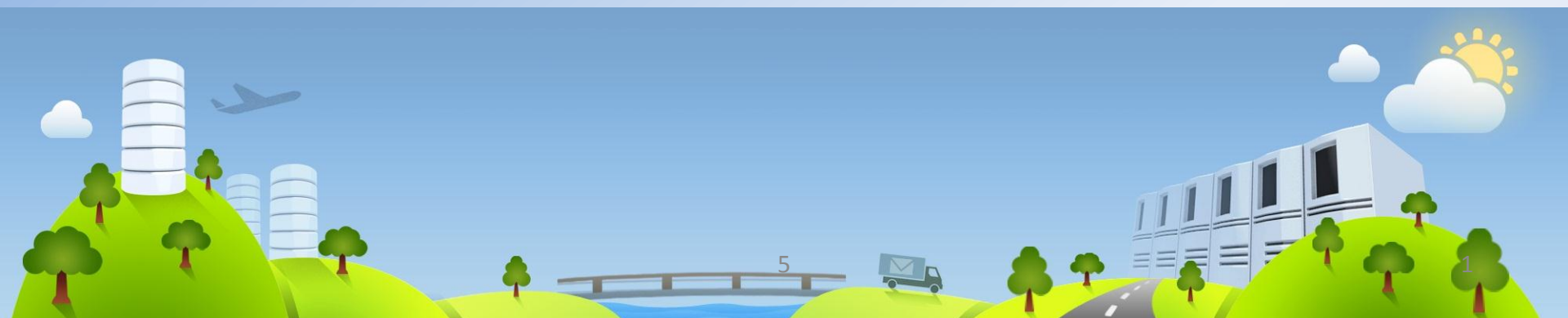




VERSÃO PARA DOWNLOAD

CURSO COMPETÊNCIAS TRANSVERSAIS

LÓGICA DE PROGRAMAÇÃO



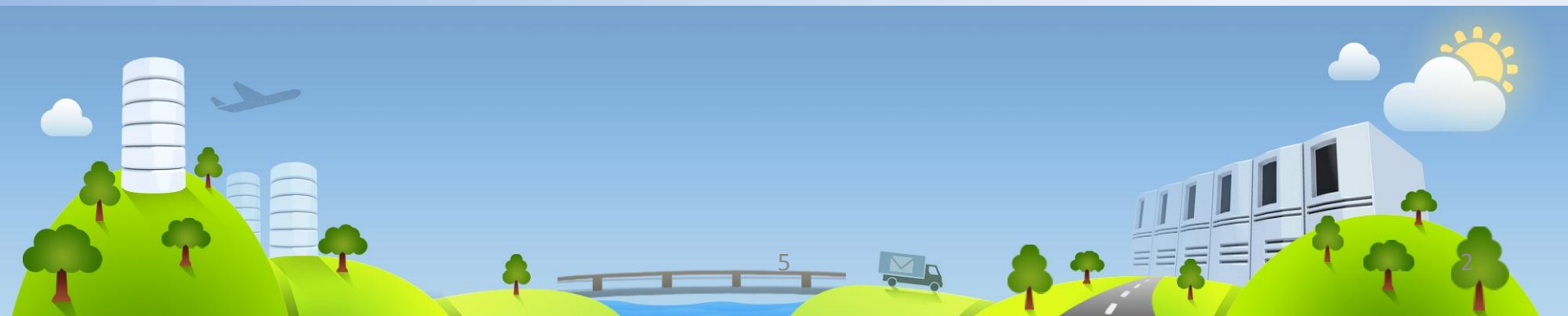


OLÁ!

SEJA BEM-VINDO AO CURSO DE
LÓGICA DE PROGRAMAÇÃO

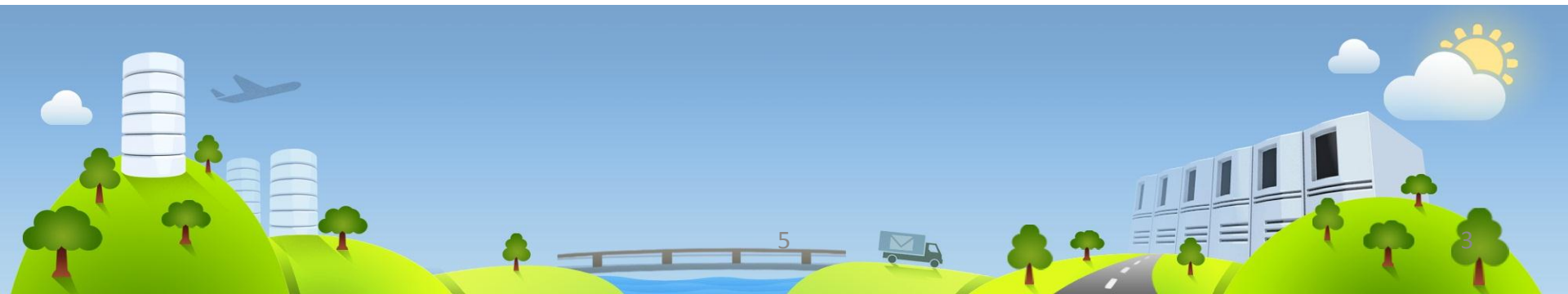
Objetivo do curso:

**Capacitar o aluno a conhecer um pouco do mundo da
lógica de programação para poder criar sites,
desenvolver games e programar robôs.**



SUMÁRIO

Introdução.....	4
Representações	
Representações de um algoritmo.....	12
VISUALG.....	18
Tipos	
Tipos de dados.....	23
Variáveis.....	26
Expressões	
Expressões aritméticas.....	33
Expressões literais.....	36
Expressões lógicas.....	38
Estruturas	
Estruturas de Condição.....	46
Estruturas de Repetição.....	50
Variáveis indexadas.....	59
Revisão.....	63
Conteúdo extra.....	70



INTRODUÇÃO

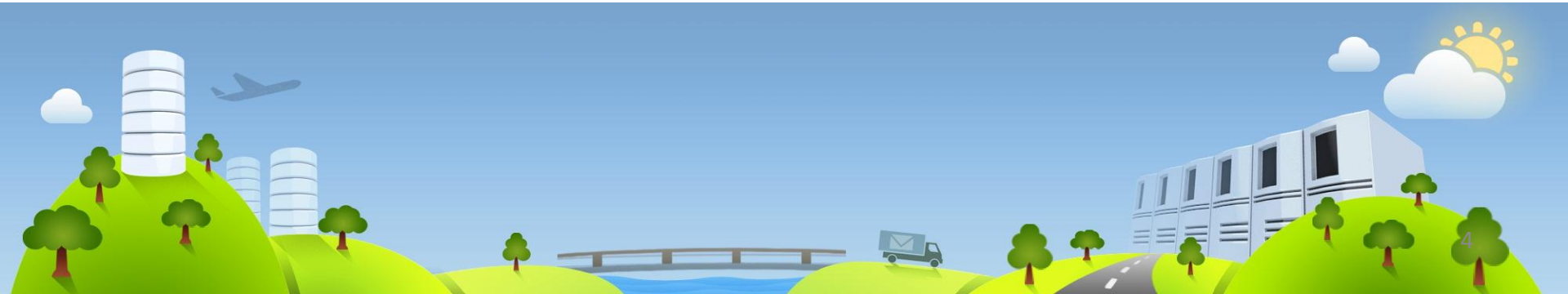
Boas Vindas!

Seja muito bem-vindo(a), a partir de agora você conhecerá um pouco do mundo da lógica de programação. Conseguirá, por meio deste curso, desenvolver sua lógica para poder:

- criar *sites*;
- desenvolver *games*;
- programar robôs;

Você será desafiado a desenvolver uma calculadora até o fim do nosso conteúdo. Então, atente-se ao que vem por aí!

Utilizando linguagens de programação temos um mundo de possibilidades, e você terá a oportunidade de entender como tudo isso funciona.



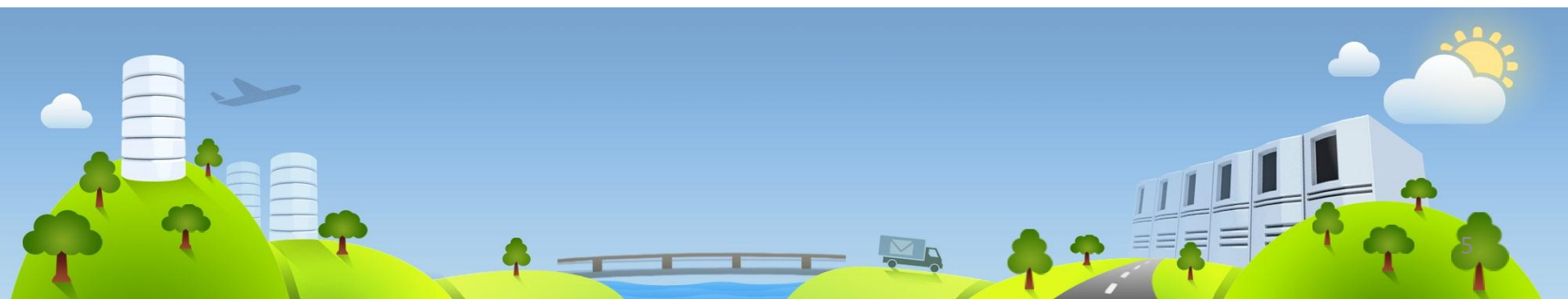
Calculadora

A nossa calculadora fará o seguinte:

- aceitará apenas dois números reais;
- fará as operações: adição, subtração, divisão e multiplicação;
- mostrará o resultado da operação ao usuário.

Esta é nossa meta até o fim do curso: ter uma calculadora pronta e funcional.

Preparado (a)? Então, que comece a nossa viagem pelo mundo da lógica de programação!



Algoritmos

Inicialmente, iremos entender o que é um algoritmo, e quais os exemplos que vivenciamos todos os dias.

Em seguida, mudaremos o foco dos algoritmos do dia a dia para algoritmos na computação e programação, veremos os tipos existentes de algoritmos e alguns exemplos práticos.

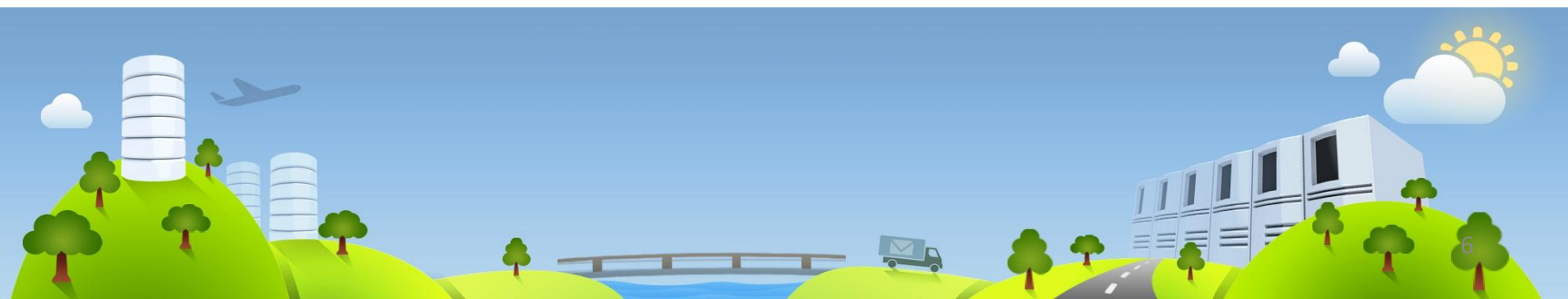
Depois que tivermos os primeiros conceitos prontos, daremos início às partes funcionais da nossa calculadora!

O que é algoritmo?

Um algoritmo é uma sequência de instruções que utilizamos para solucionar um ou vários problemas, ou até mesmo realizar tarefas do dia a dia.

Um algoritmo não é necessariamente um programa computacional, pode ser passos que iremos tomar para realizar determinada tarefa.

O algoritmo deve sempre chegar ao resultado final esperado, caso não chegue, o mesmo não pode ser considerado finalizado.



Vamos exemplificar um pouco...

Exemplos de algoritmos do dia-a-dia



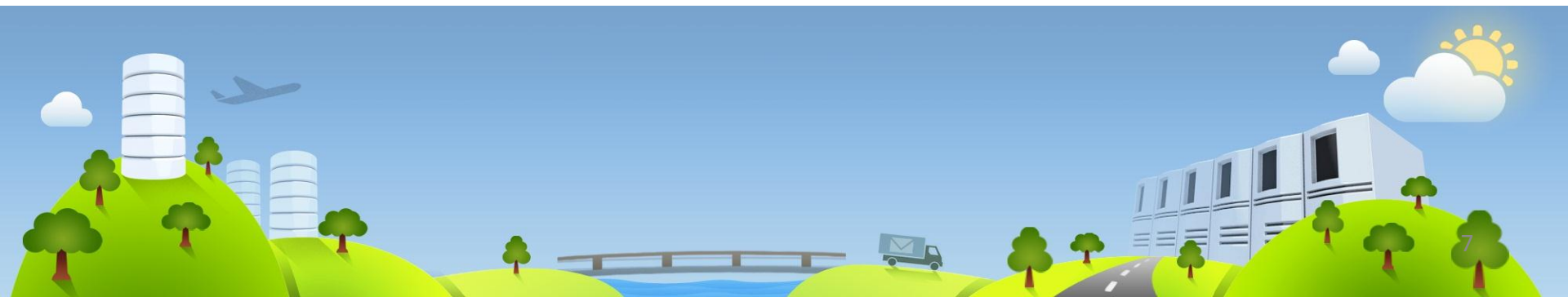
Trocar Lâmpada



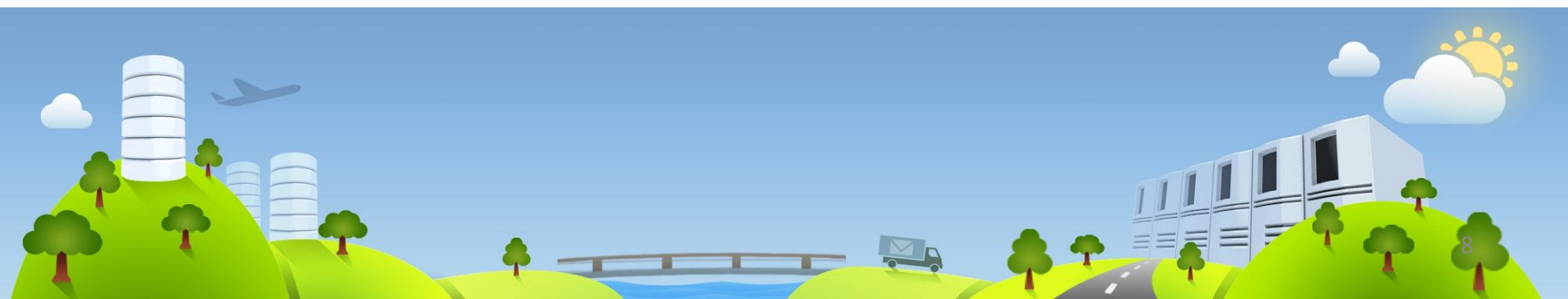
Preparar café



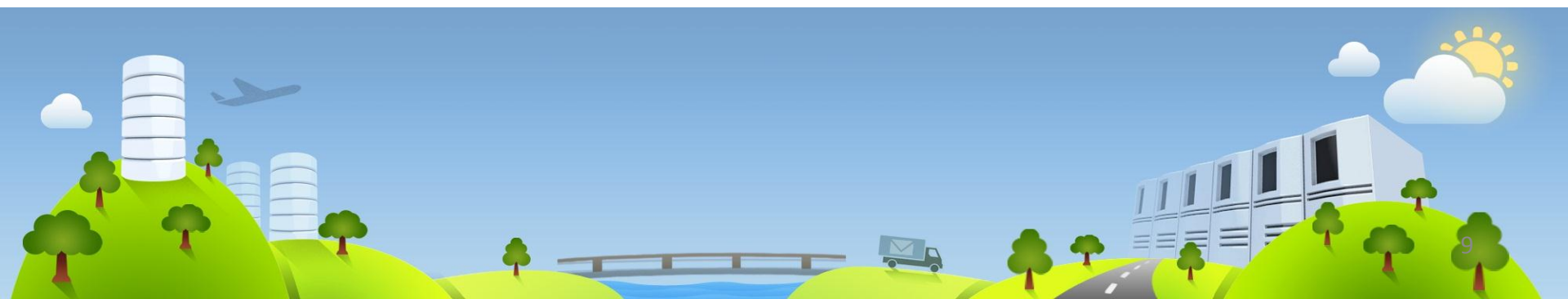
Fazer bolo



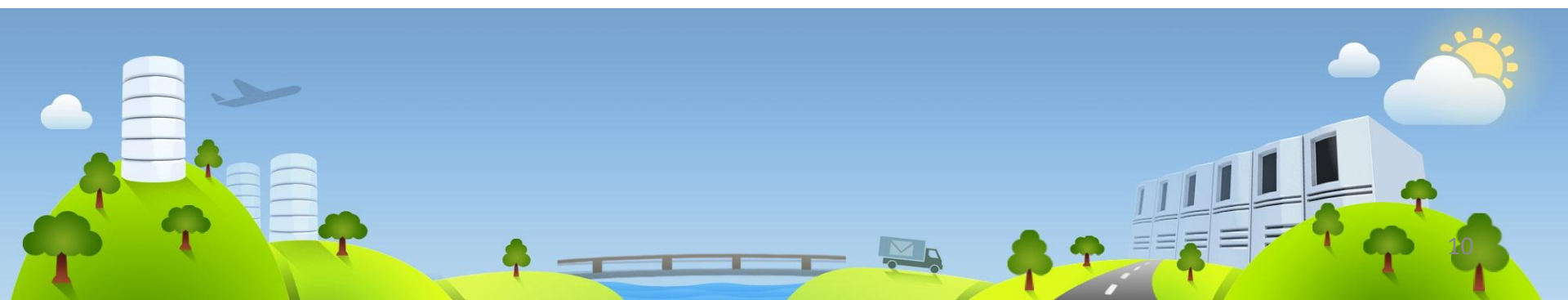
Lâmpada



Café



Bolo



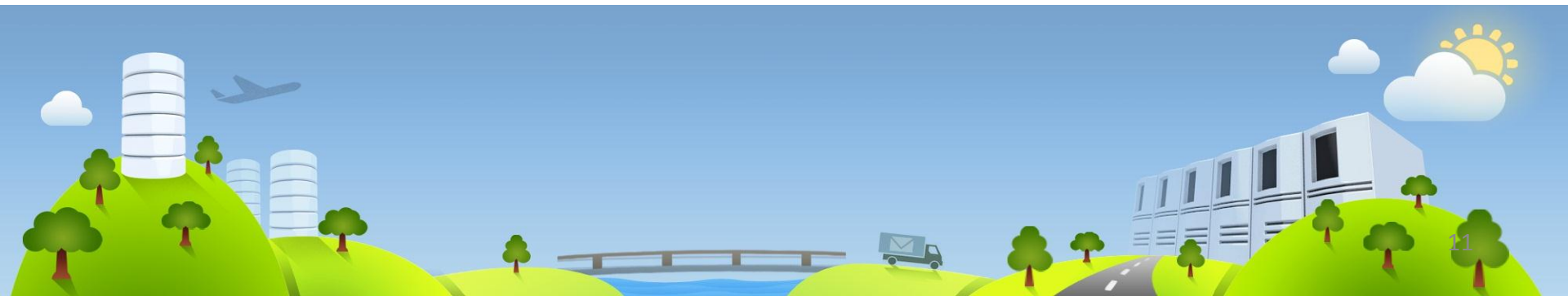
Algoritmos

Mediante a estrutura apresentada anteriormente, fica mais fácil compreender a definição de um algoritmo, caracterizando-se por ser um conjunto de instruções objetivas.

Embora a palavra algoritmo nos remeta a pensar em uma infinidade de coisas complexas, o mesmo se trata de um conjunto de instruções que tem como objetivo resolver um problema. Não quer dizer que não tenhamos algoritmos complexos, isso irá depender da complexidade do problema em questão.

Todos os exemplos vistos anteriormente fazem parte do nosso dia a dia, são algoritmos que executamos sempre que precisamos. Deixando um pouco o cotidiano de lado, iremos agora entrar de cabeça no mundo computacional.

Let's go!



REPRESENTAÇÕES

Representações de um algoritmo

Agora que tivemos uma introdução do que se trata os algoritmos, iremos entender como eles podem ser representados.

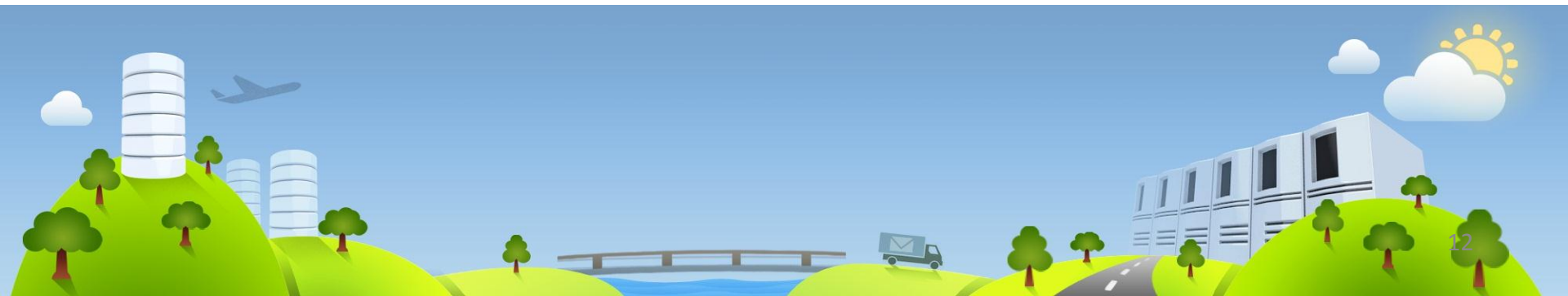
Temos várias formas de representar algoritmos, desde a mais simples, representada por formas, até as mais detalhistas, contendo regras de implementação.

