Técnicas de Programação 1

2ª parte

Linguagem C

Prof. Jobson Massollar

jobson@uniriotec.br

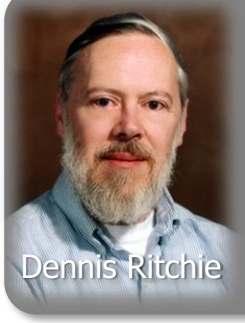
Existem dezenas de linguagens de programação.

Algumas são de propósito geral e outras são projetadas para situações específicas.

Linguagem C

Cria a linguagem BCPL em 1967

Cria a linguagem B em 1970

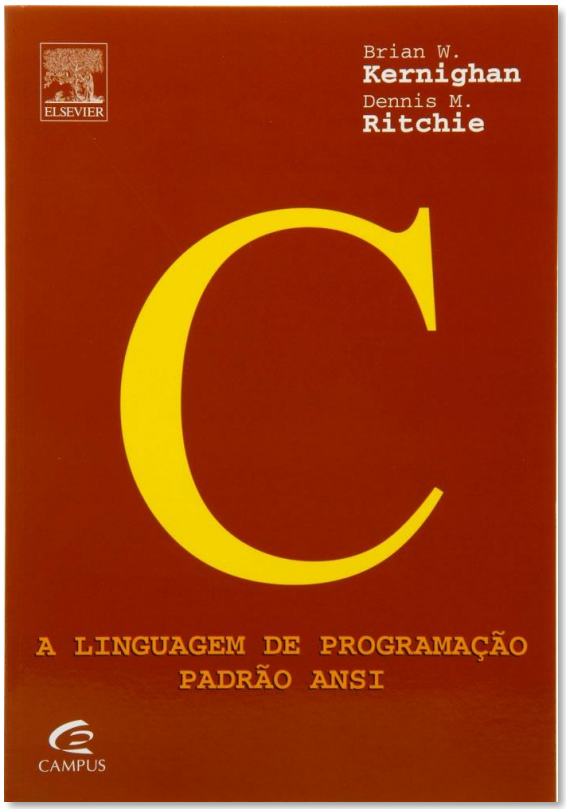


Cria a linguagem C em 1972

Linguagem C

Por muitos anos, o padrão para a linguagem C foi descrito no livro **The C Programming Language**, de Brian Kernighan e Dennis Ritchie de 1978.

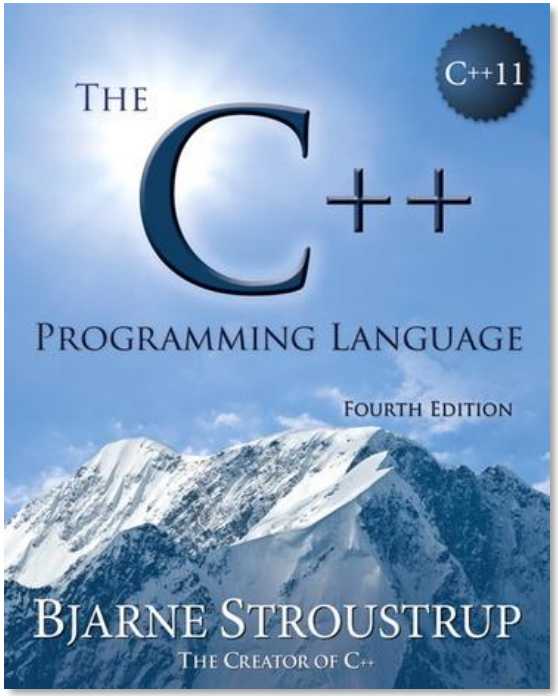
Em 1985, o ANSI (*American National Standards Institute*) estabeleceu um padrão oficial para o C, chamado **C ANSI**.



