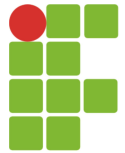
Conceitos Iniciais da Linguagem C

Algoritmos e Programação de Computadores – ABI/LFI/TAI

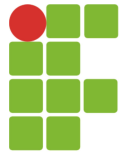
Prof. Daniel Saad Nogueira Nunes

IFB – Instituto Federal de Brasília,
Campus Taguatinga



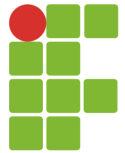
Sumário

1. Introdução
2. Variáveis
3. Tipos
4. Atribuição
5. Estrutura



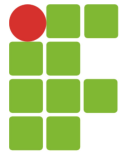
Sumário

1 Introdução



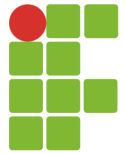
Introdução

- Nesta aula verificaremos alguns conceitos iniciais da programação na linguagem `C`.
- Através destes conceitos iniciais, poderemos dar início à implementação dos nossos primeiros algoritmos nesta linguagem de programação.
- Examinaremos:
  - Variáveis.
  - Tipos primitivos.
  - Atribuição.
  - Constantes.
  - Estrutura básica de um programa em `C`.



Sumário

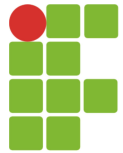
2 Variáveis



Variáveis

Variável

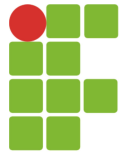
- Uma variável é um nome para uma posição de memória que armazena um valor.
- Através dos nomes das variáveis, podemos realizar operações entre valores.
- Cada variável tem um **tipo**, que determina que tipo de informação está armazenada naquele pedaço de memória indicado pelo nome da variável.
- O ato de **declarar** uma variável consiste em dizer qual o tipo e o nome referentes a ela.



Variáveis

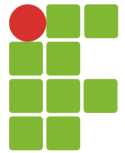
Declaração de uma Variável

```
<tipo> nome_da_variável;
```



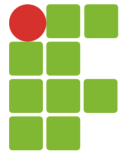
Sumário

2 Variáveis
- Regras para nomeação de variáveis
- Case sensitivity
- Múltiplas Declarações



Regras para nomeação de variáveis

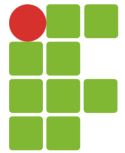
- Existem algumas regras para nomeação das variáveis.
- Caso uma variável seja nomeada com um nome que não atenda essas regras, o programa não poderá ser compilado.
- Devemos obedecer a **sintaxe** correta da linguagem.



Regras para nomeação de variáveis

Regras para nomeação de variáveis

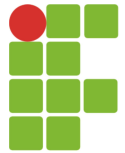
1. Deve começar com uma letra ou subscrito (_).
2. Pode ser composta de letras maiúsculas e minúsculas (sem acento), números e subscrito.
3. Não se pode utilizar caracteres especiais como: `{(+-*/\;.,?)}`
4. Não se pode utilizar um nome já declarado antes. Exceto se em escopos diferentes (veremos isso mais tarde).
5. Não se pode utilizar palavras reservadas da linguagem `C`.



Regras para nomeação de variáveis

Uso Incorreto

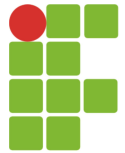
```
int 42b; // não começa com letra ou subscrito
float @variavel; // contém caractere especial
char isso#naopode; // contém caractere especial
```



Regras para nomeação de variáveis

Uso Incorreto

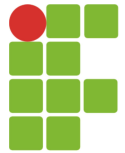
```
int a; // ok
float a; // não está ok, já existe uma variável chamada 'a'
int b; // ok
int b; // não está ok, já existe uma variável chamada b
```



Variáveis

Exemplo: Declaração de Variáveis

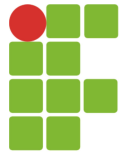
```
int numero;
float euler;
double numero_real;
char letra;
```



Nomeação de Variáveis

Boa Prática de Programação: Nomeação de Variáveis

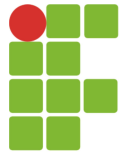
Sempre utilize nomes **mnemônicos** para suas variáveis. Isso deixará o seu programa mais legível por você e por outras pessoas. Evite utilizar nomes que não reflitam o uso daquela variável.



