Tipos em C++

Introdução

Em C++, os tipos de dados definem as características e o comportamento das variáveis. Eles determinam o tamanho e o layout da memória, o intervalo de valores que podem ser armazenados, e as operações que podem ser realizadas sobre eles.

1. Tipos Fundamentais

Tipos Inteiros

- Inteiros com Sinal: int, short, long, long long

Exemplo: int x = -42;

- Inteiros sem Sinal: unsigned int, unsigned short, unsigned long, unsigned long long

Exemplo: unsigned int y = 42;

Tipos de Ponto Flutuante

- Float: float

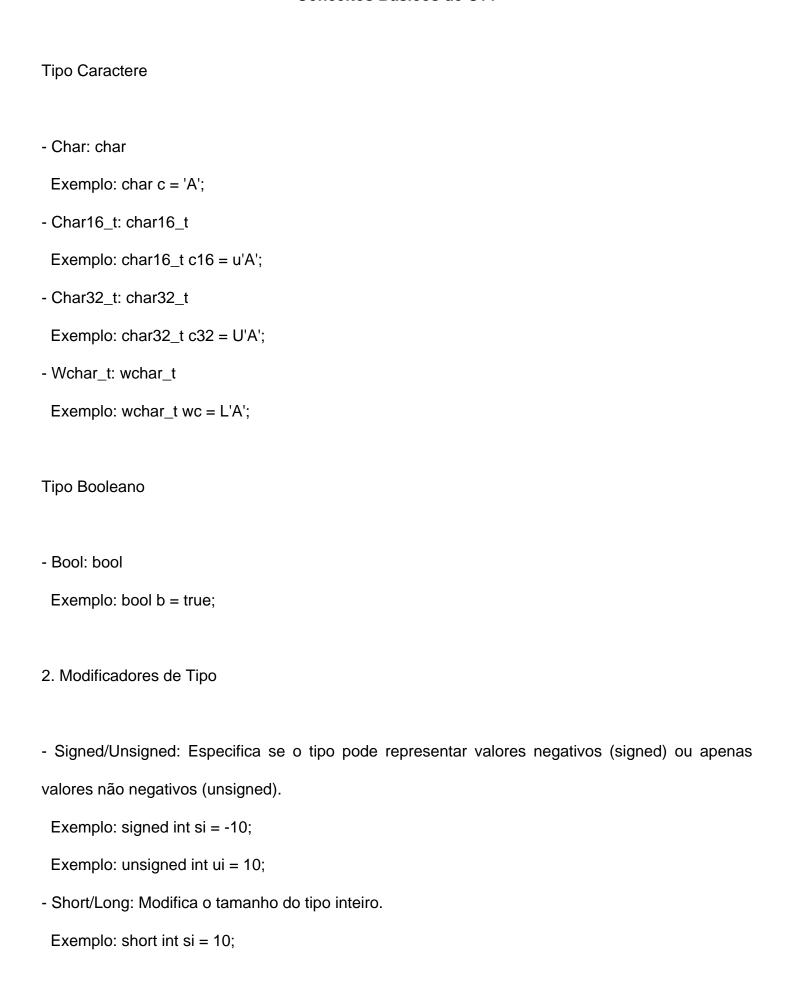
Exemplo: float f = 3.14f;

- Double: double

Exemplo: double d = 3.14;

- Long Double: long double

Exemplo: long double Id = 3.14L;



Exemplo: long int li = 1000000; 3. Tipos Derivados - Array: Coleção de elementos do mesmo tipo. Exemplo: int arr[10]; - Ponteiro: Armazena o endereço de uma variável. Exemplo: int *ptr = &x; - Referência: Referência a uma variável. Exemplo: int &ref = x; - Função: Conjunto de declarações que executa uma tarefa. Exemplo: void func(); - Enum: Conjunto de constantes inteiras nomeadas. Exemplo: enum Color { RED, GREEN, BLUE }; 4. Tipos Definidos pelo Usuário - Estruturas (struct): Conjunto de variáveis de diferentes tipos. Exemplo: struct Person { int age; float height; **}**;

- Uniões (union): Conjunto de variáveis de diferentes tipos que compartilham o mesmo espaço de

memória. Exemplo: union Data { int intValue; float floatValue; **}**; - Classes (class): Estruturas com métodos e propriedades, suporte a encapsulamento, herança e polimorfismo. Exemplo: class Car { public: int speed; void accelerate() { speed++; } **}**; 5. Tipos de Template - Permitem a definição de classes e funções parametrizadas por tipos. Exemplo: template<typename T> Tadd(Ta, Tb) { return a + b; }

Dicas de Boas Práticas

- Consistência: Use tipos consistentes para evitar erros de conversão e aumentar a legibilidade do código.
- Especificidade: Prefira tipos específicos como int32_t e uint64_t para garantir tamanhos consistentes em diferentes plataformas.
- Inicialização: Sempre inicialize variáveis para evitar comportamento indefinido.
- Uso de Auto: Use auto para deixar o compilador deduzir o tipo, reduzindo a verbosidade.
- Documentação: Documente os tipos definidos pelo usuário para facilitar o entendimento do código.

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