

Profa. Michele Fúlvia Angelo

**PGCA 028 – Tópicos Especiais em Tecnologia
Computacional I – Introdução à Programação de
Computadores
Aula 2**

Universidade Estadual de Feira de Santana

Sumário

- Linguagens de Programação
- Introdução à Linguagem C

Dos Algoritmos para os Programas

- **Programação é...**

- *A seqüência de planejamento, projeto, escrita, instalação e testes de instruções desempenhadas pelo computador.*
- *Envolve obediência às regras (léxicas, sintáticas e semânticas), organização, otimização, documentação.*

Um Programa de Computador

- Um programa nada mais é do que uma seqüência de instruções que possui significado para o computador.

Linguagens de Programação

- Permitem fornecer instruções ao computador através de comandos escritos em uma linguagem próxima à linguagem humana;
- Linguagem que o computador entende é a linguagem de máquina, composta de 0 e 1;
- Exemplos: Pascal, C, C++, Delphi, Java.

Linguagens de Programação

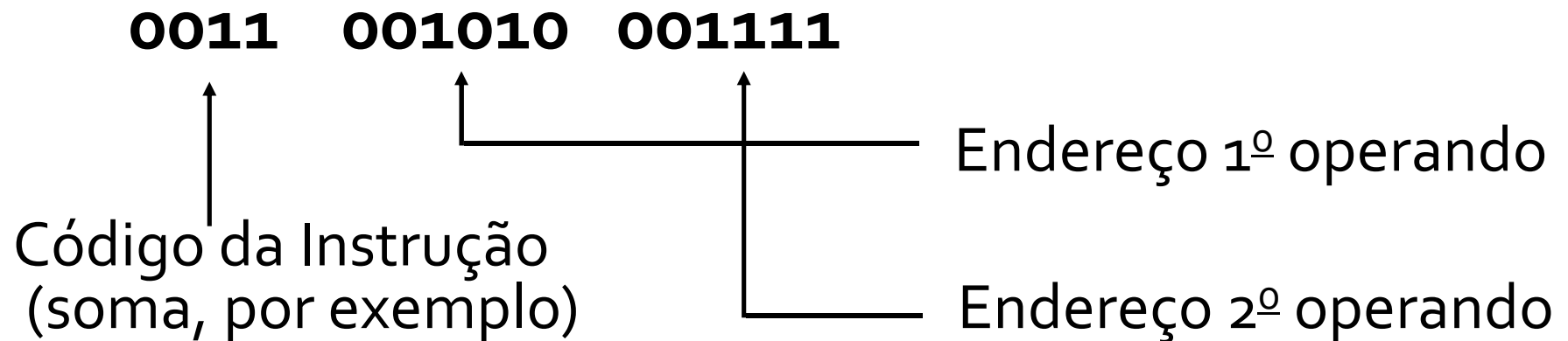
- Cada linguagem de programação obedece a regras específicas.
- As regras de sintaxe de uma linguagem de programação definem como são expressadas as instruções a serem executadas.

Linguagens de Programação

- **Baixo Nível** - Linguagem de Máquina. Mais compatíveis com o hardware do computador.
- **Alto Nível** - Linguagens de Compilação e Interpretação. Similares à nossa linguagem natural.

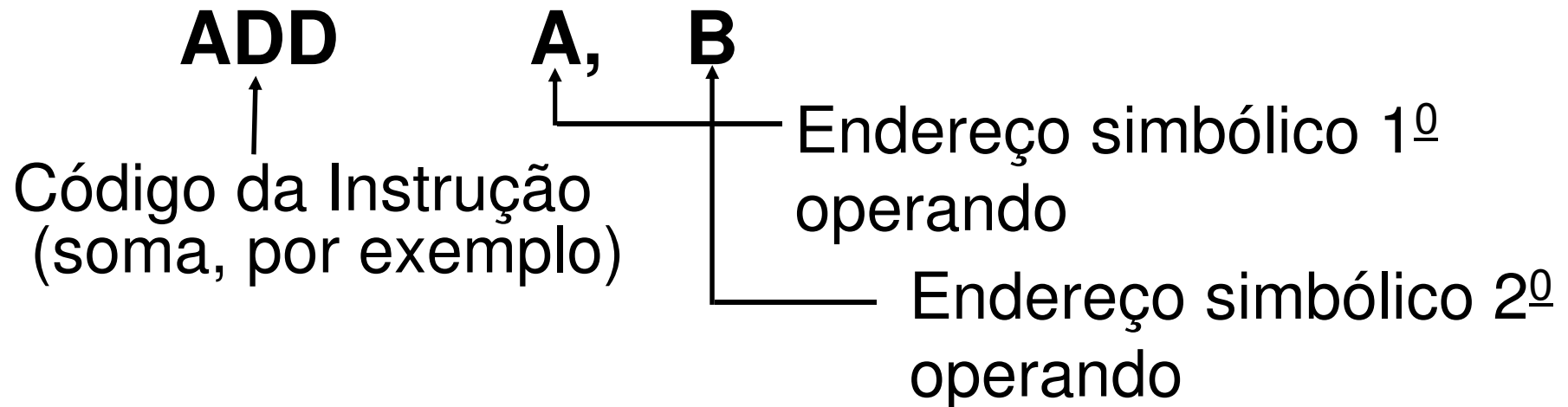
Linguagens de Baixo Nível

- Linguagem de máquina:
 - Formada por códigos binários interpretados diretamente pelo hardware do computador



