Paradigmas de Linguagens de Programação (PLP)

Professor: Carlo Marcelo Revoredo da Silva
Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO

Sumário

- **Histórico do paradigma Imperativo**
- **Conceitos Básicos**
- **Miniprojeto**

O Paradigma Imperativo

O Paradigma Imperativo

- **Arquitetura de Von Neuman**
- **Memória para armazenar os dados**
- **CPU para realizar as operações**
- **Imperativo => “Comandar”**
- **Baseadas em comandos ou orientada a atribuições**
- **Exemplos: Fortran, Pascal, C, Ada**

Conceitos Básicos

- **Variáveis**

- **Operadores e Expressões**
  - Operadores e expressões Aritméticas
  - Operadores e expressões Relacionais
  - Operadores e expressões Lógicas

- **Estruturas de Controle**
  - Estrutura de seleção
    - Simples, Composto, Encadeado e Múltipla escolha
  - Estrutura de repetição
    - Com teste no início, com teste no fim, com variável de controle

Variáveis

- São regiões de memoria reservadas para armazenar informações que serão utilizadas pelo programa
- São armazenadas na memória principal do computador
- Processo de declaração da variável
- Existe uma regra para nomear as variáveis
  - Não iniciar com números
  - Não fazer uso de caracteres especiais
  - Não fazer uso de palavras reservas (sintaxes pré-estabelecidas da linguagem)
- Tipos de variáveis
  - Inteiro, decimal, booleano, data, hora, texto
- Variáveis podem ser estáticas ou dinâmicas
  - Estáticas também são chamadas de constantes e uma vez que recebem um valor, permanecem até o final do programa
  - Dinâmicas podem ser modificadas a qualquer momento pelo programa
- Declaração e Atribuição de valores
  - int x;
  - x = 5

Operadores e Expressões

- Operadores aritméticos
  - +, -, *, ** (exponencial) e /
  - Possuem uma ordem de prioridade na execução
    - **, * e /, + e –
- Expressões aritméticas
  - Todas as expressões aritméticas resultam em um número
    - 2 + 2 (resulta em 4)
- Operadores relacionais
  - >, <, >=, <=, != (diferente)
- Expressões relacionais
  - Todas as expressões relacionais são binárias e resultam em um valor booleano.
    - 2 > 4 (resulta em falso), 2 != 3 (resulta em verdadeiro)

Operadores e Expressões

- Operadores lógicos
  - & (E => conjunção), &&, | (OU => disjunção), ||, ! (negação)
  - Curto circuito (&&, ||)
  - Tabela verdade

A	B	A e B	A ou B	A e A	B ou B	!A	!B
V	V	V	V	V	V	F	F
V	F	F	V	V	F	F	V
F	V	F	V	F	V	V	F
F	F	F	F	F	F	V	V

- Expressões lógicas
  - Quase todas as expressões lógicas são binárias, com exceção do operador ! que é unário. Resultam em valores booleanos
    - V & V = verdadeiro, !V = F
  - Como cada expressão lógica resulta em um booleano, as comparações podem ser encadeadas por meio de operadores lógicos ou relacionais
    - (V & V) & F (resulta em falso)
  - Como cada expressão relacional também resulta em um booleano, as comparações também podem ser encadeadas por meio de operadores lógicos
    - (4 > 2) & (2 < 3) (resulta em verdadeiro)

Estruturas de Controle

- Estrutura de seleção
  - Simples (explicar primeiro e concluir com limitações)
    - If End If
  - Composta
    - If Else End If
  - Encadeada
    - If If Else End If End If
  - Ternário
  - Múltipla escolha
    - Int x = 5;
    - Switch (x){
      - Case 10: //
      - break;
    - }

Estruturas de Controle

- Estrutura de repetição
  - Com teste no início (sempre motivar, expor limitações)
    - while(true){
      - //
    - }
  - Com teste no fim (não existe um término específico)
    - do{
      - //
    - }while(true);
  - Com variável de controle
    - for (int i=0; i<10; i++){(expor situações)
      - //
    - }
  - Detalhar mecânica e aplicabilidade

Resumo Geral

- Variáveis possibilitam modificar o estado da execução do programa de acordo com o contexto de execução
- Operadores e expressões são responsáveis em executar operações matemáticas, relacionais e lógicos, que em combinação com os operadores lógicos podem resultar em condições baseadas nas variáveis envolvidas
- Os resultados dessas condições podem ser manipuladas por estruturas de seleção ou repetição
- As estruturas de repetição podem estabelecer o tempo de execução do programa de acordo com as condições realizadas

Referências

- Gerson Rissetti & Sandra Puga, 2004. Lógica de programação e estrutura de dados com aplicações em Java.
- Paul Deitel & Harvey Deitel, 2016. Java: Como programar

Dúvidas?

