensagemBoleto** | String | Mensagem do boleto. | Configuração da conta |
| **mensagemArquivo** | String | Mensagem do arquivo de remessa, com as quebras de linha removidas. | Configuração da conta |
| **mensagemArquivoMultiLinha** | String | Mensagem do arquivo de remessa, sem remover as quebras de linha. | Configuração da conta |
| **codigoBanco** | String | Código do banco. | Layout |
| **codigoBancoCorrespondente** | String | Código do banco correspondente. | Layout |
| **localPagamento** | String | Local de pagamento do boleto. | Configuração da conta |
| **utilizaInfoTitulo** | boolean | Se *true*, deve-se buscar valores de juros, multa e desconto configurados no cadastro do título (no próprio objeto *cobranca*) em vez dos parâmetros da conta. | Configuração da conta |
| **valorTotal** | Number (double) | Valor total da parcela. | Calculado |
| **saldoDevedorOriginal** | Number (double) | Valor do saldo devedor original. | Calculado |
| **dataVencimento** | Date | Data de vencimento da parcela. | Parcela |
| **nossoNumeroBase** | String | Nosso número gerado originalmente na cobrança, utilizado na segunda via. É sempre vazio na geração de remessas novas. | Cobrança |
| **nossoNumeroSequencial** | Number (long) | Valor sequencial do nosso número, autoincrementado. | Configuração da conta |
| **percentJuroMora** | Number (double) | Percentual de juros. | Título |

| | | | |
|---|---|---|---|
| **percentMultaMora** | Number (double) | Percentual de multa. | Título |
| **percentDesconto** | Number (double | Percentual de desconto. | Título |
| **tipoDesconto** | String | Tipo de desconto:<br>M – Mensal<br>D – Diário<br>F – Fixo<br>A – Anual | Título |
| **valorMoraDia** | Number (double) | Valor de mora ao dia. | Título |
| **sequencialRemessa** | Number (int) | Sequencial de remessa. | Configuração da conta |
| **codigoMovimento** | String | Tipo de movimento. | Geração de remessa |
| **seuNumero** | String | Seu número. Utilizado para identificar a remessa no retorno. | Geração de remessa |
| **qtdTotalParcelas** | Number (int) | Quantidade de parcelas agrupadas. | Calculado |
| **valorTotalParcelas** | Number (double) | Valor total das parcelas agrupadas. | Calculado |
| **dataGeracao** | Date | Data de geração da remessa. | Calculado |
| **dataEmissao** | Date | Data de emissão da remessa. | Geração de remessa |
| **dataEmissaoTitulo** | Date | Data de emissão do título. | Título |
| **nomeCentroCusto** | String | Nome do centro de custo. | Centro de custo |
| **lote** | Number (int) | Número do lote. | Calculado |
| **linhaArquivo** | Number (int) | Número da linha no arquivo. | Calculado |
| **sequencialLote** | Number (int) | Número sequencial do lote. | Calculado |
| **sequencialParcela** | Number (int) | Número sequencial do lote. | Calculado |
| **documentoBoleto** | String | Número de documento do boleto. | Configuração da conta |
| **documentoRemessa** | String | Número de documento da remessa. | Configuração da conta |
| **titulo** | Number (int) | Número do título. | Título |
| **parcela** | Number (int) | Número da parcela. | Parcela |
| **tipoBoleto** | String | Tipo de boleto:<br>A – Avulso<br>C – Carnê | Geração de remessa |

## Cedente

Acesso: cobranca.cedente

O objeto *cedente* contém os dados da empresa utilizada para gerar a remessa.

| Variável | Tipo | Descrição |
|---|---|---|
| **cnpj** | String | CNPJ da empresa, formatado. |
| **cnpjSemMascara** | String | CNPJ da empresa, somente números. |
| **nome** | String | Nome da empresa. |
| **bairro** | String | Bairro da empresa. |
| **cep** | String | CEP da empresa, formatado. |
| **cepSemMascara** | String | CEP da empresa, somente números. |
| **municipio** | String | Município da empresa. |
| **endereco** | String | Endereço da empresa. |
| **numeroEndereco** | String | Número do endereço da empresa. |
| **complementoEndereco** | String | Complemento do endereço da empresa. |
| **uf** | String | Estado da empresa. |
| **tipoCpfCnpj** | String | Tipo da identificação do cedente. O Sienge possui apenas cedentes do tipo Pessoa Jurídica então este campo sempre é preenchido com o valor fixo 2. |

