Programação em C++ do básico ao avançado.

Programação em C++ do básico ao avançado.

Introdução

Este livro é um guia completo para aprender programação em C++. Ele foi escrito para iniciantes e profissionais, com o objetivo de fornecer uma compreensão sólida da linguagem e suas aplicações.

O livro é dividido em duas partes:

- A primeira parte, "Básico", cobre os fundamentos da programação C++, incluindo sintaxe, variáveis, operadores, estruturas de controle, funções e tipos de dados.
- A segunda parte, "Avançado", cobre tópicos mais avançados, como classes, objetos, herança, polimorfismo, templates e exceções.

Ao longo do livro, você encontrará muitos exemplos de código para ajudá-lo a entender os conceitos apresentados. Também há exercícios para ajudá-lo a praticar o que você aprendeu.

Requisitos

Para aproveitar ao máximo este livro, você deve ter algum conhecimento básico de matemática e lógica. Também é útil ter alguma experiência com outra linguagem de programação, como Python ou Java.

Programação em C++ do básico ao avançado.

O que você aprenderá

Ao concluir este livro, você será capaz de:

- Escrever programas em C++
- Usar os fundamentos da linguagem, como variáveis, operadores, estruturas de controle e funções
- Usar classes e objetos para criar programas orientados a objetos
- Usar herança, polimorfismo e templates para criar programas mais complexos
- Lidar com exceções para tornar seus programas mais robustos

Programação em C++ do básico ao avançado.

Como usar este livro

Este livro é projetado para ser usado como um guia passo a passo. Comece no início e siga as instruções cuidadosamente. Não hesite em consultar os exemplos de código e exercícios para ajudá-lo a entender os conceitos apresentados.

Para obter o máximo de aproveitamento deste livro, é importante praticar o que você aprendeu. Experimente escrever seus próprios programas e resolva problemas. Você também pode encontrar muitos recursos online para ajudá-lo a aprender mais sobre C++.

Boa sorte em seu aprendizado!

Programação em C++ do básico ao avançado.

Conteúdo

Exercícios

Ao longo do livro, você encontrará muitos exercícios para ajudá-lo a praticar o que você aprendeu. Esses exercícios são projetados para ajudá-lo a entender os conceitos apresentados e a desenvolver suas habilidades de programação.

Você pode encontrar as soluções para os exercícios no final do livro. No entanto, é importante tentar resolver os exercícios por conta própria antes de consultar as soluções.

Recursos adicionais

Existem muitos recursos online para ajudá-lo a aprender mais sobre C++. Aqui estão alguns recursos que você pode encontrar úteis:

- O site da linguagem C++: https://isocpp.org/
- O site da biblioteca padrão C++: https://en.cppreference.com/w/
- O site Stack Overflow: https://stackoverflow.com/
- O site GitHub: https://github.com/

Espero que você aproveite este livro e aprenda muito sobre programação C++!

Programação em C++ do básico ao avançado.

Capítulo 1: Introdução à programação

1.1 O que é programação?

Programação é o processo de escrever instruções para um computador. Essas instruções são chamadas de código, e elas são usadas para controlar o comportamento do computador.

A programação é uma ferramenta poderosa que pode ser usada para resolver uma ampla gama de problemas. Por exemplo, a programação pode ser usada para:

- Criar software, como aplicativos, jogos e sites.
- Automatizar tarefas, como o envio de e-mails ou a coleta de dados.
- Controlar dispositivos físicos, como robôs ou máquinas.

1.2 Algoritmos

Um algoritmo é um conjunto de instruções que descreve como resolver um problema. Os algoritmos são a base da programação, e eles são usados para garantir que os programas funcionem corretamente.

Um algoritmo típico consiste em uma série de passos que devem ser executados em uma determinada ordem. Por exemplo, um algoritmo para encontrar o maior número em uma lista de números poderia ser o seguinte:

1. Compare o primeiro número com o segundo.
2. Se o primeiro número for maior, então ele é o maior número.
3. Caso contrário, compare o segundo número com o terceiro.
4. Continue comparando os números até encontrar o maior.

Programação em C++ do básico ao avançado.

1.3 Linguagens de programação

Uma linguagem de programação é um conjunto de regras que definem como escrever código. Existem muitas linguagens de programação diferentes, cada uma com suas próprias características e vantagens.

