Disciplina: Introdução à Programação

Aula 6: Estrutura de seleção com múltiplas alternativas

Apresentação

Na aula anterior, vimos que existem muitas situações nas quais o fluxo de execução do algoritmo ou do programa não pode ser sequencial. Para solucionar essa dificuldade, precisamos recorrer às estruturas de seleção, ou estruturas seletivas. Da forma como estudamos até o momento, aprendemos que essas estruturas são responsáveis por avaliar condições e decidir o que acontecerá quando a condição for verdadeira e o que acontecerá quando a condição for falsa.

Há situações, entretanto, em que não existem somente duas possibilidades a serem consideradas, mas várias delas. Nesses casos, precisaremos aninhar as estruturas seletivas, ou seja, escrever várias estruturas de seleção uma dentro da outra. Algumas vezes, essa prática vai gerar estruturas seletivas muito complexas e difíceis de serem lidas e compreendidas; e é aí que vamos recorrer a outra estrutura que nos permite analisar um número maior de condições de forma mais inteligível.

Bons estudos!

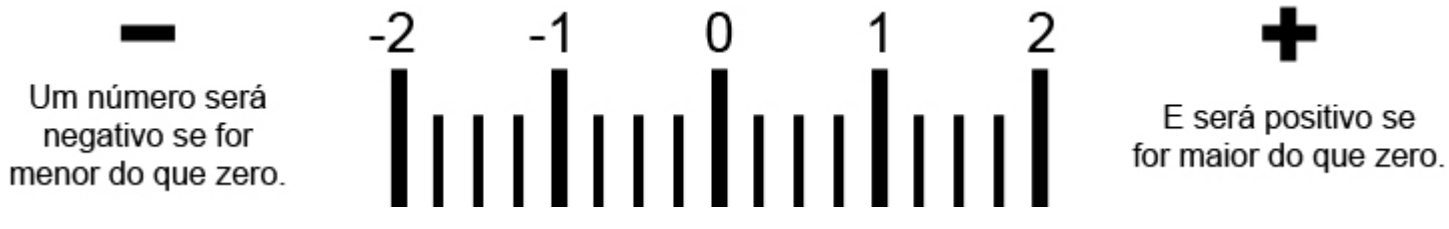
Objetivos

- Aplicar as estruturas seletivas aninhadas;
- Identificar uma alternativa à estrutura “se..então..senão”;
- Estabelecer a estrutura de seleção adequada para escrever algoritmos e programas.

Estrutura seletiva aninhada

Você se lembra do exercício que fizemos, no qual recebíamos um número e informávamos se ele era positivo ou negativo?

Sabemos que:



Mas, o que acontece se, ao pedirmos ao usuário para informar um número, ele digitar o número 0?

Da forma como fizemos a solução, avaliamos somente duas condições para o número – positivo ou negativo, mas está claro que temos que considerar uma terceira possibilidade: a de o número ser nulo.

Para dar conta das três possibilidades e ser capaz de dar uma mensagem correta seja qual for a situação – número nulo, número positivo ou número negativo – precisamos adequar o código, mais especificamente o trecho que avalia as condições. Observe:

* Portugol Studio

```
se (num>0)
  {
  escreva("Este número é positivo!")
  }
senao se (num<0)
  {
  escreva("Este número é negativo!")
  }
senao
  {
  escreva("Este número é nulo!")
  }
```

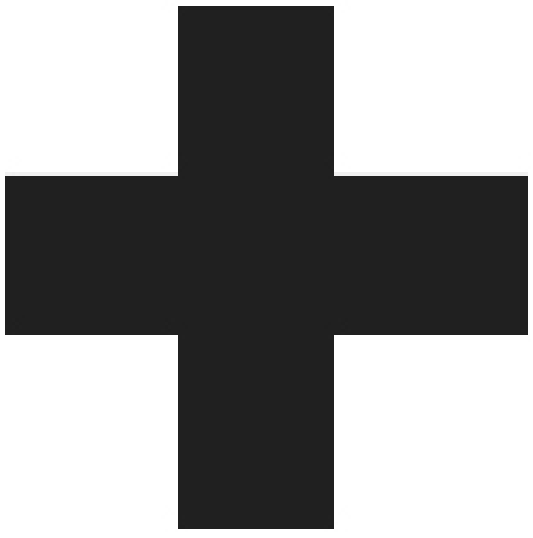
* C++

```
if (num>0)
  {
  cout <<"Este número é positivo!";
  }
else if (num<0)
  {
  cout <<"Este número é negativo!";
  }
else
  {
  cout <<"Este número é nulo!";
  }
```

Logo após o comando **senao** ou **else**, inserimos outro teste de condição, iniciado pelo comando **se** ou **if**.

