



Cheats sheet de Linguagem de Programação em C

Unidade 3 | Capítulo 1



Executores:



Coordenação:



Iniciativa:



Bem-vindo ao Cheat Sheet de Linguagem de Programação em C, uma ferramenta prática e essencial para desenvolvedores que desejam acessar rapidamente os principais conceitos, estruturas e sintaxe dessa linguagem poderosa e amplamente utilizada.

Neste guia, você encontrará resumos objetivos sobre tipos de dados, operadores, estruturas de controle, manipulação de memória e funções, tudo organizado de forma clara e acessível para que você possa consultar sempre que necessário.

Este material foi pensado para facilitar o aprendizado e o desenvolvimento rápido, sendo um recurso para programadores que desejam uma referência rápida.

Guia de Referência em Linguagem de Programação C

Estrutura básica de Programação em C :

```
1 /* Estrutura básica de uma aplicação */
2
3 #include <stdio.h> 1 // standard input-output (biblioteca padrão de entrada/saída)
4 #include <stdlib.h> // standard library (biblioteca de propósito geral)
5
6 int valor; 2 // declaração de variável global
7
8 void funcao_teste() { // declaração de função
9     printf("teste");
10 }
11
12 int main() | 4
13 {
14     printf("Hello World \n");
15     funcao_teste();
16 }
```

A imagem mostra um código de programação na linguagem C. Ele começa com dois comentários que explicam a inclusão de bibliotecas padrão, sendo a primeira, stdio.h, para entrada e saída de dados, e a segunda, stdlib.h, para funções de propósito geral. Depois, é declarada uma variável global chamada valor. O código define uma função chamada funcao_teste(), que imprime a palavra "teste". A seguir, na função principal main(), o código exibe a mensagem "Hello World" usando o comando printf, e em seguida, chama a função funcao_teste() para exibir "teste" na tela. O código ilustra a estrutura básica de uma aplicação simples em C.

Operadores:

A tabela abaixo apresenta os principais operadores utilizados na linguagem de programação C, organizados por suas funções e tipos.

Operadores são símbolos que instruem o compilador a realizar operações matemáticas, lógicas, de atribuição ou de manipulação de ponteiros, entre outras. Cada tipo de operador possui um papel específico no processamento de dados dentro de um programa, facilitando a realização de cálculos, comparações e manipulação de endereços de memória.

Tipo	Operador	Explicação	Exemplo
Aritméticos	+	Adição	valor = 3 + 5 ; // 8
	-	subtração	valor = 5 - 5 ; // 0
	*	multiplicação	valor = 3 *5 ; // 15
	/	divisão	valor = 30/5 ; // 6
	%	módulo	valor = 30 % 5; // 0
Atribuição	=	Atribuição simples	valor = 30;
Lógicos	&&	And	(valor>18) && (valor<23)
		or	(valor>18) (valor<23)
	!	negação	(valor != 15)
Relacionais	==	Igual relacional	(valor == 15)
	!=	diferente	(valor != 15)
	<	Menor que	(valor < 15)
	>	Maior que	(valor > 15)
	>=	Maior ou igual a	(valor >= 15)
	<=	Menor ou igual a	(valor <= 15)
Incremento e decremento	++	incremento	valor++
	-	decremento	valor-
Apontadores	&	Retorna o endereço de	int valor ; int *p; // ponteiro p = &valor; // atribui o endereço de valor
	*	Retorna o conteúdo de	*p = 5; /* atribui ao conteúdo apontado por p o valor 5 e p aponta para o endereço de valor, então valor recebe 5 */

Tabela 2 – Operadores em C

Entrada e saída

A tabela a seguir apresenta exemplos de funções de entrada e saída na linguagem de programação C. Essas funções são essenciais para interagir com o usuário,

Formatos de especificadores e caracteres especiais

A tabela abaixo apresenta os formatos de especificadores usados nas funções de entrada e saída, como printf e scanf, na linguagem C. Esses especificadores permitem que você indique ao programa qual tipo de dado está sendo exibido ou recebido, como números inteiros, números de ponto flutuante, caracteres e strings (sequências de caracteres)[1,2].

