UNIVERSIDADE FEDERAL DE GOIÁS
CURSO ENGENHARIA DE SOFTWARE

CONSTRUÇÃO DE SOFTWARE

PAULO ROBERTO VIEIRA

Linguagens de Programação

C, Java, R, Python

JavaScript

Goiânia

2022

C

C evoluiu de duas linguagens de programação anteriores, BCPL e B. A BCPL foi desenvolvida em 1967 por Martin Richards como uma linguagem para escrever software de sistemas operacionais e compiladores. Ken Thompson modelou muitas das características de sua linguagem B inspirado por suas correspondentes em BCPL, e utilizou B para criar as primeiras versões do sistema operacional UNIX no Bell Laboratories, em 1970. Tanto a BCPL como a B eram linguagens 'sem tipo', ou seja, sem definição de tipos de dados — todo item de dados ocupava uma 'palavra' na memória, e o trabalho de tratar um item de dados como um número inteiro ou um número real, por exemplo, era de responsabilidade do programador. Ao programar em C, você normalmente utilizará os seguintes blocos de montagem: Funções da biblioteca-padrão de C. Funções que você mesmo criar. Funções que outras pessoas criaram e tornaram disponíveis para você.

São 5 tipos de dados primitivos (pré-definidos) em C (char, int, float, double, void)

Operadores Aritméticos (+, -, /, *, %)

Operadores lógicos (!, &&, ||)

Operadores Relacionais (>, <, >=, <=, ==, !=)

FUNÇÕES

Um programa em C é formado por um conjunto de funções.

COMANDO return. Serve para retornarmos um valor calculado dentro de uma função quando chamada de alguma parte do programa.

Decisão simples (if else)

Decisão Múltiplas (switch ...case)

Laços (while …do while, for)

STRING

Strings são matrizes unidimensionais de caracteres sempre terminada em zero '\0'.

Ex: char str[11];

MATRIZ

A matriz, assim como o vetor, permite a construção de um tipo onde os valores são agregados homogêneos de um tamanho definido, isto é, seus componentes são todos de um mesmo tipo.

Em C pode-se inicializar Vetores e Matrizes globais. Não pode é inicializar matrizes locais.

ESTRUTURAS (Struct)

Estrutura é uma coleção de variáveis referenciadas por um nome. As variáveis que formam a estrutura são chamados de elementos da estrutura.

PONTEIROS

Ponteiro é uma variável que contém um endereço de memória de outra variável. Na linguagem C há uma estreita relação entre ponteiros e matrizes.

CHAMADA POR REFERÊNCIA

A linguagem C usa a chamada por valor para passar argumentos para funções. Esse método copia o valor do argumento para o parâmetro. Assim não se altera o valor das variáveis usadas para chamar a função. Pode-se alterar os valores dessas variáveis fazendo uma chamada por referência usando ponteiros. Agora o endereço da variável é passada para a função e não o seu valor.

ARQUIVOS BINÁRIOS E ARQUIVOS ASCII EM C

A linguagem C trata os arquivos de duas maneiras diferentes : como arquivos ASCII ou como arquivos binários.

Binários: A linguagem C não interpreta o sentido do arquivo quando da leitura ou escrita de dados;

Ascii: Possui duas diferenças principais - quando encontra um ^Z (ASCII 26) no arquivo que estiver sendo lido, C interpreta o ^Z como um caracter de fim de arquivo, e presume que o fim do arquivo foi alcançado;

O caracter de nova linha ‘\n’ e armazenado em disco como uma seqüência ASCII 13 10.

ALOCAÇÃO DINÂMICA

A alocação dinâmica permite ao programador criar variáveis em tempo de execução. Isto significa que é possível alocar memória para novas variáveis quando o programa está sendo executado. O padrão C ANSI define quatro funções para o sistema de alocação dinâmica, que estão disponíveis na biblioteca stdlib.h.