## Cliente

Acesso: cobranca.cliente

O objeto *cliente* contém os dados do cliente.

| Variável | Tipo | Descrição |
|---|---|---|
| **nome** | String | Nome do cliente. |
| **cpfCnpj** | String | CPF/CNPJ do cliente, formatado. |
| **cpfCnpjSemMascara** | String | CPF/CNPJ do cliente, somente números. |
| **tipoCpfCnpj** | String | Tipo do documento do cliente:<br>1 – Pessoa Física (CPF)<br>2 – Pessoa Jurídica (CNPJ) |
| **bairro** | String | Bairro do cliente. |
| **celular** | String | Número de telefone celular do cliente. |
| **cep** | String | CEP do cliente, somente números. |
| **cepFormatado** | String | CEP do cliente, formatado. |
| **municipio** | String | Município do cliente. |
| **endereco** | String | Endereço do cliente. |
| **numeroEndereco** | String | Número do endereço do cliente. |
| **complementoEndereco** | String | Completemento do endereço do cliente. |
| **uf** | String | Estado do cliente. |
| **email** | String | E-mail do cliente. |

## Avalista

Acesso: cobranca.avalista

O objeto *avalista* contém os dados do avalista.

| Variável | Tipo | Descrição |
|---|---|---|
| **nome** | String | Nome do avalista. |
| **cpfCnpj** | String | CPF/CNPJ do avalista, formatado. |
| **cpfCnpjSemMascara** | String | CPF/CNPJ do avalista, somente números. |

| | | |
|---|---|---|
| **tipoInscricao** | String | Tipo do documento do avalista:<br>1 – Pessoa Física (CPF)<br>2 – Pessoa Jurídica (CNPJ) |
| **bairro** | String | Bairro do avalista. |
| **endereco** | String | Endereço do avalista. |
| **cep** | String | CEP do avalista, somente números. |
| **cepFormatado** | String | CEP do avalista, formatado. |
| **municipio** | String | Município do avalista. |
| **uf** | String | Estado do avalista. |

### Rateios

Acesso: cobranca.rateios[i];

A lista do objeto *rateio* contém os dados do rateio.

| Variável | Tipo | Descrição |
|---|---|---|
| **beneficiario** | String | Nome do beneficiário do rateio. |
| **percentualRateio** | Number (double) | Percentual do Rateio. |
| **floating** | Number (int) | CPF/CNPJ do avalista, somente números. |
| **agencia** | String | Agencia do beneficiário. |
| **digitoAgencia** | String | Digito da Agencia do beneficiário. |
| **conta** | String | Conta do Beneficiário. |
| **digitoConta** | String | Dígito da Conta do Beneficiário. |
| **parcela** | Number (int) | Codigo da parcela. |

## Os parâmetros de conta

Os parâmetros de conta são valores informados pelo usuário na tela de configuração da conta corrente. Eles são configurados dentro da tela de configuração de layout de cobrança.

### parametrosConta

Acesso: parametrosConta

| Função | Retorno | Descrição |
|---|---|---|
| **texto** | String | Obtém o parâmetro como texto. |
| **inteiro** | Number (int) | Obtém o parâmetro como número inteiro. Corta todas as casas decimais sem arredondar. |
| **decimal** | Number (double) | Obtém o parâmetro como número decimal. |

### Funcionamento

O retorno das funções do objeto *parametrosConta* é convertido de acordo com a função utilizada. A função *inteiro* corta todas as casas decimais para retornar apenas o valor inteiro do parâmetro.

Utilize sempre a função *texto* se quiser preservar zeros à esquerda. Se um valor não numérico for lido pelas funções *inteiro* ou *decimal*, um erro será emitido.

### Utilização

Deve ser passada uma String contendo o nome do parâmetro na chamada da função:

- let carteira = parametrosConta.texto('carteira');
- let dias = parametrosConta.inteiro('diasDesconto');
- let taxa = parametrosConta.decimal('taxaJuros');

## Chamando fórmulas

É possível executar uma fórmula dentro de outra fórmula e utilizar o resultado no processamento. O objeto *chamarFormula* é responsável por esta execução.