Linguagem C

Cria o C++ em 1985:

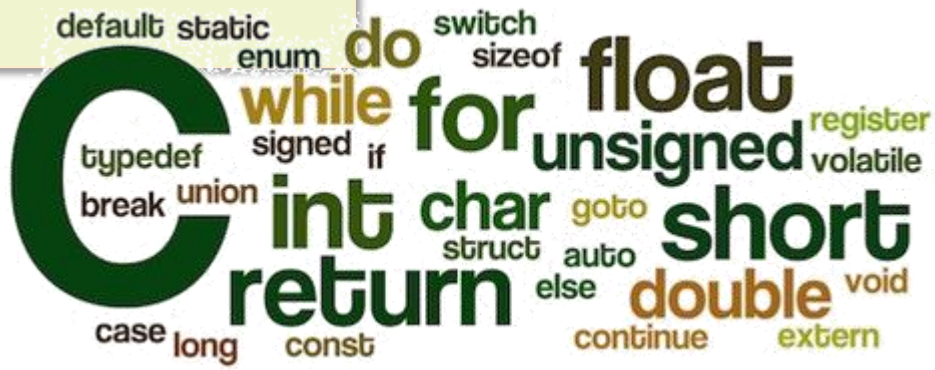
- ✓ Extensões à linguagem C
- ✓ Orientação a Objetos



Linguagem C

Principais características da linguagem C:

- ✓ Alto nível
- ✓ Estruturada
- ✓ Propósito geral
- ✓ Sintaxe simples
- ✓ Alto desempenho
- ✓ É uma das linguagens mais usadas no mundo*



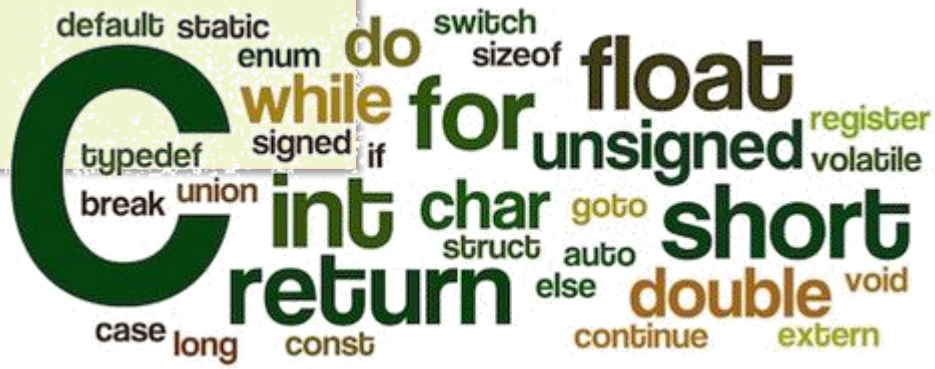
(*) https://www.tiobe.com/tiobe-index/

Linguagem C

C é usado, atualmente, para codificação de diversos tipos de softwares:

- ✓ Sistemas operacionais
- ✓ Ferramentas gráficas
- ✓ Ferramentas de programação
- ✓ Editores de texto
- ✓ Compiladores

dentre outros



A linguagem C possui poucos comandos, se comparada a outras linguagens.

Tudo em C é feito através de **funções**, como entrada e saída de dados, operações matemáticas, manipulação de data e hora, dentre outras.

Para tal, a linguagem C possui uma **biblioteca padrão**, onde já estão implementadas várias funções normalmente necessárias em um programa.

O objetivo da biblioteca de funções é oferecer aos programadores funções básicas de interesse geral.

Foto criada por freepik - br.freepik.com

Importante!