A tabela também inclui alguns caracteres especiais que podem ser usados para formatar a saída, como a criação de novas linhas, tabulações e a inserção de aspas e barras invertidas no texto. Esses especificadores são fundamentais para o controle e formatação da exibição e coleta de dados em um programa.

Formatos	
%d	Inteiro decimal
%f	Ponto flutuante
%c	Caractere
%s	String
%x	Inteiro hexadecimal

Formatos	
\n	Nova linha
\t	tabulação
\\"	Barra invertida
\"	aspas

Tabela 3 – funções de entrada e saída na linguagem de programação C

Estrutura de controle e repetição

A tabela a seguir apresenta as principais estruturas de controle de fluxo e repetição utilizadas na linguagem de programação C[1]. Essas estruturas são fundamentais para determinar a ordem em que as instruções do programa são executadas e para controlar repetições de blocos de código.

Comando	Definição	Sintaxe
Declaração de variável	Declaração de variável	<pre>tipo_de_dado nome_da_variável; // Exemplos: int idade; float altura; char inicial;</pre>
Declaração de constante	Declaração de constante	<pre>#define NOME_CONSTANTE valor // Exemplo: def float PI = 3.14159;</pre>
if	Comando condicional	<pre>f(condição){ // Código a ser executado se a condição for verdadeira } else { // Código a ser executado se a condição for falsa } // Exemplo: if(a > b){ printf("%s", "A é maior que B"); } else { printf("%s", "A é igual ou menor que B"); }</pre>
switch	Comando condicional	<pre>switch(expressão){ case valor1: // Código para o caso valor1 break; case valor2: // Código para o caso valor2 break; default: // Código para os demais casos } // Exemplo: switch(i){</pre>
while	Estrutura de controle de fluxo com pré-validação	<pre>while(condição){ // Código a ser repetido enquanto a condição for verdadeira } // Exemplo : int i = 1; while(i <= 10){ printf("%d", i++); }</pre>
do	Estrutura de controle de fluxo com pré-validação	<pre>do { // Código a ser executado pelo menos uma vez } while (condição); // Exemplo : int i = 1; do { printf("%d", i++); } while (i <= 10);</pre>

Comando	Definição	Sintaxe
for	Estrutura de controle de fluxo simples	<pre>for(inicialização; condição; incremento){ // Código a ser repetido } // Exemplo : for(i=1;i<=10;i++){ printf("\n%d", i); }</pre>
break	interrompe a execução do laço mais interno	break;
continue	Pula para a próxima iteração do laço	continue;
Sub-rotinas	Funções	<pre>tipo_de_retorno nome_da_função(lista_de_parâmetros){ // Corpo da função } // Exemplo: float area(float altura, float base){ return altura * base; }</pre>
	Procedimentos	<pre>void area(float altura, float base){ printf("A área é: %f", altura * base); }</pre>
Vetores	Variáveis unidimensionais	<pre>tipo_de_dado nome_do_vetor[tamanho]; // Exemplo: int numeros[5];</pre>
Matrizes	Variáveis multidimensionais	<pre>tipo_de_dado nome_da_matriz[linhas][colunas]; // Exemplo: float matriz[3][3];</pre>
struct	Tipos de dados compostos	<pre>struct nome_da_estrutura { tipo_de_dado membro1; tipo_de_dado membro2; // ... }; // Exemplo: struct Pessoa { char nome[50]; int idade; };</pre>
typedef	Definição de novos tipos de dados	<pre>typedef tipo_de_dado novo_nome; // Exemplo: typedef int inteiro;</pre>

Comando	Definição	Sintaxe
enum	Conjunto de constantes nomeadas	enum dias_da_semana { DOMINGO, SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO };

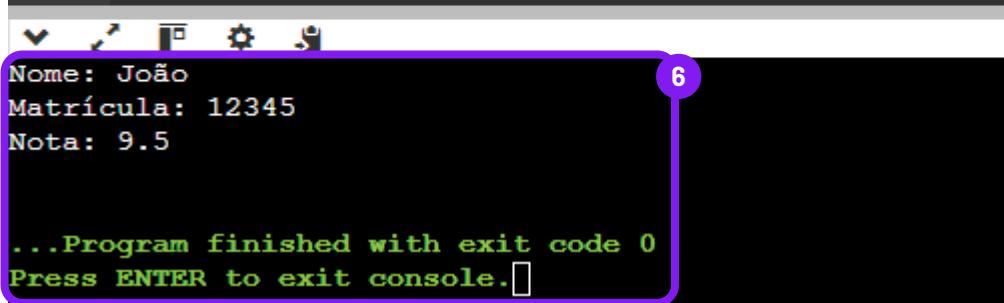
Tabela 6 - Estrutura de controle de fluxo e repetição

Exemplo básico:

```

9 #include <stdio.h> 1
10
11 struct Aluno { 2
12     char nome[50];
13     int matricula;
14     float nota;
15 };
16
17 int main() { 3
18     struct Aluno aluno1 = {"João", 12345, 9.5};
19
20     printf("Nome: %s\n", aluno1.nome); 4
21     printf("Matrícula: %d\n", aluno1.matricula);
22     printf("Nota: %.1f\n", aluno1.nota);
23
24     return 0; 5
25 }
26

```



```

Nome: João
Matrícula: 12345
Nota: 9.5

...Program finished with exit code 0
Press ENTER to exit console. 6

```

Figura 1 – Exemplo de Utilização do Código em C

Resumo da Estrutura Básica:

1. (Linha 9) Incluir

Biblioteca Padrão: O código começa com a inclusão da biblioteca stdio.h, que é essencial para a função printf (que serve para exibir texto na tela).

2. (Linhas 11 a 15) Definir a Estrutura: Aqui é definida uma struct chamada Aluno. Uma struct é uma forma de agrupar diferentes tipos de dados em uma única unidade. No caso, a estrutura Aluno possui três campos: nome: Um array de caracteres (string) com espaço para até 50 caracteres. matrícula: Um número inteiro (int), representando o número de matrícula do aluno. nota: Um número do tipo float, representando a nota do aluno (números decimais).

3. (Linhas 17 a 18) Na função principal main(), o código cria uma variável chamada aluno1 do tipo struct Aluno e inicializa seus três campos: nome é "João". matricula é 12345. nota é 9.5.

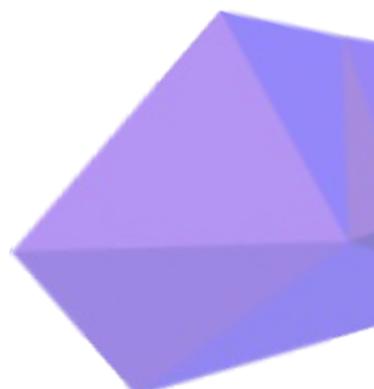
4. (Linhas 20 a 22) Nessa parte, o código usa a função printf para imprimir as informações do aluno: Nome: %s é o especificador de formato para strings. Ele vai imprimir o valor contido em aluno1.nome (que é "João"). Matrícula: %d é o especificador de formato para inteiros. Ele vai imprimir o valor de aluno1.matricula (que é 12345). Nota: %.1f é o especificador de formato para números decimais (float), com uma casa decimal. Ele vai imprimir o valor de aluno1.nota (que é 9.5).

5. (Linha 24) A função main retorna 0 para indicar que o programa foi executado com sucesso.

6. Na parte inferior da imagem, você pode ver a saída do programa no terminal. Isso confirma que o programa coletou os dados de aluno1 e os exibiu corretamente na tela.