Linguagens de Baixo Nível

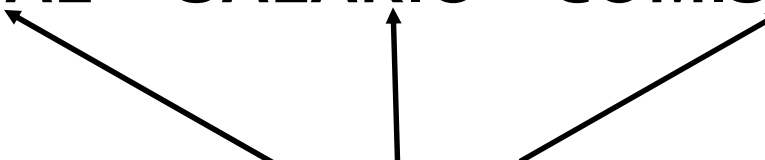
- Linguagem Assembly:
 - Assembly ou linguagem de montagem é uma notação legível por humanos para o código de máquina que uma arquitetura de computador específica usa.
 - A linguagem de máquina, que é um mero padrão de bits, torna-se legível pela substituição dos valores em bruto por símbolos chamados mnemônicos.



Linguagens de Alto Nível

- Usa sintaxe próxima da linguagem do usuário;
- Precisa ser convertida em Linguagem de Máquina, para que seja interpretada pelo hardware do computador.
 - Exemplo:

TOTAL = SALARIO + COMISSOES

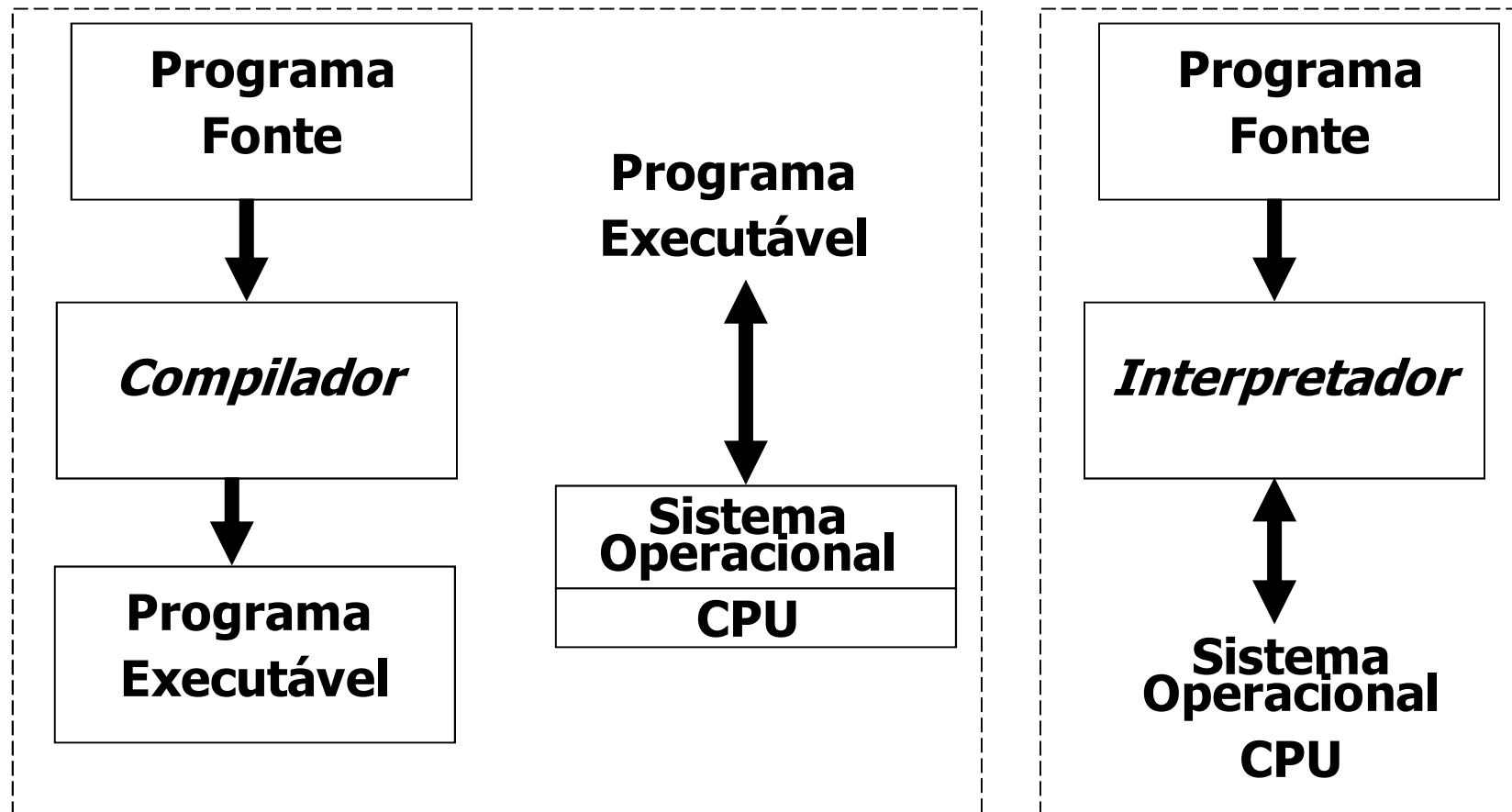


Nomes de Variáveis

The diagram illustrates the concept of variable names in a high-level programming language. It shows the equation 'TOTAL = SALARIO + COMISSOES'. Below the equation, the text 'Nomes de Variáveis' (Variable Names) is centered. Three arrows originate from this text: one points to 'TOTAL', one points to 'SALARIO', and one points to 'COMISSOES', indicating that these are the variable names in the expression.

Linguagens de Alto Nível

- Conversão em Linguagem de Máquina:



A Linguagem C



- A linguagem C foi criada por Dennis Ritchie, em 1972, no centro de Pesquisas da Bell Laboratories.
- Sua primeira utilização importante foi a reescrita do Sistema Operacional UNIX, que até então era escrito em Assembly.
- Em meados de 1973 o UNIX saiu do laboratório para ser liberado para as universidades.
- Foi o suficiente para que o sucesso da linguagem atingisse proporções tais que, por volta de 1980, já existiam várias versões de compiladores C oferecidas por várias empresas, não sendo mais restritas apenas ao ambiente UNIX, porém compatíveis com outros sistemas operacionais.
- O C é uma linguagem de propósito geral, sendo adequada à programação estruturada.

Estrutura de um Programa em Linguagem C

