Conceitos Básicos de C++

Palavras-chave em C++

Introdução

Palavras-chave em C++ são identificadores reservados que têm significados específicos na linguagem e não podem ser utilizados para outros propósitos, como nomes de variáveis ou funções. Elas formam a base da sintaxe e da estrutura do código C++.

1. Palavras-chave de Controle de Fluxo

- `if`, `else`: Utilizadas para tomar decisões com base em condições.

```
if (condicao) {
    // código
} else {
    // código
}
```

- `switch`, `case`, `default`: Utilizadas para selecionar uma entre várias opções com base no valor de uma expressão.

```
switch (valor) {
    case 1:
        // código
        break;
    default:
        // código
```

```
}
```

- `for`, `while`, `do`: Utilizadas para criar loops.

```
for (int i = 0; i < 10; ++i) {
    // código
}

while (condicao) {
    // código
}

do {
    // código
} while (condicao);
```

- `break`, `continue`: Utilizadas para controlar o fluxo de loops.

```
for (int i = 0; i < 10; ++i) {
    if (i == 5) break; // sai do loop
    if (i % 2 == 0) continue; // pula para a próxima iteração
}
```

- `return`: Utilizada para retornar um valor de uma função.

```
int soma(int a, int b) {
    return a + b;
}
```

Conceitos Básicos de C++

2. Palavras-chave de Declaração de Dados

- `int`, `float`, `double`, `char`: Tipos de dados primitivos.

```
int x = 10;
float y = 5.5f;
double z = 3.14;
char c = 'A';
```

- `bool`: Tipo de dado booleano.

```
bool flag = true;
```

- `void`: Indica que uma função não retorna valor.

```
void funcao() {
    // código
}
```

- `const`: Indica que o valor de uma variável não pode ser alterado.

```
const int tamanho = 100;
```

- `static`: Mantém o valor de uma variável entre chamadas de função ou define a duração de armazenamento de variáveis em escopo de bloco ou de classe.

```
static int contador = 0;
```

3. Palavras-chave de Controle de Acesso

Conceitos Básicos de C++

- `public`, `private`, `protected`: Definem os níveis de acesso a membros de uma classe.

```
class Exemplo {
public:
    int publico;
private:
    int privado;
protected:
    int protegido;
};
```

4. Palavras-chave de Manipulação de Memória

- `new`, `delete`: Utilizadas para alocação e desalocação dinâmica de memória.

```
int* ptr = new int;
delete ptr;
```

- `new[]`, `delete[]`: Utilizadas para alocação e desalocação dinâmica de arrays.

```
int* arr = new int[10];
delete[] arr;
```

5. Palavras-chave de Manipulação de Classes e Objetos

- `class`, `struct`: Utilizadas para definir novos tipos de dados.

```
class Exemplo {
```

```
    int x;
};

struct ExemploStruct {
    int y;
};
```

- `this`: Ponteiro para o objeto atual.

```
class Exemplo {
    int x;
public:
    void setX(int x) {
        this->x = x;
    }
};
```

- `friend`: Permite que funções ou outras classes acessem membros privados ou protegidos.

```
class Exemplo {
    friend void funcaoAmiga(Exemplo& e);
private:
    int y;
};

void funcaoAmiga(Exemplo& e) {
    e.y = 10;
```

```
}
```

6. Palavras-chave de Manipulação de Exceções

- `try`, `catch`, `throw`: Utilizadas para tratamento de exceções.

```
try {
    // código que pode lançar uma exceção
} catch (const std::exception& e) {
    // tratamento da exceção
}

void funcao() {
    throw std::runtime_error("Erro");
}
```

7. Outras Palavras-chave

- `namespace`: Define um espaço de nomes para organizar o código e evitar conflitos de nomes.

```
namespace MeuEspaco {
    int x = 10;
}

int main() {
    int y = MeuEspaco::x;
}
```

Conceitos Básicos de C++

- `using`: Introduz um nome do namespace para o escopo atual.

```
using namespace std;
```

- `typedef`: Define um alias para um tipo de dado.

```
typedef unsigned long ulong;
```

- `constexpr`: Define expressões que podem ser avaliadas em tempo de compilação.

```
constexpr int quadrado(int x) {
    return x * x;
}
```

Dicas de Boas Práticas

- Consistência: Use as palavras-chave de forma consistente para melhorar a legibilidade do código.
- Comentário: Comente o uso de palavras-chave que podem não ser imediatamente claras para outros desenvolvedores.
- Modularidade: Utilize palavras-chave como `namespace` e `class` para organizar e modularizar seu código.

Esta seção abrange os conceitos sobre palavras-chave em C++. Para mais detalhes, consulte a documentação oficial: https://en.cppreference.com/w/cpp/keyword