Algumas das linguagens de programação mais populares incluem:

- Python
- Java
- C++
- C#
- JavaScript

1.4 Como aprender a programar

A melhor maneira de aprender a programar é praticar. Há muitos recursos disponíveis para ajudá-lo a aprender, incluindo livros, tutoriais online e cursos.

Aqui estão algumas dicas para ajudá-lo a começar:

- Comece com um tutorial básico para aprender os fundamentos da programação.
- Pratique escrevendo código e resolvendo problemas.
- Junte-se a uma comunidade online de programadores para obter ajuda e apoio.

1.5 Conclusão

A programação é uma habilidade valiosa que pode ser usada para resolver uma ampla gama de problemas. Com um pouco de prática, você pode aprender a programar e criar seus próprios programas.

Programação em C++ do básico ao avançado.

Capítulo 2: Sintaxe de C++

2.1 Introdução

A sintaxe de uma linguagem de programação é o conjunto de regras que definem como escrever código. A sintaxe de C++ é baseada na sintaxe da linguagem C, mas também inclui recursos adicionais, como orientação a objetos e templates.

2.2 Componentes do código C++

O código C++ é composto por um conjunto de componentes, incluindo:

- Palavras-chave: Palavras reservadas que têm um significado especial na linguagem.
- Identificadores: Nomes usados para identificar variáveis, funções, classes, etc.
- Constantes: Valores que não podem ser alterados.
- Operadores: Símbolos usados para realizar operações matemáticas, lógicas e de fluxo de controle.
- Palavras-chave: Palavras reservadas que têm um significado especial na linguagem.
- Identificadores: Nomes usados para identificar variáveis, funções, classes, etc.
- Constantes: Valores que não podem ser alterados.
- Operadores: Símbolos usados para realizar operações matemáticas, lógicas e de fluxo de controle.
- Comentarios: Texto que não é executado pelo compilador.

2.3 Comentários

Comentários são usados para documentar o código e explicar o seu significado. Comentários podem ser incluídos em qualquer lugar do código, e eles são ignorados pelo compilador.

Programação em C++ do básico ao avançado.

2.4 Linhas de código

As linhas de código são separadas por caracteres de nova linha. O compilador lê o código linha por linha, então é importante que cada linha de código seja sintaticamente correta.

2.5 Espaços em branco

Espaços em branco são ignorados pelo compilador. No entanto, eles podem ser usados para melhorar a legibilidade do código.

2.6 Nomes de variáveis

Os nomes de variáveis devem seguir as seguintes regras:

- Devem começar com uma letra ou um caractere sublinhado.
- Podem conter letras, dígitos e caracteres sublinhados.
- Não devem ser palavras-chave ou nomes de bibliotecas.

2.7 Tipos de dados

Os tipos de dados definem o tipo de dado que pode ser armazenado em uma variável. Os tipos de dados mais comuns em C++ incluem:

- int: Números inteiros.
- float: Números de ponto flutuante.
- double: Números de ponto flutuante de precisão dupla.
- char: Caracteres.
- bool: Booleanos (verdadeiro ou falso).

Programação em C++ do básico ao avançado.

2.8 Variáveis

As variáveis são usadas para armazenar dados. Para declarar uma variável, você usa a seguinte sintaxe:

C++
```
tipo_de_dado nome_da_variavel;
```

Por exemplo, a seguinte declaração declara uma variável inteira chamada `numero`:

C++
```
int numero;
```

2.9 Atribuição de valores

Os valores podem ser atribuídos a variáveis usando o operador de atribuição (=). Por exemplo, a seguinte instrução atribui o valor 10 à variável `numero`:

C++
```
numero = 10;
```

2.10 Operadores

Os operadores são usados para realizar operações matemáticas, lógicas e de fluxo de controle. Os operadores mais comuns em C++ incluem:

- Operadores matemáticos: +, -, *, /, %
- Operadores lógicos: &&, ||, !
- Operadores de fluxo de controle: if, else, for, while, do while

Programação em C++ do básico ao avançado.

2.11 Exemplos de código

Aqui está um exemplo de código C++ simples:

C++

```
// Este programa imprime a mensagem "Olá, mundo!"

#include <iostream>

int main() {
  std::cout << "Olá, mundo!";
  return 0;
}
```

Este programa usa a biblioteca padrão `iostream` para imprimir texto na saída padrão. A função `main()` é a função principal de todo programa C++.