FUNÇÃO MALLOC

A função malloc() serve para alocar (reservar) uma área de memória.

FUNÇÃO CALLOC

A função calloc() também serve para alocar memória, mas possui funcionalidade diferente: A função aloca uma quantidade de memória igual a num * size , isto é, aloca memória suficiente para uma matriz de num objetos de tamanho size.

FUNÇÃO REALLOC

A função realloc() serve para realocar (reorganizar) a memória. A função modifica o tamanho da memória previamente alocada apontada por *ptr para aquele especificado por num.

FUNÇÃO FREE

Quando É alocada memória dinamicamente é necessário que ela seja liberada quando não for mais necessária. Para isto existe a função free() que é a responsável por este procedimento. Passar para a função free() o ponteiro que aponta para o início da memória alocada.

Java

Projetada pelo Sun Microsystems, a linguagem de programação Java é uma das mais populares do mundo, com um vasto repertório de profissionais, aplicações, documentação e comunidade. O pensamento inicial, em 1991, era bem similar ao que o Internet of Things (IoT)busca entregar: permitir que os dispositivos eletrônicos pudessem se comunicar entre si.

Características

* Sintaxe e semântica

Este tópico aborda o que é sintaxe e semântica, e como isso impacta na linguagem Java.

*O que é sintaxe de um programa?

A sintaxe de um programa é um conjunto de regras que a linguagem deve seguir para se ter um padrão de declarações ou variáveis, ou seja, ela serve como guia para se ter orações e textos corretamente estruturados nessa linguagem.

*O que é semântica de um programa?

A partir de um conjunto de sentenças é possível criar uma expressão com um significado e sentido. Isso é a semântica, o significado da expressão para uma dada oração escrita na sintaxe Java.

*Como funcionar a sintaxe e semântica do Java?

Toda linguagem possui uma sintaxe e uma semântica que deve ser respeitada para se considera correta. Um programa em Java precisa ter, pelo menos, uma classe e um método main() e existe uma sintaxe para isso, como mostra a figura.

Variáveis

Em linguagens de programação, quando é necessário armazenar valores em memória utiliza-se variáveis. Por ser uma linguagem fortemente tipada, em Java é obrigatório informar o tipo da variável. Uma vez declarada, ela não pode ser modificada.

Tipos de Dados

Primitivo: são oito tipos de dados (byte, short, int, long, float, double, boolean, char)

Não-primitivos: String, Array e outras classes (objetos).

Procedimentos e Funções

Os procedimentos ou funções são blocos de programa que executam determinada tarefa orientada pela função principal na linguagem de programação.

Procedimentos e Funções em Orientação a Objetos

A Programação Orientada a Objetos (POO) representou uma ruptura do paradigma que regia a programação clássica estruturada. Ela revolucionou os conceitos de projeto e desenvolvimento de sistemas.

R

A Linguagem R é a linguagem de programação utilizada para a manipulação de dados gráficos e estatísticos. Ela é amplamente usada para o desenvolvimento de softwares estatísticos e análise de dados. A popularidade dessa linguagem tem aumentado consideravelmente nos últimos anos.

R foi desenvolvida pela implementação da linguagem de programação S combinada com a semântica de escopo léxico e Schemas. Essa última foi criada por John Chambers, na Bell Labs. Há algumas diferenças entre R e S mas a maioria do código desse último funciona no primeiro. A Linguagem R foi criada por Ross Ihaka e Robert Gentleman na Nova Zelândia e atualmente é desenvolvida por uma equipe a qual John Chambers faz parte. O código fonte do software é feita principalmente por C, Fortran e R.

Tipos de dados

Numeric

São número inteiros ou reais e que podem ser escritos da forma convencional ou em notação científica com a utilização e.

Character

São caracteres ou trechos de textos, sempre entre aspas e não são modificados, podem ser formados por acentos, espaços e outros caracteres (apenas em caso de valores finais).