Nomeação de Variáveis

Exemplo: Variáveis Mnemônicas

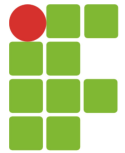
```
double preco_abacate;
int idade_usuario;
char opcao_escolhida;
```



Nomeação de Variáveis

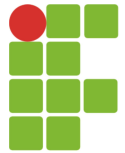
Exemplo: Variáveis não Mnemônicas

```
double fnx;
int seila;
char nanananabatman;
```



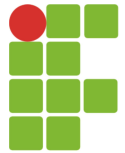
Palavras Reservadas

- Existem diversas palavras reservadas em C que não podem ser empregadas no nome de variáveis.



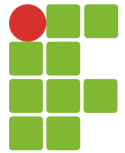
Palavras Reservadas

```
auto        else    long        switch
break       enum    register    typedef
case        extern  return      union
char        float   short       unsigned
const       for     signed      void
continue    goto    sizeof      volatile
default     if      static      while
do          int     struct      double
_Bool  _Complex  _Imaginary inline restrict // C99
```



Sumário

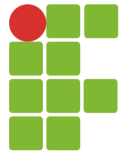
2 Variáveis
- Regras para nomeação de variáveis
- Case sensitivity
- Múltiplas Declarações



Variáveis

Case Sensitivity

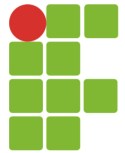
- A nomeação de variáveis e outros identificadores na linguagem C é sensível ao caso (case-sensitive), isto é, se tivermos mesma sequência de símbolos, mas com diferentes capitalizações, tratamos como variáveis (ou identificadores) distintos.
- Exemplos:
  - a $\neq$ A
  - numero $\neq$ Numero



Variáveis

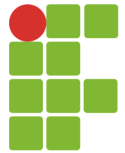
Exemplo: Declaração de Variáveis

```
int a; // ok
int A; // ok
float numero_real; // ok
double Numero_real; //ok
```



Sumário

2 Variáveis
- Regras para nomeação de variáveis
- Case sensitivity
- Múltiplas Declarações



Declaração

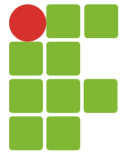
Múltiplas Declarações

Caso queiramos declarar diversas variáveis de um mesmo tipo, podemos fazê-lo em uma única linha.

Basta separar as variáveis por vírgulas.

Sintaxe:

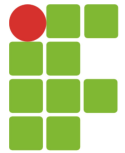
```
<tipo> <nome_1>, <nome_2>, ... , <nome_n>;
```



Atribuição

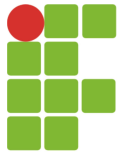
Exemplo: Múltiplas Declarações

```
int num_1,num_2,num_3;
```



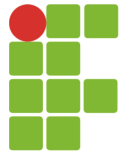
Sumário

3 Tipos



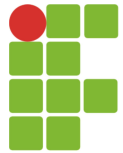
Tipos

- O tipo de uma variável determina o que pode ser armazenado por ela.
- Em C, de maneira primitiva, podemos armazenar números inteiros, números reais (ponto flutuante) e caracteres.
- O número de bits, e consequentemente a quantidade de valores que é possível armazenar em uma variável de um dado tipo, é dependente da arquitetura de computador utilizada, contudo, a linguagem C estipula o número mínimo de bits que deve ser suportado por qualquer compilador nas diferentes arquiteturas.
- Alguns tipos podem ser precedidos de modificadores como `unsigned`, `short` ou `long`, que indicam se o número tem sinal ou não e o tamanho do número. Isso interfere diretamente no intervalo de valores representáveis pela variável.



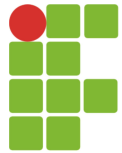
Sumário

3 Tipos
- Inteiros
- Reais
- Caracteres



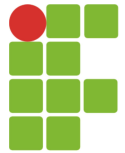
Inteiros

- Tipo: `int`.
- Normalmente as arquiteturas modernas utilizam representação binária em complemento de dois para números inteiros com sinal.
- Admitem os modificadores `short`, `long` e `unsigned`.



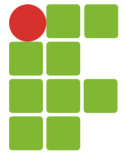
Inteiros

Tipo	Tamanho Mínimo	Tamanho Típico	Intervalo de Representação (Típico)
`short`	2 bytes	2 bytes	-32768 a 32767
`unsigned short`	2 bytes	2 bytes	0 a 65,535
`int`	2 bytes	4 bytes	-2147483648 a 2147483647
`unsigned int`	2 bytes	4 bytes	0 a 4294967295
`long int`	4 bytes	8 bytes	-9223372036854775808 a 9223372036854775807
`unsigned long int`	4 bytes	8 bytes	0 a 18446744073709551615
`long long int`	8 bytes	8 bytes	-9223372036854775808 a 9223372036854775807
`unsigned long long int`	8 bytes	8 bytes	0 a 18446744073709551615



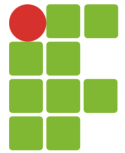
Sumário

3 Tipos
- Inteiros
- Reais
- Caracteres



Reais

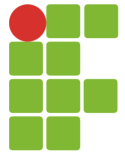
- Tipo: `float` ou `double`.
- Normalmente as arquiteturas modernas utilizam o padrão IEEE 754 para representação de números reais através de ponto flutuante.
- O tipo `double` admite o modificador `long`.