Programação em C++ do básico ao avançado.

Capítulo 3: Variáveis e operadores

3.1 Introdução

Variáveis e operadores são dois dos conceitos mais importantes da programação C++. Variáveis são usadas para armazenar dados, e operadores são usados para realizar operações com esses dados.

3.2 Variáveis

Uma variável é uma referência a um espaço na memória do computador. Esse espaço é usado para armazenar um valor de determinado tipo.

Para declarar uma variável, você usa a seguinte sintaxe:

C++

```
tipo_de_dado nome_da_variavel;
```

Por exemplo, a seguinte declaração declara uma variável inteira chamada `numero`:

C++

```
int numero;
```

Uma vez declarada, uma variável pode ser usada para armazenar um valor. Para atribuir um valor a uma variável, você usa o operador de atribuição (=).

Por exemplo, a seguinte instrução atribui o valor 10 à variável `numero`:

C++

```
numero = 10;
```

O tipo de dado de uma variável determina o tipo de valor que ela pode armazenar. Existem muitos tipos de dados diferentes em C++, incluindo:

int: Números inteiros.

float: Números de ponto flutuante.

double: Números de ponto flutuante de precisão dupla.

char: Caracteres.

bool: Booleanos (verdadeiro ou falso).

Programação em C++ do básico ao avançado.

3.3 Operadores

Os operadores são usados para realizar operações com variáveis. Existem muitos tipos diferentes de operadores em C++, incluindo:

Operadores matemáticos: +, -, *, /, %

Operadores lógicos: &&, ||, !

Operadores de fluxo de controle: if, else, for, while, do while

Os operadores matemáticos são usados para realizar operações matemáticas básicas, como soma, subtração, multiplicação, divisão e resto. Por exemplo, a seguinte expressão calcula o produto de dois números:

C++

```
int a = 10;
int b = 20;

int produto = a * b;
```

Os operadores lógicos são usados para realizar operações lógicas, como comparação, conjunção, disjunção e negação. Por exemplo, a seguinte expressão verifica se duas variáveis são iguais:

C++

```
int a = 10;
int b = 20;

bool iguais = a == b;
```

Os operadores de fluxo de controle são usados para controlar o fluxo de execução de um programa. Por exemplo, o operador `if` é usado para testar uma condição e executar um bloco de código se a condição for verdadeira.

Programação em C++ do básico ao avançado.

3.4 Exemplos de código

Aqui está um exemplo de código C++ que usa variáveis e operadores:

C++

```
// Este programa calcula a área de um círculo

#include <iostream>

int main() {

  // Declara duas variáveis para armazenar o raio e a área
  float raio = 10.0;
  float area;

  // Calcula a área
  area = 3.14159 * raio * raio;

  // Imprime a área
  std::cout << "A área do círculo é " << area << std::endl;

  return 0;
}
```

Este programa declara duas variáveis, `raio` e `area`. A variável `raio` armazena o raio do círculo, e a variável `area` armazena a área do círculo.

O programa calcula a área usando a seguinte fórmula:

```
area = pi * raio^2
```

Onde:

`pi` é uma constante que representa a constante pi.

`raio` é o raio do círculo.

O programa imprime a área na saída padrão.

Programação em C++ do básico ao avançado.

3.5 Conclusão

Variáveis e operadores são dois conceitos essenciais da programação C++. Ao entender como usar variáveis e operadores, você poderá escrever programas mais poderosos e eficientes.

Programação em C++ do básico ao avançado.

Capítulo 4: Estruturas de controle

4.1 Introdução

As estruturas de controle são usadas para controlar o fluxo de execução de um programa. Elas permitem que você execute código condicionalmente, repetidamente ou em ordem específica.

4.2 Seleção

As estruturas de seleção são usadas para executar código condicionalmente. A estrutura de seleção mais comum em C++ é o operador `if`:

C++

```
if (condição) {
  // Código executado se a condição for verdadeira
}
```

Por exemplo, a seguinte instrução executa um bloco de código se a variável `a` for maior que 10:

C++

```
if (a > 10) {
  // Código executado se a variável a for maior que 10
}
```

O operador `if` pode ser combinado com o operador `else` para formar uma estrutura de seleção `if-else`:

C++

```
if (condição) {
  // Código executado se a condição for verdadeira
} else {
  // Código executado se a condição for falsa
}
```

Programação em C++ do básico ao avançado.