Logical

São usados para resultados de testes lógicos ou para indicar opções onde há apenas 2 opções, é expresso por True(Verdadeiro) e False(Falso).

Vetores

Os vetores são as variáveis mais simples do R. Dentro delas, podemos associar diferentes classes. Pode utilizar o comando class() para verificar qual é a classe de uma variável, bastando inserir o nome da variável dentro da função.

Listas

Listas são coleções de elementos que não precisam ser da mesma classe, list é uma importante classe de objetos no R. As listas podem armazenar diferentes tipos de classes dentro dela. Sua criação acontece usando a função list().

Matrizes

Matrizes são conjuntos de dados bidimensionais sendo criados pelo comando matrix(). Dentro dele, podemos definir o número de linhas (nrow), colunas (ncol) e se os dados serão inseridos por colunas (byrow = FALSE) ou linhas (byrow = TRUE). O acesso aos dados da matriz pode ser realizado utilizandose colchetes, indicando a linha e a coluna.

Arrays

Os arrays são semelhantes às matrizes, mas podem ter mais que duas dimensões, onde o número de dimensões é configurado pelo termo dim. Os arrays são criados pela função array().

Fatores

Fatores são utilizados para criar rótulos, sendo bastante úteis em análises estatísticas, podendo ser ordenados ou não ordenados.

Data Frames

Data Frames: Data Frames são tabelas e podem agrupar dados de diferentes classes em cada coluna, sendo criados pelo comando data.frame().

Python

Características básicas da linguagem Python é uma linguagem de programação interpretada, de código-fonte aberto e disponível para vários sistemas operacionais. Diz-se que uma linguagem é interpretada se esta não ser precisar compilada (traduzida para uma linguagem da máquina), mas sim “lida” por um outro programa (chamado de interpretador) que traduzirá para a máquina o que seu programa quer dizer. O interpretador para Python é interativo, ou seja, é possível executá-lo sem fornecer um script (programa) para ele. Ao invés disso, o interpretador disponibilizará uma interface interativa onde é possível inserir os comandos desejados um por um e ver o efeito de cada um deles.

Variáveis

Variáveis são formas de se armazenar dados para uso posterior, elas podem ser classificadas em 3 tipos básicos que são mostradas logo abaixo. Quando analisarmos as listas veremos que existem outras variáveis mais complexas.

int - Um número inteiro

float - Um ponto flutuante

string - Uma sequência de caracteres

String é um tipo de objeto formado por uma sequ^encia imutavel de caracteres que nos permite trabalhar com textos.

Listas são sequências de variáveis. Após definidas, podem ser modificadas de várias maneiras, pois são mutáveis.

Estruturas de controle

Os comandos de Python são executados pelo computador, linha por linha e as estruturas de controle permitem ao programador modificar a ordem em que cada comando será executado bem como se ele será ou não executado.

O comando if direciona o computador a tomar uma decisão, baseado nas condições determinadas. Se a condição for atendida, um bloco de comandos será executado, caso contrário, o computador executa outros comandos.

While

Esta estrutura de controle tem como objetivo executar o bloco de comandos identificado nela repetidamente, enquanto a condição dada, para sua validade, for verdadeira.

For

O comando for, em Python, difere do que normalmente se vê em outras linguagens de programação, onde esse comando tem a finalidade de realizar uma iteração baseada numa progressão aritmética, percorrendo os números definidos pelo usuário, enquanto em Python a iteração é feita percorrendo os itens de uma sequência, seja ela uma lista ou até mesmo uma string.

Dicionário

Um dicionário é um conjunto de elementos que possuem índices, ou seja, dicionários são formados por chaves e seus respectivos valores, onde as chaves são os índices.

Funções

As linguagens de programação em geral têm o intuito de automatizar ações tornando-as mais rápidas. Se houver alguma ação que seja grande e utilizada com frequência, temos a opção de criar uma função que cumpra o seu objetivo, reduzindo o espaço ocupado pelo nosso programa final, além de deixá-lo com uma aparência mais limpa, visto que o tamanho do código irá diminuir.