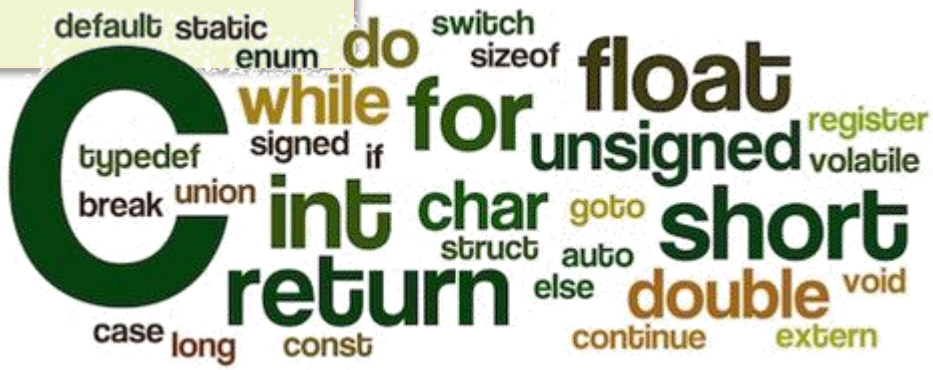
Nessa disciplina aprenderemos a linguagem C.

Entretanto, também vamos usar algumas extensões implementadas na linguagem C++ que facilitam a programação.

Estrutura Básica

Um programa em C é basicamente constituído por um conjunto de **diretivas e funções**.

Nesse conjunto de funções deve existir, obrigatoriamente, uma, e somente uma, função denominada **main**.



Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
     // Imprime uma mensagem para o usuário
     printf("Ola mundo !");
}
```

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Tudo o que vem entre os símbolos **/*** e ***/** é chamado de comentário.

Comentários são usados para:

- Documentar programas
- Ajudar as pessoas a lerem e entenderem o código

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Outra forma de criar comentários é usando o símbolo **//**.

Nesse caso, todo o conteúdo após o **//** até o fim da linha é um comentário.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Linhas iniciadas com o caracter **#** são chamadas de diretivas de compilação.

Diretivas não são comandos da linguagem.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

O **#include** é uma diretiva que diz ao compilador para incluir o arquivo indicado na compilação (nesse caso, o arquivo stdio.h).

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

O arquivo **stdio.h** (*standard input/output*) deve ser incluído em todos os programas que realizam entrada e saída de dados.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Os parênteses depois do **main** indicam que main é uma função.

A função **main** é obrigatória e define por onde a execução do programa será iniciada.

A palavra **int**, que aparece antes do **main**, é obrigatória e será explicada quando estudarmos funções.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Toda função em C/C++ tem seu corpo delimitado pelos caracteres **{** e **}**.

É no **corpo** da função que escrevemos as instruções do programa.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

A única instrução desse programa é uma chamada à função **printf**.

A função **printf** faz parte da biblioteca padrão do C/C++ e é responsável por imprimir dados na tela.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Todas as instruções em C/C++ devem finalizar com **;**

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

A linguagem C/C++ é ***case sensitive***, ou seja, palavras escritas com letras maiúsculas e minúsculas são consideradas diferentes.

As diretivas, comandos e funções do C/C++ devem ser escritos sempre em letras minúsculas.

Estrutura Básica

```
/* ------------------------------------------------
   Primeiro programa em C/C++
   Arquivo: primeiro.cpp
   ------------------------------------------------ */

#include <stdio.h>

int main()
{
    // Imprime uma mensagem para o usuário
    printf("Ola mundo !");
}
```

Note que as instruções da função main estão recuadas em relação ao **{** e **}**.

Esse recuo, também chamado de indentação, deve ser usado para melhorar a legibilidade do código.

- ✓ Use comentários para definir o objetivo do programa e para explicar trechos de código.
- ✓ Use linhas em branco para melhorar a legibilidade do programa.
- ✓ Use recuo (indentação) para melhorar a legibilidade do programa.
- ✓ Aplique o recuo de maneira uniforme.

Ambientes de Desenvolvimento

Para criar um programa em C/C++ podemos usar um editor de textos puro como o **Notepad** ou **Notepad++**.

Entretanto, por razões de facilidade e produtividade, os programadores adotam soluções mais sofisticadas chamadas de **Ambientes Integrados de Desenvolvimento** ou **IDE** (*Integrated Development Environment*).

Esses ambientes concentram, em um só lugar, todas as ferramentas necessárias para o desenvolvimento de programas.

Foto criada por senivpetro - br.freepik.com

Ambientes de Desenvolvimento

IDEs para C/C++

Ambientes Online

OnlineGDB

https://www.onlinegdb.com/

replit

https://replit.com

IDEs online C/C++

C++ Tutor

https://pythontutor.com/cpp.html#mode=display

Compilação & Linkedição