- Todo programa em C inicia sua execução chamando a função `main()`, sendo obrigatória a sua declaração no programa principal.

```
int main() {  
    instruções  
}
```

- Cada instrução encerra com `;` (ponto e vírgula).
- Exemplo:

```
#include <stdio.h>  
int main() {  
    printf("Este é um programa em C\n");  
}
```

Programando em Linguagem C

- **Palavras reservadas:** palavras que tem significado especial previamente definido, não podendo ser usadas como identificador de variáveis.
Exemplo: `scanf`, `printf`, `main`.
- **Comentários:** texto explicativo entre `/*` e `*/` (comentário de bloco) ou após `//` (comentário de linha), para auxiliar no entendimento dos diversos blocos do programa.
- A linguagem C é ***case sensitive***: diferencia maiúsculas de minúsculas.

Programando em Linguagem C

- Operadores Aritméticos

- Unários: - (valor negativo)

- Binários: +, -, *, /, %

$$10 + 3 = 13$$

$$10 - 3 = 7$$

$$10 * 3 = 30$$

$$10 / 3 = 3,3333...$$

$$10 \% 3 = 1 \text{ (resto da divisão inteira)}$$

Programando em Linguagem C

- Operadores Lógicos

- ! (NÃO)
- || (OU)
- && (E)

- Resultado: booleano

- 0 (falso)
- 1 (verdadeiro)

Programando em Linguagem C

- Operadores Relacionais

- `==, !=, >, <, >=, <=`

- Resultado: booleano

- `(10 > 5), (5 == 5), (10 != 5)`

verdadeiro

- `(10 <= 5), (5 != 5), (10 == 5)`

falso

Programando em Linguagem C

- Precedência entre os operadores:

primeiro

! (negação)

* / % (multiplicativos)

+ - (aditivos)

< > <= >= (desigualdade)

== != (igualdade)

&& (E lógico)

último

|| (OU lógico)

Programando em Linguagem C

■ Operador de Atribuição (=)

- Para armazenar um valor em uma constante ou variável.

...

```
nota = (7 + 8 + 6)/3;  
printf( "A nota é %d", nota );
```

...

- Primeiro, avalia-se o que está a direita.
- Depois, o valor é armazenado na variável à esquerda.
- À esquerda, só uma variável.
- O valor anterior da variável é perdido.
- O valor a ser atribuído deve ser compatível com o tipo da variável.