Abaixo, conheceremos os dois tipos mais utilizados: representados por formas que fazem alusão a tomadas de decisões (fluxograma) e por meio do pseudocódigo, sendo este mais utilizado para o ensino de lógica de programação; por meio de linguagem de máquina criamos nossos algoritmos para os computadores processarem e resolverem os problemas

Saiba mais: Representações de um algoritmo

Um pseudocódigo trata-se de um meio didático de se aprender a programar; não se consegue construir nenhum *software* via pseudocódigo.




No caso, teríamos que nos aprofundar para aprender alguma linguagem de programação. Ex.: Java, Python, PHP, Ruby, JavaScript, C# entre outras.

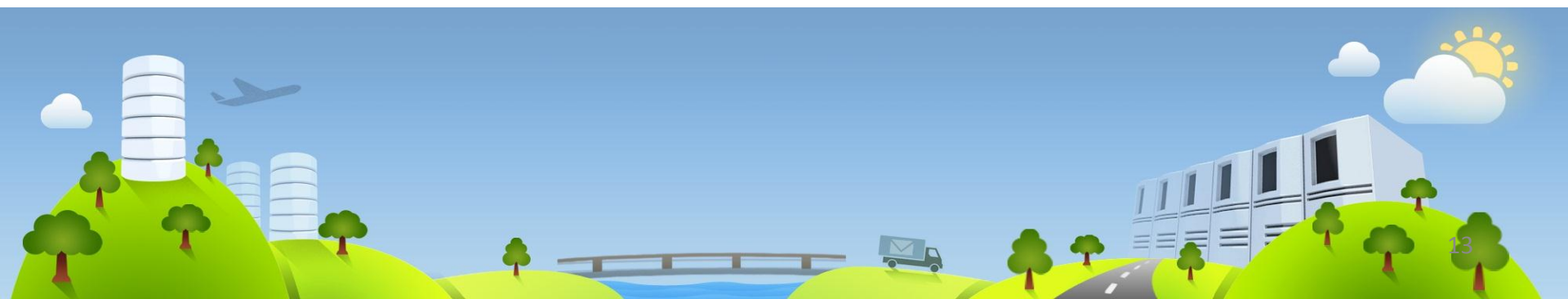


Fluxograma

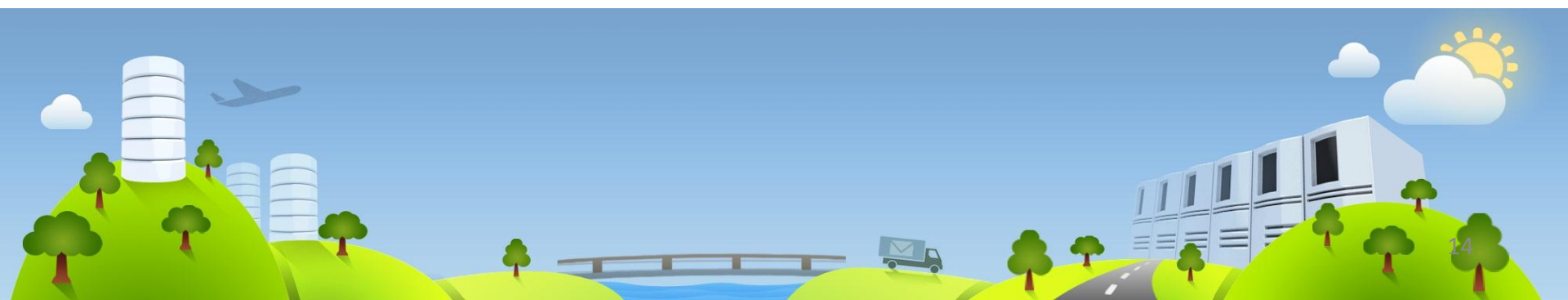
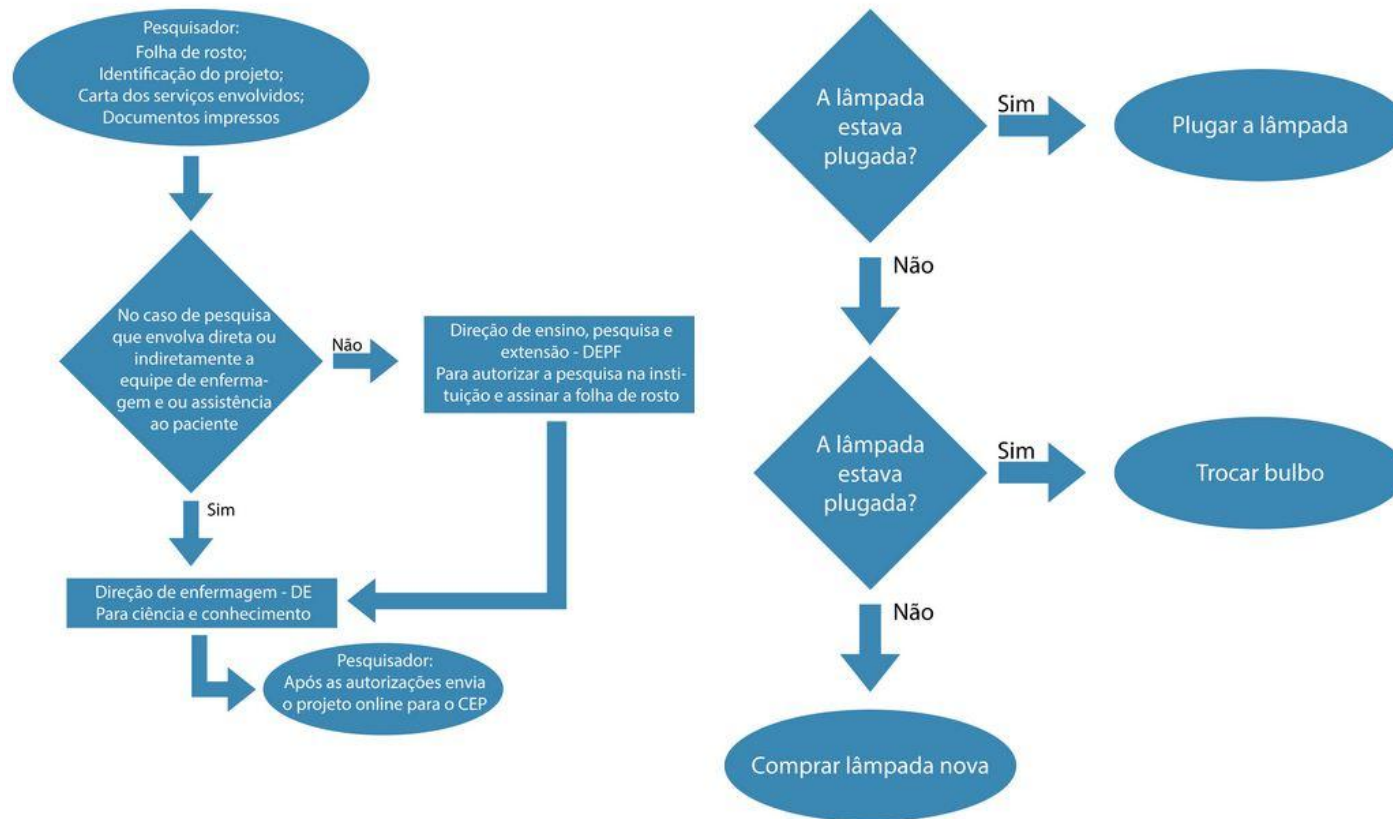
Representado por formas, tornam o entendimento de determinado algoritmo mais simples, pois uma figura é mais clara que várias palavras.

É definido por uma série de símbolos, em conjunto com desenhos geométricos que representam os passos do algoritmo, tais como: início, entrada e saída de dados, tomada de decisões, estruturas de repetição e final do algoritmo.

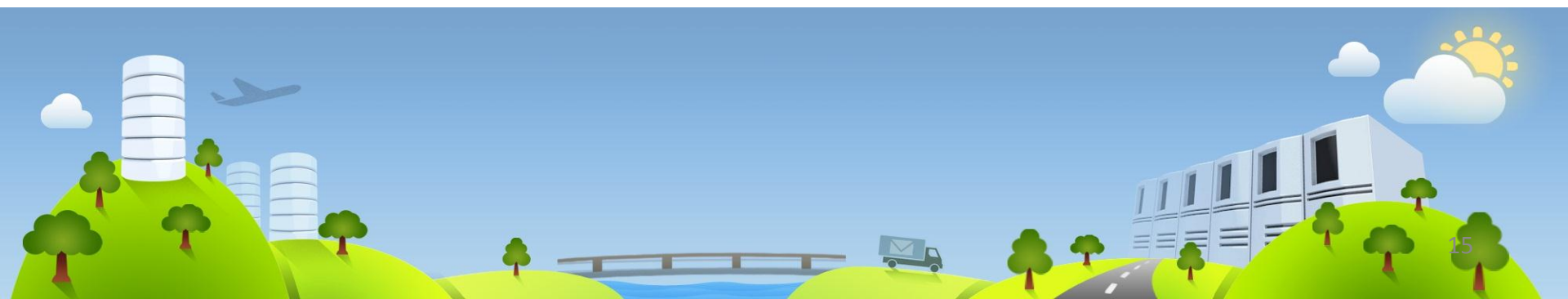
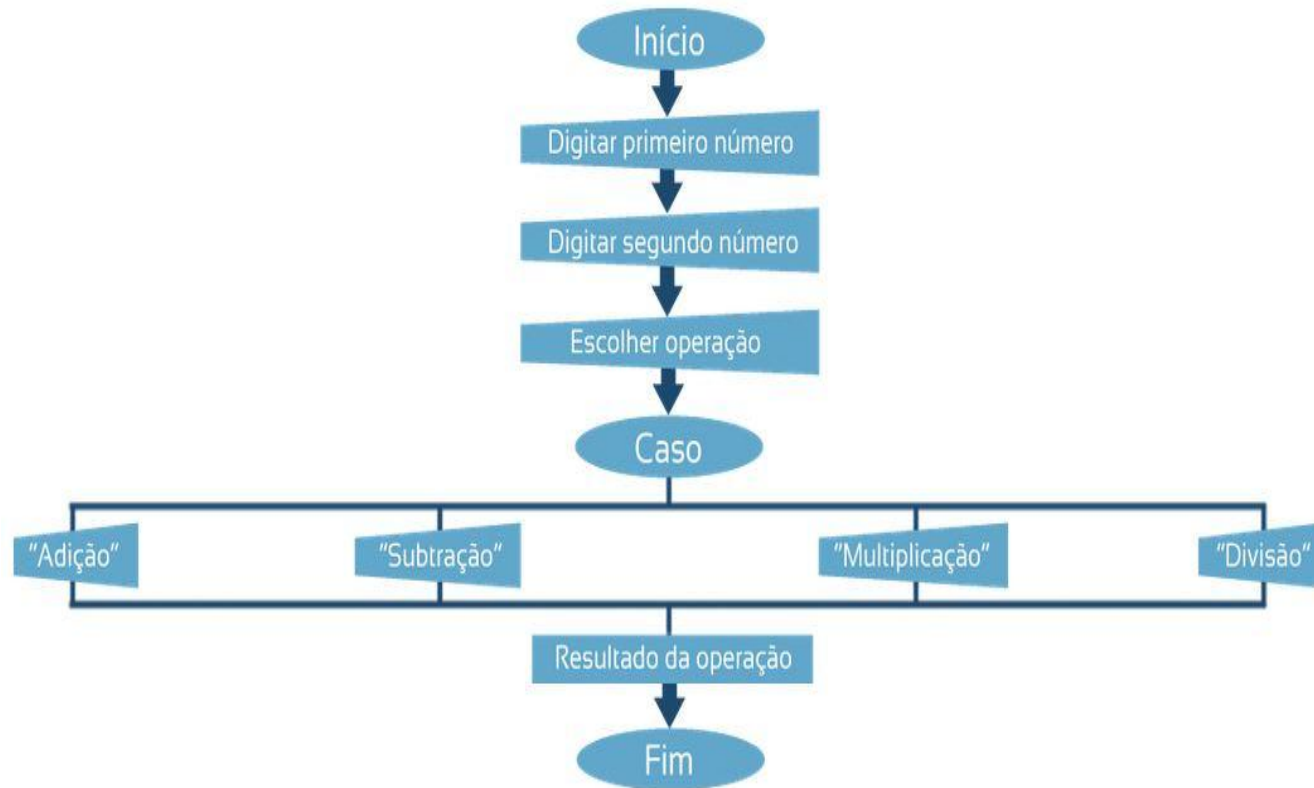
Símbolo	Descrição
	O círculo alongado é utilizado para dar início ou terminar um fluxograma.
	O retângulo é utilizado para fazer uma ação ou uma instrução no fluxograma.
	O losango é utilizado para uma decisão em um fluxograma.



Fluxograma



Fluxograma da Calculadora

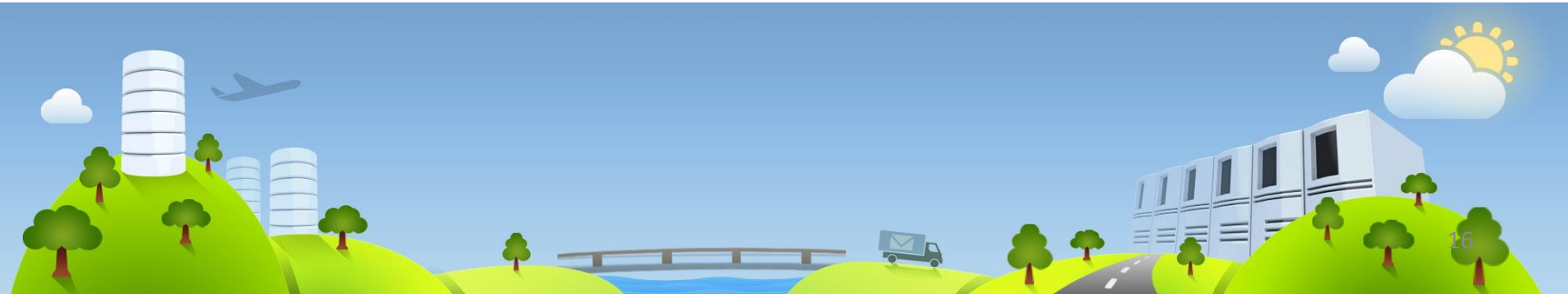


Pseudocódigo

Diferente do fluxograma, o pseudocódigo, também conhecido como portugol ou português estruturado, utiliza PDL - Program Design Language (Linguagem de Projeto de Programação). Ou seja, é uma espécie de narração do que o programa deve fazer.

Para isso, utilizaremos o pseudocódigo em um programa chamado VisuAlg, que abordaremos em nossa próxima lição.

O pseudocódigo é a base que todos devem ter para aprender uma linguagem de programação, pois é ele que intermedia a linguagem falada (humana) para a linguagem de programação (computacional).



Exercitando

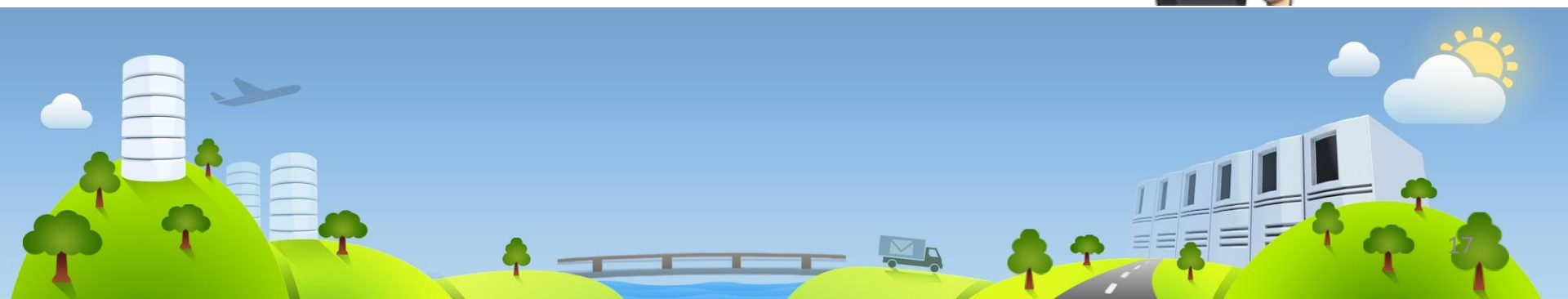
Pegue um papel e uma caneta e faça o seguinte algoritmo:

- Chupar bala - Utilizando fluxograma demonstre as etapas necessárias para chupar a bala, lembrando que você ainda não a possui.

Será que você consegue?



17



VisuAlg

Conhecemos os tipos de representações de algoritmos, e, agora, iremos conhecer o programa que executará nossos pseudocódigos.

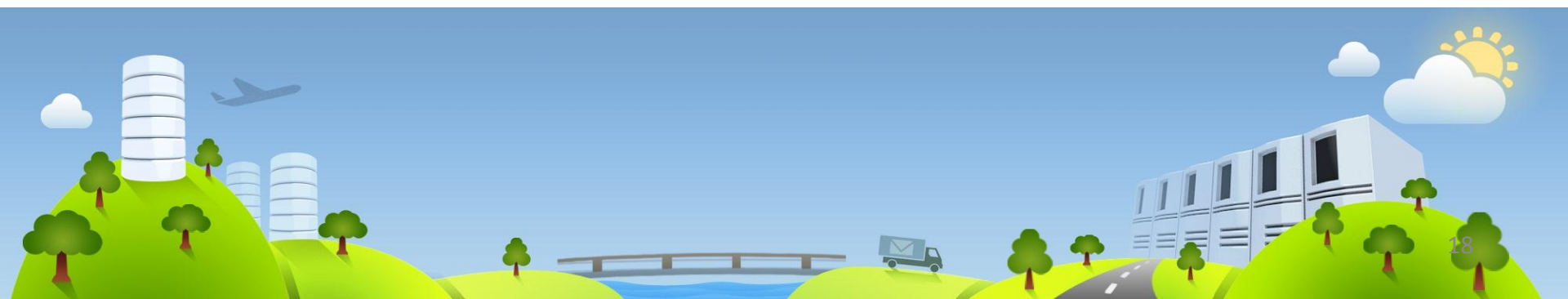
Os pseudocódigos podem ser escritos facilmente em uma folha de papel ou em um bloco de notas, mas para tornar isso prático vamos adotar o VisuAlg como nosso *software* para escrever nossos algoritmos na forma de pseudocódigo e ver na hora o seu resultado.

O que é o VisuAlg?

É um *software* criado na Universidade de Caxias do Sul no estado do Rio Grande do Sul, com a finalidade de oferecer aos alunos uma forma de exercitar o conhecimento adquirido.

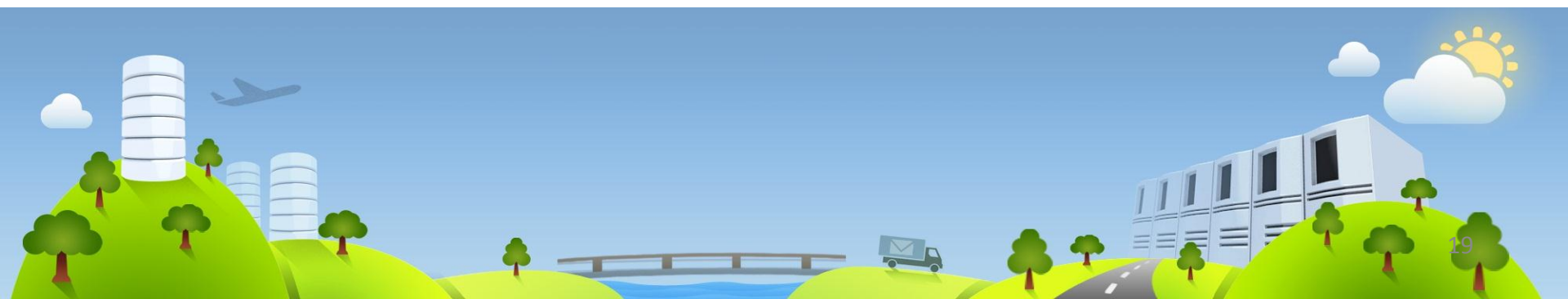
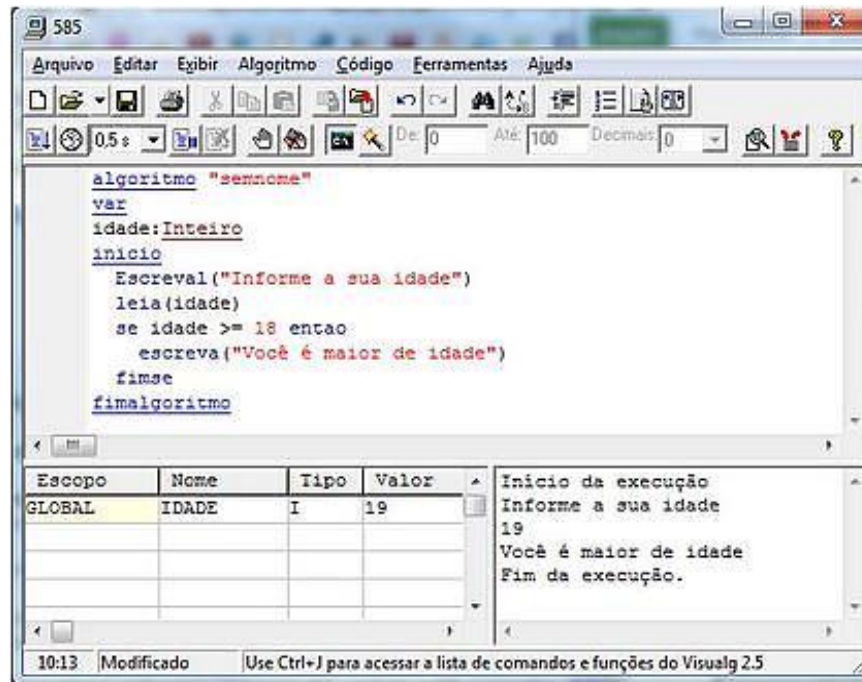
Para utilizá-lo é necessário fazer o *download* do programa. O mesmo pode ser encontrado em:

https://ava1.sp.senai.br/pluginfile.php/476890/mod_scorm/content/4/assets/modulos/representacoes/visualgv25.exe



O visuAlg é um interpretador simples de pseudocódigo utilizado por professores para o ensino de lógica de programação.

