Aula 4 - Controle de Fluxo

SECITECI
SECRETARIA DE ESTADO DE CIÊNCIA E TECNOLOGIA E INOVAÇÃO

GOVERNO DE MATO GROSSO

<Fic_Dev>
Programador de Sistemas

Roteiro

<Fic_Dev>
Programador de Sistemas

Desafio do Dia

Desenvolva um programa que determine a elegibilidade de uma pessoa para receber uma vacina. Os critérios de elegibilidade são os seguintes:
Idade: Idade igual ou superior a 18 anos.
Além da idade, pessoas com as seguintes condições médicas também são elegíveis:

- Diabetes.
- Hipertensão.
- Doenças cardíacas.

Solicite ao usuário a idade e as condições médicas (diabetes, hipertensão, doenças cardíacas). Com base nessas informações, verifique se a pessoa é elegível para a vacinação, exibindo uma mensagem indicando se a pessoa é elegível ou não para a vacinação. Certifique-se de lidar adequadamente com diferentes cenários de entrada, como valores inválidos para idade ou condições médicas.

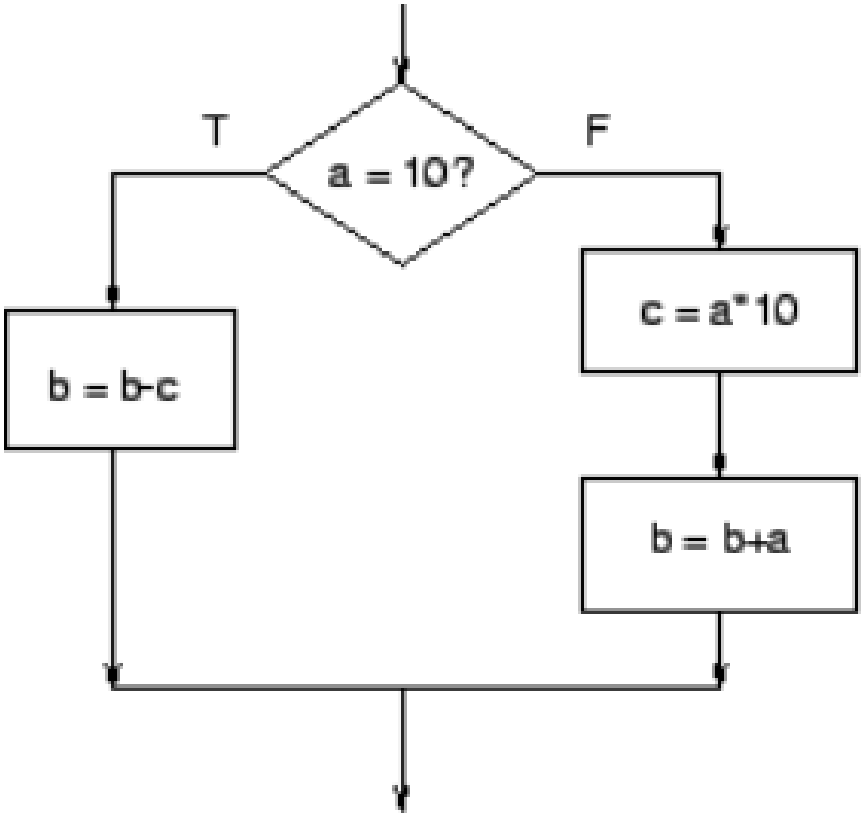
Introdução

- Sabemos que ao construir algoritmos/programas nós enfrentaremos situações em que teremos que mudar o fluxo de execução de um algoritmo.
- Isso deve-se ao fato de que um algoritmo, de forma bem simplificada, sempre será executado de cima para baixo, da esquerda para a direita, **até que algo desvie ou mude o seu fluxo de execução.**

Introdução

- Essa possibilidade de desvio/controle de fluxo dá ao programador muitas possibilidades considerando, inclusive, que alguns problemas seriam insolúveis se não fosse o desvio/controle de fluxo.
- Digamos que o processo de desvio/controle de fluxo na programação de computadores é natural, pois a maioria das linguagens conta com este recurso.
- Até as linguagens mais limitadas, contam com algum tipo de possibilidade de desvio/controle da execução dos programas.

Introdução




```
if (a == 10)
     b = b-c;
else {
     c = a*10;
     b = b+a;
}
```

Introdução

De modo geral, o controle de fluxo é realizado pelas seguintes classes de comandos

- Comandos de Seleção (Decisão)
  - if (simples, composto e aninhado)
  - switch-case (decisão múltipla)
- Comandos de Repetição
  - Delimitada
  - Baseada em Condição (Teste na entrada)
  - Baseada em Condição (Teste na saída)
  - Baseada em Coleções

Comando if - Decisão (simples, composto e aninhado)

- A decisão simples é um comando que em caso de aceitação da condição um determinado caminho é seguido, no entanto, caso a condição não seja satisfeita, **não haverá outro caminho a seguir** a não ser o fluxo do próprio algoritmo
- É o tipo de desvio/controle de fluxo mais simples.

