Expressões Lambda em C++

Introdução

Expressões lambda em C++ são funções anônimas que podem capturar variáveis do escopo circundante. Introduzidas no C++11, elas facilitam a criação de funções rápidas e concisas, especialmente úteis em contextos onde funções de callback ou funções locais são necessárias.

1. Definição e Sintaxe

- Definição: Uma expressão lambda define uma função anônima e pode capturar variáveis do escopo em que é definida.
- Sintaxe:

```
[captura](parametros) -> tipo_retorno {
    corpo
};
```

2. Exemplo de Expressão Lambda

- Exemplo:

```
auto soma = [](int a, int b) -> int {
    return a + b;
};

int resultado = soma(5, 3);  // resultado será 8
```

3. Captura de Variáveis

- Definição: Lambdas podem capturar variáveis do escopo circundante por valor (cópia) ou por referência.

- Sintaxe:

```
[captura_por_valor](parametros) { corpo }
[&captura_por_referencia](parametros) { corpo }
```

- Exemplo:

```
int x = 10;
int y = 20;

auto lambda_valor = [x, y]() {
    return x + y;
};  // Captura por valor

auto lambda_referencia = [&x, &y]() {
    x = 30;
    y = 40;
};  // Captura por referência

lambda_referencia();
int resultado = lambda_valor();  // resultado será 30
```

4. Captura Implícita

- Definição: Lambdas podem capturar todas as variáveis automaticamente por valor ou por referência.

- Sintaxe:

```
[=](parametros) { corpo }   // Captura todas por valor
[&](parametros) { corpo }   // Captura todas por referência
```

- Exemplo:

```
int a = 5, b = 10;

auto lambda_todas_valor = [=]() {
    return a + b;
};

auto lambda_todas_referencia = [&]() {
    a = 15;
    b = 20;
};

lambda_todas_referencia();
int resultado = lambda_todas_valor();  // resultado será 35
```

5. Lambdas Genéricos

- Definição: Lambdas podem ser genéricos, aceitando qualquer tipo de argumento utilizando templates.

- Sintaxe:

```
auto lambda_generico = [](auto a, auto b) {
    return a + b;
};
```

- Exemplo:

```
auto soma = [](auto a, auto b) {
    return a + b;
};

int resultado1 = soma(10, 20);        // resultado1 será 30
double resultado2 = soma(2.5, 3.5);   // resultado2 será 6.0
```

6. Utilizando `std::function` com Lambdas

- Definição: `std::function` pode armazenar lambdas, funções, e objetos funtores, permitindo flexibilidade na manipulação de funções.
- Exemplo:

```
#include <functional>
#include <iostream>

std::function<int(int, int)> funcao = [](int a, int b) {
    return a + b;
};

std::cout << funcao(2, 3) << std::endl;  // Saída: 5
```

7. Lambdas com Estado

- Definição: Lambdas podem capturar variáveis do escopo circundante, permitindo que mantenham estado entre as chamadas.
- Exemplo:

```
  auto contador = [n = 0]() mutable {
      return ++n;
  };

  int primeira = contador();  // primeira será 1
  int segunda = contador();   // segunda será 2
```

Dicas de Boas Práticas

- Captura Minimizada: Capture apenas as variáveis necessárias para evitar cópias desnecessárias ou modificações indesejadas.
- Uso de `auto`: Utilize `auto` para definir expressões lambda, simplificando a declaração de tipos.
- Legibilidade: Use lambdas para tornar o código mais legível e evitar a necessidade de definir funções adicionais em escopos limitados.

Esta seção abrange os conceitos sobre expressões lambda em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/language/lambda