Seu objetivo não é criar *softwares*, mas auxiliar o aluno a entender a execução de seu algoritmo.



Entendendo um pouco melhor

Para a criação de um algoritmo, é necessária a utilização de palavras-chave. Abaixo, seguem as três principais para o funcionamento:

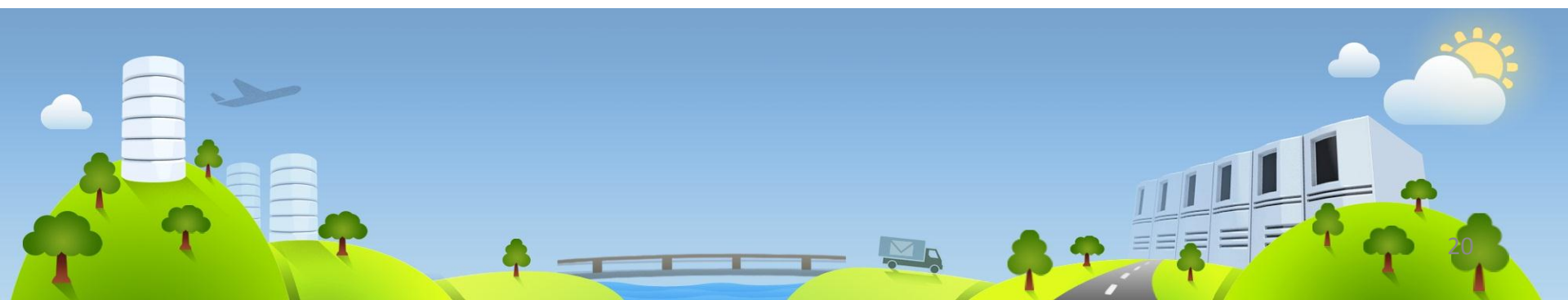
- **algoritmo:** comando que irá definir o nome do programa; deve ser feito em aspas duplas;
- **var:** comando que especifica a área em que as variáveis serão declaradas. É aqui que colocaremos as variáveis que iremos utilizar;
- **início:** comando que informa o início do programa; é nesse bloco que ficarão os comandos e a lógica que utilizaremos para criar nosso algoritmo.
- **fimalgoritmo:** comando que informa que é o final do algoritmo.

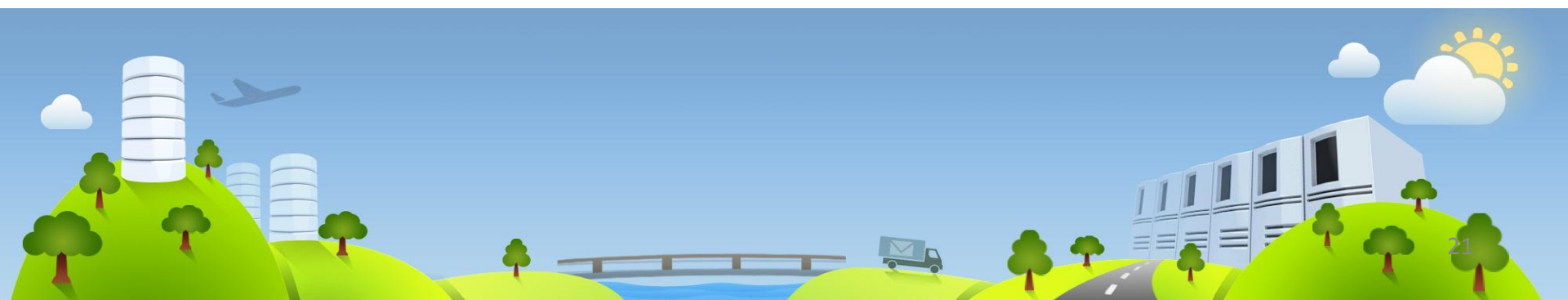
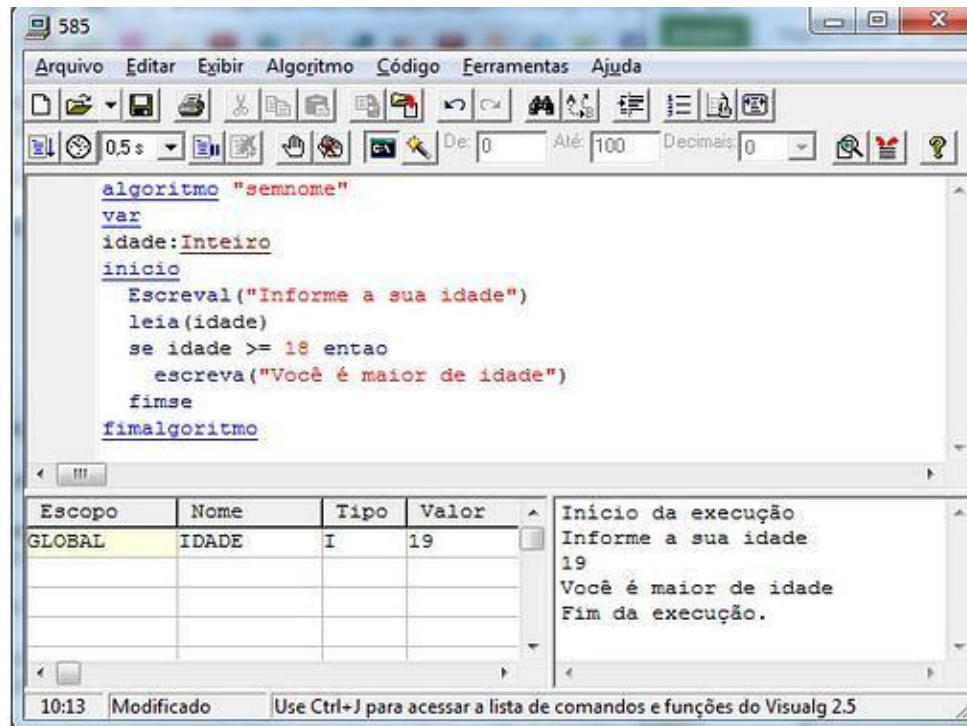
Dentro do nosso algoritmo utilizaremos palavras-chave que irão ler o que for digitado e escrever na tela para informar algo ao usuário.

escreva: esse comando irá escrever na tela alguma informação ao usuário. Ex.: `Escreva("Digite seu nome")`.

Leia: esse comando irá ler o que foi digitado pelo usuário. Ex.: `Leia(nome)`. Esse nome entre parênteses é uma das variáveis que deve estar declarada no bloco **var** que vimos anteriormente.

Escreval: esse comando é idêntico ao `escreva`, a não ser pelo fato de que ele pula uma linha. Ex.: `Escreval("Digite seu nome")`.





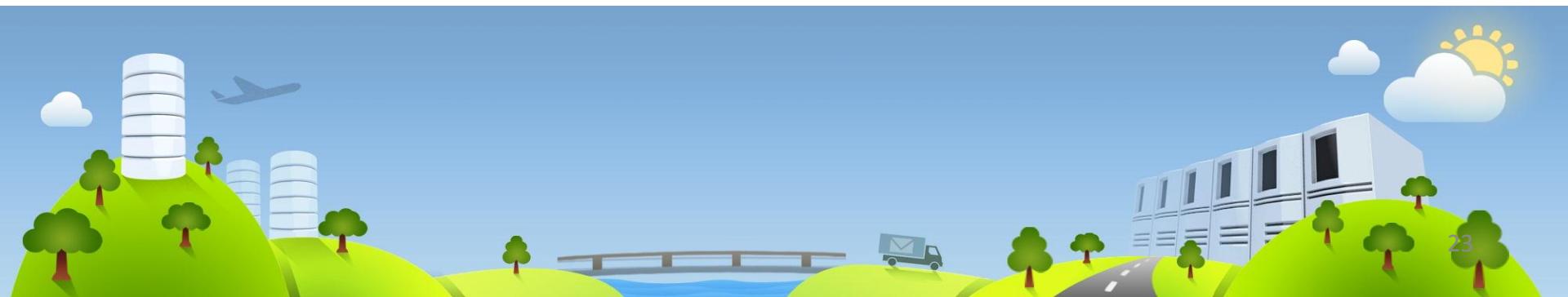
TIPOS

Tipos de Dados

Anteriormente, vimos que os algoritmos fazem parte do nosso dia a dia, também conhecemos como e o que é um algoritmo e quais as representações que ele pode ter, além da ferramenta que utilizaremos em nosso curso.

Agora começaremos a mergulhar na parte computacional, entender como os algoritmos funcionam aplicados em um sistema de computador.

Tipos de dados:	
Real	Inteiro
0.75	2
- 3.2	170
1.33	-23
...	...



Quando criamos um programa computacional, temos que levar em conta quais os tipos de dados que poderão ser inseridos no sistema. É aí que entram os Tipos de Dados. São eles que dirão o que poderá ser digitado ou quais valores nosso algoritmo aceitará.

Então, vamos ver quais são os tipos de dados e quais informações eles aceitam.

Tipo de Dado Inteiro

Aceitam somente números inteiros positivos e negativos.
Não aceitam números decimais.

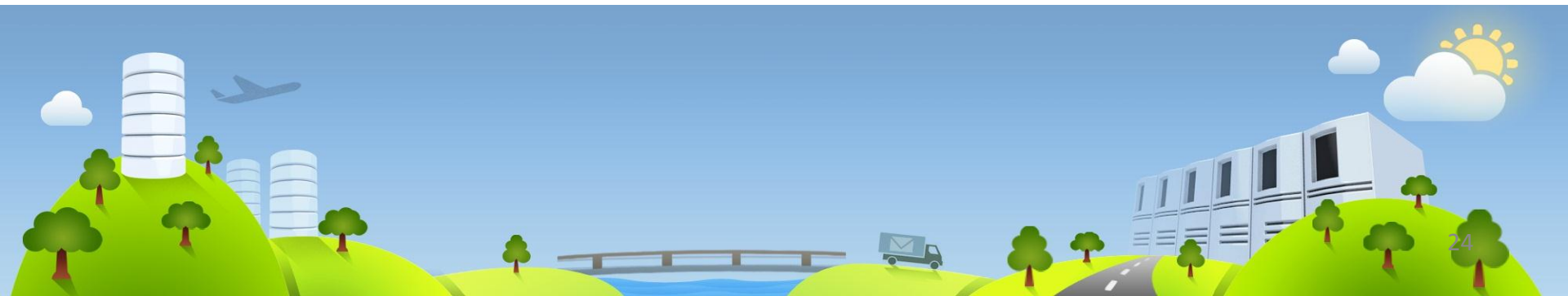
0 1 ~~4.70~~ 35 -54 -52 23 ~~59.32~~ -35 34 -31

Tipo de Dado Real

Podem ser valores positivos e negativos.
Podem ser números decimais e inteiros.

0 1 4.70 3.5 -54 -52.4 23.0 59.32 -35 34 -3.14

No VisuAlg, o separador decimal é o ponto, não a vírgula.



Tipo de Dado Literal

Podem ser letras, números ou caracteres especiais, ou seja, podem ser qualquer tipo de valor.

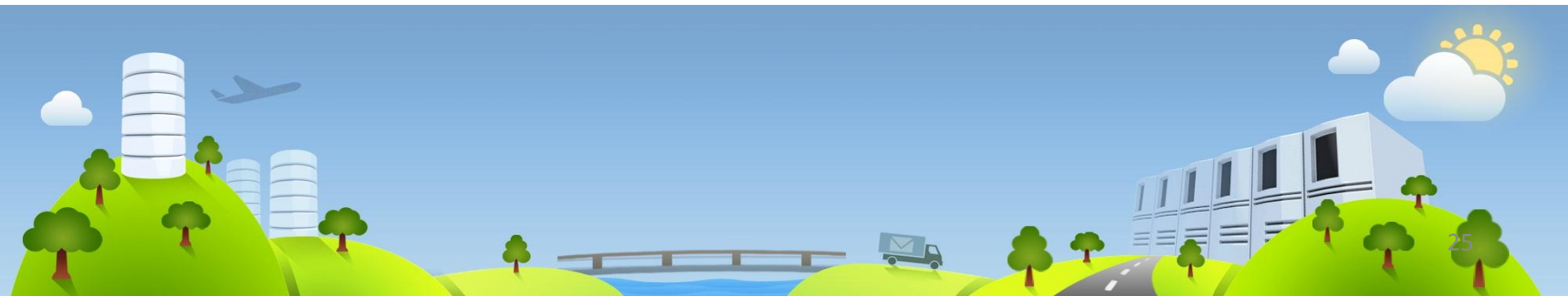
`"Algoritmo"` `"Senai"` `"3432"` `"43Ab"` `"&sp3cl@l"`

Os valores deste tipo devem estar dentro de aspas duplas ("").

Tipo de Dado Lógico

São respostas para uma pergunta, a qual deverá ter apenas duas possíveis respostas: sim ou não. Porém, sempre resultam como VERDADEIRO ou FALSO.

Pergunta	Resposta	Em algoritmo
Maior de idade?	Sim	VERDADEIRO
Está chovendo?	Não	FALSO
Maior de idade?	1	VERDADEIRO
Está chovendo?	0	FALSO



Variáveis

Variável é onde iremos guardar dados essenciais para o funcionamento do nosso algoritmo/programa.

Uma variável guarda um valor informado em uma posição de memória, seu conteúdo pode sempre ser alterado, como o próprio nome já diz "variável". Uma variável só pode ter UM valor dentro dela.

Imagine as variáveis como se fossem gavetas. Em uma gaveta podemos colocar diversas coisas, mas e se rotularmos uma gaveta, ou seja, colocarmos uma legenda, ou um papel informando o que deve ter na gaveta ?

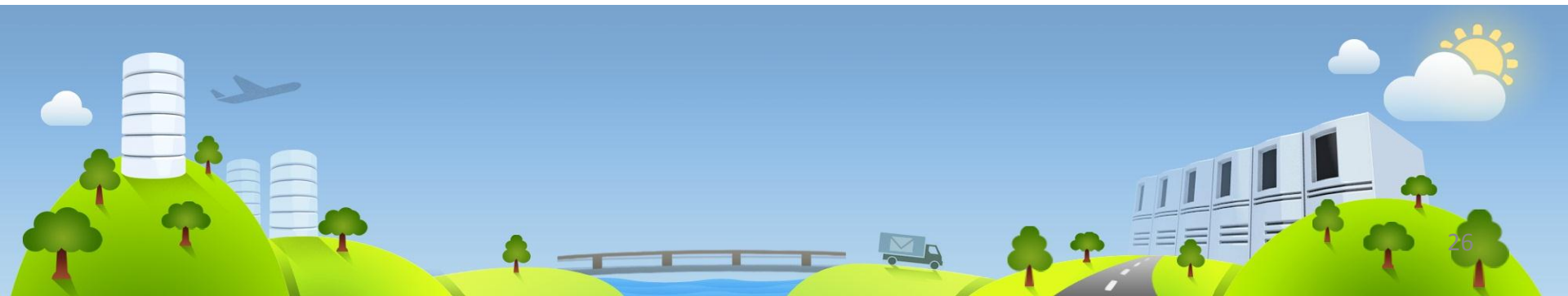
Uma gaveta informando que irá ter meias, dentro dela deverá ter apenas meias. Possivelmente, alguém poderá colocar algo diferente (camiseta), onde deveria ter apenas meias.

Nos algoritmos computacionais isso não acontece, uma vez que determinado que uma variável é do tipo inteiro, ela só poderá ter números inteiros.

Variáveis têm três passos principais para serem criadas, são eles:

1 - Ter um nome

TODA variável criada em um programa deve ter um nome para que possa ser identificada.



2 - Ter um tipo de dado vinculado

Vimos, na lição anterior, os tipos de dados, e é aqui que iremos utilizá-los. Ou seja, é aqui que diremos qual tipo de dado pode conter em nossa variável.

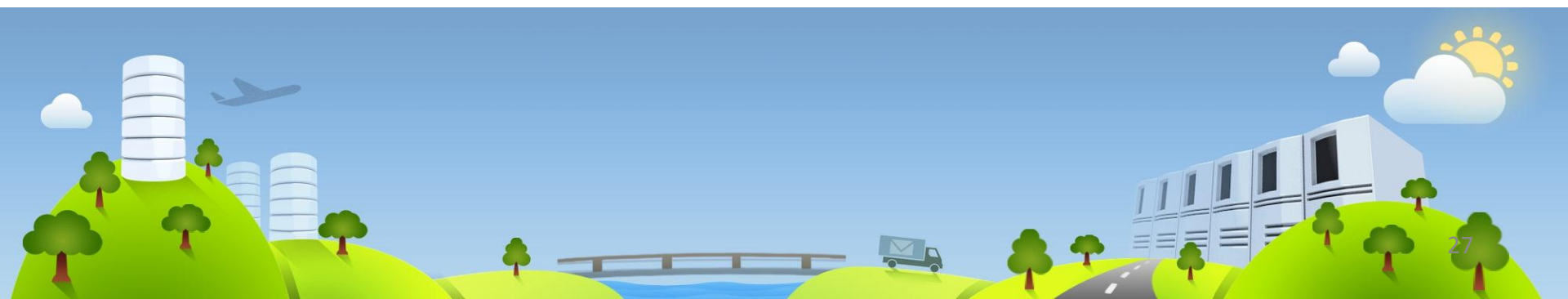
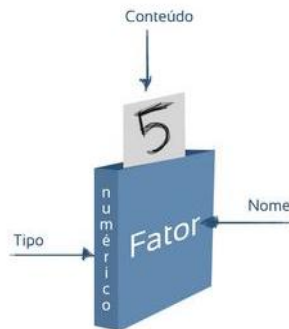
3 - Ter um valor atribuído

Não tem por que eu criar uma variável sem ter a intenção de colocar algum dado dentro dela.

Agora vamos ver isso na prática:

- O que são: espaços da memória do computador destinados ao armazenamento de dados.
- Como usar: no programa, deve-se dar um nome a ela e informar qual o seu tipo de dado. E, no decorrer do algoritmo, atribuir o valor a ela necessário.

```
var  
nomeDaVariavel: TipoDeDado
```



Variáveis – Regras

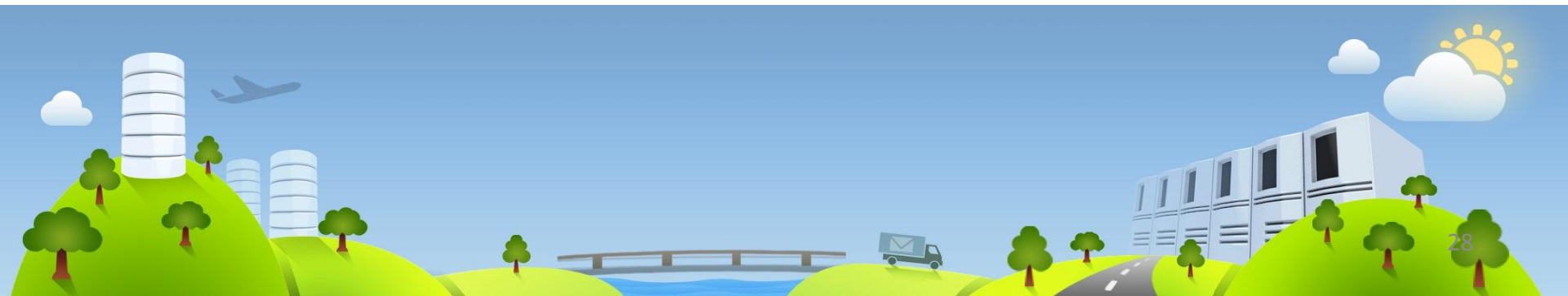
Regras para o nome da variável:

- nomes de variáveis não podem ser iguais a palavras reservadas;
- nomes de variáveis devem possuir como primeiro caractere uma letra ou sublinhado '_' (os outros caracteres podem ser letras, números e sublinhado);
- nomes de variáveis devem ter no máximo 127 caracteres;
- nomes de variáveis não podem conter espaços em branco.

Saiba mais: Palavras Reservadas

Palavras reservadas são comandos que o algoritmo processa como ação e não como valor, por exemplo, não podemos criar uma variável com o nome de algoritmo, var, entre outros, pois estas estão reservadas para uso do algoritmo.

