Rodrigo Henrich

rodrigohenrich@faccat.br

Manipulando dados em C

Variáveis

- Aqui, assim como nos algoritmos temos variáveis, elas podem assumir determinado valor e ser modificadas ao longo da execução do programa
- Porém como o C é uma linguagem fortemente tipada, temos de declarar a existência de nossas variáveis antes de usá-las.
- Para fazer isso em C e na grande maioria das linguagens usamos a sintaxe

  tipo_variável nome_variável

Variáveis

```
int x;
```

- No exemplo acima criamos uma variável do tipo inteiro que pode receber valores numéricos inteiros. Ex, 1, 2, 193, etc.

```
1  #include <stdio.h>
2  int main(){
3    int x;
4    return 0;
5  }
```

Nomeando variáveis

- Existem algumas palavras reservadas da linguagem que não podem ser usados como nomes de variáveis, tais como

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

Nomeando variáveis

- Além disso variáveis não podem ter espaços em branco no nome, então

```
int numero casa;
```

```
int numeroCasa;
```

```
int numero_casa;
```

Nomeando variáveis

- Também não podem iniciar com um número

```
int 1numero;

int numero1;
```

Nomeando variáveis

- O C é uma linguagem case sensitive, o que significa que

  A não é a mesma coisa de **a**

  ```
  int A;
  ```

  ```
  int a;
  ```

- São duas variáveis diferentes

Nomeando variáveis

- Considera-se uma boa prática de programação dar nomes que façam sentido para o contexto da variável, por exemplo
  - Três notas de um aluno
    - Ao invés de usar x, y e z
    - Usar n1, n2 e n3 ou nota1, nota2, nota3

Tipos de dados em C

- As variáveis podem assumir determinado domínio de informação, por exemplo no domínio dos inteiros não cabe o número 1.5
- Em C temos 4 tipos essenciais de variáveis
  - **char** pode armazenar um caractere ‘a’, ‘b’, ‘2’,...
  - **int** pode armazenar números inteiros 1, 2, 3,...
  - **float** pode armazenar números com ponto flutuante 1.5, 0.5, 3.14,...
  - **double** têm a mesma função que o **float**, porém com maior capacidade

Tipos de dados em C

Tipo	Intervalo de valores	Exemplo
char	-127 a 127	chat letra = ‘a’;
int	-32.767 a 32.767	int numero = 25;
float	$3,4E^{-38}$ a $3.4E^{+38}$ (6 dígitos precisão)	float area = 5.4;
double	$1,7E^{-308}$ a $1,7E^{+308}$ (10 dígitos precisão)	double valor = 10.50

Modificadores de tipo

- **signed** é o padrão da linguagem, determina que as variáveis podem assumir valores negativos e positivos
- **unsigned** determina que a variável poderá assumir valores apenas positivos
- **long** uma variável do tipo long, tem sua faixa de valores estendida, com o preço de ocupar mais espaço na memória do computador
- **short** reduz o tamanho ocupado por uma variável, geralmente empregado apenas em inteiros, ao preço de suportar números menores.

Tipos de dados em C

Tipo	Intervalo de valores	Tamanho (bits)
char	-128 a 127	8
int	-32.767 a 32.767	16
unsigned int	0 a 65.535	16
long int	-2.147.483.647 a 2.147.483.647	32
unsigned long int	0 a 4.294.967.295	32

Tipos de dados em C

Tipo	Intervalo de valores	Tamanho (bits)
float	$3,4E^{-38}$ a $3.4E^{+38}$	32
double	$1,7\ E^{-308}$ a $1,7E^{+308}$	64
long double	$3,4E^{-4932}$ à $3,4E^{+4932}$	80

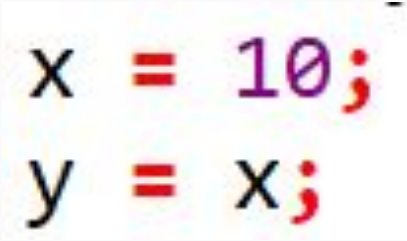
Inicialização de variáveis

- É possível dar um valor a determinada variável já na hora que ela é declarada
  - Ex **int numero = 0;** declara a variável numero do tipo inteiro que recebe o valor 0;
  - Ex **char letra = 'B';** declara a variável letra do tipo caracter que recebe o valor B;
  - Ex. **float valor = 3.50;** declara a variável valor do tipo float e a inicializa com o valor 3,50

```
int numero = 0;
char letra = 'B';
```