### chamarFormula

Acesso: chamarFormula

| Função | Retorno | Descrição |
|---|---|---|
| **texto** | String | Executa a fórmula e obtém o resultado como texto. |
| **inteiro** | Number (int) | Executa a fórmula e obtém o resultado como número inteiro. |
| **decimal** | Number (double) | Executa a fórmula e obtém o resultado como número decimal. |

### Funcionamento

O retorno das chamadas de fórmula é convertido de acordo com a função utilizada. Utilize sempre a função *texto* se quiser preservar zeros à esquerda. Se um valor não numérico for lido pelas funções *inteiro* ou *decimal* ou um valor não-inteiro pela função *inteiro*, um erro será emitido.

### Utilização

Deve ser passada uma String contendo o nome da fórmula na chamada da função:

- let nossoNumero  = chamarFormula.texto('nossoNumero');
- let dias = chamarFormula.inteiro('diasDesconto');
- let valorJuros = chamarFormula.decimal('vlJuro');

## Funções especiais

Por meio do objeto *funcoes*, é possível executar ações comuns como manipulação de texto, números e datas, ou cálculo de módulo para dígito verificador.

### funcoes

Acesso: funcoes

| Valor | Descrição |
|---|---|
| **texto** | Funções para manipulação de texto como corte e preenchimento. |
| **numero** | Funções para manipulação de números, como formatação e arredondamento. |
| **data** | Funções para manipulação de datas, como formatação, comparação e adição. |
| **moduloDez** | Funções para cálculo de dígito verificador utilizando módulo 10. |
| **moduloOnze** | Funções para cálculo de dígito verificador utilizando módulo 11. |

## texto

Acesso: funcoes.texto

### cortar

Acesso: funcoes.texto.cortar(textoOriginal, posicaoInicial, posicaoFinal)

Corta um texto nas posições informadas.

Parâmetros:

- textoOriginal: O texto a ser cortado.
- posicaoInicial: A posição inicial, zero-based, do início do texto resultante.
- posicaoFinal: A posição final de corte do texto, zero-based. O caractere situado nesta posição não será incluído no resultado.

### truncar

Acesso: funcoes.texto.truncar(textoOriginal, tamanho)

Corta o final de um texto de acordo com o tamanho desejado.

Parâmetros:

- textoOriginal: O texto a ser cortado.
- tamanho: O tamanho final do texto. Caso seja maior que o texto original, o texto original é retornado.

### cortarInicioPosicao

Acesso: funcoes.texto.cortarInicioPosicao(textoOriginal, posicaoInicio)

Corta o início de um texto a partir da posição informada.

Parâmetros:

- textoOriginal: O texto a ser cortado.
- posicaoInicio: A posição inicial de corte do texto, zero-based.

### cortarInicioTamanho

Acesso: funcoes.texto.cortarInicioTamanho(textoOriginal, tamanho)

Corta o início de um texto de forma que o texto resultante fique do tamanho desejado.

Parâmetros:

- textoOriginal: O texto a ser cortado.
- tamanho: O tamanho final do texto, contando a partir do último caractere.

### preencherInicio

Acesso: funcoes.texto.preencherInicio(textoOriginal, preenchimento, tamanhoMinimo)

Preenche o início de um texto com os caracteres desejados, até que ele atinja ou ultrapasse o tamanho mínimo desejado.

Parâmetros:

- textoOriginal: O texto a ser preenchido.
- preenchimento: O texto a ser utilizado como preenchimento.
- tamanhoMinimo: Tamanho mínimo final do texto resultante. Se o tamanho mínimo for menor que o tamanho original do texto, nada será feito. Se o preenchimento conter mais de um caractere, é possível que o resultado final seja maior que o tamanho mínimo. Se este não for o comportamento desejado deve ser utilizada a função cortarOuPreencherInicio em vez desta.

### cortarOuPreencherInicio

Acesso: funcoes.texto.cortarOuPreencherInicio(textoOriginal, preenchimento, tamanhoFinal)

Preenche o início de um texto com os caracteres desejados, até que ele atinja o tamanho final desejado, cortando o início do texto caso seja necessário.

Parâmetros:

- textoOriginal: O texto a ser preenchido.
- preenchimento: O texto a ser utilizado como preenchimento.
- tamanhoMinimo: Tamanho final do texto resultante. Se o tamanho mínimo for menor que o tamanho original do texto, ele será apenas cortado. Se o preenchimento conter mais de um caractere e ultrapassar o tamanho final, o texto resultante será cortado no início para atingir o tamanho desejado.