Com a reformulação da estrutura de seleção, passamos às seguintes análises:

1



Avaliamos a possibilidade de o número ser positivo.

2

Se ele não for positivo, avaliamos a possibilidade de o número ser negativo.

3

Caso o número não seja nem positivo nem negativo, sabemos que ele será nulo, pois não há nenhuma outra possibilidade a ser avaliada; portanto, a última possibilidade é tratada somente por um comando **senao** ou **else**.

Atenção

É importante que você saiba que não há um limite para aninhamento de estruturas seletivas.

Agora imagine que você deseja criar uma calculadora básica, capaz de realizar as operações matemáticas básicas entre dois números. Veja como poderia ser o algoritmo para solucionar esse problema:

* Portugol Studio

```
programa
{
  funcao inicio()
  {
    real num1, num2, resultado, op
    escreva("Informe o 1o. número: ")
    leia(num1)
    escreva("Informe o 2o. número: ")
    leia(num2)
    escreva("-----------------------------------\n")
    escreva("Escolha a operação matemática: \n")
    escreva("1 - Adição\n")
    escreva("2 - Subtração\n")
    escreva("3 - Divisão\n")
    escreva("4 - Multiplicação\n")
    escreva("Opção: ")
    leia(op)
    se (op==1)
    {
      resultado=num1+num2
      escreva(num1, " + ", num2, " = ", resultado)
    }
    senao se (op==2)
    {
      resultado=num1-num2
      escreva(num1, " - ", num2, " = ", resultado)
    }
    senao se (op==3)
    {
      resultado=num1/num2
      escreva(num1, " / ", num2, " = ", resultado)
    }
    senao se (op==4)
    {
      resultado=num1*num2
      escreva(num1, " * ", num2, " = ", resultado)
    }
    senao
    {
      escreva("Opção inválida!!!")
    }
  }
}
```

No exemplo anterior, avaliamos cinco possibilidades de escolha de opção: 1, 2, 3, 4 ou qualquer outro número diferente desses.

Dessa forma, se o usuário escolher:

Opção 1

Efetuamos a soma entre os números informados.

Opção 2

O resultado será a subtração entre o primeiro e o segundo números.

Opção 3

O algoritmo calculará a divisão entre o primeiro e o segundo números.

Opção 4

A operação de multiplicação será efetuada entre os números.

Um número diferente

O algoritmo exibirá a mensagem de erro “Opção inválida!!!”.

Vamos ver como fica o programa em C++?

* Programa em C++

```
#include <iostream>
using namespace std;
int main()
{
  float num1, num2, resultado;
  int op;
  cout <<"Informe o 1o. número: ";
  cin >>num1;
  cout <<"Informe o 2o. número: ";
  cin >>num2;
  cout <<"-----------------------------------\n";
  cout <<"Escolha a operação matemática: \n";
  cout <<"1 - Adição\n";
  cout <<"2 - Subtração\n";
  cout <<"3 - Divisão\n";
  cout <<"4 - Multiplicação\n";
  cout <<"Opção: ";
  cin >>op;
  if (op==1)
  {
    resultado=num1+num2;
    cout <<num1 << " + " << num2 << " = " << resultado;
  }
  else if (op==2)
  {
    resultado=num1-num2;
    cout <<num1 << " - " << num2 << " = " << resultado;
  }
  else if (op==3)
  {
    resultado=num1/num2;
    cout <<num1 << " / " << num2 << " = " << resultado;
  }
  else if (op==4)
  {
    resultado=num1*num2;
    cout <<num1 << " * " << num2 << " = " << resultado;
  }
  else
  {
    cout <<"Opção inválida!!!";
  }
}
```

Imagine que você precisa criar um algoritmo que, a partir de um número de mês, escreva o nome do mês por extenso. Utilizando estruturas de condição aninhadas, teríamos algo assim:

* Portugol Studio

```
programa
{
  funcao inicio()
  {
    inteiro numMes
    cadeia mesExtenso=""
    escreva("Informe o número do mês: ")
    leia(numMes)
    se (numMes==1)
    {
      mesExtenso="janeiro"
    }
    senao se (numMes==2)
    {
      mesExtenso="fevereiro"
    }
    senao se (numMes==3)
    {
      mesExtenso="março"
    }
    senao se (numMes==4)
    {
      mesExtenso="abril"
    }
    senao se (numMes==5)
    {
      mesExtenso="maio"
    }
    senao se (numMes==6)
    {
      mesExtenso="junho"
    }
    senao se (numMes==7)
    {
      mesExtenso="julho"
    }
    senao se (numMes==8)
    {
      mesExtenso="agosto"
    }
    senao se (numMes==9)
    {
      mesExtenso="setembro"
    }
    senao se (numMes==10)
    {
      mesExtenso="outubro"
    }
    senao se (numMes==11)
    {
      mesExtenso="novembro"
    }
    senao se (numMes==12)
    {
      mesExtenso="dezembro"
    }
    senao
    {
```

```
        escreva("Mês inválido!!!")
      }
      escreva("O número ", numMes, " equivale ao mês de ", mesExtenso)
    }
  }
```

Um algoritmo imenso, não é mesmo?

Tudo porque temos doze possibilidades de meses, mais o caso de o usuário digitar um número de mês inválido.

Em situações nas quais existem muitas condições a serem avaliadas, temos a alternativa de utilizar outra estrutura condicional, conforme veremos a seguir.

Atividade

1 - Antes de seguir para o próximo tópico, execute o algoritmo anterior e informe um número de mês inválido, como 15, por exemplo. Observe que o algoritmo exibe a mensagem “Mês inválido!!!”, mas exibe também “O número 15 equivale ao mês de”. Isso acontece porque a linha responsável por exibir esta última mensagem está fora das condições e será executada independentemente do número de mês digitado. Vamos resolver esse problema? Crie uma variável lógica chamada **invalido** e inicialize-a com o valor **falso**. Caso o mês digitado seja inválido, troque o valor dessa variável para **verdadeiro**. Agora, crie um teste de condição capaz de avaliar o valor da variável **invalido** e exibir a mensagem do mês por extenso somente quando o mesmo não for inválido.

2 - Leia os enunciados a seguir e crie os algoritmos e os programas em C++ que solucionem os problemas propostos.

a) Peça ao usuário que informe a medida de cada um dos três lados de um triângulo e informe se ele é equilátero, escaleno ou isósceles. Veja como é a classificação dos triângulos:

Equilátero: Três lados com a mesma medida;

Escaleno: Dois lados quaisquer com a mesma medida;

Isósceles: Três lados com medidas diferentes.

b) O Jockey Clube está organizando uma competição de hipismo e criou as seguintes categorias:

INFANTIL – competidores entre 7 e 12 anos;

JUVENIL – competidores entre 13 e 17 anos;

ADULTO – competidores entre 18 e 49 anos;

SENIOR – competidores com 50 anos ou mais.

Para ajudar na classificação correta, crie um código capaz de informar a categoria do competidor a partir de sua idade.

Estrutura de seleção CASO ou SWITCH

A estrutura CASO (nos algoritmos) ou SWITCH (nos programas em C++) são semelhantes à estrutura SE ou IF, mas possui algumas restrições:

Uso dos operadores lógicos para avaliar mais de uma condição em um mesmo teste: isso não é possível na estrutura CASO ou SWITCH.

Os operadores relacionais (como **maior que** ou **menor ou igual a**) também não são permitidos. Aqui, os valores a serem testados precisam ser definidos e a variável será igual a eles ou diferente dos mesmos.

Observe a sintaxe desta estrutura no Portugol Studio e no C++:

* Portugol Studio

```
escolha (variável_a_ser_testada)
{
  caso valor1:
    // Comandos a serem executados.
    pare
  caso valor2:
    // Comandos a serem executados.
    pare
  caso valorN:
    // Comandos a serem executados.
    pare
  caso contrario:
    // Comandos a serem executados caso a variável testada não seja
    // igual a nenhum dos valores testados nos “casos”.
}
```

* C++

```
switch (variável_a_ser_testada)
{
  case valor1:
    // Comandos a serem executados.
    break;
  case valor2:
    // Comandos a serem executados.
    break;
  case valorN:
    // Comandos a serem executados.
    break;
  default:
    // Comandos a serem executados caso a variável testada não seja
    // igual a nenhum dos valores testados nos “cases”.
}
```

Atividade:

3 - Os exercícios do triângulo e da competição de hipismo poderiam ser escritos utilizando a estrutura CASO ou SWITCH? Por quê?