```c
#include <stdio.h>
#include <stdlib.h>

float funcaoIMC(float a, float p){ /*Definição da função*/

        float calculo = p/(a*a);
        return calculo;
}

int main(){ /*Função main onde os comandos são executados*/

    float peso, altura;

    printf("Digite o peso: ");
    scanf("%f",&peso);
    printf("Digite a altura: ");
    scanf("%f",&altura);

    float imc = funcaoIMC(altura, peso); /*Chamada à função*/

    printf("Valor do imc %0.2f \n", imc);

}
```

Computadores só trabalham com **códigos binários** (0s e 1s).

Computadores não conseguem executar diretamente o código escrito em C.

É preciso **traduzir** o código em C para código binário.

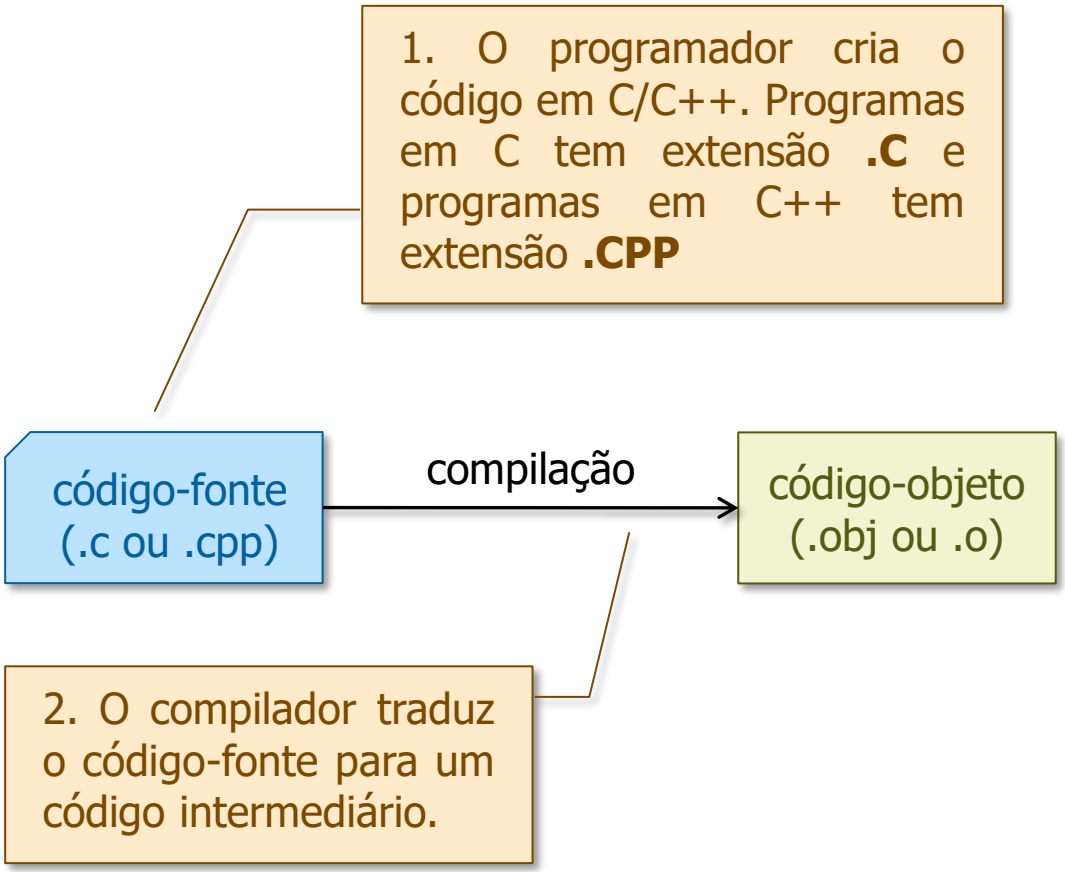
Esse processo de tradução é chamado de **compilação**.

Compilação & Linkedição