### preencher

Acesso: funcoes.texto.preencher(textoOriginal, preenchimento, posicaoInicial)

Preenche o texto na posição informada com o preenchimento.

Parâmetros:

- textoOriginal: O texto a ser preenchido.
- preenchimento: O texto a ser utilizado como preenchimento.
- posicaoInicial: Posição inicial, zero-based, onde o preenchimento será inserido.

### tratarNulo

Acesso: funcoes.texto.tratarNulo(texto)

Garante que um texto não é nulo, retornando o valor informado se presente, ou um texto vazio se nulo ou *undefined*.

Parâmetros:

- texto: O texto a ser testado.

## numero

Acesso: funcoes.numero

### formatar

Acesso: funcoes.numero.formatar(numero, casasDecimais)

Formata o número para exibição.

Parâmetros:

- numero: O número a ser formatado.
- casasDecimais: A quantidade de casas decimais do número resultante. Se informado zero, o resultado será um número inteiro. Se as casas decimais do número original excederem este parâmetro, ele será truncado de acordo sem arredondamento.

### formatarDinheiro

Acesso: funcoes.numero.formatarDinheiro(numero)

Formata o número para exibição com duas casas decimais, preenchendo ou truncando sem arredondamento caso o número informado possua menos ou mais de duas casas decimais, respectivamente.

Parâmetros:

- numero: O número a ser formatado.

### arredondar

Acesso: funcoes.numero.arredondar(numero, casasDecimais)

Arredonda um número para o número de casas decimais informado. O método utilizado é o *half up,* onde o número 5 é arredondado para cima.

Parâmetros:

- numero: O número a ser arredondado.
- casasDecimais: A quantidade de casas decimais do número resultante.

### arredondarABNT

Acesso: funcoes.numero.arredondarABNT(numero)

Arredonda um número para 2 casas decimais utilizando a norma ABNT. O método utilizado é o *half even,* onde o número 5 é arredondado para o número par mais próximo.

Parâmetros:

- numero: O número a ser arredondado.

### somenteDigitos

Acesso: funcoes.numero.somenteDigitos(numeroFormatado)

Remove todos os separadores e sinais do número informado, mantendo apenas os algarismos.

Parâmetros:

- numeroFormatado: O número a ser convertido.

### removerZerosIniciais

Acesso: funcoes.numero.removerZerosIniciais(numero)

Remove todos os zeros à esquerda do número informado.

Parâmetros:

- numero: O número a ser convertido.

### numeroAleatorio

Acesso: funcoes.numero.numeroAleatorio(min, max)

Obtém um número aleatoriamente, entre o mínimo e o máximo informado.

Parâmetros:

- min: O valor mínimo do número gerado.
- max: O valor máximo do número gerado.

## data

Acesso: funcoes.data

### formatarData

Acesso: funcoes.data.formatarData(data, formato)

Formata uma data de acordo com o formato fornecido.

As seguintes sequências são substituídas pelos valores correspondentes:

| Sequência | Substituído por |
|---|---|
| **YYYY** | Ano com 4 dígitos |
| **YY** | Ano com 2 dígitos |
| **MM** | Mês |
| **DD** | Dia |
| **hh** | Hora |
| **mm** | Minuto |
| **ss** | Segundo |

Parâmetros:

- data: A data a ser formatada.
- formato: A máscara do formato da data. As chaves da máscara serão substituídas de acordo com a tabela acima.

### dataAtual

Acesso: funcoes.data.dataAtual(formato)

Obtém a data atual, formatada de acordo com o formato fornecido.

Parâmetros:

- formato: A máscara do formato da data. As chaves da máscara serão substituídas de acordo com a tabela da função formatarData.

### somarDias

Acesso: funcoes.data.somarDias(data, dias)

Adiciona o número de dias especificado à data.

Parâmetros:

- data: A data base.
- dias: O número de dias a somar na data base.

### subtrairDias

Acesso: funcoes.data.subtrairDias(data, dias)

Subtrai o número de dias especificado da data.

Parâmetros:

- data: A data base.
- dias: O número de dias a subtrair da data base.

### igualOuDepois

Acesso: funcoes.data.igualOuDepois(dataBase, outraData)

Verifica se uma data é no mesmo dia ou posterior à data base. Retorna *true* se outraData >= dataBase, desconsiderando a hora.