Sintaxe:
if (<condição>)
<comandos>

Escreva um pequeno código para ler uma idade e julgar se a mesma é maior ou menor de 18. Se for menor, escreva no log "menoridade"

<Fic_Dev>
Programador de Sistemas

Comando if - Decisão (simples, composto e aninhado)

- A decisão composta é aquela que oferece outro caminho a seguir e, então, segue o fluxo do algoritmo.
- Amplamente utilizada, devido a quantidade de caminhos que possibilita.

Sintaxe:
if (<condição>)
<comandos>
else
<comandos>

Altere o programa anterior e caso a idade lida seja maior ou igual, escreva no log "maioridade"

<Fic_Dev>
Programador de Sistemas

Comando if - Decisão (simples, composto e aninhado)

- Decisões aninhadas ocorrem quando colocamos em um dos caminhos da decisão outro comando de decisão, ligando uma decisão à outra.
- De modo similar à Composta, é amplamente utilizada, devido a quantidade de caminhos que possibilita.

Sintaxe:
if (<condição>)
<comandos>
else if (<condição>)
<comandos>
else if (<condição>)
<comandos>

Altere o programa anterior para tratar os casos de menor que 16, maior que 16 e menor que 18 e maior que 18

<Fic_Dev>
Programador de Sistemas

Um pequeno problema - Calculadora de bônus salarial

Você foi contratado para desenvolver um algoritmo que calcule o bônus salarial dos funcionários de uma empresa com base em determinadas regras. O algoritmo deve receber como entrada o salário mensal do funcionário e o seu tempo de serviço em anos.
Regras para o cálculo do bônus salarial:

- Se o tempo de serviço for inferior a 1 ano, não há direito a bônus.
- Se o tempo de serviço for de 1 ano ou mais, mas inferior a 5 anos, o bônus será de 5% sobre o salário mensal.
- Se o tempo de serviço for de 5 anos ou mais, o bônus será de 10% sobre o salário mensal.

switch-case (decisão múltipla)

A decisão multipla é um comando que facilita muito a construção do código. No entanto, é apenas uma alternativa, pois é similar ao uso de decisões compostas ou aninhadas. Fica apenas a preferência pela organização do código.
Sintaxe:
switch (<expressão avaliada>) {
<case valor 1>
<comando>
break;
<case valor 2>
<comando>
break;
default }
Altere o programa da idade para usar o switch-case.

<Fic_Dev>
Programador de Sistemas

Repetição Delimitada (Contada)

- A repetição delimitada ou contada existente em JavaScript é o **for**.
- Esse comando é muito similiar ao de outras linguagens e sua sintaxe é bem simples.

Sintaxe:
for (int i = 0; i <= 10; i++)
<bloco de comandos>
Vamos escrever um simples bloco de repetição para imprimir os números dos meses do ano.

<Fic_Dev>
Programador de Sistemas

Repetição Delimitada (Contada)

Considerações:

- Dimensão da repetição.
- Inicialização da variável de controle.
- Condição de Parada/Continuidade.
- Incremento/Decremento da variável de controle.

Reescreva o comando anterior para seguir de modo inverso.

Baseada em Condição (Teste na entrada)

- A repetição baseada em condição é o **while**. Em algumas outras linguagens, existe o comando **repeat...util**
- Esse comando é bem mais flexível que a repetição contada, no entanto, exige maior controle por parte do programador, pois a saída/continuidade na repetição dependerá da avaliação da condição.

Sintaxe:
while (condição)
<bloco de comandos>
Agora, usando while, vamos escrever um simples bloco de repetição para imprimir os números dos meses do ano.

<Fic_Dev>
Programador de Sistemas

Baseada em Coleções

A repetição baseada em coleções é uma grande aliada dos desenvolvedores, pois facilita a construção de códigos otimizados e organizados. Permite iterar sobre elementos iteráveis, como arrays, strings, objetos do tipo Map e Set, entre outros.
Sintaxe:
for (variável of iterável) {
<bloco de comandos> }

Baseada em Coleções

A cada iteração, a variável assumirá o valor do item da coleção.
Sintaxe:

```
const array = [1, 2, 3];
for (const elemento of array) {
console.log(elemento);
}
```

É importante notar que o for...of não permite acessar o índice de cada elemento do iterável

Vamos implementar agora, a impressão dos meses do ano usando array de string

<Fic_Dev>
Programador de Sistemas

Baseada em Coleções

Este último comando que percorre uma coleção é mais específico. Ele busca percorrer todas as propriedades de um determinado objeto. O comando é o **for..in**

Sintaxe:

```
const objeto = {
nome: "João",
idade: 30,
profissao: "Desenvolvedor"};
for (const propriedade in objeto) {
console.log(‘propriedade :objeto[propriedade]‘);
}
```

Conclusão

- Muito provavelmente as decisões e as repetições são os comandos mais utilizados pelos programadores. Não somente pela simplificade, mas sim pela utilidade.
- O ideal é que cada programador conheça bem essas ferramentas com o intuito de produzir código enxuto e eficiente.
- O uso das repetições baseas em coleções também são ferramentas incríveis, que facilitam a vida do programador.
- **Vamos ao Desafio do Dia!**

<Fic_Dev>
Programador de Sistemas