Disciplina: Introdução a Programação

Aula 5: Estrutura de seleção simples e composta

Apresentação

Na aula anterior, começamos a escrever códigos simples utilizando os comandos de entrada e saída. Você deve ter percebido que os algoritmos e os programas que vimos como exemplos são executados sequencialmente, ou seja, da primeira linha à ultima, sem nenhum desvio.

Nesta aula, veremos que existem muitas situações em que o fluxo de execução não pode ser sequencial. Há muitos casos em que a solução somente pode passar por determinado trecho do código quando uma condição é satisfeita.

Para criar testes que determinam o caminho a ser seguido na execução do programa ou algoritmo, você precisará recorrer às estruturas de seleção, ou estruturas seletivas. Essas estruturas são responsáveis por avaliar expressões relacionais ou lógicas denominadas condições e, assim, criar uma restrição para que uma parte do programa possa ser executada.

Os programas ficam mais interessantes à medida que novas estruturas são conhecidas. Ao aprender um novo conceito, certifique-se de que não tem dúvidas antes de seguir adiante.

Objetivos

- Identificar as estruturas seletivas;
- Aplicar as estruturas de seleção para escrever algoritmos e programas;
- Analisar os resultados produzidos por códigos que empreguem estruturas seletivas.

Estrutura Seletiva

Você se lembra do código que escrevemos na aula anterior para calcular a idade de um indivíduo? Imagine que, além de informar a idade, você deseje dizer ao usuário se ele já pode votar.

Para ser um eleitor, é preciso ter a idade mínima de 16 anos. Isso significa que você precisar verificar se o usuário tem 16 anos ou mais para poder informar que o mesmo pode votar. Para que isso seja possível, será preciso fazer uso de uma **estrutura seletiva**, conhecida também como **estrutura de seleção**.

Uma estrutura seletiva restringe a execução de um trecho do código à veracidade, ou não, de uma ou mais condições. No seu formato mais simples, uma estrutura seletiva, no Portugol Studio e no C++, tem a seguinte sintaxe:

```
* Portugol Studio
se (condição)
{
// Comandos a serem executados se a condição for verdadeira.
}

* C++
if (condição)
{
// Comandos a serem executados se a condição for verdadeira.
}
```

Você se lembra de quando falamos das expressões relacionais e lógicas?

Uma condição será sempre uma expressão desse tipo. Retomando a situação proposta no início desta aula, na qual desejamos informar ao usuário se ele pode votar, poderíamos ter uma estrutura seletiva escrita conforme vemos a seguir:

* Portugol Studio

```
se (idade>=16)
{
escreva("Você já pode votar!")
}
```

Observe que estamos avaliando a condição (idade>=16) – que é uma expressão relacional – e, se ela for verdadeira, exibimos a mensagem informando ao usuário que ele já pode votar. Em C++, o trecho seria escrito da seguinte maneira:

* C++

```
if (idade>=16)
{
cout <<"Você já pode votar!";
}
```

Observe como ficaria, então, a solução completa no Portugol Studio:

```
1. programa
2. {
3. funcao inicio()
4. {
5. inteiro anoNasc, anoAtual, idade
6. escreva("Ano atual: ")
7. leia(anoAtual)
8. escreva("Ano de nascimento: ")
9. leia(anoNasc)
10. idade=anoAtual-anoNasc
11. escreva("Você tem ou fará ", idade, " anos")
12. se (idade>=16)
13. {
14. escreva("Você já pode votar!")
15. }
16. }
17. }
```

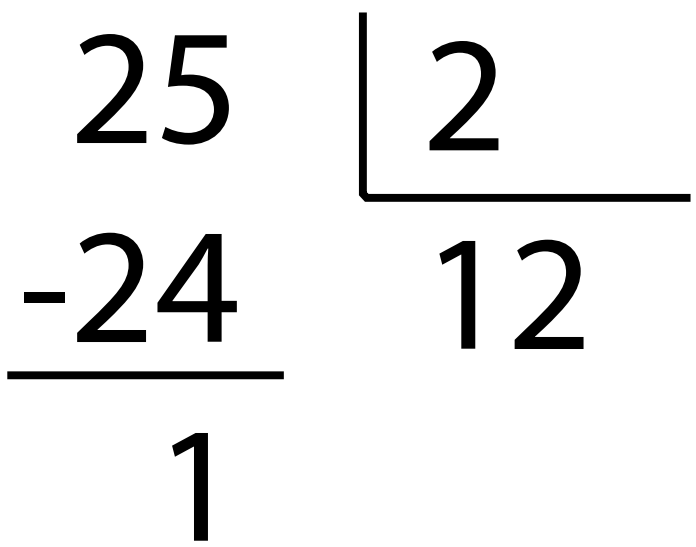
Agora veja o programa completo em C++:

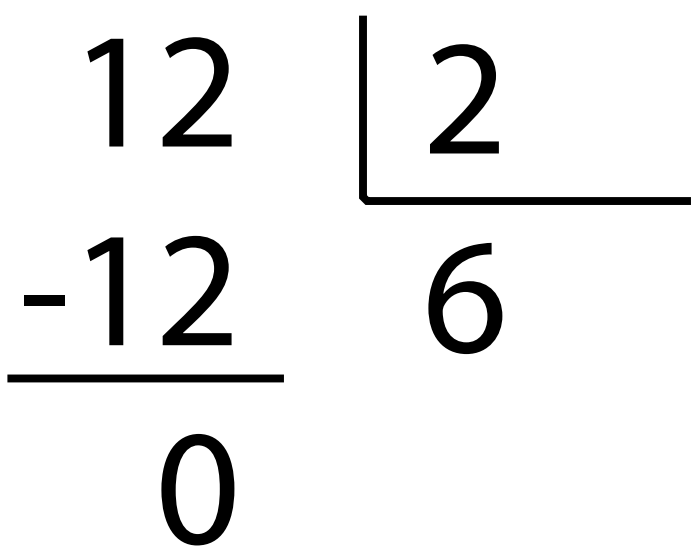
```
1. #include 2. using namespace std;
3. int main()
4. {
5. int anoAtual, anoNasc, idade;
6. cout <<"Ano atual: ";
7. cin >> anoAtual;
8. cout <<"Ano de nascimento:";
9. cin >> anoNasc;
10. idade=anoAtual-anoNasc;
11. cout <<"Você tem ou fará " << idade << " anos.";
12. if (idade>=16)
13. {
14. cout <<"Você já pode votar!";
15. }
16. }
```

Exemplo

Vamos ver mais um exemplo?

Imagine que você deseja verificar se um número qualquer informado pelo usuário é múltiplo de 2. Para que seja múltiplo de 2, o resto da divisão do número por 2 precisa ser igual a 0. Observe:





A divisão do número 25 por 2 tem resto igual a 1: o que significa que 25 não é múltiplo de 2.

A divisão do número 12 por 2 tem resto igual a 0; o que significa que 12 é múltiplo de 2.

Precisamos, então, receber o número informado pelo usuário, efetuar sua divisão por 2 e descobrir qual é o resto. Para isso, utilizaremos a função módulo, expressa pelo caractere %.

Estudamos esse operador na aula 2, você se lembra?

Veja como ficaria o algoritmo para esse problema:

```
1. programa
2. {
3. funcao inicio()
4. {
5. inteiro num, resto
6. escreva("Informe um número qualquer: ")
7. leia(num)
8. resto=num%2
9. se (resto==0)
10. {
11. escreva("Este número é múltiplo de 2!")
12. }
13. }
14. }
```

Agora veja a solução em C++

```
1. #include
2. using namespace std;
3. int main()
4. {
5. int num, resto;
6. cout <<"Informe um número qualquer: ";
7. cin >> num;
8. resto=num%2;
9. if (resto==0)
10. {
11. cout <<"Este número é múltiplo de 2!";
12. }
13. }
```

Atividade

1. Leia os enunciados a seguir e crie os algoritmos e os programas em C++ que solucionem os problemas propostos. Exiba o quadrado de um número inteiro qualquer informado pelo usuário e diga se ele é maior do que 25. Receba a quantidade de dias que uma pessoa já viveu e informe quantos anos ela tem. Desconsidere os anos bissextos. Informe se a pessoa já é maior de idade.

Nos dois exemplos que vimos – o primeiro que informa se o usuário pode votar e o segundo que diz se o número informado é múltiplo de 2 – a mensagem somente é exibida quando a condição avaliada é verdadeira.

No primeiro exemplo, se o usuário ainda não tiver 16 anos e, por isso, não puder votar, nada será exibido depois da idade dele.

No segundo exemplo, nenhuma mensagem é exibida quando o número não é múltiplo de 2, mas seria interessante que algo fosse dito ao usuário quando esse for o caso, ou ele vai informar um número e nada acontecerá.