Parâmetros:

- dataBase: A data base.
- outraData: A data a ser comparada.

### calcularFatorVencimento

Acesso: funcoes.data.calcularFatorVencimento(data)

Calcula o fator de vencimento de um boleto com base na data informada, de acordo com as especificações da FEBRABAN.

Parâmetros:

- data: A data a ser calculada.

## moduloDez

Acesso: funcoes.moduloDez

### calcularModuloDez

Acesso: funcoes.moduloDez.calcularModuloDez(valor)

Efetua cálculo de módulo 10. Cada algarismo, da direita para a esquerda, é multiplicado alternadamente por 2 e 1 e os resultados são somados. A soma resultante é dividida por 10 e o resto, subtraído de 10, é retornado.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 10.

### calcularModuloDezSepararDigitos

Acesso: funcoes.moduloDez.calcularModuloDezSepararDigitos(valor)

Efetua cálculo de módulo 10. Diferente da função calcularModuloDez, a soma é feita utilizando os dígitos das multiplicações separadamente, em vez de somar a multiplicação diretamente.

Por exemplo, se o valor de entrada é *47*, as multiplicações são 4 x 1 = 4 e 7 x 2 = 14. Na função calcularModuloDez a soma seria 4 + 14 = 18, enquanto nesta função a soma seria 4 + 1 + 4 = 9.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 10.

## moduloOnze

Acesso: funcoes.moduloOnze

### calcularModuloOnze

Acesso: funcoes.moduloOnze.calcularModuloOnze(valor)

Efetua cálculo de módulo 11. Cada algarismo, da direita para a esquerda, é multiplicado alternadamente por números crescentes de 2 a 9  e os resultados são somados. A soma resultante é dividida por 11 e o resto, subtraído de 11, é retornado. Se o resto for igual a 0 ou 1, o valor retornado é 1.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.

### calcularModuloOnzeInvertido

Acesso: funcoes.moduloOnze.calcularModuloOnzeInvertido(valor)

Efetua o cálculo do mótulo 11 como na função calcularModuloOnze, porém a multiplicação é feita descrescente, de 9 a 2.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.

### calcularModuloOnzeZeroOuResto

Acesso: funcoes.moduloOnze.calcularModuloOnzeZeroOuResto(valor)

Efetua o cálculo do mótulo 11 como na função calcularModuloOnze, porém retorna 0 se o resto for igual a 0 ou 1.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.

### calcularModuloOnzeZeroOuRestoComSequencia

Acesso: funcoes.moduloOnze
.calcularModuloOnzeZeroOuRestoComSequencia(valor, sequenciaMultiplicadora)

Efetua o cálculo do mótulo 11 como na função calcularModuloOnzeZeroOuResto, utilizando uma sequência multiplicadora customizada, da esquerda para a direita.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.
- sequenciaMultiplicadora: A sequência de dígitos utilizados na multiplicação.

### calcularModuloOnzeInvertidoZeroOuRestoComSequencia

Acesso: funcoes.moduloOnze
.calcularModuloOnzeZeroOuRestoComSequencia(valor, sequenciaMultiplicadora)

Efetua o cálculo do mótulo 11 como na função calcularModuloOnzeZeroOuResto, utilizando uma sequência multiplicadora customizada, da direita para a esquerda.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.
- sequenciaMultiplicadora: A sequência de dígitos utilizados na multiplicação.

### calcularModuloOnzeVariavel

Acesso: funcoes.moduloOnze.calcularModuloOnzeVariavel(valor, maximoMultiplicador)

Efetua o cálculo do módulo 11 como na função calcularModuloOnze, utilizando o máximo multiplicador. Se o resultado é igual a zero, retorna o valor acrescido de 0. Se o resultado é maior que 2, retorna o valor acrescido do resultado. Caso contrário, o último dígito do valor é somado a 1 e o processo é feito novamente, até que o resultado seja igual a 0 ou maior que 2. Em seguida este dígito somado é substituído no valor original e acrescido do resultado, então retornado.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 11.

### calcularModuloCem

Acesso: funcoes.moduloOnze.calcularModuloCem(valor, maximoMultiplicador)

Efetua o cálculo do mótulo como na função calcularModuloOnze, porém utilizando o resto da divisão por cem (modulo 100) em vez de 11.

Parâmetros:

- valor: O valor a ser utilizado no cálculo do módulo 100.