Por exemplo, a seguinte instrução executa um bloco de código se a variável `a` for maior que 10, ou outro bloco de código se a variável `a` for menor ou igual a 10:

C++

```
if (a > 10) {
  // Código executado se a variável a for maior que 10
} else {
  // Código executado se a variável a for menor ou igual a 10
}
```

O operador `if` também pode ser combinado com o operador `else if` para formar uma estrutura de seleção `if-else-if`:

C++

```
if (condição 1) {
  // Código executado se a condição 1 for verdadeira
} else if (condição 2) {
  // Código executado se a condição 2 for verdadeira
} else {
  // Código executado se as condições 1 e 2 forem falsas
}
```

Por exemplo, a seguinte instrução executa um bloco de código se a variável `a` for igual a 10, outro bloco de código se a variável `a` for igual a 20, ou outro bloco de código se a variável `a` for diferente de 10 e 20:

C++

```
if (a == 10) {
  // Código executado se a variável a for igual a 10
} else if (a == 20) {
  // Código executado se a variável a for igual a 20
} else {
  // Código executado se a variável a for diferente de 10 e 20
}
```

Programação em C++ do básico ao avançado.

4.3 Repetição

As estruturas de repetição são usadas para executar código repetidamente. A estrutura de repetição mais comum em C++ é o laço `for`:

C++

```
for (int i = 0; i < 10; i++) {
  // Código executado 10 vezes
}
```

content_copy

Este laço executa o bloco de código 10 vezes, com o valor da variável `i` começando em 0 e incrementando em 1 a cada iteração.

O laço `for` pode ser usado para executar código repetidamente até que uma condição seja satisfeita. Por exemplo, o seguinte laço executa o bloco de código enquanto a variável `a` for menor que 10:

C++

```
for (int a = 0; a < 10; a++) {
  // Código executado enquanto a variável a for menor que 10
}
```

O laço `while` é outra estrutura de repetição que pode ser usada para executar código repetidamente. O laço `while` executa o bloco de código enquanto uma condição for verdadeira:

C++

```
int a = 0;

while (a < 10) {
  // Código executado enquanto a variável a for menor que 10
  a++;
}
```

Programação em C++ do básico ao avançado.

O laço `do while` é uma estrutura de repetição que executa o bloco de código pelo menos uma vez, e então executa o bloco de código novamente enquanto uma condição for verdadeira:

C++

```
int a = 0;


do {
  // Código executado pelo menos uma vez
  a++;
} while (a < 10);
```

4.4 Conclusão

As estruturas de controle são uma parte essencial da programação C++. Ao entender como usar estruturas de controle, você poderá escrever programas mais poderosos e eficientes.

Programação em C++ do básico ao avançado.

Capítulo 5: Funções

5.1 Introdução

As funções são blocos de código que podem ser reutilizados em diferentes partes de um programa. As funções tornam o código mais modular e reutilizável, e elas ajudam a organizar o código de forma mais lógica.

5.2 Declaração de funções

Para declarar uma função, você usa a seguinte sintaxe:

C++
```
tipo_de_retorno nome_da_funcao(lista_de_parametros) {
  // Código da função
}
```

Onde:

- `tipo_de_retorno` é o tipo de valor que a função retorna.
- `nome_da_funcao` é o nome da função.
- `lista_de_parametros` é uma lista de parâmetros que a função aceita.

Os parâmetros são usados para passar dados para a função. Os parâmetros podem ser de qualquer tipo de dado, incluindo variáveis, constantes e expressões.

Programação em C++ do básico ao avançado.

5.3 Chamada de funções

Para chamar uma função, você usa o seguinte operador:

C++
```
nome_da_funcao(lista_de_argumentos)
```

Onde:

- `nome_da_funcao` é o nome da função que você deseja chamar.
- `lista_de_argumentos` é uma lista de valores que você deseja passar para a função.

Os argumentos são usados para fornecer dados para a função. Os argumentos devem corresponder ao tipo de dados dos parâmetros da função.

5.4 Exemplos de código

Aqui está um exemplo de código C++ que declara e chama uma função:

C++
```
// Declaração da função
int soma(int a, int b) {
  return a + b;
}

// Chamada da função
int x = 10;
int y = 20;

int resultado = soma(x, y);

// Imprime o resultado
std::cout << "O resultado da soma é " << resultado << std::endl;
```

Esta função declara uma função chamada `soma()` que retorna o valor da soma de dois números inteiros. A função `soma()` aceita dois parâmetros, `a` e `b`, que são os números a serem somados.

