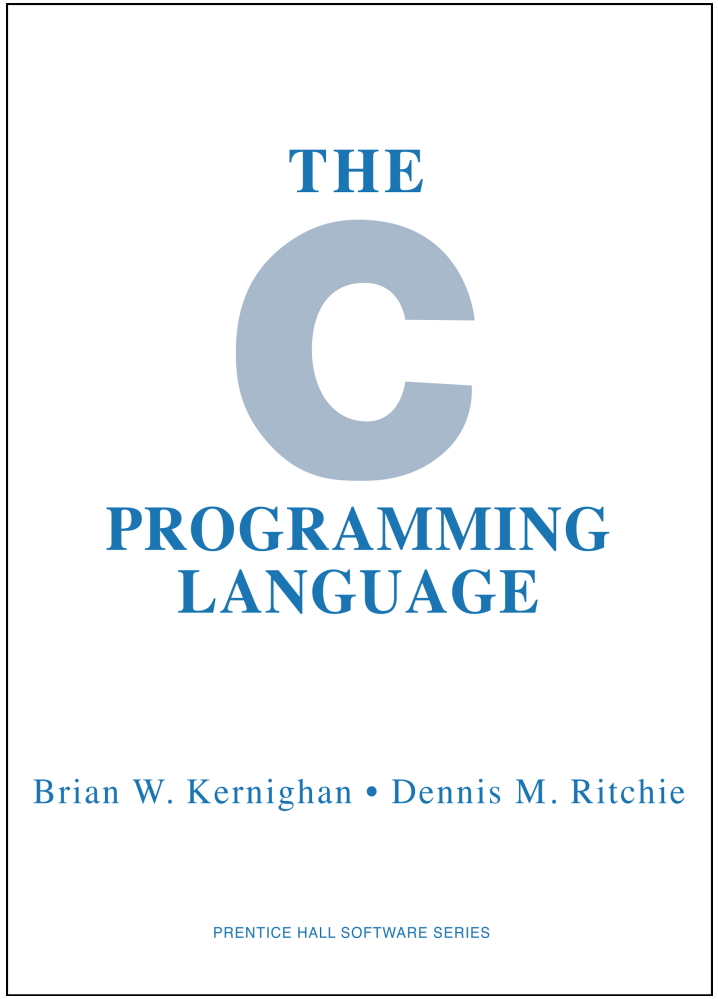
AULA 02

{introcomp}

CONCEITOS BÁSICOS I

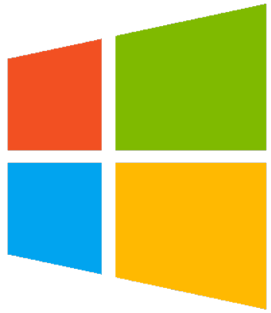
Histórico

- Criada em 1972 por Dennis Ritchie
- Usada no desenvolvimento do sistema operacional Unix no Bell Labs
- C foi derivada da linguagem B, desenvolvida por Ken Thompson
- Linguagem procedural de alto nível
- Rapidez de execução e eficiência em utilização de recursos do sistema operacional.



Utilização da Linguagem C

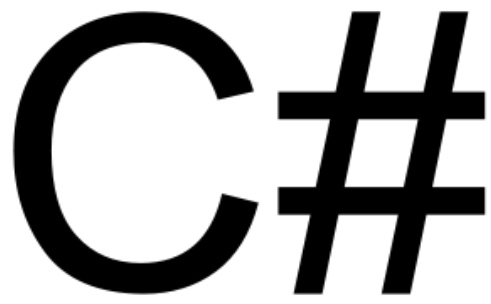
- Desenvolvimento de Sistemas Operacionais. (Microsoft Windows, Mac OS, GNU/Linux)



- Influenciou diversas outras linguagens (C++, JAVA,C# ...)
- Utilizado em microcontroladores (sistemas embarcados) como PIC, Arduino, MSP430, etc.

Por que aprender C?

- Sintaxe simples.
- Linguagem base para o aprendizado de diversas outras linguagens.
- 



- Adquire capacidade para tratar com problemas de linguagens de alto nível.
- C é uma linguagem muito utilizada.