Vamos ver, então, como ficaria o trecho da estrutura seletiva do algoritmo que escreve o mês por extenso utilizando a estrutura **CASO**?

Observe:

```
escolha (numMes)
  {
    caso 1:
      mesExtenso="janeiro"
      pare
    caso 2:
      mesExtenso="fevereiro"
      pare
    caso 3:
      mesExtenso="março"
      pare
    caso 4:
      mesExtenso="abril"
      pare
  caso 5:
    mesExtenso="maio"
    pare
  caso 6:
    mesExtenso="junho"
    pare
  caso 7:
    mesExtenso="julho"
    pare
  caso 8:
    mesExtenso="agosto"
    pare
  caso 9:
    mesExtenso="setembro"
    pare
  caso 10:
    mesExtenso="outubro"
    pare
    caso 11:
      mesExtenso="novembro"
      pare
    caso 12:
      mesExtenso="dezembro"
      pare
    caso contrario:
      escreva("Mês inválido!!!")
}
```

Você não acha que esta estrutura está mais inteligível do que a do primeiro exemplo?

Veja que não há necessidade de repetir a variável a cada novo teste. Ela é escrita somente na primeira linha da estrutura.

Dica

Para o programa em C++, declare as variáveis da seguinte maneira:

int numMes;
char *mesExtenso;
bool invalido=false;

A linha **int numMes;** declara a variável **numMes**, responsável por armazenar o número do mês informado via teclado pelo usuário.
A linha **char *mesExtenso;** declara a variável **mesExtenso**, que armazenará o nome do mês equivalente ao número digitado.
O asterisco (*) que antecede o nome da variável informa ao C++ que a mesma não tem um tamanho definido. Fazemos isso porque o nome de cada mês é formado por uma quantidade diferente de caracteres: “janeiro” é composto por sete caracteres, enquanto que “fevereiro” é composto por nove e “março” é composto por somente cinco caracteres.
A linha **bool invalido=false;** declara a variável lógica **invalido**, que utilizamos anteriormente para evitar que o programa exiba a mensagem do nome do mês por extenso quando o usuário informar um mês inválido.

Atividade:

4 - Como seria o programa em C++? Você conseguiria escrevê-lo? Lembre-se de fazer a correção vista anteriormente utilizando a variável lógica.

Muito bem, chegamos ao final de mais uma aula. Não deixe de realizar as atividades propostas conforme sugerido nos enunciados. É importante que você perceba as diferenças entre as estruturas seletivas apresentadas e saiba quando escolher uma em detrimento da outra.

Atividade

5 - Uma lanchonete está disponibilizando para seus clientes um terminal por meio do qual poderão fazer os pedidos. Para que isso aconteça, o terminal deve exibir o menu de itens disponíveis conforme se vê a seguir:

*** Cardápio ***

100 – Hambúrguer – R$5,50

101 – Cachorro-quente – R$4,50

102 – Milkshake – R$7,00

103 – Pizza brotinho – R$8,00

104 - Cheeseburguer – R$8,50

Informe o código do seu pedido:

Uma vez que o cliente informe o código do item desejado, o terminal deverá perguntar a quantidade de itens que ele deseja pedir. Ao final, o usuário deverá informar o pedido do cliente e o valor a pagar.

Escreva o algoritmo e seu equivalente em C++ que possam resolver o problema da lanchonete. Utilize as estruturas CASO e SWITCH.

Notas
Referências

MANZANO, J. A. N. G., OLIVEIRA, J. F. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 28.ed. São Paulo: Érica, 2016.

PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados com aplicações em Java**. 2.ed. São Paulo: Prentice Hall, 2005.

Próxima aula

- Estruturas de repetição;
- Estruturas de repetição com pré-teste e com pós-teste;
- Algoritmos e programas com a estrutura de repetição adequada.

Explore mais

Quando você precisar testar programas em C++ e não dispuser do DEV C++ instalado na máquina que estiver utilizando, acesse o ambiente onlinegdb <https://www.onlinegdb.com/online_c++_compiler> .