Atribuição de variáveis

- Para atribuir um valor a uma variável usamos o sinal de =
  - Ex. **a = 30**; a variável a recebe o valor 30
  - Ex. **b = a**; a variável **b** recebe o valor da variável **a**



Escrevendo na tela

- Para escrever na tela temos que usar a função **printf()**;
- Para escrever um texto na tela usamos a seguinte sintaxe
  - printf(“Texto”);

```
printf("Texto a ser escrito na tela");
```

Escrevendo na tela

- Exemplo

```c
#include <stdio.h>
int main(){
  printf("Texto a ser escrito na tela");
  return 0;
}
```

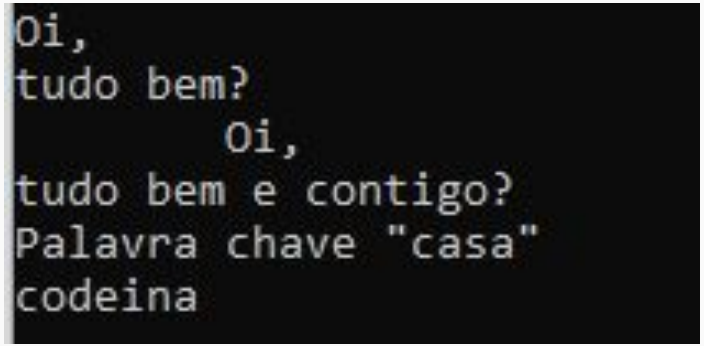
Algumas opções para formatar texto

Caractere	efeito
\n	quebra de linha
\t	tabulação horizontal
\r	volta o cursor para o início da linha
\”	mostra aspas duplas na tela
\’	mostra aspas simples na tela
\\	mostra contra barra na tela

Exemplo

```c
#include <stdio.h>
int main(){
   printf("Oi, \ntudo bem?\n");
   printf("\tOi,\ntudo bem e contigo?\n");
   printf("Palavra chave \"casa\"\n");
   printf("cafeina");
   printf("\rcode");
}
```

Exemplo saída do código



Escrevendo valores formatados na tela

- Nem sempre vamos escrever apenas texto, na maioria dos casos vamos precisar dar alguma resposta ao usuário
- Para isso vamos precisar escrever o resultado de um cálculo
- Neste caso a função printf, permite escolher alguns tipos de saída
- Cada um destes tipos de saída está associado a um tipo de dado específico
- Os tipos de saída sempre são precedidos de um **%**, desta forma o compilador entende que ali deverá aparecer um determinado valor.

Escrevendo valores formatados na tela

- Usamos a seguinte sintaxe para expressar os valores na tela
  - printf(“%tipo_saida”,expressão);

- E se forem duas expressões
  - printf(“%tipo1 %tipo2”, expressão1, expressão2);

Escrevendo valores formatados na tela

- Alguns exemplos de tipos de saída

%c	escrita de um caractere
%d ou %i	escrita de números inteiros
%u	escrita de números inteiros sem sinal
%f	escrita de números reais
%.2f	escrita de números reais limitando a 2 casas decimais, o 2 é um exemplo, podemos usar quantas casas desejarmos

Escrevendo valores formatados na tela

- Alguns exemplos de tipos de saída

%s	Escrita de vários caracteres
%p	Escrita de um endereço de memória
%e ou %E	Escrita de um número em notação científica

Exemplo 1

```c
#include <stdio.h>
int main(){
   int x = 10;
   //Escrita de um valor inteiro
   printf("%d",x);
   return 0;
}
```

Exemplo 2

```c
#include <stdio.h>
int main(){
   float y = 5.4;
   //Escrita de um valor real
   printf("%f",y);
   return 0;
}
```