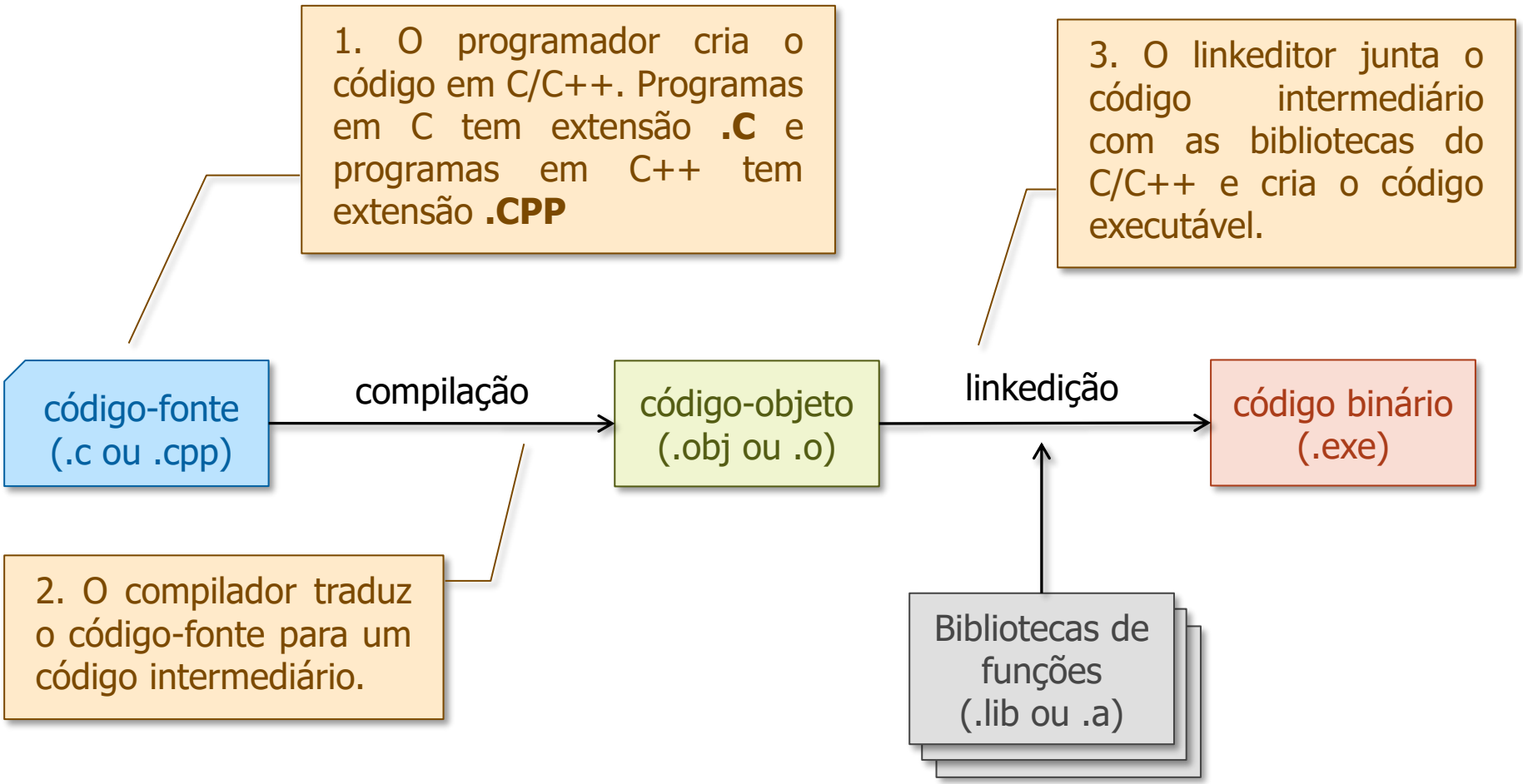
1. O programador cria o código em C/C++. Programas em C tem extensão **.C** e programas em C++ tem extensão **.CPP**

código-fonte
(.c ou .cpp)

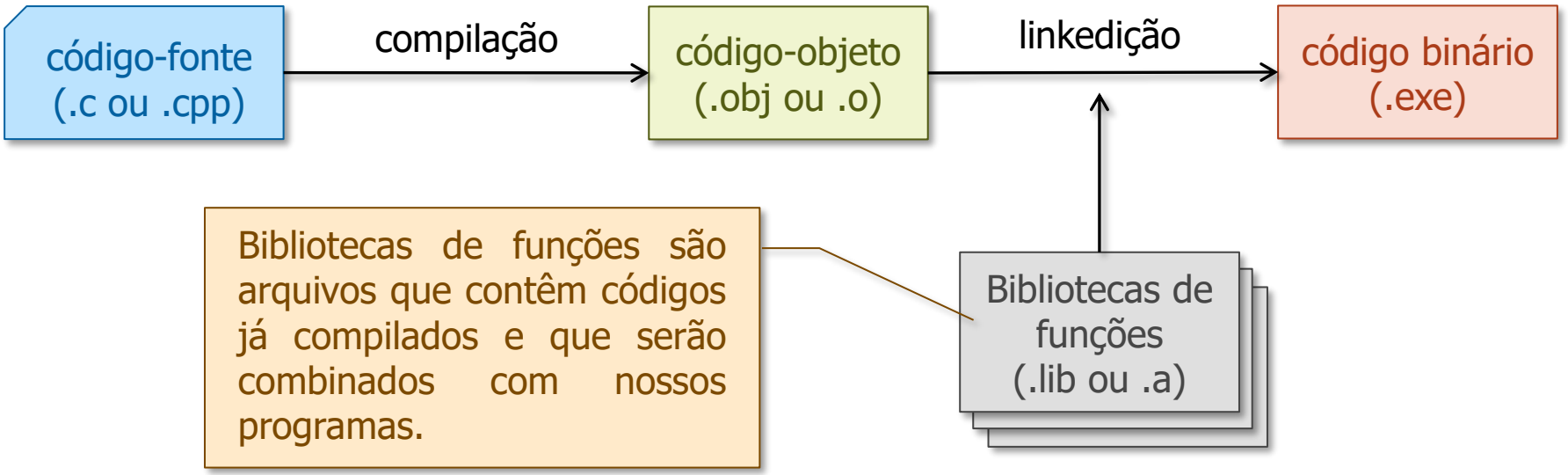
Compilação & Linkedição



Compilação & Linkedição



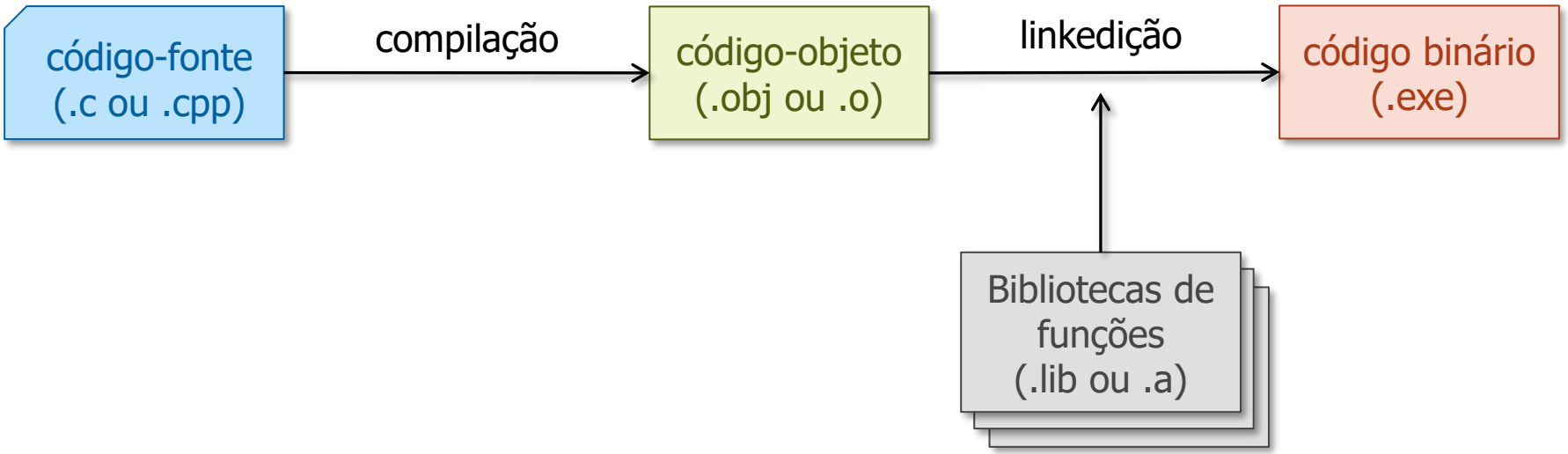
Compilação & Linkedição



Compilação & Linkedição

Importante!

Caso seja encontrado algum erro durante o processo de compilação ou linkedição, a mensagem de erro será apresentada e o código executável **não** será gerado.



Compilação & Linkedição

Erro de Compilação

Ocorre quando o compilador encontra um erro de sintaxe.

Exemplos:

- Instruções digitadas de forma errada
- Nome de variável errada ou não declarada
- Falta do ponto-e-vírgula no final da instrução
- Caracteres "estranhos"