A função `soma()` é chamada duas vezes, uma vez com os valores `10` e `20` e outra vez com os valores `5` e `6`. O resultado da soma é impresso na saída padrão.

Programação em C++ do básico ao avançado.

5.5 Conclusão

As funções são uma parte essencial da programação C++. Ao entender como usar funções, você poderá escrever programas mais modulares, reutilizáveis e eficientes.

Programação em C++ do básico ao avançado.

Capítulo 6: Tipos de dados

6.1 Introdução

Os tipos de dados definem o tipo de valor que pode ser armazenado em uma variável. Os tipos de dados mais comuns em C++ incluem:

- int: Números inteiros.
- float: Números de ponto flutuante.
- double: Números de ponto flutuante de precisão dupla.
- char: Caracteres.
- bool: Booleanos (verdadeiro ou falso).

6.2 Números inteiros

Os números inteiros são números que não têm parte decimal. Os tipos de dados inteiros mais comuns em C++ são:

- int: Um número inteiro de 32 bits.
- long: Um número inteiro de 64 bits.
- short: Um número inteiro de 16 bits.
- char: Um número inteiro de 8 bits que pode representar um único caractere.

6.3 Números de ponto flutuante

Os números de ponto flutuante são números que têm uma parte decimal. Os tipos de dados de ponto flutuante mais comuns em C++ são:

- float: Um número de ponto flutuante de 32 bits.
- double: Um número de ponto flutuante de 64 bits.

Programação em C++ do básico ao avançado.

6.4 Caracteres

Os caracteres são usados para representar letras, números e outros símbolos. O tipo de dados de caracteres mais comum em C++ é:

- char: Um número inteiro de 8 bits que pode representar um único caractere.

6.5 Booleanos

Os booleanos são usados para representar valores lógicos. O tipo de dados booleano mais comum em C++ é:

- bool: Um valor que pode ser `true` ou `false`.

6.6 Declaração de variáveis

Para declarar uma variável, você usa a seguinte sintaxe:

C++
```
tipo_de_dado nome_da_variavel;
```

Onde:

- `tipo_de_dado` é o tipo de dado da variável.
- `nome_da_variavel` é o nome da variável.

6.7 Atribuição de valores

Os valores podem ser atribuídos a variáveis usando o operador de atribuição (=). Por exemplo, a seguinte instrução atribui o valor 10 à variável `numero`:

C++
```
int numero;
numero = 10;
```

Programação em C++ do básico ao avançado.

6.8 Exemplos de código

Aqui está um exemplo de código C++ que declara e usa variáveis de diferentes tipos de dados:

C++
```
// Declaração de variáveis
int numero = 10;
float numero_real = 3.14159;
char letra = 'A';
bool verdadeiro = true;
bool falso = false;

// Impressão dos valores das variáveis
std::cout << "O número é " << numero << std::endl;
std::cout << "O número real é " << numero_real << std::endl;
std::cout << "A letra é " << letra << std::endl;
std::cout << "O valor verdadeiro é " << verdadeiro << std::endl;
std::cout << "O valor falso é " << falso << std::endl;
```

Este código declara seis variáveis de diferentes tipos de dados. As variáveis são inicializadas com os seguintes valores:

- `numero` é um número inteiro de 10.
- `numero_real` é um número de ponto flutuante de 3.14159.
- `letra` é um caractere `A`.
- `verdadeiro` é um booleano `true`.
- `falso` é um booleano `false`.

Os valores das variáveis são impressos na saída padrão.

6.9 Conclusão

Os tipos de dados são uma parte essencial da programação C++. Ao entender como usar tipos de dados, você poderá escrever programas mais precisos e eficientes.

Programação em C++ do básico ao avançado.

Capítulo 7: Classes e objetos

7.1 Introdução

Classes são um dos conceitos mais importantes da programação orientada a objetos. Classes são modelos que podem ser usados para criar objetos.

7.2 Declaração de classes

Para declarar uma classe, você usa a seguinte sintaxe:

C++
```
class nome_da_classe {
  // Declaração dos membros da classe
};
```

Onde:

- `nome_da_classe` é o nome da classe.
- `declaração dos membros da classe` é uma lista de membros da classe.