Exemplo 3

```c
#include <stdio.h>
int main(){
   int x = 10;
   float y = 5.4;
   //Escrita de um valor inteiro e outro real
   printf("%d %f",x,y);
   return 0;
}
```

Exemplo 4

- Escreva um programa que crie uma variável do tipo inteiro
- Atribua a ela o valor 20
- Crie outra variável do tipo char com a letra ‘d’
- Escreva seus respectivos valores na tela

Frase e valores ao mesmo tempo

- Para escrever uma frase e determinado valor ao mesmo tempo a sintaxe é muito parecida
  - printf(“texto %tipo_saida”,expressão);
  - printf(“texto %tipo_saida1 mais texto %tipo_saida2”,expressao1, expressao2);

Exemplo 5

```c
#include <stdio.h>
int main(){
   int idade = 10;
   float nota;
   nota = 6.0;
   printf("A aluna Ana tem %d anos e teve nota %f",idade,nota);
}
```

Exemplo 5 saída

```
A aluna Ana tem 10 anos e teve nota 6.000000
```

A função putchar

- A função putchar em c serve para escrever um único caractere na tela
- Ela recebe como parâmetro um valor inteiro ou caractere que será mostrado na tela
- Os números correspondentes a determinados caracteres pode ser encontrado na tabela ASCII

Exemplo

```c
#include<stdio.h>

int main(){
   putchar(65);
   putchar('\n');
   putchar('a');
}
```

Tabela ASCII

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]

Lendo dados do teclado

- Para ler dados do teclado em C temos que usar a função scanf
- Seu nome tem origem parecida com o nome da função printf
  - printf - **print f**ormatetted
  - scanf - **scan f**ormatted
- A função **scanf** espera dois parâmetros
  - tipo de entrada de dados
  - lista de variáveis

Lendo dados do teclado

- Lendo uma variável
  - scanf(“%tipo_de_entrada”, &nome_variavel);
- Lendo duas variáveis
  - scanf(“%tipo1 %tipo2”, &var1, &var2);
- O **&** deve aparecer porque o scanf espera o endereço da variável na memória e não apenas o nome dela.
- **&nome_variavel** equivale ao endereço da variável na memória

Lendo dados do teclado

Alguns tipos de entrada	
%c	leitura de um caractere
%d ou %i	leitura de números inteiros
%f	leitura de números reais
%s	leitura de vários caracteres

Lendo dados do teclado

```c
#include<stdio.h>
int main(){
  int num1, num2;
  float num3;
  char letra;
  //Leitura de um caractere
  printf("\nDigite um caractere: ");
  scanf("%c", &letra);
  //Leitura de um número inteiro
  printf("\nDigite um numero inteiro: ");
  scanf("%d", &num1);
  //Leitura de um número real
  printf("\nDigite um numero real. Ex: 1.5 ");
  scanf("%f", &num3);
  //Leitura de dois números ao mesmo tempo um inteiro e outro real
  printf("\nDigite um numero real e outro inteiro: ");
  scanf("%f %i",&num3, &num2);
  return 0;
}
```

Comando getchar

- No C assim como existe o comando putchar() que mostra um caractere na tela
- Existe o comando getchar(), que tem a função de ler apenas um caractere do teclado e retorná-lo.
- Sua sintaxe não pede nenhum parâmetro, e retorna um caractere
  - char getchar();

Comando getchar

```c
#include<stdio.h>

int main(){
   char letra;
   letra = getchar();
}
```

Comando getchar

```c
#include<stdio.h>

int main(){
   char letra;
   letra = getchar();
   printf("Foi digitado o caractere: %c",letra);
}
```

Operadores matemáticos em C

- = atribuição
  - x = 10;
  - y = getchar();
- + soma
  - a = b + c;
  - a = a + 5;
- - subtração
  - a = b - c;
  - a = b - 2;

Operadores matemáticos em C

- * multiplicação
  - x = a*b;
  - x = 2*2;
- / divisão
  - x = a/b;
  - **O divisor precisa ser diferente de 0**;
- % resto da divisão
  - x = 4%2 x irá valer 0
  - x = 5%2 x irá valer 1
  - **O divisor precisa ser diferente de 0**;