Reais

float

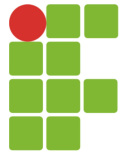
- 32-bits;
- A grosso modo, possui uma precisão de 6 casas decimais.
- Intervalo de representação está contido em : $[10^{-38}, 10^{38}]$.



Reais

double

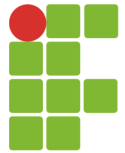
- 64-bits;
- A grosso modo, possui uma precisão de 15 casas decimais.
- Intervalo de representação está contido em : $[10^{-308}, 10^{308}]$.



Reais

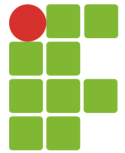
long double

- 80-bits (tipicamente), 96-bits ou 128-bits;
- Intervalo de representação está contido em : $[10^{-4951}, 10^{4932}]$.



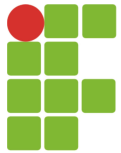
Sumário

3 Tipos
- Inteiros
- Reais
- Caracteres



Caracteres

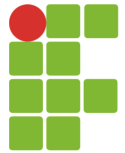
- Caracteres em C são representados internamente da mesma forma que inteiros, mas utilizando apenas 1 byte (8 bits).
- Um valor de um caractere é um inteiro que corresponde ao índice de uma letra na tabela ASCII.
- Apenas um caractere pode ser armazenado em uma variável do tipo `char`.
- O tipo `char` admite o modificador `unsigned`.
- Para representar uma sequência de caracteres (uma palavra), temos que utilizar uma *string* (veremos mais tarde).



Caracteres

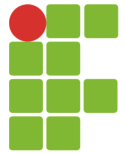
ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



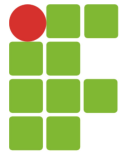
Caracteres

Tipo	Tamanho	**Intervalo de Representação**
`char`	1 byte	-128 a 127
`unsigned char`	1 byte	0 a 255



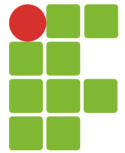
Sumário

4 Atribuição



Atribuição

- Até o presente momento verificamos os tipos primitivos da linguagem C e como declarar variáveis utilizando estes tipos.
- Contudo, ainda não aprendemos a designar valores as nossas variáveis.
- Para isso, utilizaremos o comando de atribuição (=).



Atribuição

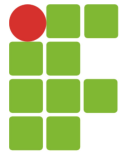
Atribuição

O comando de atribuição designa um valor para uma variável previamente declarada.

Para isto, utilizamos da seguinte sintaxe:

```
<variavel> = <expressão>
```

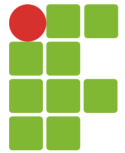
É importante declarar todas as variáveis antes de sua atribuição.



Atribuição

Exemplo: Atribuição

```
double pi;
pi = 3.1415;
```

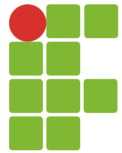


Atribuição

Exemplo: Atribuição

```
double pi;
pi = 3.1415;
```

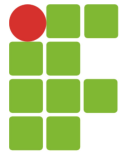
- Note que a separação entre a parte inteira e a parte fracionária é feita através do símbolo de '.'.



Atribuição

Exemplo: Atribuição

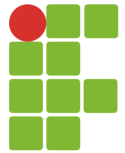
```
char c;
c = 65; /* C recebe o valor 65,
           que corresponde ao caractere 'A'.*/
```



Atribuição

Atribuição e Caracteres

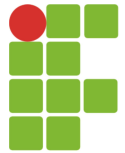
- Apesar de os caracteres se comportarem como inteiros de 1 byte na linguagem C, podemos usar o operador de atribuição indicando o valor do caractere entre aspas simples, sem a necessidade de consultar a tabela ASCII.
- Isto deixa o programa mais legível e menos propenso a erros.



Atribuição

Exemplo: Atribuição

```
char c;
c = 'A'; // C recebe o valor 'A', que internamente é 65.
```



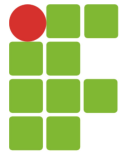
Atribuição

Declaração e Atribuição

É possível atribuir um valor imediatamente após a declaração de uma variável em C.