Essas são as regras para nossas variáveis funcionarem no programa.



Abaixo estão listados exemplos de variáveis:

nome1

meuNome (Sempre que tivermos palavras separadas, removemos o espaço e deixamos a primeira letra de cada palavra em maiúsculo).

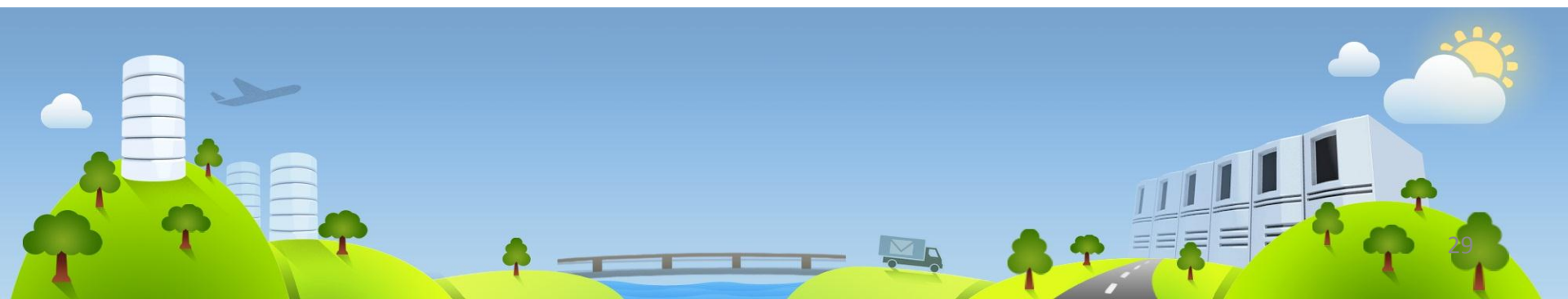
Para definir o tipo de dado de uma variável, primeiro declaramos a variável dentro do bloco de variáveis do nosso programa, em seguida colocamos dois pontos (:) e o tipo de dado (inteiro, literal, lógico ou real).

Exemplo:

```
var  
nomeDaVariavel: TipoDeDado  
var  
idade: inteiro
```

É assim que devemos declarar nossas variáveis no VisuAlg para que funcione.

- | | |
|------------------|------------------|
| ✓ Minha_Variavel | ✓ _MinhaVariavel |
| ✓ MinhaVariavel | ✗ Minha Variavel |
| ✓ MinhaVariavel2 | ✗ IMinhaVariavel |
| ✓ Minha2Variavel | ✗ Inicio |



Variáveis - Atribuindo Valor

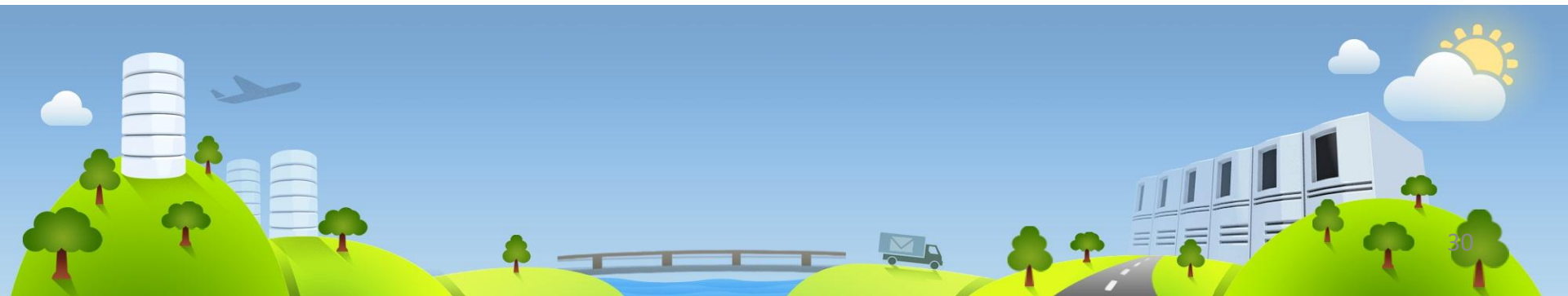
Para atribuir um valor a uma variável, precisamos usar o operador de atribuição que é <-

Exemplo:

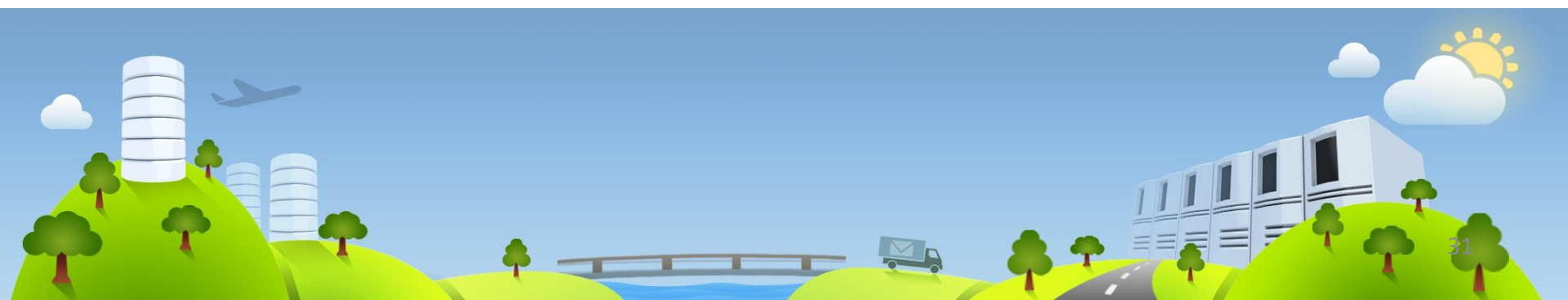
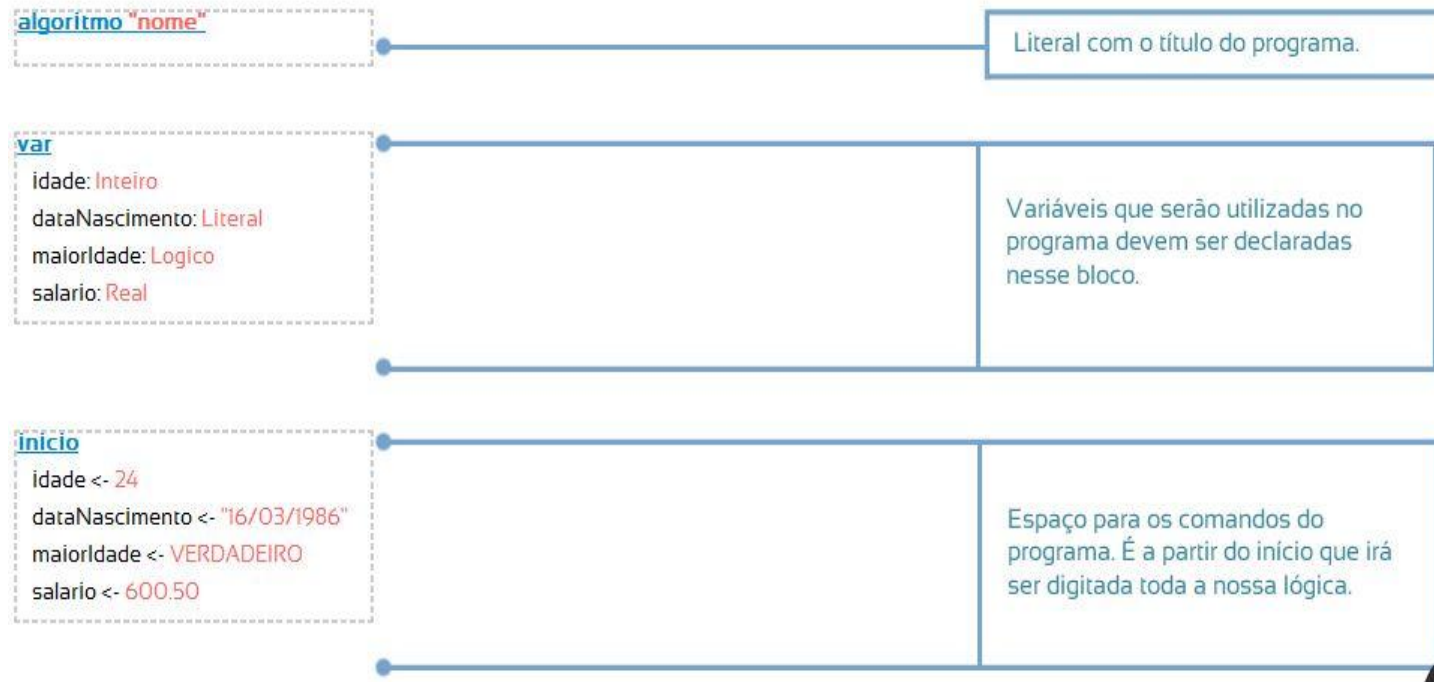
```
var  
nomeDaVariavel <- "SENAI"
```

Nesse exemplo eu estou dizendo que a variável "nomeDaVariavel" irá receber o valor "SENAI" como literal.

A variável deve ter sido declarada com o tipo literal para que aceite tal tipo.



Exemplo no VisuAlg:



Variáveis - Entrada de dados

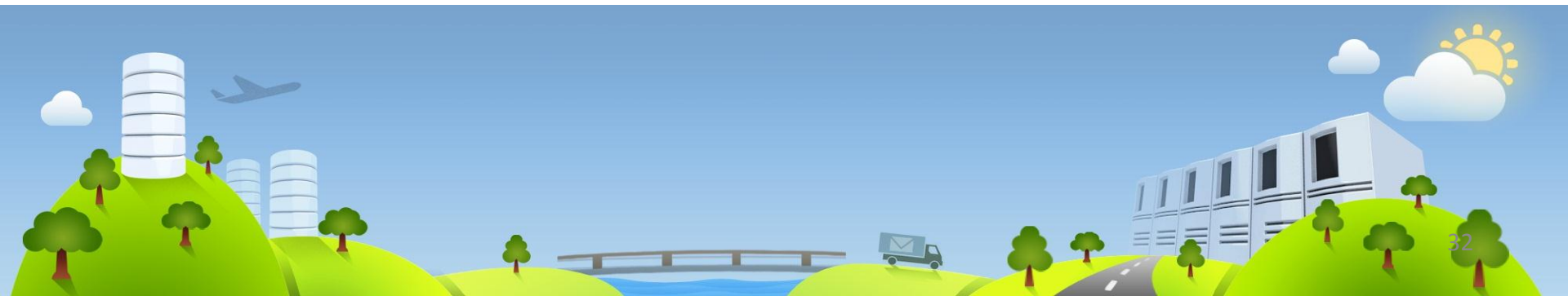
Para atribuir um valor a uma variável, precisamos usar o operador de atribuição que é <-



Exemplo:



Ele irá jogar o valor que o usuário digitar para dentro da variável "nome".



EXPRESSÕES

Expressões Aritméticas

Agora faremos a relação da lógica de programação com a tão temida matemática. Você pode estar se perguntando, qual o sentido dessa relação?

Veja bem, no algoritmo resolvemos problemas humanamente difíceis, custosos ou repetitivos, sendo, em sua maioria, rotinas de cálculos. Portanto a matemática estará inserida em nossas instruções a fim de nos auxiliar a resolver problemas. Por exemplo, um computador consegue fazer a média de $15+18+43+50+2+44+78$ muito mais rápido do que um humano.

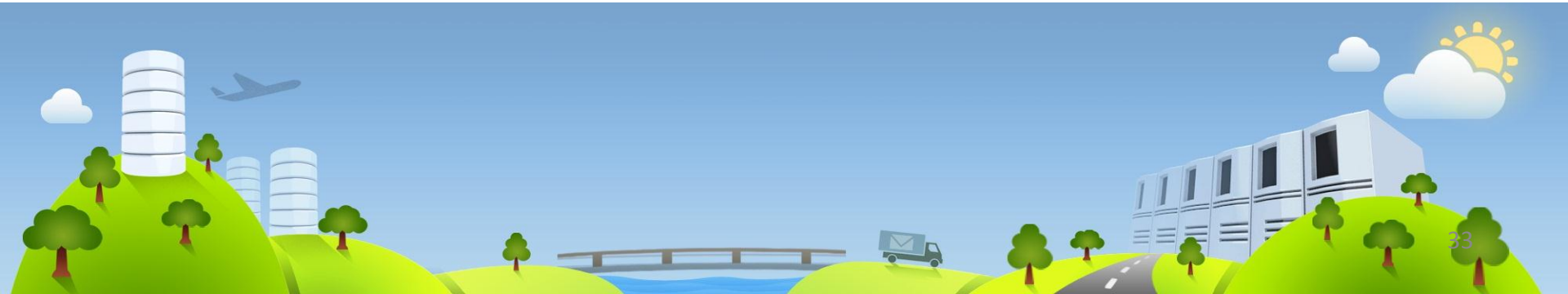
Então, assim como na matemática, expressões aritméticas em algoritmo são basicamente a mesma coisa, ou seja, são um conjunto de símbolos que representam as operações básicas da matemática, que são:

+ Adição

- Subtração

*** Multiplicação**

/ Divisão



Ok, então recapitulamos o que são os operadores e do que se tratam as expressões aritméticas. Agora vamos aos exemplos:

$$2+3+4+1-9 = 1$$

$$10*3-5 = 25$$

$$(3-1)*(3+2)/2 = 5$$

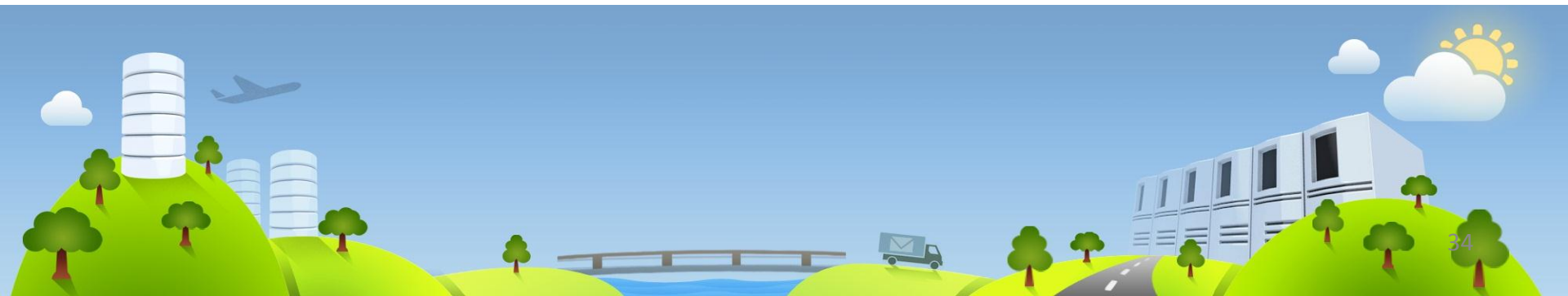
$$22-10*(8-4) = -18$$

Até agora nada de novo, não é? Então, veremos como podemos utilizar essas expressões em nossos algoritmos utilizando as variáveis.

```
var  
n1,n2,n3,media: Real
```

1) Primeiramente, declaramos as variáveis que iremos utilizar no nosso algoritmo.

```
Início  
n1 <- 10  
n2 <- 8  
n3 <- 7.5
```



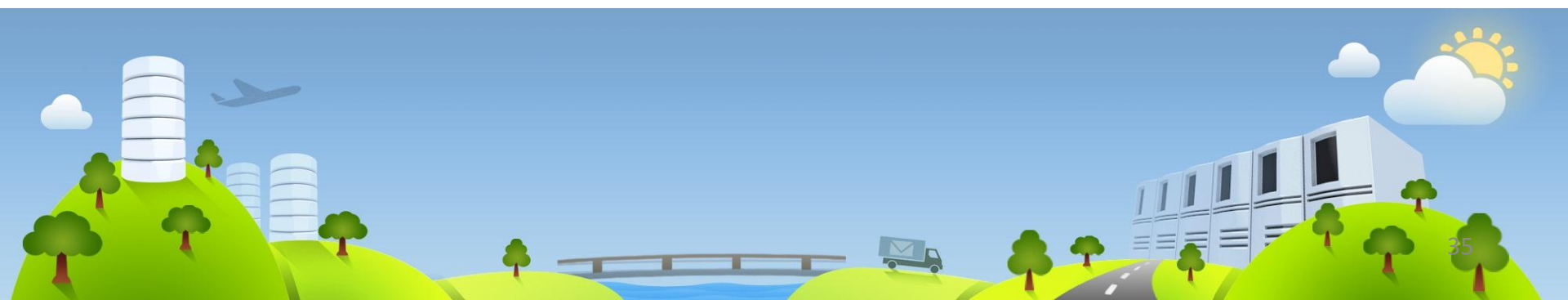
2) Em seguida, inicio meu algoritmo e atribuo os valores que eu quero para as minhas variáveis.

```
media <- (n1+n2+n3) / 3  
escreva(media)  
finalgoritmo
```

3) Feito isso, eu posso criar minha expressão aritmética, no exemplo acima estou fazendo a média de três notas, em seguida mostrando na tela.

Então nosso algoritmo ficará da seguinte forma:

```
algoritmo "aritmética"  
var  
  n1,n2,n3,media: Real  
inicio  
  n1 <- 10  
  n2 <- 8  
  n3 <- 7.5  
  media <- (n1+n2+n3) / 3  
  escreva (media)  
finalgoritmo
```



Expressões aritméticas são aquelas que, utilizando operadores aritméticos, resultam em um valor inteiro ou real. Ou seja, o resultado sempre será um número, seja ele do tipo real ou do tipo inteiro.

Então, vimos como são e como aplicar expressões aritméticas em nossos algoritmos.

Nossa calculadora será baseada em expressões aritméticas com todos os operadores vistos aqui.

Expressões Literais

Expressões literais são expressões de concatenação, ou seja, elas juntam valores por meio do operador de concatenação, que, no VisuAlg, é a vírgula (,).

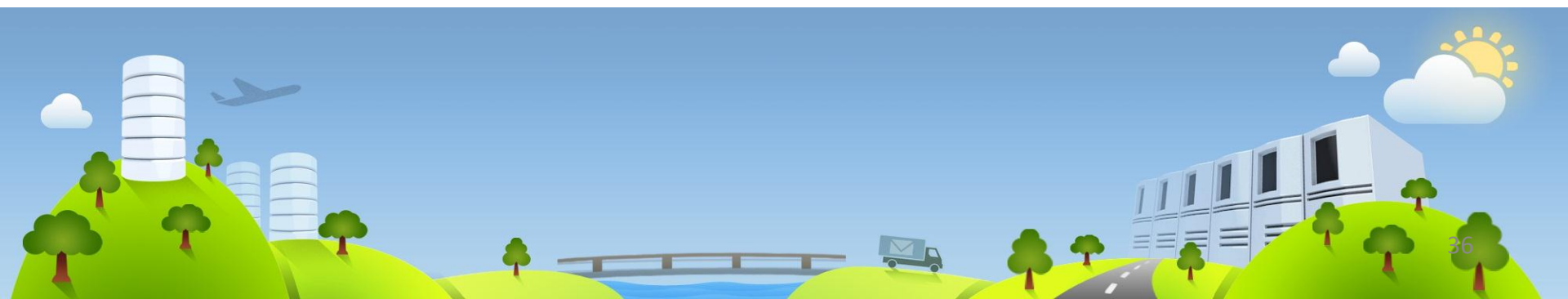
Exemplo:

Meu nome é Anderson

Meu sobrenome é Aguiar

Meu nome completo é Anderson Aguiar

O nome completo é resultado de uma expressão literal.



```

var
  nome, sobrenome, completo: Literal
inicio
  nome <- "Anderson"
  sobrenome <- "Aguiar"
  completo <- nome, " ", sobrenome

```

Expressões literais recebem mais de um valor e resultam SEMPRE em um valor do tipo literal.

```

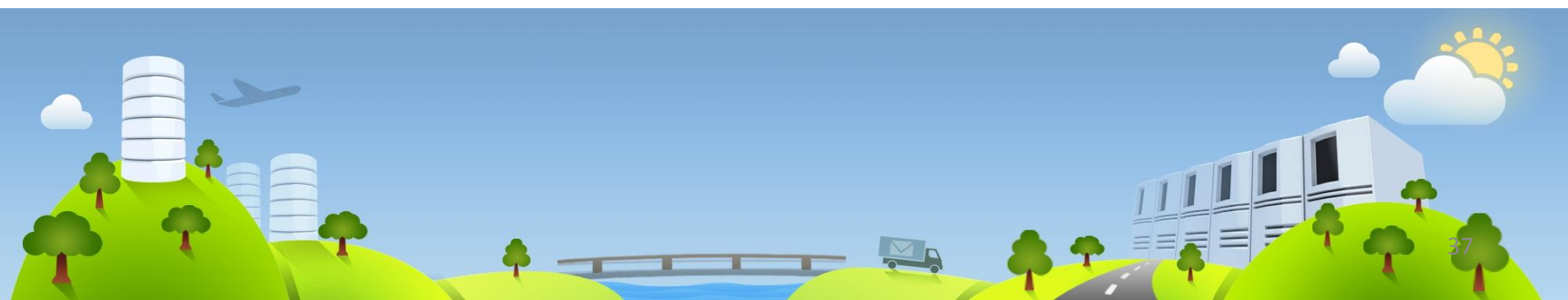
algoritmo "concatenação"
var
  nome: Literal
  idade: Inteiro
inicio
  escreva ("Digite seu nome: ")
  leia(nome)
  escreva ("Qual sua idade: ")
  leia(idade)
  escreva (nome, " você tem", idade, " anos")
finalgoritmo

```

```

Digite seu nome: Anderson
Qual sua idade: 26
Anderson você tem 26 anos
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.