Estruturação

- Processo de tradução de um programa descrito em linguagem de **alto nível** para um equivalente em linguagem de **baixo nível**.
- No GNU/LINUX existe o GCC (*GNU Compiler Collection*)

Estrutura Básica

```
//Inclusão de bibliotecas

int main()
{
   //Declaração de variáveis locais
   ... //Processamento de dados
   return 0;
}
```

Bibliotecas

- `#include` permite incluir uma biblioteca
- Bibliotecas contêm funções pré-definidas utilizadas nos programas
- Exemplos:

`#include <stdio.h>`	Funções de entrada e saída
`#include <stdlib.h>`	Funções de sistema
`#include <math.h>`	Funções matemáticas
`#include <string.h>`	Funções de texto

Variáveis

- Armazenar dados fornecidos pelo usuário.
- Manipular os dados durante
a execução do programa.

Variáveis

- Abstração para o endereço de memória.
- Células de memória são referenciadas por meio de rótulos (nomes de variáveis).

Endereço	Nome da Variável	Conteúdo
1001	var1	x
1002	var2	3.14
1003	var3	24
...	...	...

Identificadores

- Os identificadores são os “nomes” das variáveis
- Elemento definido pelo programador
- Regras:
  a. Diferença entre minúsculas e maiúsculas (case sensitive)
  b. “Não podem ter acentuação”
  c. Apenas os caracteres “_” (sublinha), e “$” são aceitos, além das letras do alfabeto e dígitos
  d. Não podem conter espaços
  e. Podem começar com letras, “_” e “$”
  f. Não podem ser palavras reservadas

Identificadores

Quais identificadores
estão
incorretos?

Identificadores

- X, a, z, fila, numero, LucroFinal
- 123, %cont, num#
- primeira_letra, prim_nome
- !dep, @asdf, ?alfa
- y1, x1, fila_11, z1, cont1a
- Con!ato, *resp, ?alfa, 1resp2
- Número, +ou-, Lucro Final