Os membros da classe podem ser variáveis, funções ou métodos.

7.3 Objetos

Objetos são instâncias de classes. Objetos são criados usando o operador `new`:

C++
```
nome_da_classe *objeto = new nome_da_classe;
```

Onde:

- `nome_da_classe` é o nome da classe.
- `objeto` é o nome da variável que armazenará o ponteiro para o objeto.

Programação em C++ do básico ao avançado.

7.4 Acesso aos membros da classe

Os membros da classe podem ser acessados usando o operador de ponto:

C++
```
objeto->membro_da_classe;
```

Onde:

- `objeto` é o nome da variável que armazena o ponteiro para o objeto.
- `membro_da_classe` é o nome do membro da classe que você deseja acessar.

Programação em C++ do básico ao avançado.

7.5 Exemplos de código

Aqui está um exemplo de código C++ que declara e usa uma classe:

C++
```
class Carro {
public:
  // Declaração de uma variável membro
  int numero_de_portas;

  // Declaração de um método membro
  void imprimir() {
    std::cout << "O carro tem " << numero_de_portas << " portas." <<
std::endl;
  }
};

int main() {
  // Cria um objeto da classe Carro
  Carro *carro = new Carro();

  // Atribui o valor 5 à variável membro numero_de_portas
  carro->numero_de_portas = 5;

  // Chama o método membro imprimir()
  carro->imprimir();

  // Deleta o objeto da memória
  delete carro;

  return 0;
}
```

Este código declara uma classe chamada `Carro`. A classe `Carro` tem uma variável membro chamada `numero_de_portas` e um método membro chamado `imprimir()`.

O programa cria um objeto da classe `Carro` e atribui o valor 5 à variável membro `numero_de_portas`. Em seguida, o programa chama o método membro `imprimir()`.

7.6 Conclusão

Classes são uma ferramenta poderosa que pode ser usada para organizar código e criar objetos com comportamentos complexos. Ao entender como usar classes, você poderá escrever programas mais modulares e reutilizáveis.

Programação em C++ do básico ao avançado.

Capítulo 8: Herança

8.1 Introdução

Herança é um dos conceitos mais importantes da programação orientada a objetos. Herança permite que você crie classes que herdem as propriedades de outras classes.

8.2 Declaração de classes herdadas

Para declarar uma classe herdada, você usa a seguinte sintaxe:

C++
```
class nome_da_classe : public nome_da_classe_base {
  // Declaração dos membros da classe
};
```

Onde:

- `nome_da_classe` é o nome da classe herdada.
- `nome_da_classe_base` é o nome da classe base.
- `declaração dos membros da classe` é uma lista de membros da classe.

8.3 Membros da classe base

Os membros da classe base são herdados pela classe herdada. Os membros da classe base podem ser acessados na classe herdada usando o operador de ponto.

8.4 Sobrescrita de métodos

Os métodos da classe base podem ser sobrescritos na classe herdada. A sobrescrita de métodos permite que você reimplemente o comportamento de um método da classe base.

Programação em C++ do básico ao avançado.

8.5 Exemplos de código

Aqui está um exemplo de código C++ que declara duas classes: uma classe base chamada `Carro` e uma classe herdada chamada `CarroPopular`:

C++

```
class Carro {
public:
  // Declaração de uma variável membro
  int numero_de_portas;

  // Declaração de um método membro
  void imprimir() {
    std::cout << "O carro tem " << numero_de_portas << " portas." <<
std::endl;
  }
};

class CarroPopular : public Carro {
public:
  // Declaração de uma variável membro
  bool ar_condicionado;

  // Declaração de um método membro
  void imprimir() {
    Carro::imprimir();
    std::cout << "O carro popular tem ar condicionado." << std::endl;
  }
};

int main() {
  // Cria um objeto da classe CarroPopular
  CarroPopular *carro = new CarroPopular();

  // Atribui o valor 5 à variável membro numero_de_portas
  carro->numero_de_portas = 5;

  // Atribui o valor true à variável membro ar_condicionado
  carro->ar_condicionado = true;

  // Chama o método membro imprimir()
  carro->imprimir();

  // Deleta o objeto da memória
  delete carro;

  return 0;
}
```

Este código declara a classe base `Carro` com uma variável membro `numero_de_portas` e um método membro `imprimir()`. A classe herdada `CarroPopular` herda a variável membro `numero_de_portas` e o método membro `imprimir()` da classe base `Carro`.