```



Expressões Lógicas

Vamos agora conhecer nosso último tipo de expressão, que são as expressões lógicas.

Essa expressão é a que irá requerer uma atenção especial, pois iremos trabalhar com alguns operadores lógicos, e é de extrema importância que, ao final desta lição, saibamos o que significa cada um deles.

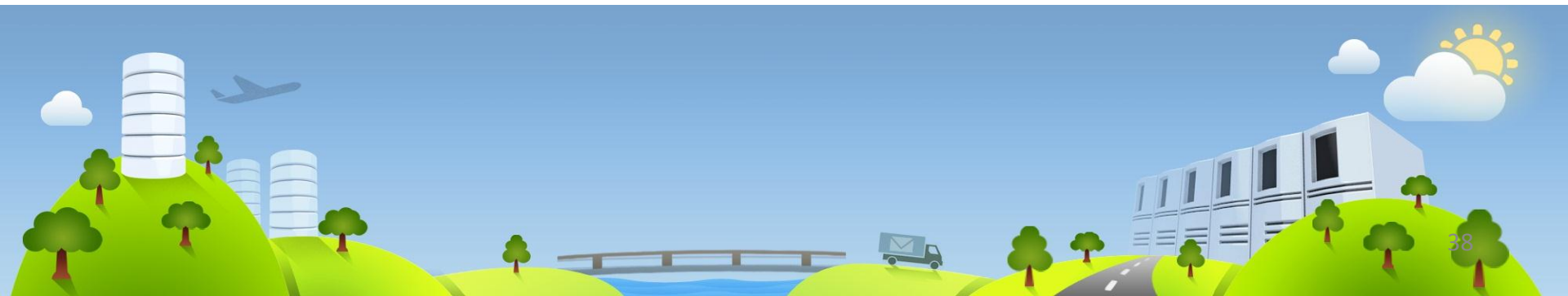
Ansioso para conhecer? Então vamos lá!

As expressões lógicas servem para compararmos valores e dizer se o resultado é VERDADEIRO ou FALSO.

Essas expressões SEMPRE resultam em um dado do tipo lógico. Utilizam os operadores relacionais lógicos (maior, menor, igual, diferente etc.).

Além disso, expressões lógicas podem conter expressões aritméticas dentro delas, veremos a seguir.

É muito importante que consigamos entender cada um dos operadores lógicos.



Operadores relacionais lógicos:

Símbolo	Operador
>	Maior
<	Menor
>=	Maior igual
<=	Menor Igual
=	Igual
<>	Diferente

Exemplo com inteiros:

23 > 42

O inteiro 23 é maior que o inteiro 42?
Falso

84 = 24

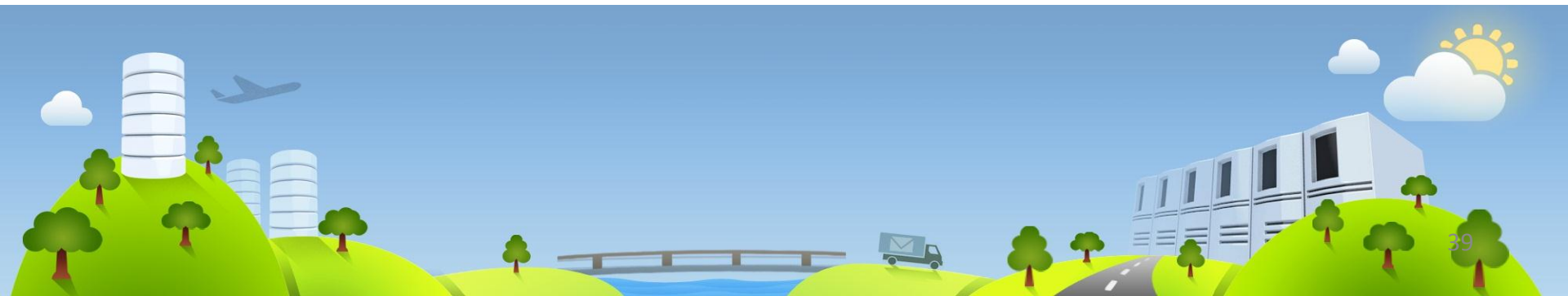
O inteiro 84 é igual ao inteiro 24?
Falso

12 < 12

O inteiro 12 é menor que o inteiro 12?
Falso

44 < 64

O inteiro 44 é menor que o inteiro 64?
Verdadeiro



Exemplo com inteiros:

$$23+10 > 42-2$$

Falso

$$84/2 = 24$$

Falso

$$640+3 \leq 643$$

Falso

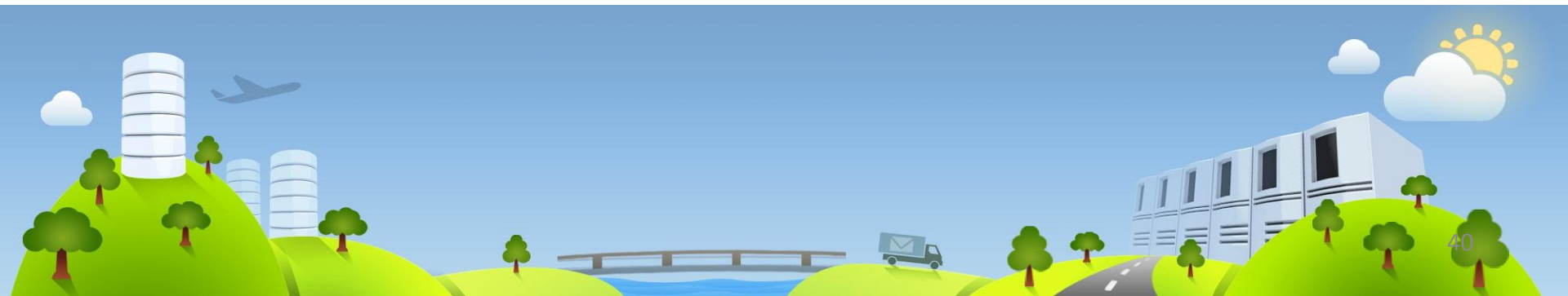
$$54+5 \neq 52-5$$

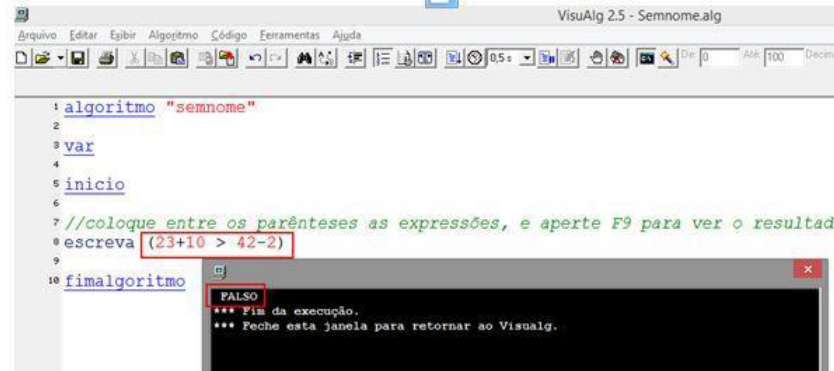
Verdadeiro

Todos esses exemplos nós podemos colocar no VisuAlg e ver se o resultado é realmente o esperado.

Veremos na tela a seguir.

Abra o VisuAlg e coloque as expressões lógicas dentro dos parênteses e aperte a tecla F9 para executar o algoritmo e exibir o resultado.



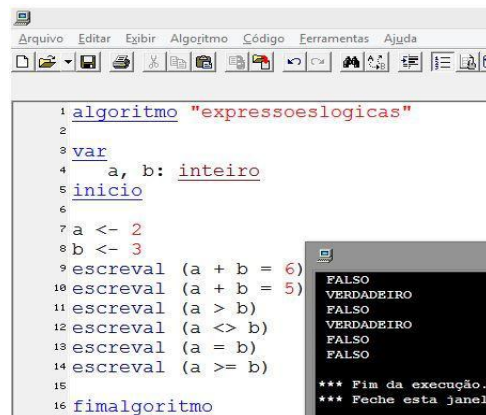


```
1 algoritmo "semnome"
2
3 var
4
5 inicio
6
7 //coloque entre os parênteses as expressões, e aperte F9 para ver o resultado
8 escreva (23+10 > 42-2)
9
10 fimalgoritmo
```

Output: FALSO
*** Fim da execução.
*** Feche esta janela para retornar ao VisuAlg.

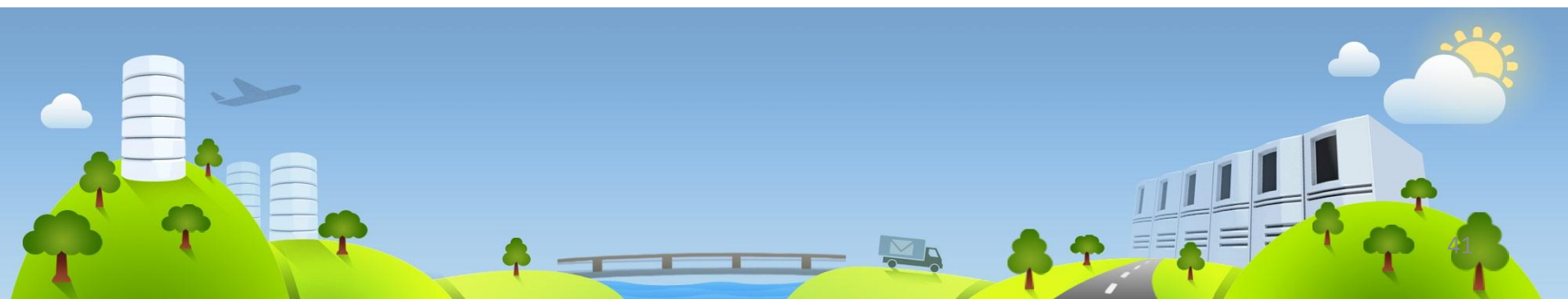
Agora, para fazer o teste dos operadores lógicos, montaremos um pequeno programa que faz a leitura de dois números fazendo as devidas comparações entre eles.

Exemplo com variáveis:



```
1 algoritmo "expressoeslogicas"
2
3 var
4   a, b: inteiro
5 inicio
6
7   a <- 2
8   b <- 3
9   escreval (a + b = 6)
10  escreval (a + b = 5)
11  escreval (a > b)
12  escreval (a <> b)
13  escreval (a = b)
14  escreval (a >= b)
15
16 fimalgoritmo
```

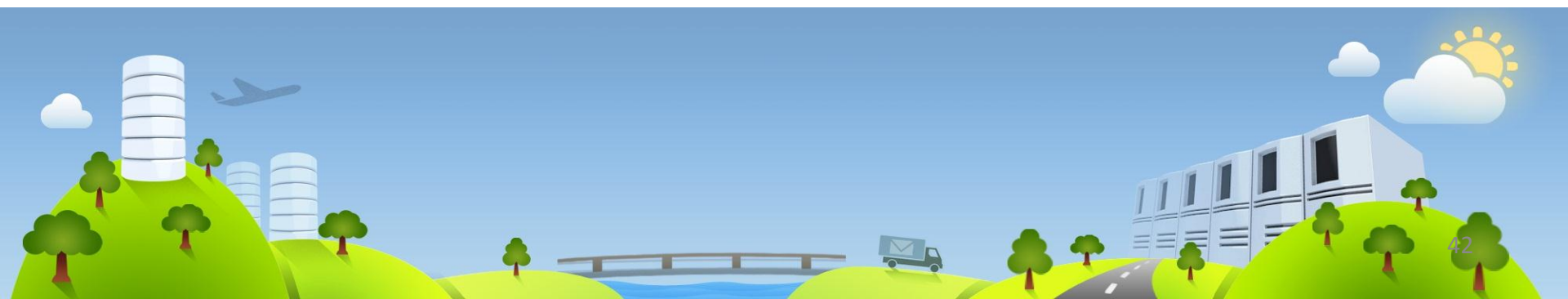
Output: FALSO
VERDADEIRO
FALSO
VERDADEIRO
FALSO
FALSO
*** Fim da execução.
*** Feche esta janela



Expressões lógicas são expressões que recebem dois ou mais valores, além de poder ter outras expressões dentro dela, mas no final elas sempre retornam dados lógicos (VERDADEIRO ou FALSO).

Na tabela abaixo, mostramos de forma detalhada a execução do algoritmo, e, também, o motivo do resultado alcançado.

Expressão 1	Comparador	Expressão 2	Resultado	Motivo
2 + 3	=	6	FALSO	5 é diferente de 6
2 + 3	=	5	VERDADEIRO	5 é igual a 5
2	>	3	FALSO	2 é menor que 3
2	<>	3	VERDADEIRO	2 é diferente de 3
2	=	3	FALSO	2 é diferente de 3
2	>=	3	FALSO	2 não é maior 3



Operadores de sentença: E

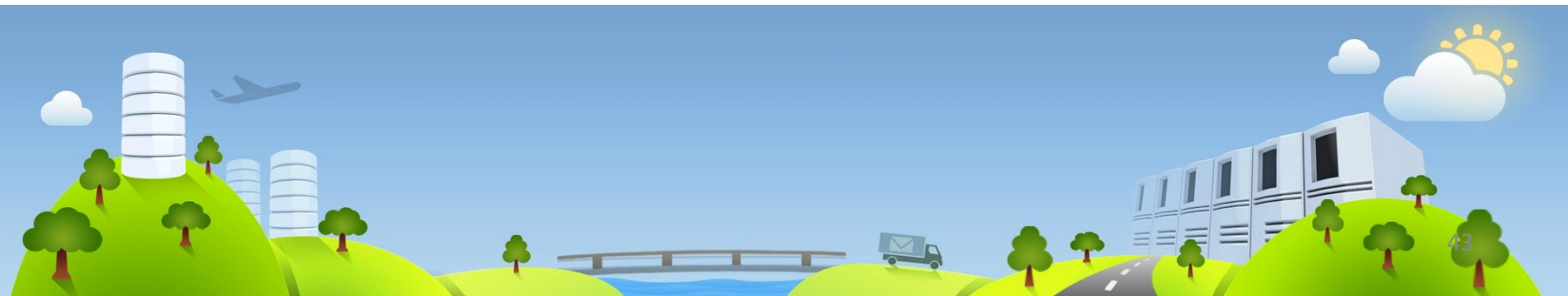
O operador E refere-se à comparação binária de termos, ou seja, $1+1$ E $3-1$ é verdadeiro ou falso? Verdadeiro, ambos os resultados são 2. Remete à combinação verdadeiro + verdadeiro = verdadeiro.

Sentença	Resultado
Verdadeiro (E) Verdadeiro	Verdadeiro
verdadeiro (E) Falso	Falso
Falso (E) Verdadeiro	Falso
Falso (E) Falso	Falso

Sentença	Resultado
0 (=) 0	Verdadeiro
1 (=) 0	Falso
0 (=) 1	Falso
1 (=) 1	Verdadeiro

Exemplo com inteiros:

Operação	Resultado
$(5=5)$ e $(6<8)$	Verdadeiro
$(5>5+9)$ e $(1<7)$	Falso
$(2>1)$ e $(2>=2)$ e $(1<>11)$	Verdadeiro
$(2=1)$ e $(5<>5)$	Falso



Operadores de sentença: OU

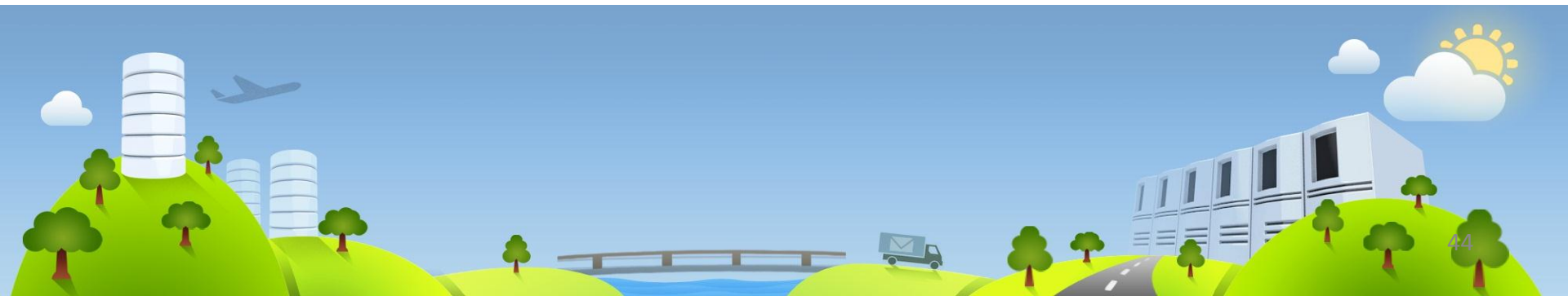
O operador OU refere-se à comparação binária de termos. Remete à combinação verdadeiro + falso = verdadeiro. Se qualquer um dos termos estiver correto a expressão é verdadeira $1+1=2$ OU $3+1=6$?

Sentença	Resultado
Verdadeiro (OU) Verdadeiro	Verdadeiro
verdadeiro (OU) Falso	Verdadeiro
Falso (OU) Verdadeiro	Verdadeiro
Falso (OU) Falso	Falso

Sentença	Resultado
0 (=) 0	Falso
0 (=) 1	Verdadeiro
1 (=) 0	Verdadeiro
1 (=) 1	Verdadeiro

Exemplo com inteiros:

Operação	Resultado
(5=4) ou (6<8)	Verdadeiro
(15>5+9) ou (1<0)	Verdadeiro
(2>1) ou (2>2) ou (1<>11)	Verdadeiro
(2=1) ou (5<>5)	Falso



Colocou todas as expressões no VisuAlg?

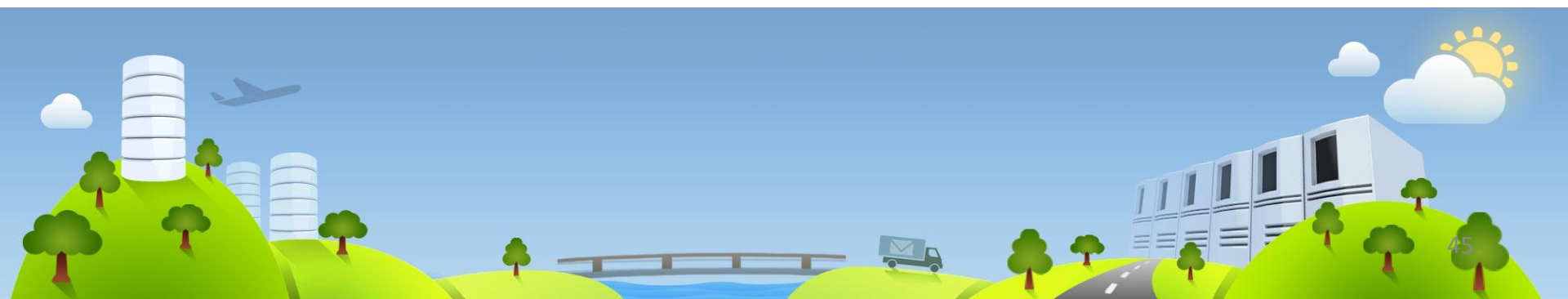
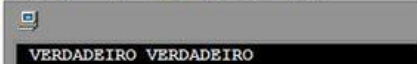
É isso aí! Em nossas perguntas valendo medalha é recomendado que você coloque-as no VisuAlg para obter os resultados.

Serão dadas cinco expressões para serem colocadas no VisuAlg. Com isso você deverá informar seu resultado. (Verdadeiro ou Falso).

Preparado(a) para conquistar mais uma medalha?

Vamos às perguntas então!

```
algoritmo "expressoeslogicas"
var
  a, b: inteiro
inicio
  a <- 4
  b <- 2
  //4 maior que 2? e 2 diferente de 4?
  escreva ((a > b) e (b <> a))
  //4 igual a 2? ou 2 diferente de 4?
  escreva ((a = b) ou (b <> a))
finalgoritmo
```



ESTRUTURA

Estruturas de Condição

Agora iremos conhecer as estruturas de condição, que servem para nos auxiliar na tomada de alguma decisão. São definidas por expressões lógicas, em que o programa irá executar a estrutura de acordo com as respostas (VERDADEIRO ou FALSO).

Considerando um cenário real e cotidiano, elas são as várias decisões que tomamos no nosso dia a dia.

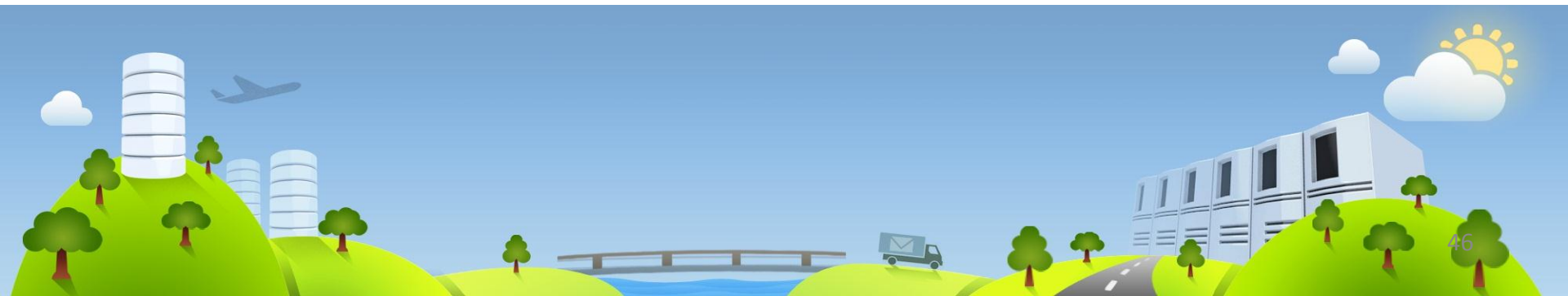
Vamos imaginar que estamos saindo de casa, e, de repente, começa a chover. O que devemos fazer?

