

Revisão: conceitos básicos C++

Processo de compilação

Os elementos envolvidos no processo de compilação são [1]:

- Compilador
- Link-editor

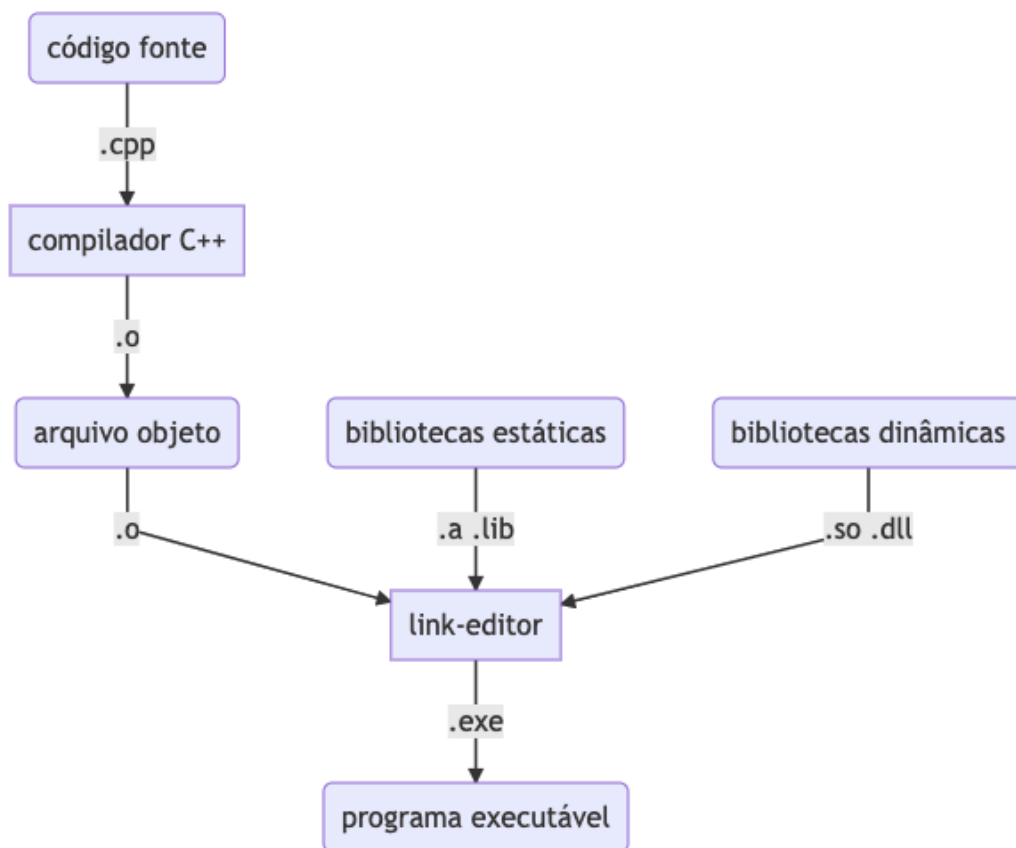


Figura 1: Processo de compilação de um programa C++.

Diretivas de pré-compilação

As diretivas de pré-compilação são executadas antes do processo de compilação. Por esse motivo, erros de sintaxe só serão validados posteriormente [1].

#include

Esta diretiva é usada para incluir cabeçalhos. Ela informa ao pré-processador para incluir o conteúdo dos arquivos no ponto do texto onde se encontra no arquivo fonte [\[1\]](#).

```
// Inclui um arquivo que está nos diretórios padrão
#include <iostream>

// Inclui um arquivo que está nos diretórios locais
#include "arquivo.h"
```

#define

Esta diretiva é utilizada para substituir valores no arquivo fonte [\[1\]](#).

```
#define PI 3.14
#define REAL double

REAL x = PI;
```

Também é possível definir macros que recebem parâmetros. A convenção para nomes de macros é utilizar todas letras maiúsculas separadas por *underscore*. Por exemplo:

```
#define MAX(x, y) ((x) > (y)) ? (x) : (y))

int x = MAX(10, 300);
```

Na definição das macros é importante envolver os valores entre parênteses para evitar problemas na ordem de precedência dos operadores [\[1\]](#). Por exemplo:

```
#define MULT(x, y) x * y

int x = MULT(3, 10); // 30
int y = MULT(1 + 2, 10); // 21!
int z = MULT(3, 5 + 5); // 20!
```

Outras diretivas

- `#undef`
- `#ifdef` e `#ifndef`
- `#if`
- `#else`
- `#elif`
- `#endif`

Apelidos

Em C++ é possível criar um apelido para um nome. Os apelidos são usados durante o período de compilação para se referir a nomes. Este mecanismo não cria um novo tipo, apenas um nome novo para um tipo [\[1\]](#).

```
typedef int Inteiro;
typedef char* Pchar;

Inteiro x = 5;
Pchar y = "y é um char*";
```

Tipos de dados

- Tipos primitivos
 - `bool`
 - `char`
 - `int`
 - `float`
 - `double`
- Modificadores de tipo: sinal
 - `signed`
 - `unsigned`

- Modificadores de tipo: tamanho
 - `short`
 - `long`

Comandos

- Comandos de decisão
 - `if`
 - `if / else`
 - `if / else if`
 - `switch / case`
- Comandos de repetição
 - `while`
 - `do / while`
 - `for`

Operadores

Operadores aritméticos

- `<Número> <operador> <Número> = <Número>`
 - `+`
 - `-`
 - `*`
 - `/`
 - `%`

Operadores relacionais

- `<Número> <operador> <Número> = <Booleano>`
 - `==`
 - `!=`
 - `>`
 - `>=`
 - `<`
 - `<=`

Operadores lógicos

- `<Booleano> <operador> <Booleano> = <Booleano>`
 - `||` ou `or`
 - `&&` ou `and`
 - `!` ou `not`

Atividade prática

1. Escreva um programa que leia três números a , b e c , e encontre as raízes x_1 e x_2 de uma equação do segundo grau no formato $ax^2 + bx + c = 0$. Para encontrar as raízes da equação utilize a fórmula de Bháskara.
 - Crie uma macro $DELTA(a, b, c)$ para calcular o valor do delta da equação.
 - Crie o apelido `Real` para o tipo `double`.
2. Escreva um programa que leia um número x e imprima o fatorial de todos os números no intervalo de 1 até x . Crie uma macro para realizar o cálculo do fatorial.

Atividade teórica

1. Qual é a finalidade das diretivas de pré-compilação?
2. Quais são as vantagens e desvantagens de apelidos?

3. Descreva a finalidade das seguintes diretivas:
- a. `#undef`
 - b. `#ifdef` e `#ifndef`
 - c. `#if`
 - d. `#else`
 - e. `#elif`
4. Qual a finalidade da diretiva `#pragma` ? Quando ela pode ser utilizada?
5. Pesquise outras diretivas de pré-compilação e explique as suas finalidades.
6. Quantos tipos de dados podemos compor em C++ combinando os tipos primitivos, modificadores de sinal e tamanho?

Referências bibliográficas

- [1] STROUSTRUP, Bjarne; LISBÔA, Maria Lúcia Blanck; LISBÔA, Carlos Arthur Lang (Trad.). **A linguagem de programação C++**. 3 ed. Porto Alegre, RS: Bookman, 2000.