Referências e Saiba Mais

- ASCENCIO, Ana Fernanda Gomes; DE CAMPOS, Edilene Aparecida Veneruchi. Fundamentos da programação de computadores. Pearson Educación, 2008.
- DE ALMEIDA, Rodrigo Maximiano Antunes; DE MORAES, Carlos Henrique Valério; SERAPHIM, Thatyana de Faria Piola. Programação de sistemas embarcados: Desenvolvendo software para microcontroladores em linguagem C. Elsevier Brasil, 2017.
- GOMES, Ruan Victor; PINHEIRO, Bruno Lima. Git e GitHub: Desenvolvendo Habilidades Essenciais para Colaboração e Controle de Versões. Disponível em <<https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/132/576/888-1>> Acesso em: 25 jul. 2024.
- TSITOARA, Mariot. Beginning Git and GitHub. Springer, New York, 2020.



Tipo		
Saída	printf(formato , argumentos)	<pre>#include <stdio.h> int main(){ int idade = 30; float altura = 1.75; char nome[] = "João"; printf("Olá, meu nome é %s e tenho %d anos. \n", nome, idade); printf("Minha altura é %.2f metros. \n", altura); return 0; }</pre>
Entrada	Scanf(formato, lista de endereços)	<pre>#include <stdio.h> #include <string.h> int main(){ char nome[50]; int idade; float altura;</pre>

Obrigado

Resumo da Estrutura Básica:

- 1. #include:** Carrega bibliotecas para usar ferramentas no programa.
- 2. Variáveis:** São "caixinhas" onde guardamos dados (números, textos, etc.).
- 3. Funções:** São "blocos de tarefas" que o programa pode usar várias vezes.
- 4. main:** É o coração do programa, onde ele começa a funcionar.

Tipos Primitivos:

Os tipos primitivos em C são a base para armazenar diferentes tipos de dados no seu programa. Eles incluem números inteiros, números com casas decimais e caracteres. Saber quando usar cada tipo e como modificar seus tamanhos e sinais é uma habilidade importante ao programar em C.

Tipos Primitivos			
Palavra-chave	Tipo	Tamanho (bytes)	Valores válidos
char	Caractere	1	-128 a 127
signed char	Caractere com sinal	1	-128 a 127
unsigned char	Caractere sem sinal	1	0 a 255
int	inteiro	1	-32.768 a 32.767
signed int	Inteiro com sinal	2	-32.768 a 32.767
unsigned int	Inteiro sem sinal	2	0 a 65.535
short int	Inteiro curto	2	-32.768 a 32.767
signed short int	Inteiro curto com sinal	2	-32.768 a 32.767
unsigned short int	Inteiro curto sem sinal	2	0 a 65.535
long int	Inteiro longo	4	-2.147.483.648 a 2.147.483.647
signed long int	Inteiro longo com sinal	4	-2.147.483.648 a 2.147.483.647
unsigned long int	Inteiro longo sem sinal	4	0 a 4.294.967.295
float	Ponto flutuante com precisão simples	4	10^{-38} a 10^{38}
double	Ponto flutuante com precisão dupla	8	10^{-308} a 10^{308}
long double	Ponto flutuante com precisão dupla longo	16	10^{-4932} a 10^{4932}

Tabela 1 – Tipos Primitivos em C

permitindo que o programa receba dados de entrada (como informações digitadas) e exiba informações na tela.

No C, as funções mais comumente usadas para esse propósito são printf (para exibir saídas) e scanf (para ler entradas).

A tabela fornece a sintaxe básica de cada função, juntamente com exemplos de código prático que mostram como utilizar printf para exibir variáveis formatadas e scanf para coletar dados de diferentes tipos, como inteiros, strings e números com ponto flutuante. Essas funções são essenciais para criar programas que precisam de interações dinâmicas com o usuário.