Se está chovendo = pegar um guarda-chuva.

Se não está chovendo = não preciso de um guarda-chuva.

Isto foi uma condição que determinará o que eu deveria levar comigo.

Saindo um pouco do cenário real e indo para o ambiente computacional, temos a seguinte tarefa: Cadastrar pessoas maiores de idade no sistema.

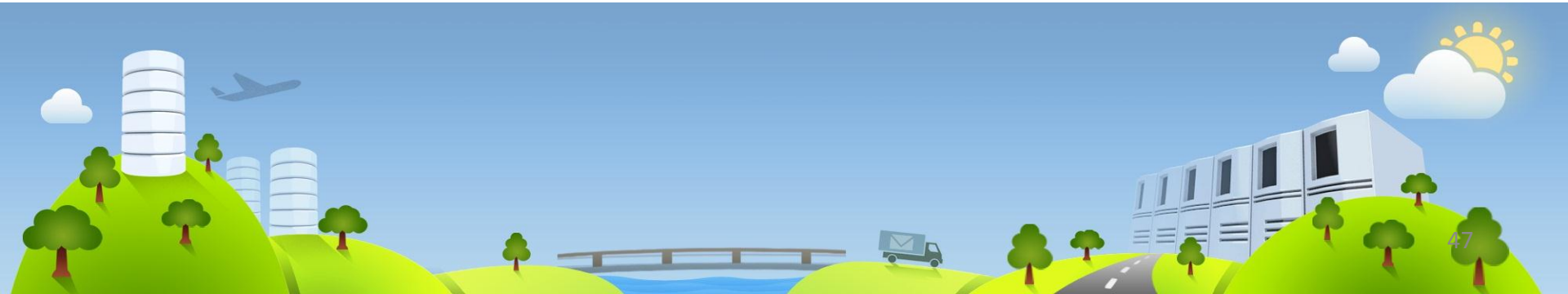


Para isso devemos fazer uma estrutura de condição para verificar a idade digitada. Veja no fluxograma abaixo um exemplo de condição.

No cadastro não podem ser cadastradas pessoas menores de idade.



Demonstrada a importância de fazer comparações em seu algoritmo, iremos evoluir para aprender como faremos em linguagem de máquina, pois o que sabemos até agora é em relação aos operadores, estes, por sua vez, nos auxiliam a comparar. Por exemplo: podemos comparar se o 15 é maior do que 16; o nosso algoritmo irá retornar falso.



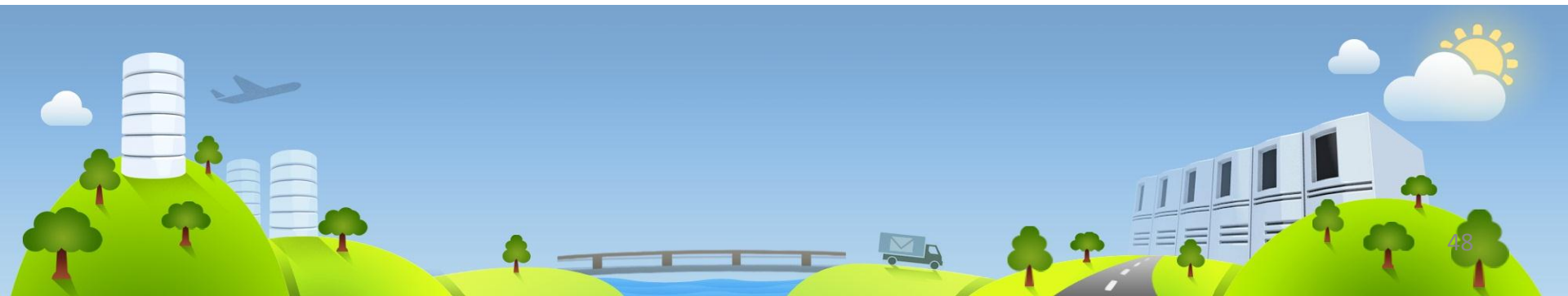
Como escrever uma estrutura condicional em algoritmo?

```
se <expressão lógica> entao  
    //ações para executar caso a expressão seja verdadeira  
fimse
```

```
1 algoritmo "condicionais"  
2  
3 var  
4 idade: Inteiro  
5  
6 inicio  
7     escreva("Informe a sua idade: ")  
8     leia(idade)  
9     se idade >= 18 entao  
10        escreva("Você é Maior de Idade")  
11    fimse  
12 fimalgoritmo
```

```
Informe a sua idade:  
18  
Você é Maior de Idade  
*** Fim da execução.  
*** Feche esta janela p
```

No exemplo anterior, vimos como entrar em uma condição caso ela dê VERDADEIRA, e se for necessário tratar se a expressão for falsa? Para isso utilizamos a cláusula SENAO.



```
se <expressão lógica> entao
    //ações para executar caso a expressão seja verdadeira
senao
    //ações para executar caso a expressão seja verdadeira
fimse
```

```
algoritmo "idade"
var
idade: Inteiro

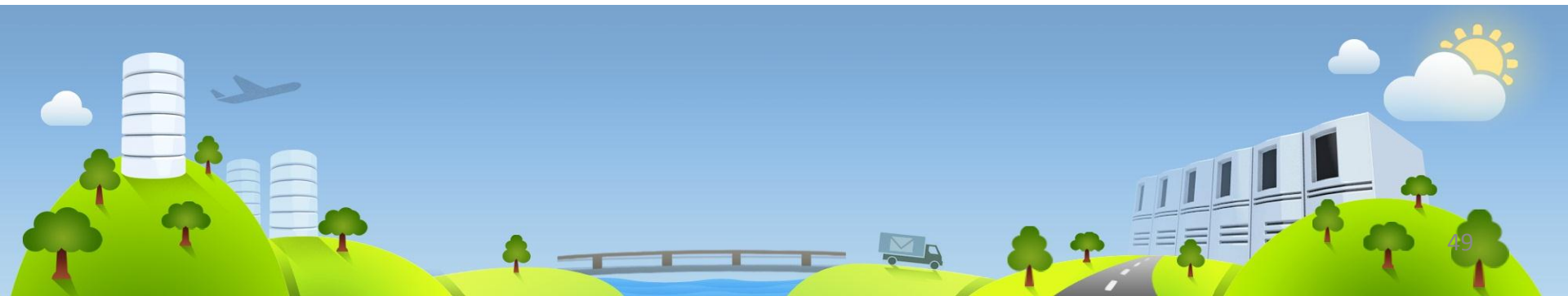
Inicio
    Escreva("Informe a sua idade: ")
    Leia(idade)
    se idade >= 18 entao
        Escreval("Você é Maior de Idade")
    senao
        Escreval("Você é Menor de Idade")
    FimSe
Fimalgoritmo
```

```
Informe a sua idade: 12
Você é Menor de Idade

*** Fim da execução.
*** Feche esta janela ps
```

Nas estruturas condicionais, temos a possibilidade de fazer condições sequenciais/consecutivas sempre que necessário.

Podemos utilizar qualquer expressão lógica, dessa forma, as estruturas de condições servem para controle de ações, valores e variáveis do sistema.



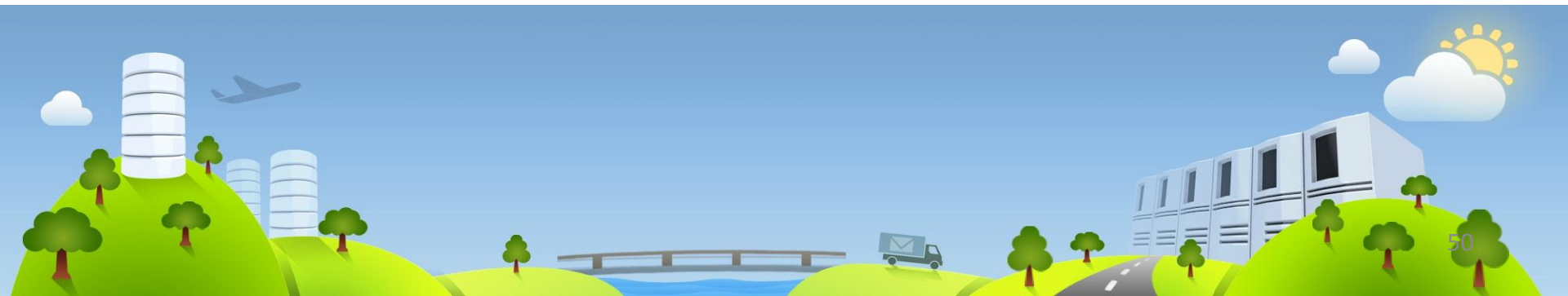


Estruturas de Repetição

Estamos chegando ao final do nosso curso, esta será nossa penúltima lição. Será que você vai conseguir o troféu?!

Acredito que, se você chegou até aqui, não está para brincadeira!

Agora, iremos aprender sobre as estruturas de repetição, elas irão nos economizar MUITO código na hora de escrevermos nossos algoritmos, então vamos lá!



No VisuAlg temos disponíveis três estruturas de repetição para utilizarmos em nossos algoritmos. São elas:

- Para...faça
- Enquanto...faça
- Repita...até

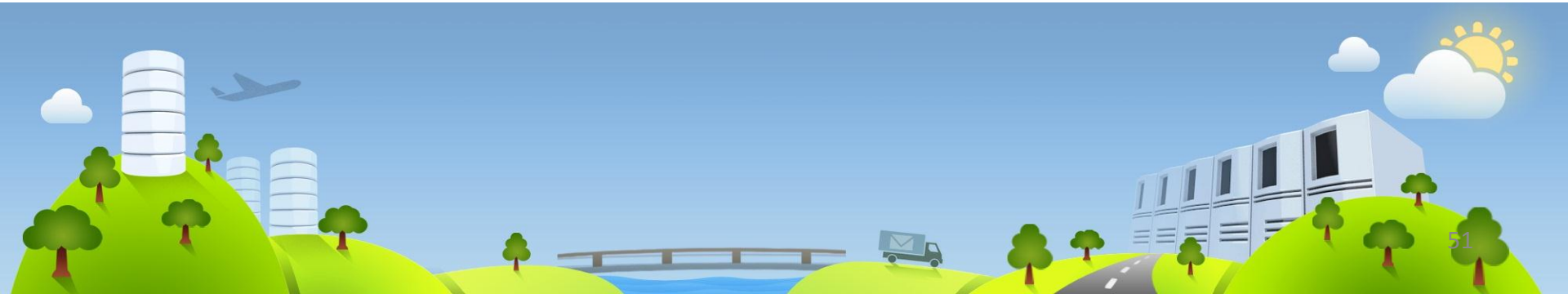
Então o que são exatamente estruturas de repetição?

São conjuntos de comandos que repetem instruções em uma quantidade predeterminada ou não. Ou seja, são blocos de códigos que se repetirão sempre que se enquadrem nos blocos condicionais a eles atribuídos.

Quando utilizaremos?

Sempre que precisarmos fazer coisas repetitivas.

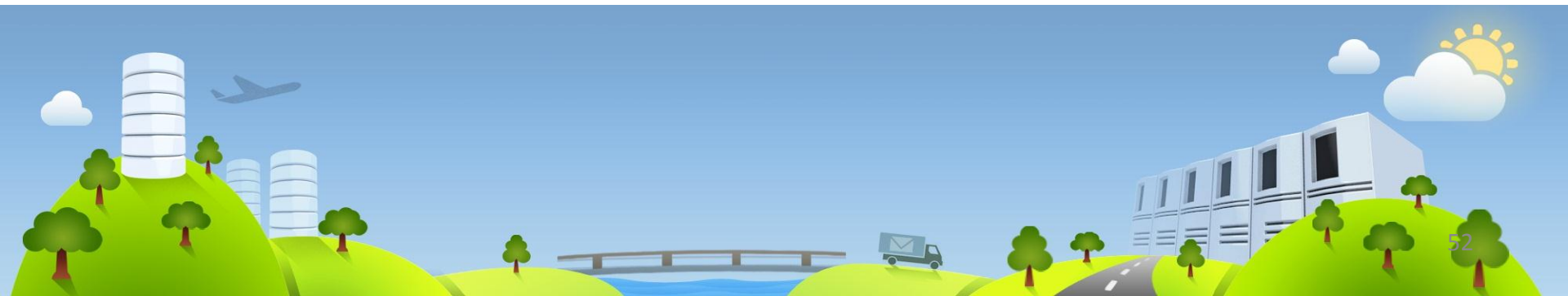
Para exemplificar, usaremos uma variável chamada `cont` para ser o nosso contador de vezes que o algoritmo repetirá um determinado bloco de código.



Vamos utilizar o nosso exemplo de verificar se determinada pessoa é maior ou menor de idade.

```
var
  idade: inteiro
início
  escreva("Informe sua idade:")
  leia(idade)
  se(idade) >= 18 então
    escreva("Você é maior de idade!")
  senão
    escreva("Você é menor de idade!")
fimse
```

Mas, se fossem três pessoas?



Exemplo: Verificar se três pessoas são maiores ou menores de idade.

Início

```
escreva("Informe sua idade:")
leia(idade1)
se(idade1) >= 18 entao
    escreva("Você é maior de idade!")
senao
    escreva("Você é menor de idade!")
fimse
escreva("Informe sua idade:")
leia(idade2)
se(idade2) >= 18 entao
    escreva("Você é maior de idade!")
senao
    escreva("Você é menor de idade!")
fimse
```

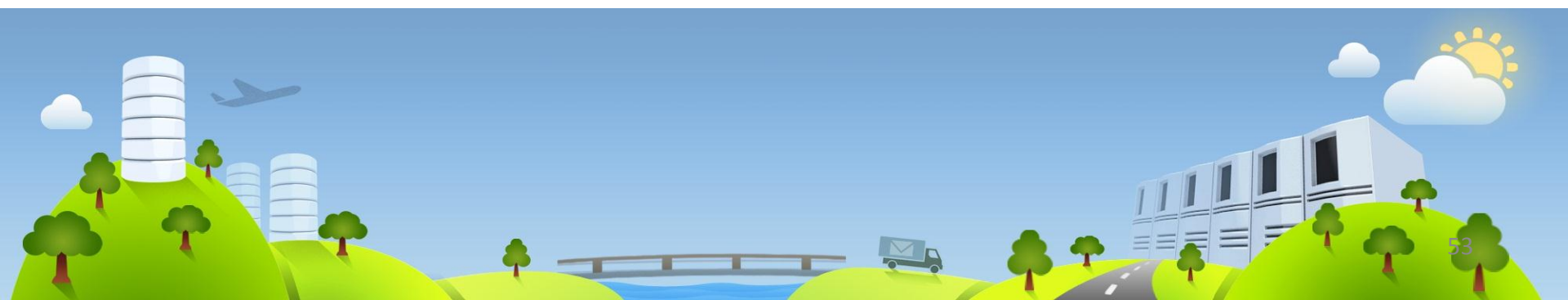
```
escreva("Informe sua idade:")
leia(idade3)
se(idade3) >= 18 entao
    escreva("Você é maior de idade!")
senao
    escreva("Você é menor de idade!")
fimse
fimalgoritmo
```



Note a quantidade de código que tivemos que digitar para verificar a idade de três pessoas. Poderia ser mais simples, não é?

Com certeza!

É aí que entram as estruturas de repetição.



Estrutura de Repetição - Para...faça

A estrutura de repetição para..faça é utilizada da seguinte forma:

```
para variavel* de inicio** ate final*** faca  
    // Instruções para repetir  
fimpara
```

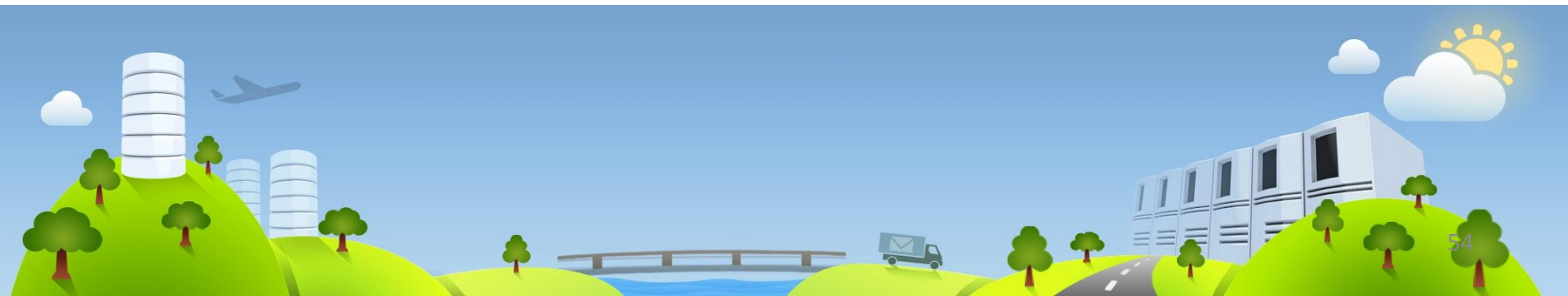
* Nome da variável que irá controlar a repetição

** Valor da variável onde iniciará a repetição

*** Valor da variável onde terminará a repetição

Exemplo:

```
para i de 1 ate 3 faca  
    // Instruções para repetir  
fimpara
```

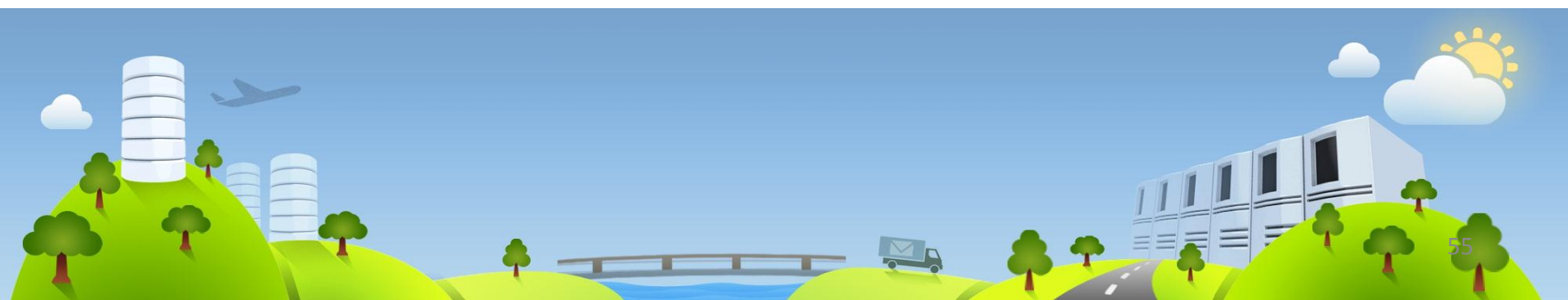


A variável que irá controlar a repetição deve estar definida no bloco de variáveis.

```
algoritmo "semnome"  
var  
  cont, idade: Inteiro  
inicio  
  para cont de 1 ate 10 faça  
    escreval("Informe sua idade:")  
    leia(idade)  
    se(idade) < 20 entao  
      se(idade) < 15 entao  
        escreval("Você é uma criança!")  
      fimse  
    fimse  
  fimpara  
fimalgoritmo
```

Lembra do exemplo que utilizamos lá no início para verificar a idade de três pessoas e dizer se é maior ou menor de idade?

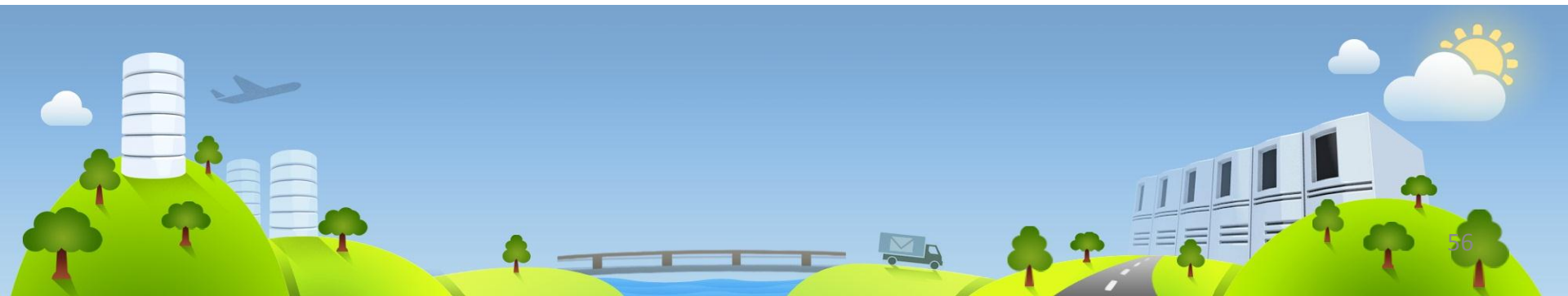
Vejamos como fica utilizando a estrutura de repetição para..faça:



```
var  
  idade,i:inteiro  
inicio  
  para i de 1 ate 3 faca  
    escreva("Informe sua idade:")  
    leia(idade)  
    se(idade) >= 18 entao  
      escreva("Você é maior de idade!")  
    senao  
      escreva("Você é menor de idade!")  
    fimse  
  fimpara  
finalgoritmo
```

Bem melhor, não acha?