Módulos

Pensando na reutilização de código, a linguagem Python já possui um conjunto de funções prontas para serem usadas ou agregadas em seus programas. Essas funções estão agrupadas em estruturas denominadas módulos. Para a utilização desses módulos é preciso utilizar o comando import nome_do_modulo.

JavaScript

A linguagem JavaScript foi criada pela Netscape Communications Corporation e foi desenvolvida com o nome de Mocha, depois passou a se chamar LiveScript e foi finalmente lançada como JavaScript em 1995 integrando a versão 2.0B3 do navegador Netscape e visava implementar uma tecnologia de processamento modo cliente. A denominação da linguagem, JavaScript, se deve a similaridades com a sintaxe do Java e embora as duas linguagens não tenham nenhuma outra relação além desta, os nomes ainda causam confusão para alguns usuários. Mais tarde, a linguagem tornou-se um padrão da ECMA (European Computer Manufacturers Association) que atualmente é seguido por outros desenvolvedores como os da Adobe com a linguagem ActionScript.

JavaScript é uma linguagem de programação de tipagem dinâmica. Funções de ordem superior - são funções que recebem uma ou mais funções como argumentos ou que têm uma função como saída. Com isso, é possível criar o que são chamadas function factories que são funções que a partir de outras funções simples são capazes de realizar ações mais complexas. JavaScript é uma linguagem de programação que possibilita a definição de funções de ordem superior. Programação Client-side vs. Server-side - JavaScript é uma linguagem que nasceu como Client-side (que roda no computador cliente) e tem sido muito mais usada essa forma atualmente. Quando o programa é criado com esta característica ele é enviado para o computador cliente ainda na forma de código-fonte, que só então é interpretado e executado, dependendo assim unicamente da capacidade de processamento do cliente.

Variáveis

Var

Declarar variáveis em javascript é bem simples, diferente de linguagens tradicionais como C++, em javascript não informamos o tipo de dados e por isso não precisamos de operações como typecast, basta usar a palavra reservada “var” e o nome da variável, veja a seguir o código básico para se declarar uma variável.

Const

Constantes tem uma diferença importante em relação às variáveis, elas não podem ser modificadas, uma vez declarada a atribuída uma constantes, seu valor não pode

mais ser modificado. A sintaxe de declaração é a mesma da variável com exceção da palavra “var” que será substituída pela palavra “const”, veja o código a seguir.

Let

Foi pensando em trazer o escopo de bloco (tão conhecido em outras linguagens) que o ECMAScript 6 destinou-se a disponibilizar essa mesma flexibilidade (e uniformidade) para a linguagem. Através da palavra-chave let podemos declarar variáveis com escopo de bloco.

Caixas de mensagens

Alert

A caixa de mensagem mais simples é a caixa “alert”, mostra uma mensagem simples com um botão somente o botão OK, o comando é bem simples, basta inserir o conteúdo que será mostrado na caixa de texto dentro dos parênteses.

Prompt

A caixa de mensagens prompt se diferencia por possibilitar a entrada de texto, permitindo que esse texto seja coletado e passado a uma variável, desta maneira é a primeira forma de coletar dados externos que iremos aprender. Sintaxe = prompt ("Texto principal a ser mostrado","Valor inicial mostrado na caixa de texto");

Tipos de dados

Tipos **numéricos** (Inteiros, flutuante)

Em JavaScript os números são representados pelo padrão IEEE 754. Todos os valores numéricos são "declarados" pela simples atribuição dos valores a uma variável.

Booleano

Uma variável do tipo booleano pode assumir apenas dois valores: true e false. Os valores deste tipo são em geral usados pela linguagem como resultado de comparações e podem ser usados pelo usuário para valores de teste ou para atributos que possuam apenas dois estados.