Tipo		
Saída	printf(formato , argumentos)	<pre>#include <stdio.h> int main(){ int idade = 30; float altura = 1.75; char nome[] = "João"; printf("Olá, meu nome é %s e tenho %d anos. \n", nome, idade); printf("Minha altura é %.2f metros.\n", altura); return 0; }</pre>
Entrada	Scanf(formato, lista de endereços)	<pre>#include <stdio.h> #include <string.h> int main(){ char nome[50]; int idade; float altura; printf("Digite seu nome: "); scanf("%s", nome); printf("Digite sua idade: "); scanf("%d", &idade); printf("Digite sua altura: "); scanf("%f", &altura); printf("\nSeus dados:\n"); printf("Nome: %-20s Idade: %d\n", nome, idade); printf("Altura: %.2f metros\n", altura); return 0; }</pre>

Tabela 3 – funções de entrada e saída na linguagem de programação C

Linguagens e Ambientes de Programação

Unidade 3 | Capítulo 1



Executores:



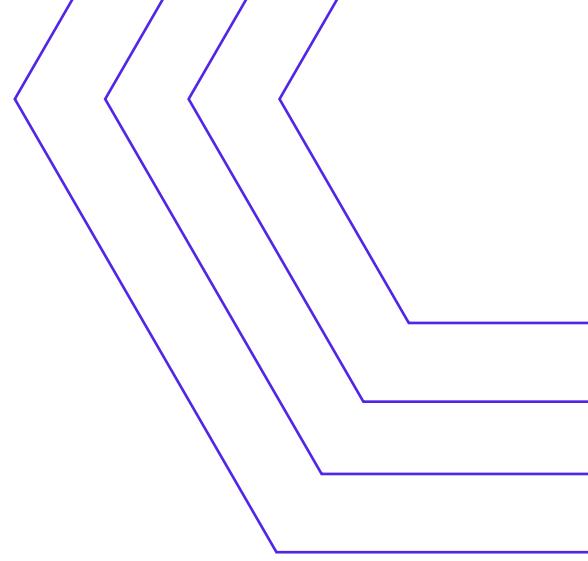
Coordenação:



Iniciativa:

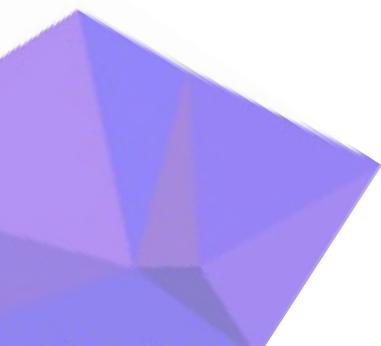


Mapa Mental da Estrutura de programação em Linguagem C



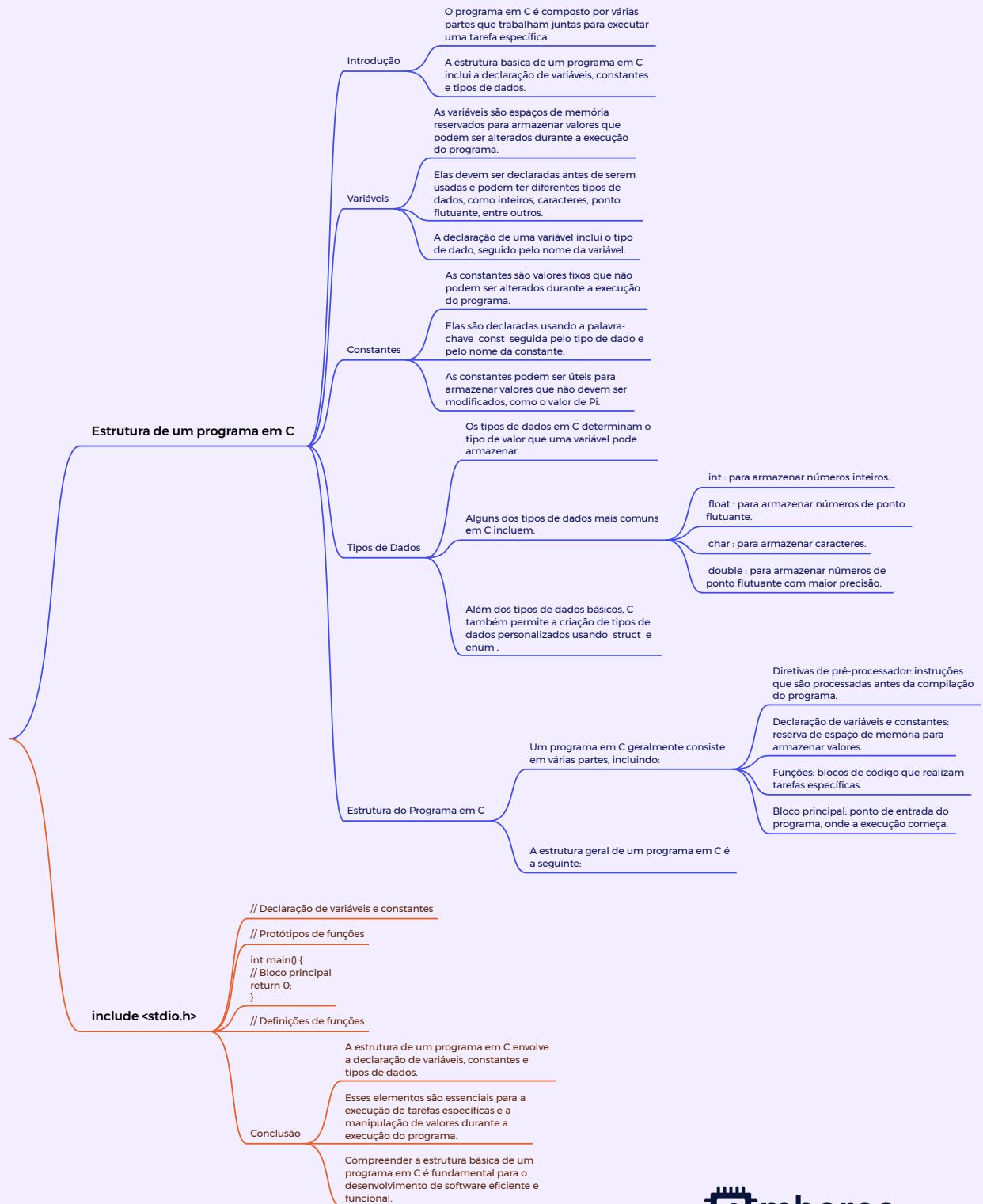
Olá, hoje vamos explorar a estrutura da linguagem C de uma forma visual e interativa. Através deste mapa mental, teremos uma visão geral dos principais elementos que compõem um programa em C[1,2].

Visualizaremos a organização hierárquica da linguagem, desde os componentes mais básicos, como variáveis e tipos de dados, até os elementos mais complexos, como estruturas de controle e funções. Essa representação gráfica nos permitirá compreender de forma clara e concisa as relações entre os diferentes elementos da linguagem, facilitando assim o aprendizado e a memorização dos conceitos.



Estrutura de um programa em C

Estrutura de um programa em C



• <https://mapify.so/pt/share-link/yzw5x9ENhy>



Referências

[1] ASCENCIO, Ana Fernanda Gomes; DE CAMPOS, Edilene Aparecida Veneruchi. Fundamentos da programação de computadores. Pearson Educación, 2008.

[2] DE ALMEIDA, Rodrigo Maximiano Antunes; DE MORAES, Carlos Henrique Valério; SERAPHIM, Thatyana de Faria Piola. Programação de sistemas embarcados: Desenvolvendo software para microcontroladores em linguagem C. Elsevier Brasil, 2017.

Obrigado