Então, a essência para se utilizar as estruturas de repetição é identificar onde há blocos que se repetem mais de uma vez no nosso algoritmo.



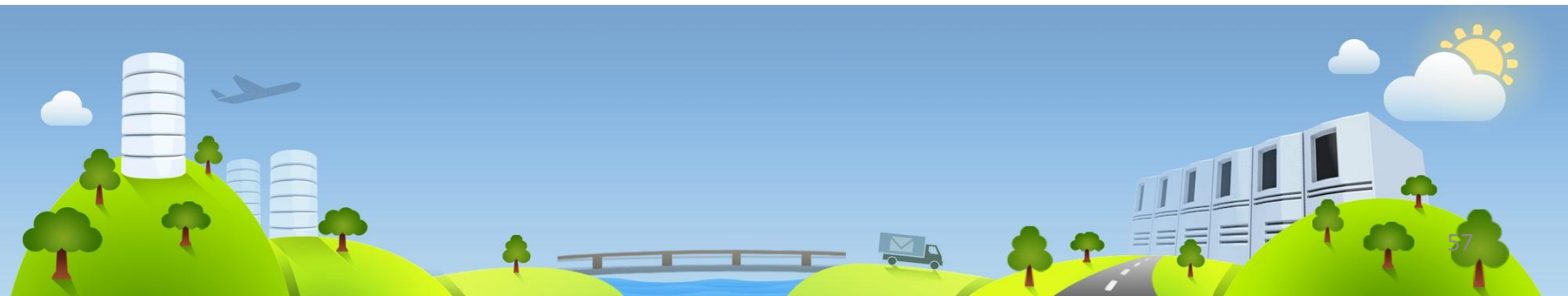
Estruturas de Repetição - Enquanto...faça

Na estrutura "Para Faça", ela repete durante a iteração de um valor, ou seja, repete-se até chegar no número informado. Há outras estruturas que repetem a partir de uma condição (expressão lógica), que é o caso do enquanto..faça. Estrutura "Enquanto Faça":

```
enquanto <expressão lógica> faça  
    // Comandos para repetir  
fimenquanto
```

Os comandos são executados enquanto a expressão lógica for verdadeira.

```
algoritmo "triplo"  
var  
    numero: Inteiro  
inicio  
    numero <- -1  
    enquanto ( numero <> 0) faça  
        escreva("Digite um número: ")  
        leia(numero)  
        escreva("Triplo: ", numero*3)  
    fimenquanto  
fimalgoritmo
```



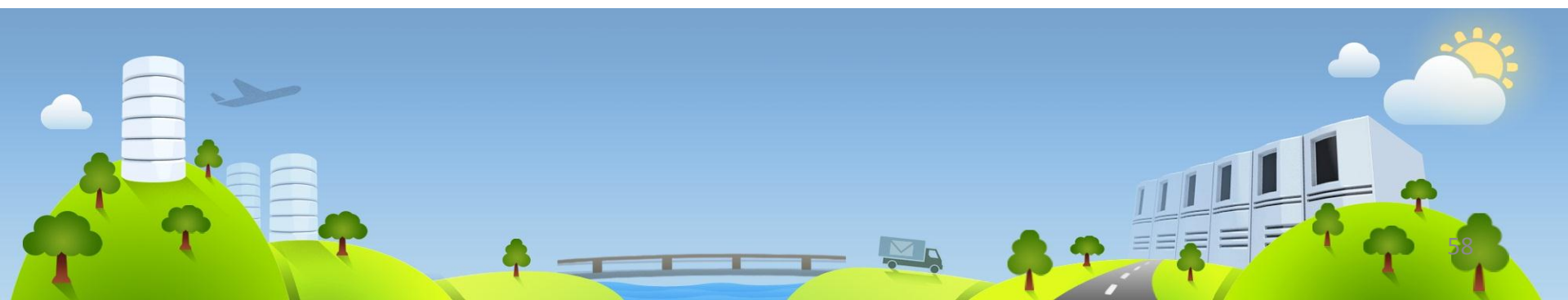
Estruturas de Repetição - Repita...até

A estrutura "Repita...até" segue o mesmo princípio da "Enquanto Faça", porém a verificação condicional é realizada no final da repetição, deste modo pelo menos uma vez a estrutura é executada.

```
repita  
    // Comandos que serão repetidos  
até <expressão lógica>
```

Um algoritmo que calcula a soma de um número digitado e mostre para o usuário até ele sair do programa.

```
algoritmo "repita"  
var  
    op:Literal  
    num, soma:Real  
  
início  
    soma <- 0  
    repita  
        escreva("Informe o número para o cálculo:")  
        leia(num)  
        soma <- soma + num  
        escreva("Soma = ", soma)  
        escreva("Deseja continuar? (s/n) ")  
        leia(op)  
    até ( op = "n" ) OU op = "N"  
fimalgoritmo
```



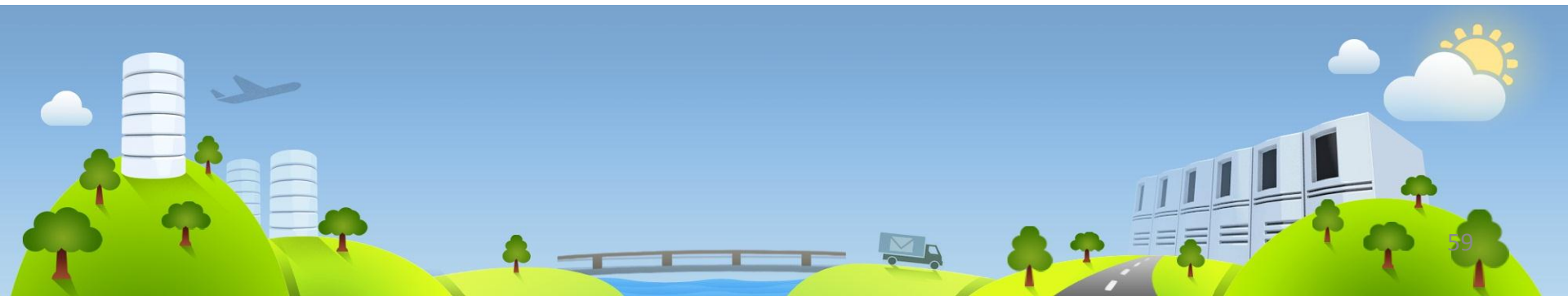
Variáveis Indexadas

Variáveis indexadas são variáveis que contêm mais de um espaço alocado em memória. Variáveis comuns possuem apenas um valor.

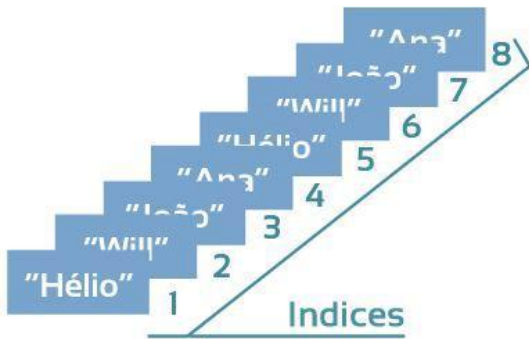
Idade Inteiro	Nome Literal	Salário Real	Estado Civil Lógico
15	"Ana"	784.0	Falso

Variáveis indexadas possuem mais de um valor.

Idade Inteiro	Nome Literal	Salário Real	Estado Civil Lógico
23 54 15	"Ana" "João" "Hélio"	734.21 734.43 738.40	Verdadeiro Falso Falso

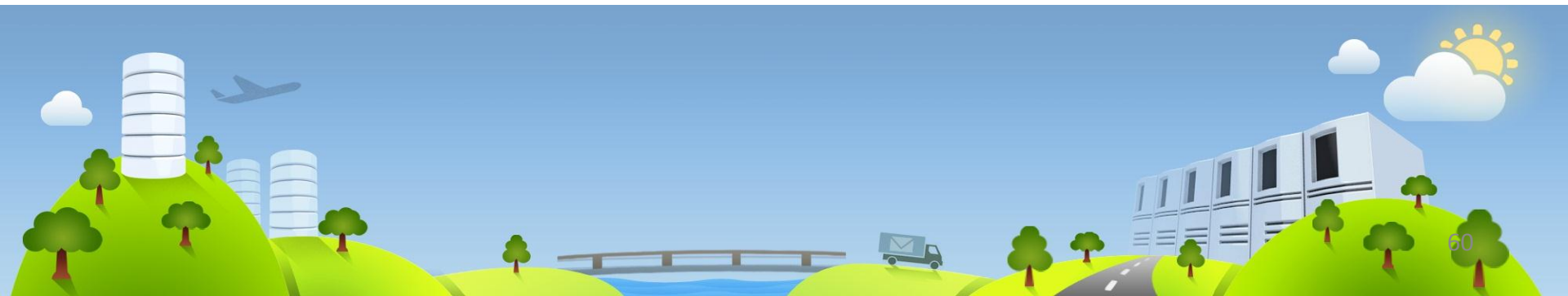


Os valores contidos nas variáveis indexadas são organizados por índices. Ou seja, são identificados por um número que corresponde ao seu endereço em memória.



Vamos considerar que estamos criando um algoritmo que precise cadastrar oito nomes de pessoas, três salários e seis idades. Para declarar essas variáveis, seria mais ou menos dessa forma:

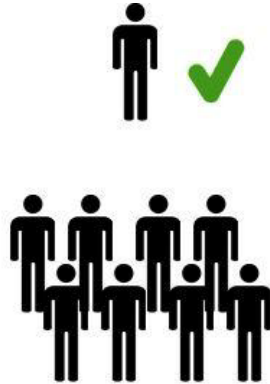
```
var  
nome1, nome2, nome3, nome4, nome5, nome6, nome7, nome8 : Literal  
salario1, salario2, salario3 : Real  
idade1, idade2, idade3, idade4, idade5, idade6 : Inteiro
```



Correto? Teríamos que criar várias variáveis para receber os diversos valores. E, se eu não soubesse o limite? Se fossem 100 variáveis, 1000 variáveis? Já pensou no trabalho que teríamos? É aí que entram as variáveis com índices, as chamadas variáveis indexadas.

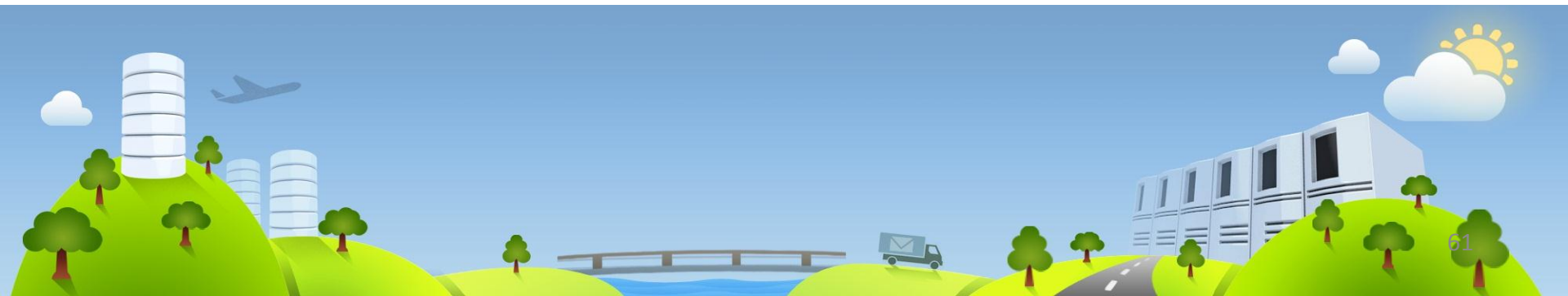
É aí que entram as variáveis com índices, as chamadas variáveis indexadas.

Vamos ver como elas funcionam?



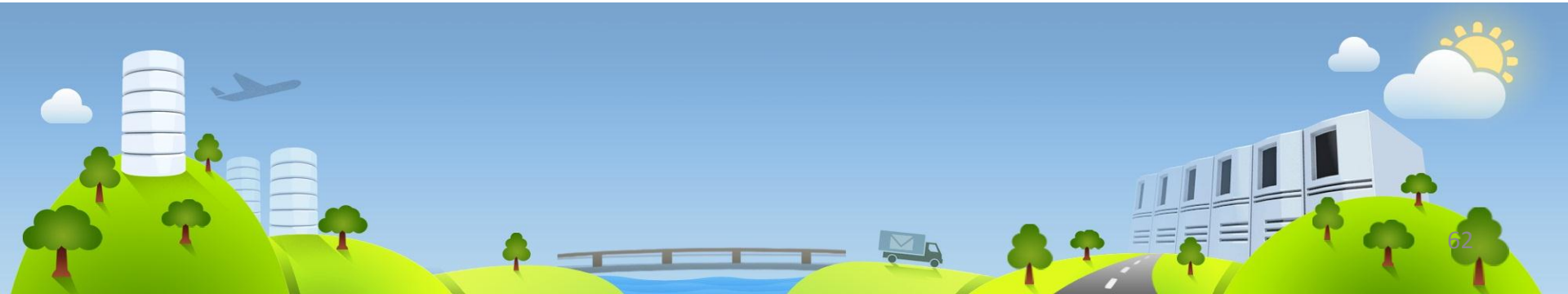
Então uma variável indexada deve ser declarada da mesma forma que uma variável simples, como já foi visto na lição de variáveis.

Sua declaração ficaria da seguinte forma:




```
var  
nomes: vetor [1..8] de Literal  
salarios: vetor [1..3] de Real  
idades: vetor [1..6] de Inteiro
```

Melhor? Menos códigos? Economizamos mais códigos e deixamos bem mais organizadas nossas variáveis.



REVISÃO

Antes de mostrar tudo que você aprendeu no curso, que tal fazer uma breve revisão de tudo que vimos? Topa? Então vamos lá!

Representação de um algoritmo

Temos dois tipos de representações, que são:

Fluxograma e Pseudocódigo

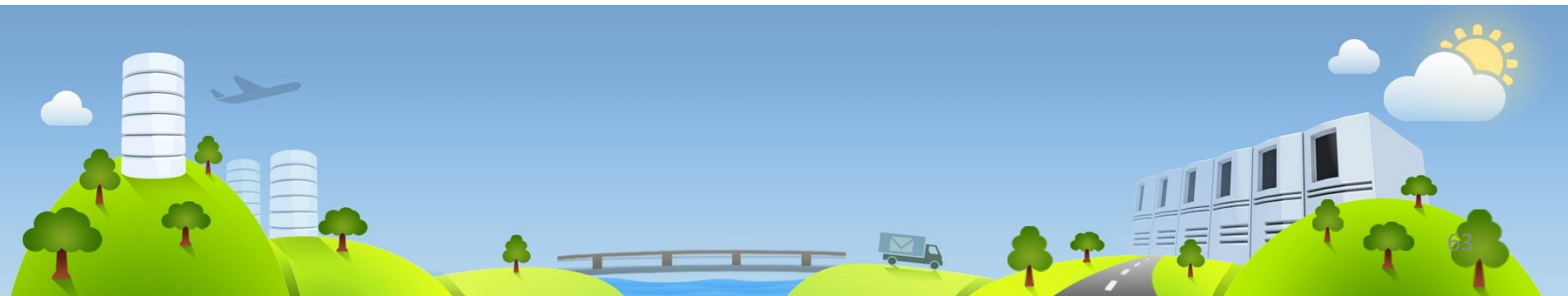
Fluxograma é representado por formas gráficas, e o pseudocódigo é representado como uma narrativa do que está acontecendo, ele é a ponte entre a linguagem falada e a linguagem de máquina(programação).

Tipo de dados

Temos quatro tipos principais de dados, que são:

inteiro, real, lógico e literal.

INTEIRO	Valores não decimais	1, 20, -39, 3324, -32, 0
REAL	Valores decimais ou inteiros	32.3, -34.3, 20, 132.34, 34.0, -3
LITERAL	Sequência de caracteres	"SENAI", "34", "AULA 1"
LÓGICO	Valores lógicos	VERDADEIRO ou FALSO



Inteiro: aceitam somente números inteiros positivos e negativos e NÃO aceitam números reais. Ex.: 0 1 4 35 -54.

Reais: aceitam valores positivos e negativos Podem ser números reais, ou inteiros. Ex.: 1.3 2 4.0 -5.4.

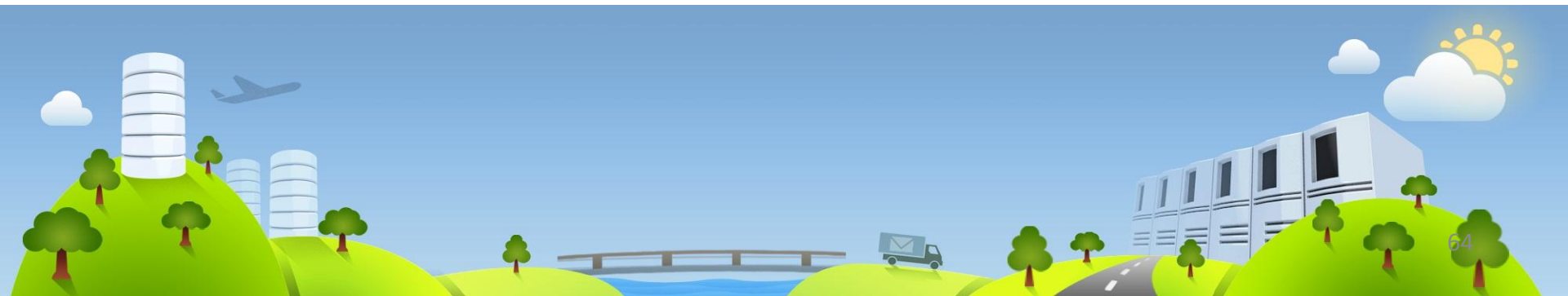
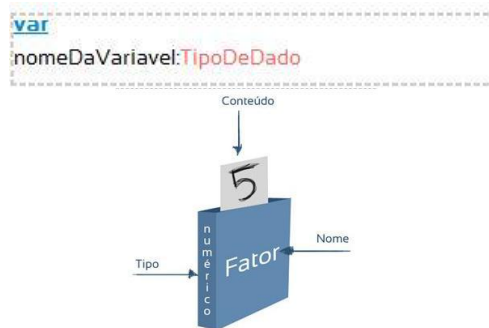
Literal: podem ser letras, números ou caracteres especiais. Ou seja, pode ser qualquer tipo de valor desde que esteja entre aspas duplas. Ex.: "Algoritmo" "Senai" "3432" "43Ab" "&sp3c1@l".

Lógico: são respostas para uma pergunta, em que deverá ter apenas duas possíveis respostas: sim ou não. Porém sempre resultam como VERDADEIRO ou FALSO. Exemplo: Está Chovendo? Sim (Verdadeiro), Maior de idade? 0 (Falso).

Variáveis

O que são: espaços de memória do computador destinados ao armazenamento de dados.

Como usar: no programa, eu devo dar um nome a ela e informar qual o seu tipo de dado. E, no decorrer do algoritmo, atribuir o valor a ela quando necessário.



Expressões Aritméticas

Podemos trabalhar com as principais operações matemáticas, que são:

Símbolo	Operador
+	Adição
-	Subtração
*	Multiplificação
/	Divisão

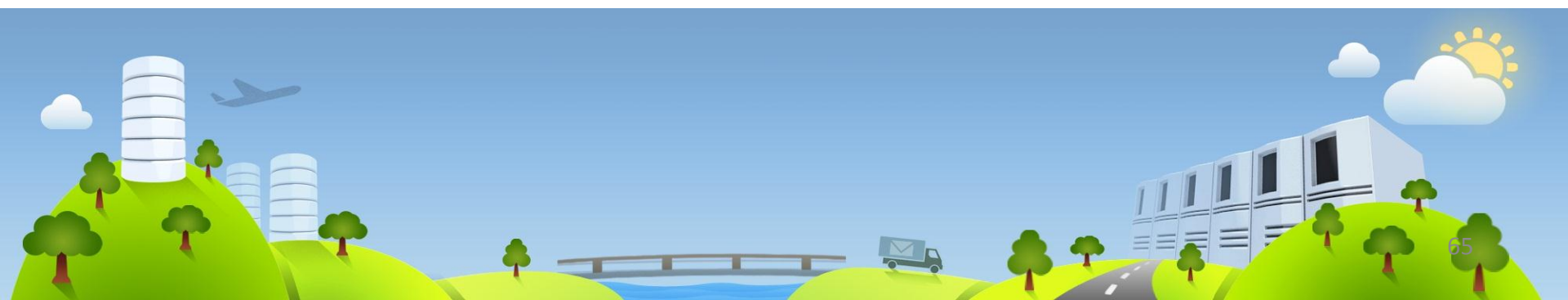
Expressões aritméticas são as expressões que, utilizando operadores aritméticos, resultam em um valor inteiro ou real. Ou seja, o resultado sempre será um número, seja ele do tipo real ou do tipo inteiro.

Exemplos:

$$2+3+4+1-9=1$$

$$10*3-5=25$$

$$(3-1)*(3+2)/2=5$$



Expressões Literais

Expressões literais são expressões de concatenação, ou seja, elas juntam valores através do operador de concatenação que, no VisuAlg, é a virgula (,).