Indefinido

Uma variável é indefinida quando ela foi declarada de alguma forma mas não possui nenhum valor concreto armazenado. Quando tentamos acessar uma variável

que não teve nenhum valor associado a ela teremos como retorno "undefined" (indefinido).

Null

O null é a ausência de valor; quando atribuimos null a um objeto ou variável significa que essa variável ou objeto não possui valor válido. Para efeito de comparação, se usarmos o operador de igualdade "==", JavaScript irá considerar iguais os valores null e undefined. E isso não afeta o uso da comparação (var.metodo == null) quando queremos descobrir se um objeto possui determinado método. No entanto, se for necessário diferenciar os dois valores é recomendável o uso do operador "===" de identicidade.

Strings

São sequências de caracteres. Em JavaScript a string pode ser tanto um tipo primitivo de dado como um objeto; no entanto, ao manipulá-la temos a impressão de que sejam objetos pois as strings em JavaScript possuem métodos que podemos invocar para realizar determinadas operações sobre elas. Essa confusão ocorre porque quando criamos uma string primitiva, o JavaScript cria também um objeto string e converte automaticamente entre esses tipos quando necessário.

Arrays

Os Arrays são pares do tipo inteiro-valor para se mapear valores a partir de um índice numérico. Em JavaScript os Arrays são objetos com métodos próprios. Um objeto do tipo Array serve para se guardar uma coleção de ítens em uma única variável.

Operadores

Aritméticos (+, -, /, *, %, ++, --)

Bit a bit (!, &&, ||, ^, ~, >>, <<, >>>)

Comparação (>, <, >=, <=, ==, !=, ===)

Lógicos (&&, ||, !)

Estruturas de controle

If ... else

A estrutura if é usada quando se deseja verificar se determinada expressão é verdadeira ou não, e executar comandos específicos para cada caso.

switch ... case

As estruturas do tipo switch são usadas quando queremos selecionar uma opção dentre várias disponíveis.

while

Os laços do tipo while são usados quando se deseja que uma sequência de ações seja executada apenas no caso da expressão de condição ser válida. Assim, primeiro a expressão é testada, para depois o conteúdo do laço ser executado ou não.

do ... while

Diferentemente do while, o do ... while primeiro executa o conteúdo do laço uma vez e, depois disso, realiza o teste da expressão pare decidir se continuará executando o laço ou irá seguir o resto do programa.

for

Na maioria das vezes, quando usamos um laço do tipo while também construímos uma estrutura com um contador que é incrementado a cada passo para controle do laço e manipulação interna de objetos, arrays como nos exemplos anteriores. Os laços for oferecem a vantagem de já possuírem em sua estrutura essa variável de contador e incrementá-la de maneira implícita.

for ... in

Existe uma segunda forma de se utilizar os laços for para percorrer propriedades de um objeto.

Funções

Possuem um papel muito importante na programação estrutural pelo fato de ajudar muito na modularização no programa, ou seja, viabiliza a divisão do programa em partes menores e logicamente relacionadas.

Um ponto importante é que em JavaScript as funções são consideradas como dados, ou seja, podemos atribuir uma função a uma variável ou propriedade de um objeto e a partir desde momento usar a variável ou a propriedade da mesma forma que se usaria a função.

Elas também podem ser passadas como argumentos para outras funções e por isso funções de JavaScript são chamadas funções de alta ordem, elas podem tanto receber funções como argumento quanto retornar uma função.

Expressão function

A primeira maneira de se declarar uma função é através do uso da palavra chave function de maneira similar a como elas são declaradas na linguagem C, com as diferenças de que em JavaScript não definimos o tipo de retorno e nem mesmo o tipo dos argumentos. Uma função complexa pode ser capaz de tratar argumentos diferentes e retornar argumentos diferentes dependendo das circunstâncias nas quais foi invocada. Deve-se definir seu nome e seus argumentos conforme mostra o exemplo a seguir.