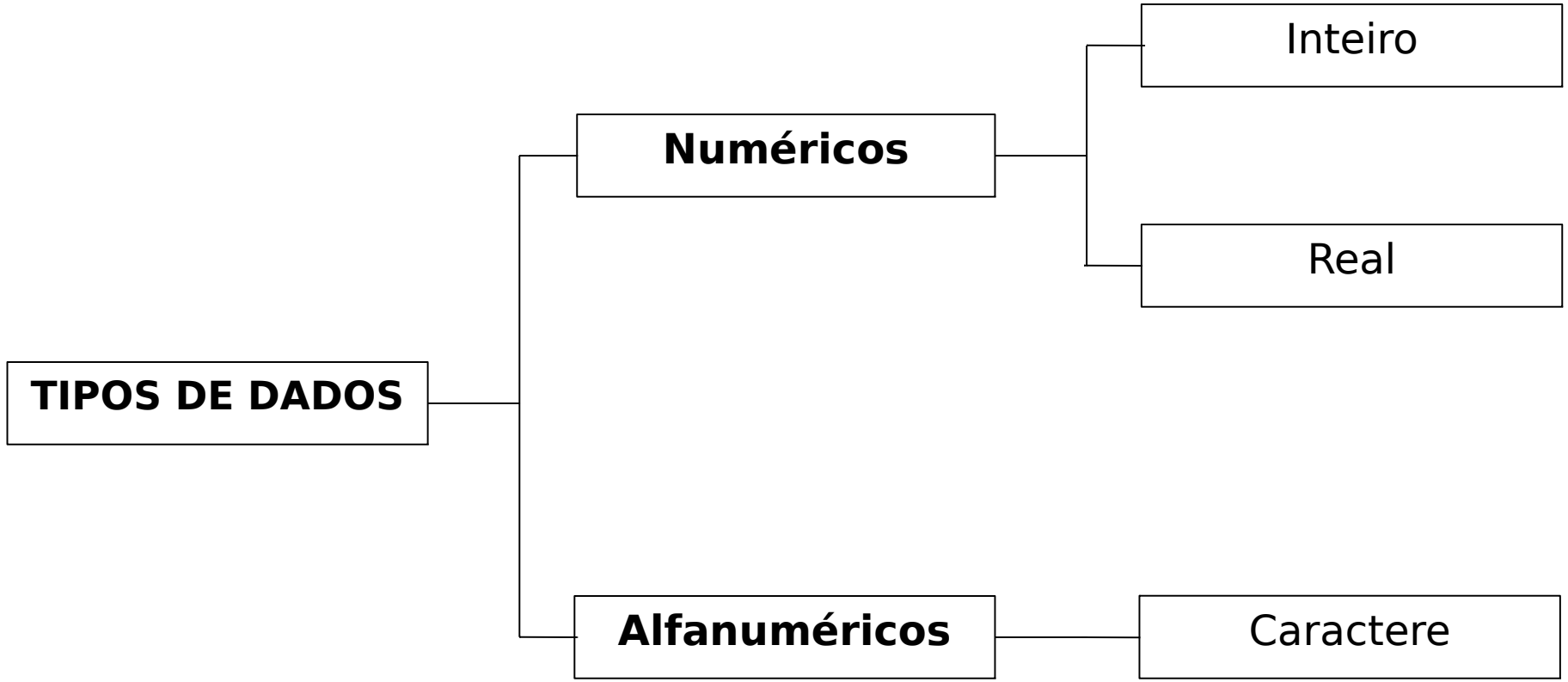
Identificadores

- X, a, z, fila, numero, LucroFinal
- 123, %cont, num#
- primeira_letra, prim_nome
- !dep, @asdf, ?alfa
- y1, x1, fila_11, z1, cont1a
- Con!ato, *resp, ?alfa, 1resp2
- Número, +ou-, Lucro Final

Palavras Reservadas

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

Tipos de Dados



Tipos de Dados

Inteiro: `int`

- Tamanho: 4 bytes = 32 bits
- Intervalo: -2.147.483.648 até 2.147.483.647

1 byte = 8 bits

Com isso temos que o numero total de valores a serem apresentados com o tipo inteiro é igual a

2^32 = 4.294.967.296

Tipos de Dados

Real: `float e double`

- Float
  - Tamanho = 4 bytes → 32 bits
  - ±3,4E-38 até ±3,4+38
  - Seis dígitos de precisão decimal
- Double
  - Tamanho = 8 bytes → 64 bits
  - ±1,7E-308 até ±1,7E+308
  - Dez dígitos de precisão decimal

Tipos de Dados

Alfanumérico: `char`

- Tamanho: 1 byte

1 byte = 8 bits → $2^8 = 256$

- Tipo utilizado para armazenar os caracteres
- Podem representar até 256 caracteres distintos

Tipos de Dados

Tabela ASCII

Decimal	Binário	Glifo
97	0110 0001	a
98	0110 0010	b
99	0110 0011	c
100	0110 0100	d
101	0110 0101	e
102	0110 0110	f
103	0110 0111	g

Tipos de Dados

Tabela ASCII

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

Tipos de Dados

Tabela dos tipos básicos

TIPO	TAMANHO	INTERVALO	VALOR
char	1 byte	-128 a 127	caractere
int	4 bytes	-2.147.483.648 a 2.147.483.647	inteiro
float	4 bytes	Seis dígitos de precisão	real
double	8 bytes	Dez dígitos de precisão	real

Tipos de Dados

Modificadores de tipos

- Modificadores de tipos
- Modificadores de tipos podem ser aplicados a tipos básicos para variar os intervalos
- Exemplos:
  - long long int
  - unsigned int
  - signed int
  - unsigned long long int
  - long double

Tipos de Dados

Modificadores de tipos

TIPO	TAMANHO	INTERVALO
short int	2 byte	-32.768 a 32.767
long int	4 bytes	-2.147.483.648 a 2.147.483.647
signed int	4 bytes	-2.147.483.648 a 2.147.483.647
unsigned int	4 bytes	0 a 4.294.967.295
signed short int	2 bytes	-32.768 a 32.767
unsigned short int	2 bytes	0 a 65.535
long long int	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
signed long long int	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
unsigned long long int	8 bytes	0 a 18.446.744.073.709.551.615

Variáveis

Declaração de variável

Sintaxe:

```
<tipo> <nome_da_variável>;
```

Exemplo:

```
int main()
{
  int x;
  ...
  return 0;
}
```

DECLARAÇÃO DA VARIÁVEL

Exemplo

Declaração de variáveis

```
#include <stdio.h> //Inclusão de bibliotecas
```

DECLARAÇÃO DAS VARIÁVEIS

```
int main()
{
   int x, y; //Declaração de inteiros
   char letra; //Declaração de caracter
   float num; //Declaração de ponto flutuante simples
   double soma, sub; //Declaração de ponto flutuante estendido
   ... //Processamento de dados
   return 0;
}
```