Miniprojeto (Let’s Code!)

Descrição conceitual

- Desenvolver uma versão simples do jogo Elifoot (“Futebyte”)
- Serão dois times se confrontando e cada um possui um fator de técnica (1 até 5)
- A partida terá um placar e cada time acumula tentativas de gol
- A rotina do jogo deverá ter um delay entre os eventos para maior experiencia do usuário (por exemplo, Thread.sleep(1000) no mínimo)
- Pode assumir um minuto como um segundo (45 segundos cada tempo de jogo)
- Os eventos podem prolongar o total de tempo do jogo

Descrição conceitual

- **Tempo de partida**
  - O jogo é baseado em narrativas, ou seja, o usuário deve ser informado a cada instante o que está ocorrendo na partida;
    - Por exemplo: bola rolando, pênalti, lance de perigo, gol confirmado, gol perdido.
  - Deve ser dividido em 2 tempos (1 tempo, intervalo e 2 tempo)
  - Ao atingir o final do primeiro tempo, deverá ser exibido ao usuário:
    - Informar que a partida está em intervalo (tempo aproximado de 10 segundos)
    - Os jogadores foram ao vestiário
    - Quem teve mais chance de marcar (quantidade de tentativas de gol), caso a quantidade seja igual, comentar que o jogo está equilibrado, etc
    - Os jogadores estão aquecendo no campo
    - Iniciado o segundo tempo
  - Ao iniciar o termino do jogo, deve informar ao usuário a conclusão e placar final.

Descrição conceitual

- **Lance de perigo**
  - É uma tentativa de gol durante a partida
  - A cada 5 segundos (em média), será executada uma rotina para confirmar ou não um lance de perigo. Entre dois números, 1 e 2.
  - **Cálculo de possibilidade de perigo:**
    - int perigo = random.nextInt(2)+1
  - Se perigo for 2, dispara o evento, caso contrário, o jogo segue normalmente.
  - Uma vez que seja disparado o lance de perigo, o próximo passo é saber quem será o autor do mesmo, definido pelo cálculo de fator de técnica.
  - Fator de técnica * valor randômico entre os números 1, 2 e 3.
  - **Cálculo de fator de técnica**:
    - calcTime1 = fator de técnica1 * random.nextInt(3)+1
    - calcTime2 = fator de técnica2 * random.nextInt(3)+1
    - Se calcTime1 > calcTime2, o autor é o time1, e vice-versa.
    - Caso seja empate, o jogo segue sem lance de perigo.

Descrição conceitual

- **Lance de pênalti**
  - Durante a partida, a cada 10 minutos (de cada tempo) será disparada uma rotina para avaliar se ocorrerá ou não um lance de pênalti.
  - O lance de pênalti será na probabilidade de ser sorteado o número 3 entre os números 1 até 5. (sugestão)
  - Cálculo: int penalty = random.nextInt(5)+1
  - Caso se confirme, será realizado o evento de lance de pênalti
  - Ao confirmar o evento de lance de pênalti, será disparada a rotina de tentativa de gol.

Descrição conceitual

- **Tentativa de Gol**
  - É um lance de perigo confirmado (com autor atribuído)
  - Uma vez que ocorra, o time autor contabiliza o lance de perigo.
  - Será o mesmo **cálculo de fator de técnica**
  - Caso o time1 tenha maior cálculo, confirma-se um gol ao seu favor
  - Caso seja menor ou igual, o time perde a chance de marcar.
- **Gol marcado e Gol perdido**
  - Ambos os eventos devem ser notificados ao usuário.
  - Em caso de gol, deve ser exibido em seguida o placar atual.

Grato pela atenção

Disciplina: Paradigmas de Linguagens de Programação (PLP)

Professor: Carlo Marcelo Revoredo da Silva

Contato: revoredo@gmail.com

UNIVERSIDADE DE PERNAMBUCO