Isto pode ser feito através da seguinte sintaxe:

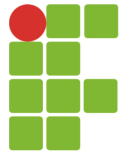
```
<tipo> <nome_da_variavel> = <valor>
```



Atribuição

Exemplo: Atribuição em Linha Única

```c
char c = 'A'; // C recebe o valor 'A'
```



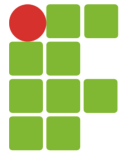
Atribuição

Declaração e Atribuição

O mesmo pode ser feito com múltiplas declarações em uma única linha.

Sintaxe:

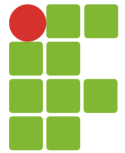
```
<tipo> <nome_1> = <valor_1>, ... , <nome_n> = <valor_n>;
```



Atribuição

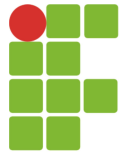
Exemplo: Atribuição

```
int primeiro_numero = 2, segundo_numero = 3;
```



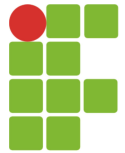
Sumário

4 Atribuição
- Constantes



Constantes

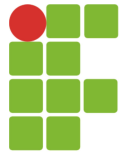
- Uma variável pode ser declarada com o modificador `const`.
- Isso efetivamente torna a variável uma constante, ou seja, não é possível alterar o valor dela.
- Somente é possível designar um valor à uma constante no momento de sua declaração.
- Caso o valor de uma constante seja alterado, o compilador indicará um erro de semântica durante a compilação.



Constantes

Exemplo: Constante

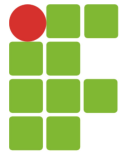
```
const double pi = 3.1415;
```



Constantes

Uso Incorreto de Constante

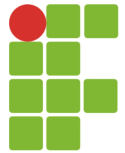
```
const double pi = 3.1415;
pi = 3.1415926; // não é possível alterar o valor de uma constante.
```



Constantes

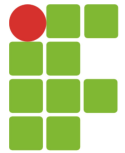
Boa Prática de Programação: Constantes

Caso você saiba de antemão que o valor de uma variável sempre será o mesmo, você pode declará-la como `const`. Isso deixa o código mais legível, uma vez que a variável está sendo explicitamente sinalizada como imutável.



Sumário

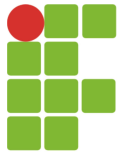
5 Estrutura



Estrutura Básica de um Programa em C

- A estrutura básica de um programa na linguagem C é a seguinte:

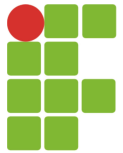
```
int main(void){
        declaração de variáveis
        ...
        comando_1
        comando_2
        ...
        comando_n
        return 0;
}
```



Estrutura Básica de um Programa em C

Exemplo: Programa de Soma

```
int main(void){
        int num_1 = 2, num_2 = 3, num_3;
        num_3 = num_1 + num_2;
        return 0;
}
```

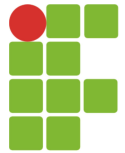


Estrutura Básica de um Programa em C

Exemplo: Programa de Soma

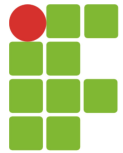
```
int main(void){
        int num_1 = 2, num_2 = 3, num_3;
        num_3 = num_1 + num_2;
        return 0;
}
```

- Qual o valor de num_3 ao fim do programa?



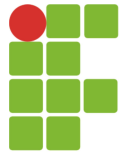
Sumário

5 Estrutura
- Comentários



Comentários

- Para documentar o nosso código e deixá-lo mais legível, tanto para nós como para as outras pessoas, é importante escrever comentários.
- Os comentários tem como missão descrever trechos complexos, realizar anotações no código ou até mesmo especificar a licença do software e o autor.
- Eles são completamente ignorados por compiladores ou interpretadores.
- Na linguagem `C`, comentários de várias linhas devem estar presentes entre os símbolos `/*` e `*/`.
- Comentários de uma única linha podem ser adicionados após os símbolos `//`.



Comentários

```
1  /**
2   * Autor: Daniel Saad
3   *
4   * Este programa realiza a soma entre dois inteiros com valor 2 e 3,
5   * armazenados nas variáveis num_1 e num_2 e armazena o valor desta
6   * soma na variável num_3
7   */
8  int main(void){
9      int num_1=2,num_2=3,num_3;
10     num_3 = num_1+ num_2; // a variável num_3 recebe a soma das outras duas
11     return 0;
12 }
```