Exemplos:

Meu nome é Anderson

Meu sobrenome é Aguiar

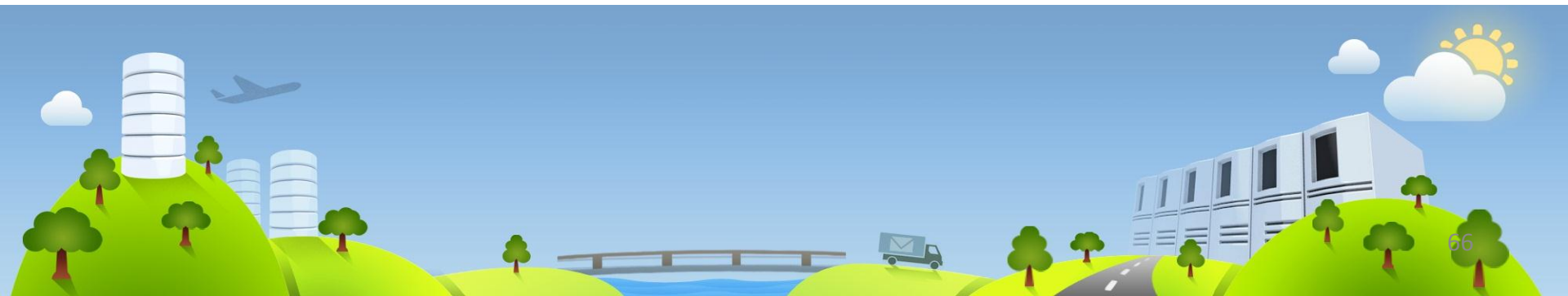
Meu nome completo é Anderson Aguiar

Expressões Lógicas

As expressões lógicas servem para compararmos valores e dizer se o resultado é VERDADEIRO ou FALSO.

No programa, eu devo dar um nome a ela e informar qual o seu tipo de dado. E, no decorrer do algoritmo, atribuir o valor a ela quando necessário.

Além disso, expressões lógicas podem conter expressões aritméticas dentro delas.



Operadores relacionais lógicos:

Símbolo	Operador
>	Maior
<	Menor
>=	Maior igual
<=	Menor Igual
=	Igual
<>	Diferente

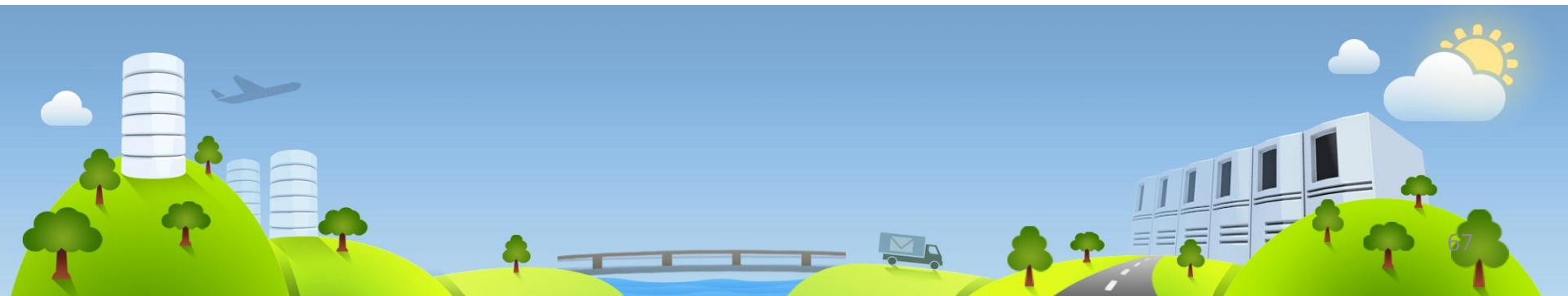
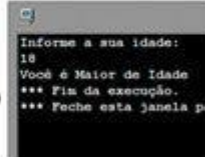
Estrutura de Condição

São ações que devemos tomar, e são definidas por expressões lógicas, em que o programa irá executar a estrutura de acordo com as respostas (VERDADEIRO ou FALSO).

Se está chovendo = pegar um guarda-chuva.

Senão = não preciso de um guarda-chuva.

```
1 algoritmo "condicionais"
2
3 var
4 idade: Inteiro
5
6 inicio
7   escreva("Informe a sua idade: ")
8   leia(idade)
9   se idade >= 18 entao
10    escreva("Você é Maior de Idade")
11  fimSe
12 fimalgoritmo
```



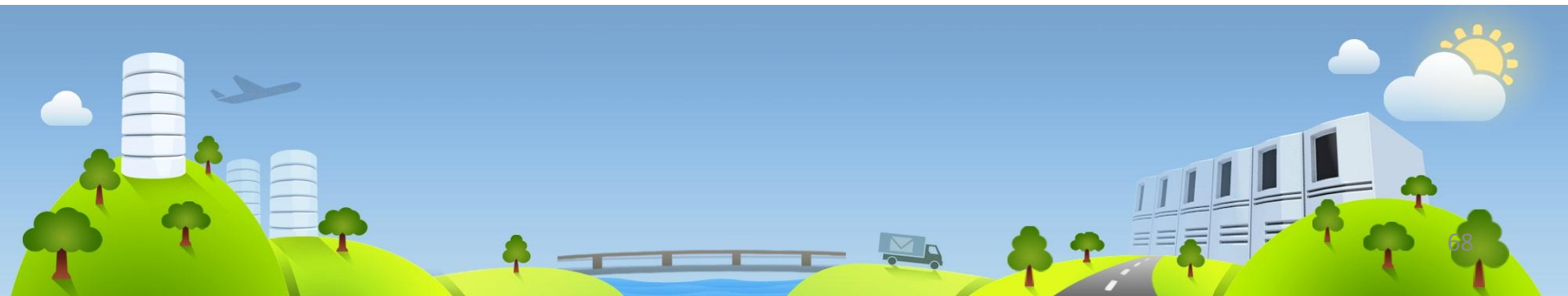
Estrutura de Repetição

Temos três principais estruturas de repetição, que são:

- Para...faça
- Enquanto...faça
- Repita...até

O que são? São conjuntos de comandos que repetem instruções em uma quantidade determinada ou não. Ou seja, são blocos de códigos que se repetirão sempre que se enquadrarem nos blocos condicionais a eles atribuídos.

Quando utilizaremos? Sempre que precisarmos fazer coisas repetitivas.



Variáveis Indexadas

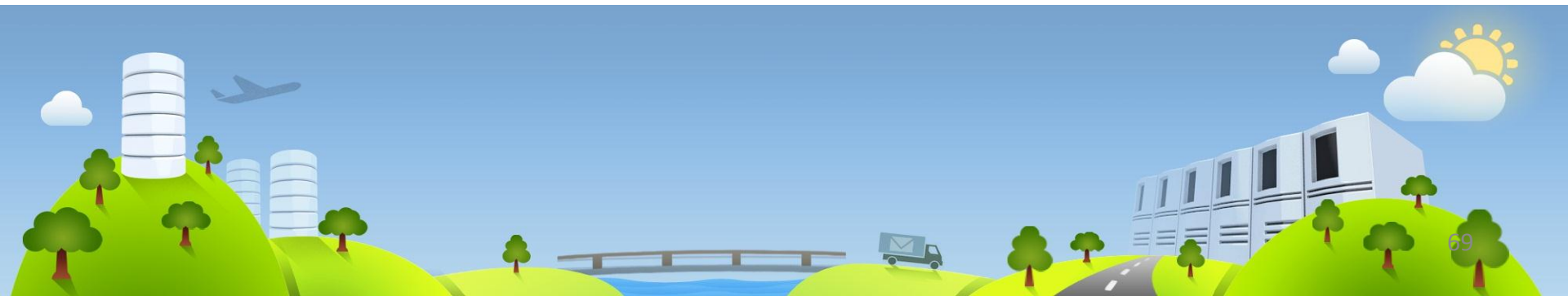
Variáveis indexadas são variáveis que contém mais de um espaço alocado em memória.

```
[tamanhoInicial..tamanhoFinal]: tipoDeDado
```



Exemplo:

```
var
  nomes:vetor[1..8] de Literal
  salarios:vetor[1..3] de Real
  idades:vetor[1..6] de Inteiro
inicio
  nomes[1] <- "Anderson"
  nomes[2] <- "Senai"
  nomes[3] <- "João"
  nomes[4] <- "Maria"
  escreva(nomes[1]) //irá imprimir "Anderson"
```



Conteúdo Extra

Funções

Imagine que as funções são um algoritmo dentro de outro, que irão nos auxiliar a executar alguma tarefa comum.

Exemplo:

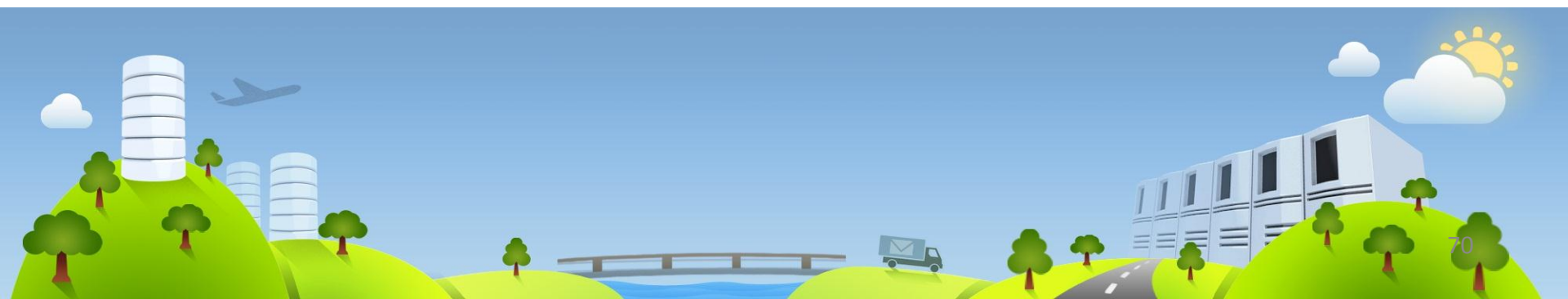
- verificar se determinada variável está preenchida;
- imprimir os números de 1 a 10;
- verificar se determinada idade pertence a uma pessoa;
- maior ou menor de idade.

As vantagens de se utilizar funções são a reutilização e a economia de código que teremos, ou seja, não precisamos ficar digitando sempre o mesmo bloco de código para imprimir os números de 1 a 10 por exemplo, basta chamar a função que faz isso.

Pelo jeito você gostou mesmo de trabalhar sua lógica, não é mesmo? Você é um campeão só por ter chegado até aqui.

Então, como recompensa, você conhecerá o mundo das funções, conteúdo bônus para agregar ainda mais ao seu conhecimento.

Então, aqui, iremos entender como funcionam os subalgoritmos ou funções.



Vamos ao que interessa: O conteúdo!

Existem dois tipos de funções:

1. Funções predefinidas ou nativas

São funções que já existem na linguagem, são criadas com o propósito de nos poupar tempo e trabalho. Ex.: raiz quadrada, potencialização, quantidade de caracteres em determinado literal.

2. Funções definidas pelo usuário

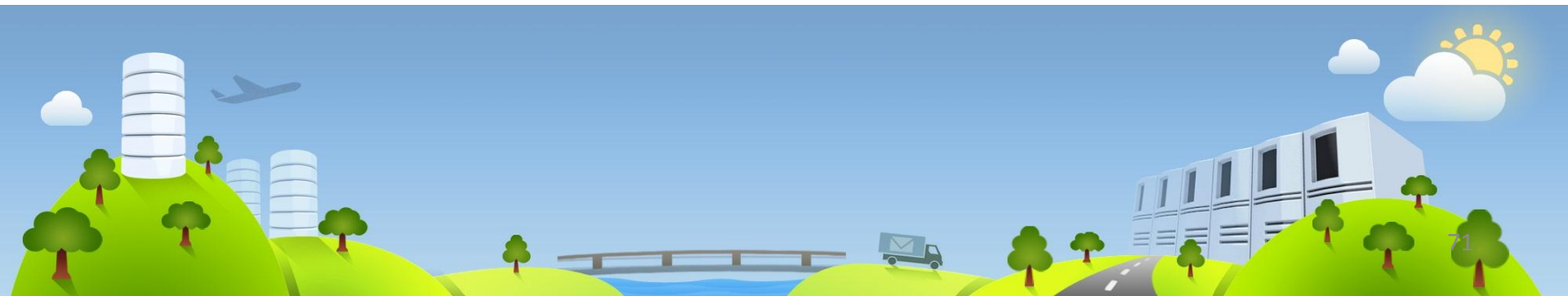
Como o próprio nome já diz, são funções que o usuário irá criar. A maioria das linguagens de programação nos dá a possibilidade para ter tal autonomia, o que facilita e muito o nosso trabalho. Ex.: somar um número, emitir mensagens, verificar se um número é maior-menor-igual a outro, entre outros.

No Visualg


Agora vamos ver na prática como elas funcionam.

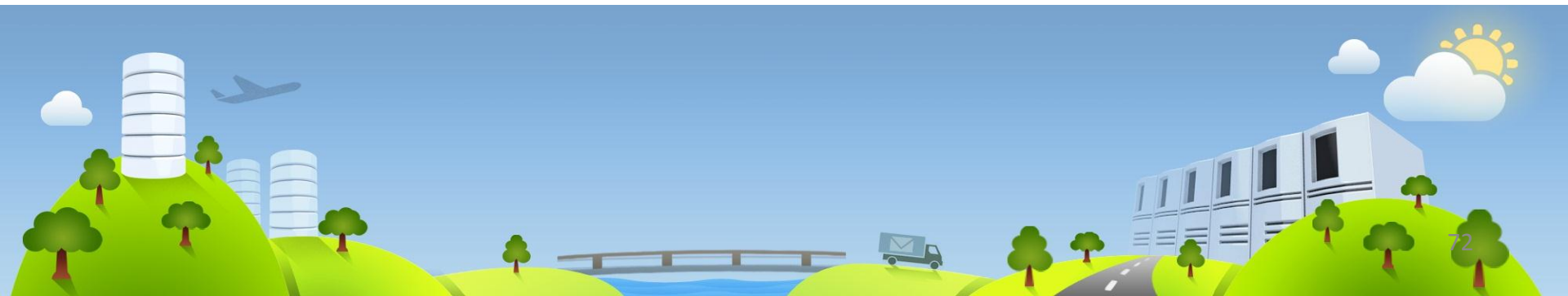
Vamos começar pelas funções nativas e depois veremos as definidas por nós.

No VisuAlg, está disponível uma série de funções que não temos a necessidade de criar.



Numpcarac(n: inteiro ou real): caractere	Converte um numero inteiro ou real para caractere
PI: real	Valor de Pi
Pos(sub,c: caractere): inteiro	Retorna a posição do caractere
Quad(valor: real): real	Elevado ao quadrado
Radpgrau(valor: real): real	Converte Radiano para grau
Raizq(valor: real): real	Raiz quadrada
Rand: real	Gerador de números aleatórios entre 0 e 1
Randi(limite: inteiro): inteiro	Gerador de números aleatórios inteiros com um limite determinado
Sen(valor: real): real	Seno
Tan(valor: real): real	Tangente

COMPETÊNCIAS TRANSVERSAIS
LÓGICA DE PROGRAMAÇÃO




Funções

Para utilizá-las basta chamar qualquer uma delas. Para exemplificar, utilizaremos uma função que sorteia números, como listado abaixo:

```
algoritmo "funcoesnativas"  
inicio  
  //utilizará a função nativa de sorteio.  
  //Sorteará números de 0 à 10  
  escreval( randi(11))  
fimalgoritmo
```

Faça o teste escrevendo esse algoritmo no VisuAlg. Agora vamos entender as funções definidas pelo usuário.

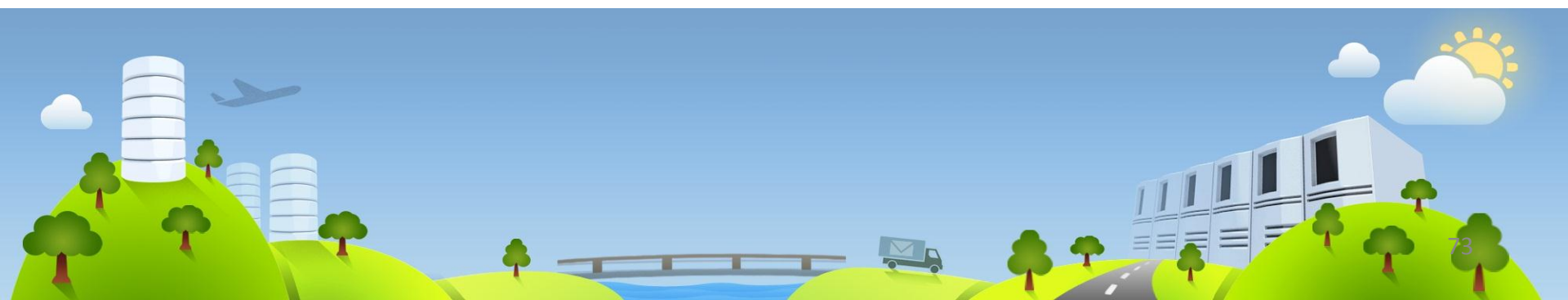
Como são compostas

Identificador: nome da função.

Parâmetros: valores que nossa função irá receber.

Tipo de retorno da função: real, inteiro, lógico, literal.

Declaração de variáveis locais: idêntica à declaração de variáveis globais. As variáveis declaradas localmente têm validade apenas dentro do escopo da função, ou seja, fora dela não conseguimos utilizá-la.



Retorne: local onde é colocada a variável de retorno que enviará o valor para onde a função foi chamada.

Função <identificador>(<lista de parâmetros>: <tipo dos parâmetros>): <tipo de retorno>.

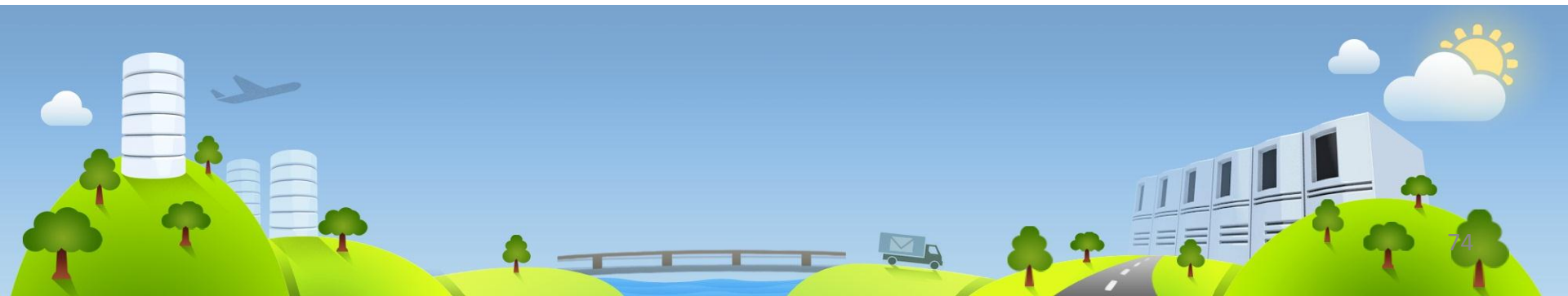
Var <declaração das variáveis locais da função> //início da execução da função.

Início <instruções> //valor a ser retornado no algoritmo, ao chamar a função.

Retorne <valor de retorno>fimfuncao

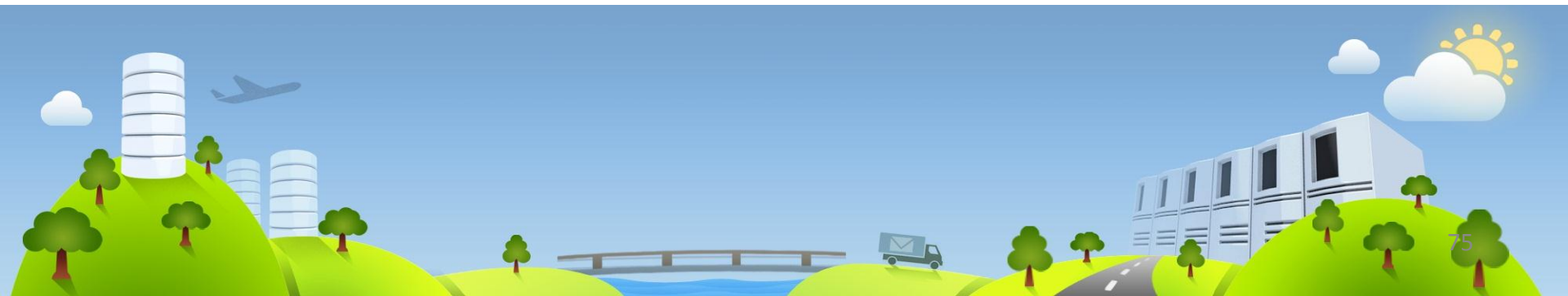
Vamos utilizar o nosso exemplo de verificar se determinada pessoa é maior ou menor de idade.

```
algoritmo "funcoes"
var
  numero1, numero2: inteiro
  //aqui começa a nossa função, é aqui que ela é declarada
  funcao somar(numero1, numero2: inteiro): inteiro
  var
  inicio
    retorne numero1+numero2
  fimfuncao
```



```
início  
leia(numero1)  
leia(numero2)  
//aqui começa a nossa função, é aqui que ela é declarada  
escreval( somar(numero1,numero2))  
fimalgoritmo
```

Agora que vimos o que são e como funcionam as funções, vamos exercitar!



**Após concluir seus estudos, acesse o ambiente virtual para
realizar o Desafio Final!**