Comando de Atribuição

OPERAÇÃO	PSEUDOCÓDIGO	PROGRAMA EM C
Atribuição	<-	=

```
auxiliar <- a
a <- b
b <- auxiliar
```

```
auxiliar = a;
 a = b;
 b = auxiliar;
```

Comando de Atribuição

DECLARAÇÃO E ATRIBUIÇÃO DAS VARIÁVEIS

```
int main()
{
    double a = 5, b = 4, c = 3 ;

    a = 7;
    b = a;
    c = 11;

    return 0;
}
```

ATRIBUIÇÃO DAS VARIÁVEIS

Constantes

- Necessidade de utilização de um valor constante diversas vezes
- Sintaxe:

```
#define <nome_da_constante> <valor>
```

- Exemplo:

```
#include <stdio.h> //Inclusão de bibliotecas
#define MAXVALOR 10000
int main()
{
   int valor; //Declaração de variável inteira
   valor = MAXVALOR;
   return 0;
}
```

DEFINIÇÃO DE CONSTANTES

Expressões Aritméticas

- Expressões de tipos e operadores numéricos
- Comparando:

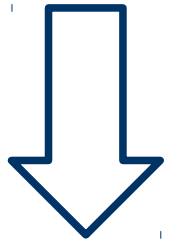
OPERAÇÃO	PSEUDOCÓDIGO	PROGRAMA EM C
Soma	+	+
Subtração	-	-
Multiplicação	*	*
Divisão inteira	/	/
Divisão real	/.	/
Resto da divisão	%	%

Expressões Aritméticas

Precedência

- Primeiro: multiplicação, divisão e resto de divisão
- Segundo: adição e subtração
- Não deve-se utilizar colchetes e chaves, apenas parênteses

```
{[3 + 4] * 5 + [25/5 + (12 * 5)]}
```



```
((3 + 4) * 5 + (25/5 + (12 * 5)))
```

Expressões Aritméticas

Exemplo

```
int main()
{
   int x, y;
   x = 10 + 10 * 2 ; //x = 30
   y = (10 + 10) * 2; // y = 40
   return 0;
 }
```

Expressões Aritméticas

- Funções matemáticas (`#include <math.h>`)
- Sintaxe: `<nome_da_função> (<valor>);`

NOME	DESCRIÇÃO
abs	Valor absoluto de um valor inteiro
fabs	Valor absoluto de um valor racional
sin	Seno de um ângulo em radianos
cos	Cosseno de um ângulo em radianos
sqrt	Raiz quadrada de um número
pow	Potenciação
floor	Arredondamento abaixo
ceil	Arredondamento acima
log	Logaritmo neperiano
exp	Exponencial

