Tipos Fundamentais em C++

| 1 1 | . ~ |
|--------|------|
| Introd | いたつい |
| Introd | ucao |

Os tipos fundamentais em C++ formam a base para a manipulação de dados na linguagem. Eles incluem inteiros, ponto flutuante, caracteres, booleanos, e o tipo `void`.

1. Tipos Inteiros

Inteiros com Sinal

- `int`: Tipo inteiro padrão.

int a = 10;

- `short`: Tipo inteiro curto.

short b = 5;

- `long`: Tipo inteiro longo.

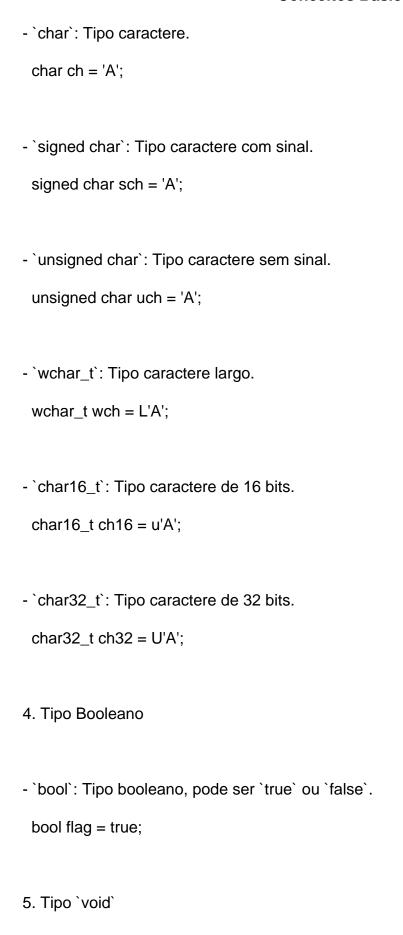
long c = 100000;

- `long long`: Tipo inteiro muito longo.

long long d = 10000000000LL;

Inteiros sem Sinal

| - `unsigned int`: Tipo inteiro sem sinal. |
|---|
| unsigned int e = 10U; |
| |
| - `unsigned short`: Tipo inteiro curto sem sinal. |
| unsigned short $f = 5U$; |
| |
| - `unsigned long`: Tipo inteiro longo sem sinal. |
| unsigned long $g = 100000UL$; |
| |
| - `unsigned long long`: Tipo inteiro muito longo sem sinal. |
| unsigned long long h = 100000000ULL; |
| |
| 2. Tipos de Ponto Flutuante |
| |
| - `float`: Tipo de ponto flutuante de precisão simples. |
| float $x = 3.14f$; |
| - `double`: Tipo de ponto flutuante de precisão dupla. |
| |
| double y = 3.141592653589793; |
| - `long double`: Tipo de ponto flutuante de precisão estendida. |
| |
| long double z = 3.141592653589793238L; |
| 3. Tipo Caractere |
| 5p5 Garagio.6 |



| - `void`: Representa a ausência de tipo. Utilizado principalmente para funções que não retornam |
|---|
| valor. |
| void funcao() { |
| // código |
| } |
| 6. Modificadores de Tipo |
| `signed` e `unsigned` |
| |
| - `signed`: Pode representar valores positivos e negativos. |
| signed int i = -10; |
| |
| - `unsigned`: Pode representar apenas valores não negativos. |
| unsigned int j = 10U; |
| |
| `short` e `long` |
| - `short`: Reduz o tamanho do tipo inteiro. |
| short s = 32767; |
| |
| - `long`: Aumenta o tamanho do tipo inteiro. |
| long I = 2147483647L; |
| |
| Dicas de Boas Práticas |

- Consistência: Use tipos consistentes para variáveis relacionadas para evitar erros de conversão.
- Especificidade: Utilize tipos específicos como `int32_t` e `uint64_t` para garantir tamanhos consistentes em diferentes plataformas.
- Legibilidade: Prefira tipos que tornam o código mais legível e claro para outros desenvolvedores.
- Inicialização: Sempre inicialize variáveis para evitar comportamento indefinido.

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