A classe `CarroPopular` também declara uma variável membro `ar_condicionado` e um método membro `imprimir()` que sobrescreve o método membro `imprimir()` da classe base `Carro`.

O programa cria um objeto da classe `CarroPopular` e atribui os valores 5 e true às variáveis membros `numero_de_portas` e `ar_condicionado`, respectivamente. Em seguida, o programa chama o método membro `imprimir()`.

A saída do programa é a seguinte:

```
O carro tem 5 portas.
O carro popular tem ar condicionado.
```

8.6 Conclusão

Herança é uma ferramenta poderosa que pode ser usada para reusar código e criar classes mais complexas. Ao entender como usar herança, você poderá escrever programas mais modulares e eficientes.

Programação em C++ do básico ao avançado.

Capítulo 10: Templates

10.1 Introdução

Templates são uma ferramenta poderosa que pode ser usada para criar código genérico que pode ser usado com diferentes tipos de dados.

10.2 Declaração de templates

Para declarar um template, você usa a seguinte sintaxe:

C++
```
template <typename T>
// Declaração do template
```

Onde:

- `T` é o nome do parâmetro do template.
- `declaração do template` é a declaração do código genérico.

10.3 Uso de templates

Para usar um template, você usa a seguinte sintaxe:

C++
```
// Declaração do objeto
T objeto;

// Inicialização do objeto
objeto = valor;

// Uso do objeto
// ...
```

Onde:

- `T` é o nome do parâmetro do template.
- `valor` é o valor a ser atribuído ao objeto.

Programação em C++ do básico ao avançado.

10.4 Exemplos de código

Aqui está um exemplo de código C++ que declara um template chamado `max()` que calcula o maior valor de dois números:

C++
```
template <typename T>
T max(T a, T b) {
  return a > b ? a : b;
}

int main() {
  // Chama o método max() com dois números inteiros
  int maior_inteiro = max(10, 20);

  // Chama o método max() com dois números de ponto flutuante
  double maior_real = max(10.0, 20.0);

  // Imprime os resultados
  std::cout << "O maior inteiro é " << maior_inteiro << std::endl;
  std::cout << "O maior real é " << maior_real << std::endl;

  return 0;
}
```

Este código declara o template `max()` que aceita dois parâmetros de tipo `T`. O método `max()` retorna o maior dos dois valores.

O programa chama o método `max()` com dois números inteiros e dois números de ponto flutuante. O programa imprime os resultados.

A saída do programa é a seguinte:

```
O maior inteiro é 20
O maior real é 20.0
```

10.5 Conclusão

Templates são uma ferramenta poderosa que pode ser usada para escrever código mais modular e eficiente. Ao entender como usar templates, você poderá escrever programas mais genéricos e reutilizáveis.

Programação em C++ do básico ao avançado.

Capítulo 11: Exceções

11.1 Introdução

Exceções são uma ferramenta poderosa que pode ser usada para lidar com erros durante a execução de um programa.

11.2 Lançamento de exceções

Para lançar uma exceção, você usa a seguinte sintaxe:

C++
```
throw exceção;
```

Onde:

- `exceção` é o objeto exceção que será lançado.

11.3 Captura de exceções

Para capturar uma exceção, você usa a seguinte sintaxe:

C++
```
try {
  // Código que pode lançar uma exceção
} catch (Exception e) {
  // Tratamento da exceção
}
```

Onde:

- `try` é a cláusula try.
- `catch` é a cláusula catch.
- `e` é a variável que armazena o objeto exceção.

11.4 Exemplos de código

Aqui está um exemplo de código C++ que lança uma exceção se o valor de uma variável for menor que zero:

C++
```
int main() {
  int valor = -1;

  // Lança uma exceção se o valor for menor que zero
  if (valor < 0) {
    throw std::invalid_argument("O valor deve ser maior que zero");
  }

  // O código abaixo não será executado se a exceção for lançada
  std::cout << "O valor é válido." << std::endl;

  return 0;
}
```

Este código declara uma variável chamada `valor` e atribui o valor `-1` a ela. A função `main()` lança uma exceção se o valor de `valor` for menor que zero.