Compilação - GCC

gcc <nome_do_arquivo>.c
Gera um arquivo executável: a.out

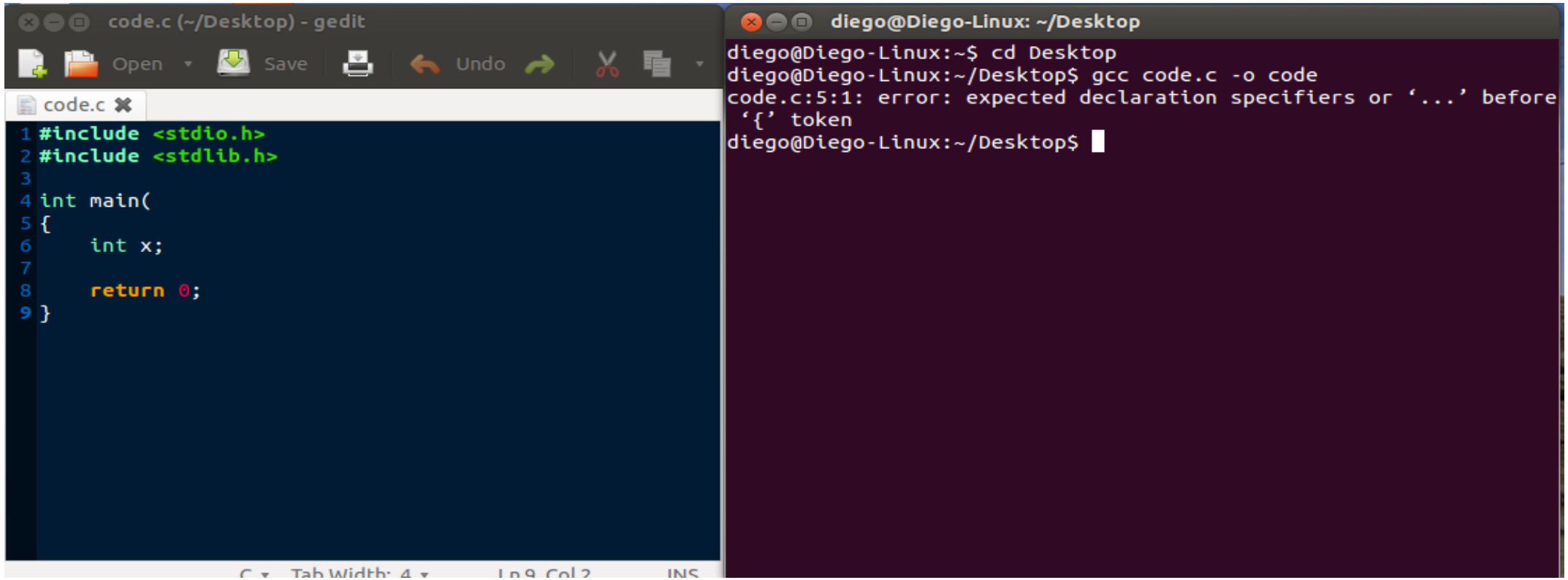
gcc <nome_do_arquivo>.c -o <nome_do_executável>
Gera um arquivo executável com o nome dado

Ex.:
gcc introcomp.c
gcc introcomp.c -o teste

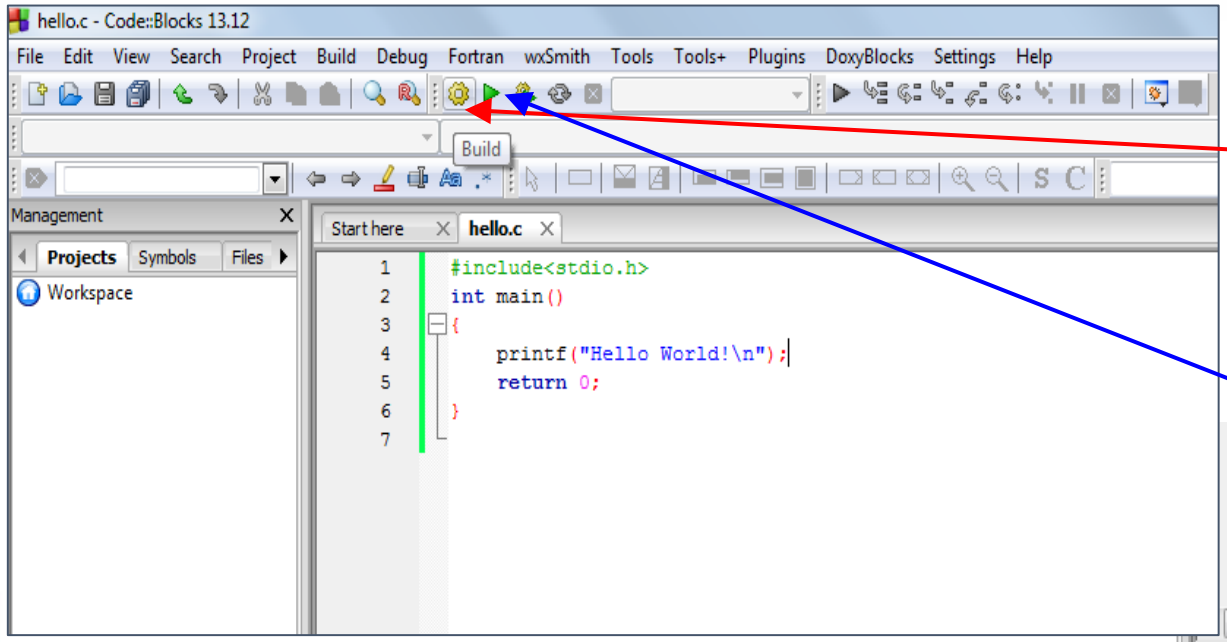
nome do executável
a.out
teste

Compilação - GCC

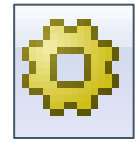
- O compilador indicará a presença de erros de sintaxe, etc.