- Chaves ou parênteses desbalanceados (abriu, mas não fechou ou fechou sem abrir)

Erro de Linkedição

Ocorre quando o linkeditor não encontra a função ou a chamada da função não corresponde ao que está definido na biblioteca.

Exemplos:

- Nome de função errada
- Parâmetros errados

Erro de Execução

Ocorre quando uma situação inesperada acontece durante a execução do programa.

Exemplos:

- Divisão por zero
- Raiz quadrada de número negativo
- Falta de espaço em memória
- Falta de espaço em disco
- Perda de conexão de rede

Compilação & Linkedição

```
1  /* ----------------------------------------------
2     Primeiro programa em C
3     Arquivo: primeiro.cpp
4     ---------------------------------------------- */
5
6  #include <stdio.h>
7
8  int main()
9  {
10    // Imprime uma mensagem para o usuário
11    printf("Ola mundo !")
12 }
```

Line 12: expected ';' before '}' token

Essa mensagem é estranha...não consigo entender o que está errado na linha 12.

Foto criada por asierromero - br.freepik.com

Compilação & Linkedição

Importante!

Assim como a língua portuguesa, as linguagens de programação também tem regras de sintaxe e semântica.

Erros de compilação e linkedição indicam, em quase 100% dos casos, que **você digitou algo errado no seu programa**, ou seja, algo que não está de acordo com as regras sintáticas ou semânticas da linguagem.

Leia a mensagem de erro e confira seu código com calma e atenção.

No início as mensagens podem parecer "estranhas", mas com a prática você vai ser capaz de interpretá-las e corrigir o erro cada vez mais rapidamente.