O construtor Function()

A segunda forma de se declarar uma função é utilizando o construtor Function() e o operador new, pois em JavaScript funções e objetos são interligados.

Funções como literais

Uma terceira e última forma de se declarar uma função em JavaScript é através de literais. Essa forma é basicamente a mesma que declarar através do construtor Function(). No entanto, ela é melhor porque os comandos podem ser declarados com

a sintaxe normal de JavaScript ao invés de ser uma string como é o caso do construtor.

Objetos

Ao contrário de uma variável, um objeto pode conter diversos valores e de tipos diferentes armazenados nele (atributos) e também possuir funções que operem sobre esses valores (métodos). Tanto os atributos, quanto os métodos, são chamados de propriedades do objeto.

Métodos

No paradigma de orientação a objetos, os métodos são simplesmente funções que são invocadas por meio de um objeto! E em JavaScript isso é levado tão a sério que a maneira de se criar métodos para seus objetos leva isso ao pé da letra. Basta criarmos uma função e atribuí-la a uma propriedade do objeto.

Prototypes

Quando declaramos ou atribuímos um método no construtor de um objeto ele ficará disponível para todas as instâncias criadas a partir desse construtor. No entanto, existe um modo muito mais eficiente de se fazer isso, que é com o uso da propriedade prototype. Tudo o que for definido no prototype de um objeto poderá ser referenciado por todas as instâncias desse objeto.

Arrays Associativos

Para finalizar nossa discussão sobre objetos, vamos mostrar como eles podem ser usados como arrays associativos, ou seja, um array com objetos indexados por valores não numéricos. Isso só pode ser feito porque é possível acessarmos atributos de um objeto usando MeuObjeto["atributo"]. Assim podemos simular o comportamento de um array associativo armazenando cada item em um atributo.

Objeto String

Vamos nos ater agora aos métodos das Strings, e embora existam outros, aqui serão relacionados apenas os que fazem parte da ECMA 262-3 que equivale ao JavaScript 1.6, pois estes métodos são comuns a uma grande variedade de browsers como FireFox, Netscape e Internet Explorer.

valueOf() - retona o valor primitivo do objeto string;

charAt(pos) - retorna uma string contendo o caractere de posição pos da string.;

concat(string1,string2, ... ,stringN) - este método retorma uma string contendo a própria string;

indexOf(padrao,pos) - procura a ocorrência da string contida em padrao a partir da posição pos dentro da string;

lastIndexOf(padrao,pos) - retorna o índice da última ocorrência de padrao na string sobre a qual se utilizou o método;

replace(velho,novo) - busca na string uma substring que seja igual ao conteúdo de velho e a substitui pelo conteúdo de novo e retorna a string resultante da substituição;

search(padrao) - busca o conteúdo de padrao e retorna o índice de início desse padrão;

slice(inicio,fim) - retona a substring contendo os caractéres da posição inicio até a posição fim, mas sem a incluir, ou até o final da string se fim for indefinido;

split(separador,limite) - retorna um objeto Array contendo as substrings que resultaram da separação da string original pelo conteúdo de separador, sem incluí-lo.;
substring(inicio,fim) - este método funciona da mesma maneira e para os mesmos fins que slice(inicio,fim);
toLowerCase() - os caractéres da string são todos convertidos para letras minúsculas;
toUpperCase() - semelhante ao método toLowerCase(), no entanto, ele converte todas as letras da string para letras maiúsculas;
length() - valor inteiro que contém o número de caractéres que compõem a string.

Exceções

Nas versões atuais, comandos para manipulação de exceções foram incluídos em JavaScript, de forma similar aos que a linguagem Java oferece. Temos os comandos throw, try, catch e finally. Com eles é possível desenvolver uma aplicação em JavaScript capaz de tratar possíveis erros em tempo de execução, aumentando de maneira considerável sua robustez.