Para casos assim, nos quais desejamos que um bloco de comandos seja executado quando a condição for verdadeira e outro bloco seja executado quando a condição for falsa, precisamos complementar a estrutura seletiva conforme vemos a seguir:

* Portugol Studio

```
se (condição)
{
// Comandos a serem executados se a condição for verdadeira.
}
senao
{
// Comandos a serem executados se a condição for falsa.
}
```

* C++

```
if (condição)
{
// Comandos a serem executados se a condição for verdadeira.
}
else
{
// Comandos a serem executados se a condição for falsa.
}
```

Revendo as estruturas de seleção utilizadas no primeiro exemplo e adaptando-as com a nova possibilidade apresentada, ou seja, de definir o que será feito quando a condição for falsa, temos:

* Portugol Studio

```
se (idade>=16)
{
escreva("Você já pode votar!")
}
senao
{
escreva("Você ainda não pode votar!")
}
```

* C++

```
if (idade>=16)
{
cout <<"Você já pode votar!";
}
else
{
Cout <<"Você ainda não pode votar!";
}
```

Que tal vermos como ficaria a solução que verifica se o número é múltiplo de 2? Observe a seguir:

```
* Portugol Studio

   se (resto==0)
   {
   escreva("Este número é múltiplo de 2!")
   }
   senao
   {
   escreva("Este número não é múltiplo de 2!")
   }
```

Atividade

2. Escreva os algoritmos e os programas em C++ para os enunciados a seguir.
Receba um número qualquer e informe se ele é par ou ímpar.
Receba um número qualquer e informe se ele é positivo ou negativo.
Receba o valor de um salário e informe se ele é maior ou menor do que o salário mínimo. Considere o valor de 1000,00 como sendo o mínimo.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Em todos os exemplos vistos até agora, analisamos somente uma condição nas estruturas de condição.

O que fazer quando precisarmos avaliar duas ou mais condições?

Imagine que você deseja receber um número do usuário e informar se o número é **par** e **positivo**.
Para que seja par, o número precisa ser divisível por **2**; e para ser positivo, o número precisa ser maior do que **0**.

Veja que temos, então, duas situações distintas para avaliar; o que nos deixa com duas condições a serem testadas.

Como faríamos neste caso?

Observe o algoritmo a seguir:

```
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14. programa
{
funcao inicio()
{
inteiro num, resto
escreva("Informe um número qualquer: ")
leia(num)
resto=num%2
se (resto==0 e num>0)
{
escreva("Este número é par e positivo!")
}
}
}
```

Agora veja a solução em C++

```
1. #include
2. using namespace std;
3. int main()
4. {
5. int num, resto;
6. cout <<"Informe um número qualquer: ";
7. cin >> num;
8. resto=num%2;
9. if (resto==0 && num>0)
10. {
11. cout <<"Este número é par e positivo!";
12. }
13. }
```

Observe que, na linha inicial da estrutura seletiva, onde incluímos o teste condicional, vemos não só uma, mas duas condições unidas pelo operador lógico **E**.

Você deve se lembrar que o resultado final de uma expressão lógica utilizando esse operador somente será verdadeira quando todos os termos avaliados forem verdadeiros.

Isso significa que, para que a mensagem **Este número é par e positivo!** seja impressa, o número precisa ser par e o número precisa ser positivo. Se uma dessas condições for falsa, a mensagem não será impressa.

Saiba que, assim como utilizamos o operador **E** no exemplo anterior, você pode utilizar o operador **OU** e o operador **NÃO**; a escolha dependerá da situação a ser avaliada.

Atividade

3. Escreva um algoritmo e seu equivalente em C++ que receba uma hora qualquer (sem os minutos e os zeros) e informe se é dia ou noite. Será dia se a hora estiver entre 6 e 18.

Esta foi a primeira vez que falamos sobre estruturas seletivas, mas continuaremos falando sobre elas na próxima aula, quando você vai ver que, muitas vezes, precisamos avaliar não somente uma ou duas condições, mas várias delas.

Imagine quantas opções existem em uma calculadora, por exemplo.

Você pode somar, subtrair, multiplicar, dividir, calcular potências, guardar o número na memória de cálculo e várias outras coisas mais!

Como será que um programa faz para dar conta de tantas alternativas?

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Notas

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

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 seletivas aninhadas;
- Alternativa à estrutura “se..então..senão”;
- Estrutura de seleção adequada.

Explore mais

- Acesse o nível Masmorra Kithgard <https://br.codecombat.com/play> para jogar as primeiras etapas gratuitas, enquanto pratica conceitos de programação.