Se a exceção for lançada, o programa exibirá a seguinte mensagem de erro:

```
Exception: O valor deve ser maior que zero
```

Conclusão

Exceções são uma ferramenta poderosa que pode ser usada para escrever código mais robusto e fácil de manter. Ao entender como usar exceções, você poderá escrever programas mais seguros e confiáveis.

Aqui estão alguns exemplos de exceções que podem ser lançadas em C++:

- `std::invalid_argument`: Argumento inválido.
- `std::out_of_range`: Índice fora do intervalo.
- `std::overflow_error`: Sobrecarga.
- `std::underflow_error`: Subcarga.
- `std::runtime_error`: Erro de tempo de execução.

Você pode encontrar mais informações sobre exceções na documentação da linguagem C++.

Programação em C++ do básico ao avançado.

Apêndice

- Referência de linguagem

A referência de linguagem C++ é um documento que fornece informações sobre a linguagem C++. A referência de linguagem é uma fonte essencial para qualquer desenvolvedor que deseja aprender C++.

A referência de linguagem C++ está disponível online e em formato impresso. A referência de linguagem online pode ser acessada no site da ISO. A referência de linguagem impressa pode ser adquirida em livrarias.

A referência de linguagem C++ é dividida em várias partes. A primeira parte da referência de linguagem fornece uma visão geral da linguagem C++. A segunda parte da referência de linguagem fornece informações sobre os tipos de dados em C++. A terceira parte da referência de linguagem fornece informações sobre as expressões em C++. A quarta parte da referência de linguagem fornece informações sobre as declarações em C++. A quinta parte da referência de linguagem fornece informações sobre as funções em C++. A sexta parte da referência de linguagem fornece informações sobre os operadores em C++. A sétima parte da referência de linguagem fornece informações sobre as classes e objetos em C++. A oitava parte da referência de linguagem fornece informações sobre a herança em C++. A nona parte da referência de linguagem fornece informações sobre o polimorfismo em C++. A décima parte da referência de linguagem fornece informações sobre os templates em C++. A décima primeira parte da referência de linguagem fornece informações sobre as exceções em C++.

A referência de linguagem C++ é um documento abrangente que fornece informações sobre todos os aspectos da linguagem C++. A referência de linguagem é uma fonte essencial para qualquer desenvolvedor que deseja aprender C++.

Aqui estão alguns recursos adicionais que podem ser úteis para aprender C++:

- Livros sobre C++
- Tutoriais sobre C++
- Cursos online sobre C++
- Comunidades online sobre C++

Programação em C++ do básico ao avançado.

Glossário

Aqui está um glossário de alguns dos termos mais comuns usados em C++:

- Atribuição: É o processo de atribuir um valor a uma variável.
- Argumento: É um valor que é passado para uma função.
- Arranjo: É uma coleção de dados que são armazenados em uma ordem específica.
- Booleano: É um tipo de dado que pode ter um dos dois valores: verdadeiro ou falso.
- Classe: É um modelo que pode ser usado para criar objetos.
- Comentário: É uma parte de código que não é interpretada pelo compilador.
- Declaração: É uma instrução que define uma variável, função ou outra entidade.
- Expressão: É uma combinação de operandos e operadores que produz um valor.
- Função: É um bloco de código que pode ser chamado por outro bloco de código.
- Herança: É um mecanismo que permite que uma classe herde as propriedades de outra classe.
- Infixo: É um operador que é colocado entre dois operandos.
- Objeto: É uma instância de uma classe.
- Operador: É um símbolo que é usado para realizar operações matemáticas, lógicas ou de outras naturezas.
- Programa: É um conjunto de instruções que são executadas pelo computador.
- Saída: É o resultado de um programa.
- Tipo de dado: É um tipo de informação que pode ser armazenada em uma variável.
- Variável: É uma área de memória que é usada para armazenar dados.

Programação em C++ do básico ao avançado.

Agradecimentos

Ao escrever este livro, meu objetivo era ajudar as pessoas que estão estudando programação. Se você chegou até aqui, significa que meu objetivo foi alcançado.

Agradeço a você por ter dedicado seu tempo para ler este livro. Espero que ele tenha sido útil e que você tenha aprendido algo novo.

Sou um grande fã de programação e estou sempre procurando maneiras de aprender mais sobre ela. Se você tiver comentários ou sugestões, não hesite em me contatar.

Obrigado novamente por sua leitura.

Yann Xavier