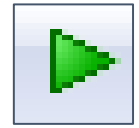
Compilação - Code::Blocks



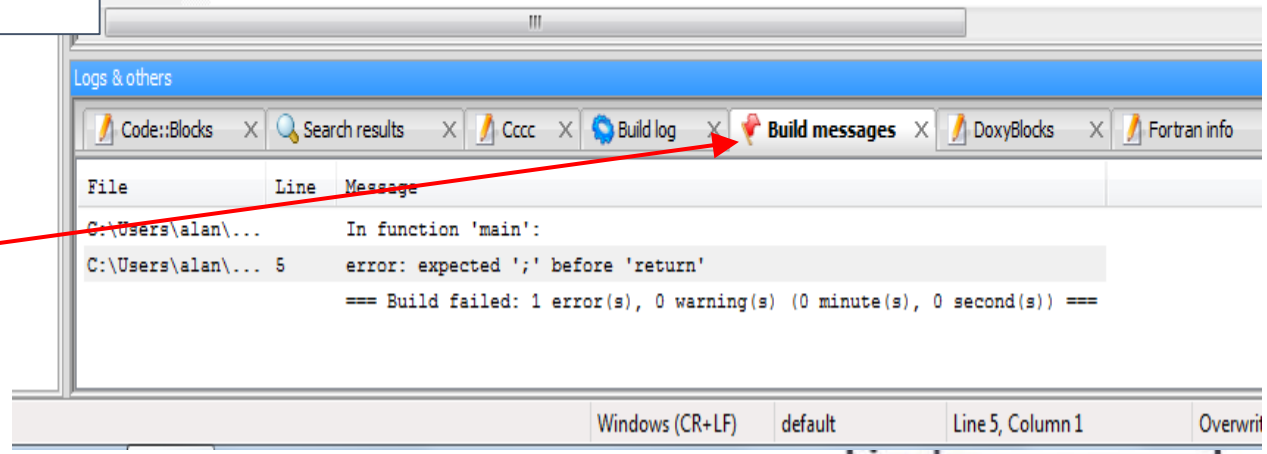
CLICAR EM *BUILD*
PARA COMPILAR



PARA EXECUTAR
CLIQUE EM *RUN*



BUILD MESSAGES
LISTA OS ERROS
DE COMPILAÇÃO



{introcomp}
INTROCOMP.PET.INF.UFES.BR

Estrutura Básica

FINALMENTE!!!

```
#include <stdio.h>

int main()
{
    printf (“Hello world!\n”);
    return 0;
}
```

Expressões Aritméticas

Faça um programa que calcule a área do círculo de raio 4

Expressões Aritméticas

Expressões Aritméticas

```
#include <math.h> //Inclusão de biblioteca
#define PI 3.14 //Definição de constante

int main()
{



   return 0;
}
```

ESTRUTURA INICIAL

Expressões Aritméticas

```
#include <math.h> //Inclusão de biblioteca
#define PI 3.14 //Definição de constante

int main()
{
   int raio = 4; //Declaração de variável inteira
   double area; //Declaração de variável ponto flutuante

   return 0;
}
```

DECLARAÇÃO DAS VARIÁVEIS

Expressões Aritméticas

```
#include <math.h> //Inclusão de biblioteca
#define PI 3.14 //Definição de constante

int main()
{
   int raio = 4; //Declaração de variável inteira
   double area; //Declaração de variável ponto flutuante
   area = PI * pow(raio,2); //Atribuição a variável
   return 0;

}
```

Expressões Aritméticas

Faça uma programa que calcule a área do losango de diagonais 5 e 4

Expressões Aritméticas

Expressões Aritméticas

```
int main()
{



   return 0;
}
```

ESTRUTURA INICIAL

Expressões Aritméticas

```
int main()
{
    int D, d; //Declaração de variável inteira
    double area; //Declaração de variável ponto flutuante



    return 0;
}
```

DECLARAÇÃO DAS VARIÁVEIS

Expressões Aritméticas

```
int main()
{
   int D, d; //Declaração de variável inteira
   double area; //Declaração de variável ponto flutuante
   D = 5;
   d = 4;
   area = (D * d)/2.0; //Atribuição a variável
   return 0;
}
```

Expressões Aritméticas

Quantos metros o som percorre em 10 segundos? Dados: velocidade do som = 340,3m/s

Dado: Velocidade = espaço percorrido / tempo decorrido

Expressões Aritméticas

Expressões Aritméticas

```
#define SOM 340.3

int main()
{



   return 0;
}
```

Expressões Aritméticas

```
#define SOM 340.3

int main()
{
   double dist;
   dist= SOM * 10;


   return 0;
}
```

DECLARAÇÃO DAS VARIÁVEIS

Tipos de Dados

O QUE ACONTECE QUANDO COLOCAMOS UM VALOR REAL EM UMA VARIÁVEL DE TIPO INTEIRO EM C?

Tipos de Dados

Conversão de tipos

- Conversão de real para inteiro

```
int main()
{
   int teste;
   teste = 3.534;
   return 0;
}
```

Tipos de Dados

Conversão de tipos

- Conversão de real para inteiro

```
int main()
{
   int teste;
   teste = 3.534;
   return 0;
}
   //teste = 3
```

Tipos de Dados

Conversão de tipos

- Divisão entre dois inteiros

```
int main()
{
   double teste;
   teste = 3/2;
   return 0;
}
```

Tipos de Dados

Conversão de tipos

- Divisão entre dois inteiros

```
int main()
{
   double teste;
   teste = 3/2;
   return 0;
}
  //teste = 1
```

Tipos de Dados

Conversão de tipos - casting

- Divisão entre dois inteiros

```
int main()
{
   double teste;
   teste = (double)3/2;
   return 0;
}
```

Tipos de Dados

Conversão de tipos - casting

- Divisão entre dois inteiros
-

```
int main()
{
   double teste;
   teste = (double)3/2;
   return 0;
}
  //teste = 1.5
```

Comandos de Entrada e Saída de Dados

Comando de Entrada

- Serve para captar do usuário do programa um ou mais valores
- Sintaxe:

```
scanf ("<formato1> ... <formatoN>", &var1, ..., &varN);
```

Pseudocódigo	Linguagem C
Leia(a)	`scanf("%d", &a);`
Escreva(a)	`printf("%d\n", a);`

Comandos de Entrada e Saída de Dados

Comando de Entrada

Argumento de controle	Uso na linguagem C

%d – Usado para referenciar tipo inteiro.	`int a;`
	`scanf("%d", &a);`
	`printf("%d", a);`

%f – Usado para referenciar tipos flutuantes/reais.	`float a;`
	`scanf("%f", &a);`
	`printf("%f", a);`

%c – Usado para referenciar caracteres.	`char a;`
	`scanf("%c", &a);`
	`printf("%c", a);`

%lf – Usado para referenciar tipos flutuantes/reais com maior precisão.	`double a;`
	`scanf("%lf", &a);`
	`printf("%lf", a);`

Comandos de Entrada e Saída de Dados

Comando de Saída

- Serve para escrever mensagens e exibir valores de variáveis
- Sintaxe:

```
printf ("<formato1> ... <formatoN>", var1, ..., varN);
```

FORMATO	SIGNIFICADO
`%c`	Imprimir um único caracter
`%d`	Imprimir um número inteiro (`int`)
`%f`	Imprimir um número real (`float`)
`%lf`	Imprimir um número real (`double`)
`%%`	Insere o símbolo "%"

NÃO PRECISA DO SÍMBOLO "&"

Exercícios

Faça um programa que calcule o valor da conta de água, dado a **leitura anterior**, e **atual** de metros cúbicos consumidos, e o **valor do m³**

Exercícios

Exercícios

```
#include <stdio.h>

int main()
{




return 0;
}
```

Exercícios

```
#include <stdio.h>

int main()
{
    double atual, anterior, diferenca;
    double valor_unit, valor_conta;




return 0;
}
```

DECLARAÇÃO DAS VARIÁVEIS

Exercícios

```
#include <stdio.h>

int main()
{
   double atual, anterior, diferenca;
   double valor_unit, valor_conta;

   scanf (“%lf %lf %lf”, &atual, &anterior, &valor_unit);



return 0;
```

LEITURA DOS VALORES

Exercícios

```
#include <stdio.h>

int main()
{
   double atual, anterior, diferenca;
   double valor_unit, valor_conta;

   scanf ("%lf %lf %lf", &atual, &anterior, &valor_unit);
   diferenca = atual - anterior;
   valor_conta = diferenca * valor_unit;

   printf ("Valor da conta: %lf\n", valor_conta);

   return 0;
}
```

SAÍDA DO PROGRAMA

Exercícios

Faça um programa para calcular a comissão (7%) de um vendedor, dado o total de vendas

Exercícios

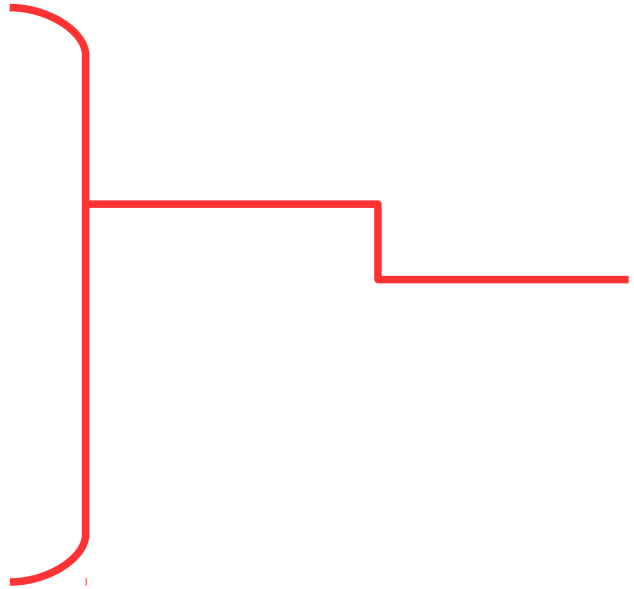
Exercícios

```
#include <stdio.h>

int main()
{




   return 0;
}
```



Exercícios

```
#include <stdio.h>

int main()
{
   double total,comissao;




   return 0;
}
```

DECLARAÇÃO DAS VARIÁVEIS

Exercícios

```
#include <stdio.h>

int main()
{
   double total,comissao;
   printf (“Digite o total de vendas: ”);
   scanf (“%lf”, &total);



   return 0;
}
```

LEITURA DO
DADO DE ENTRADA

Exercícios

```
#include <stdio.h>

int main()
{
   double total,comissao;
   printf (“Digite o total de vendas: ”);
   scanf (“%lf”, &total);
    comissao = total*0.07;
   printf (“Total: %lf\nPercentual: 7%%\nComissão: %lf”, total,
comissao);

   return 0;
}
```

SAÍDA DO PROGRAMA

Exercícios

Faça um programa para calcular a distância entre dois pontos

Exercícios

Exercícios

```
#include <stdio.h>
#include <math.h>
int main()
{




   return 0;
}
```

ESTRUTURA INICIAL

Exercícios

```
#include <stdio.h>
#include <math.h>
int main()
{
   double x1, y1, x2, y2, dx, dy, dist;




   return 0;
}
```

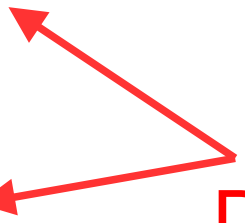
DECLARAÇÃO DAS VARIÁVEIS

Exercícios

```
#include <stdio.h>
#include <math.h>
int main()
{
   double x1, y1, x2, y2, dx, dy, dist;
   printf (“Digite um ponto: ”);
   scanf (“%lf %lf”, &x1, &y1);
   printf (“Digite um ponto: ”);
   scanf (“%lf %lf”, &x2, &y2);




   return 0;
}
```



LEITURA DOS
DADOS DE ENTRADA

Exercícios

```
#include <stdio.h>
#include <math.h>
int main()
{
   double x1, y1, x2, y2, dx, dy, dist;
   printf (“Digite um ponto: ”);
   scanf (“%lf %lf”, &x1, &y1);
   printf (“Digite um ponto: ”);
   scanf (“%lf %lf”, &x2, &y2);
   dx = x2 – x1;
   dy = y2 – y1;
   dist = sqrt (pow(dx,2) + pow(dy,2));
   printf (“Distância entre os pontos: %.2lf”, dist);

   return 0;
}
```

SAÍDA DO PROGRAMA

AULA 02

{introcomp}

CONCEITOS BÁSICOS I