Programando em Linguagem C

- **Operadores de Incremento e Decremento**
 - O operador incremento soma 1 ao seu operando, e o decremento subtrai 1.
 - `x++;` equivale a `x = x + 1;`
 - `x--;` equivale a `x = x - 1;`
 - Podem ser usados como operadores pré-fixo(`++x`) ou pós-fixo(`x++`):
 - `++x` incrementa `x` antes de utilizar o seu valor;
 - `x++` incrementa `x` depois de ser utilizado;

```
x = 23;  
y = x++;  
no final tem-se y = 23 e x = 24
```

```
x = 23;  
y = ++x;  
no final tem-se y = 24 e x = 24
```

Programando em Linguagem C

- Exemplo:

```
main() {  
    int x=0;  
    printf("x= %d\n", x++);  
    printf("x= %d\n", x);  
    printf("x= %d\n", ++x);  
    printf("x= %d\n", x);  
}
```

Programando em Linguagem C

- **Atribuições especiais com operadores**
 - `num1 += 10;` equivale a `num1 = num1 + 10;`
 - `num2 *= 10;` equivale a `num2 = num2 * 10;`
 - `num2 -= 10;` equivale a ?
 - `num2 /= 10;` equivale a ?

Primeiros Programas

```
#include <stdio.h>
int main() {
    printf("Oi, mundo!\n");
}
```

```
#include <stdio.h>
int main() {
    printf("%d", 21);
}
```

Bibliotecas

#include <nome biblioteca>

- Avisa ao compilador que serão usados procedimentos, funções, variáveis ou constantes declarados no arquivo especificado (no exemplo anterior, stdio.h)
- Outras bibliotecas de uso comum em C
 - **#include** <math.h> /* Funções matemáticas */
 - **#include** <stdlib.h> /* Funções de gerência de memória */
 - **#include** <string.h> /* Funções de manipulação de strings */

Variáveis

- Declarações de variáveis

`tipo_de_dado nome_variável;`

```
int main(){  
/*declaração de variável para números inteiros*/  
  int nota;  
  int prova1, prova2;  
  ...  
}
```



Inicialização de Variáveis

```
int main() {  
    int nota;  
    printf("%d", nota );  
}
```

- O que será mostrado?
 - Declaração de variável = Reserva de espaço na memória.
 - Espaço pode estar limpo (zerado) ou não (lixo).

Inicialização de Variáveis

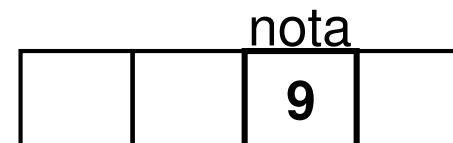
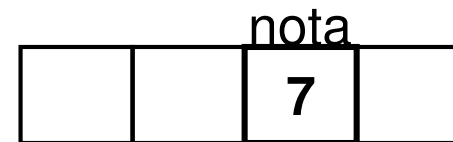
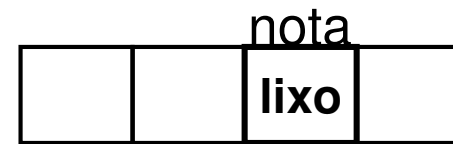
```
int main() {  
    int nota1, nota2, media;  
  
    nota1 = 10;  
    nota2 = 5;  
    media = nota1 + nota2;  
    printf("%d", media );  
}
```

```
int main() {  
    int nota1 = 10;  
    int nota2 = 5;  
    int media;  
  
    media = nota1 + nota2;  
    printf("%d", media );  
}
```

Variáveis na Memória

■ Exemplo

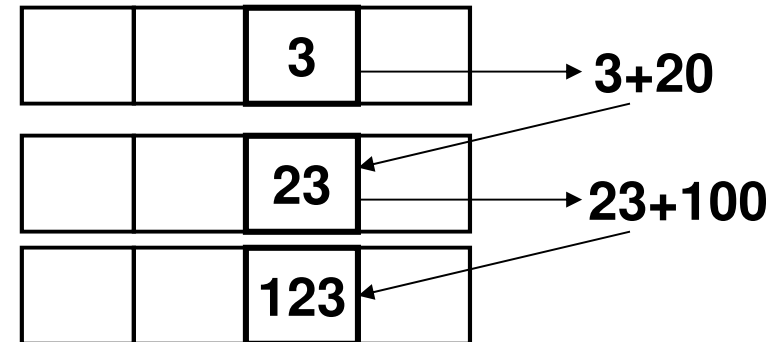
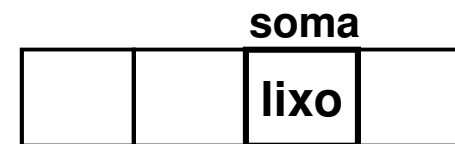
```
int main() {  
    int nota;  
  
    nota = 4 + 3;  
    printf("%d", nota );  
  
    nota = 9;  
    printf("%d", nota );  
  
}
```



Variáveis na Memória

■ Exemplo

```
int main() {  
    int soma;  
  
    soma = 3;  
    soma = soma + 20;  
    soma = soma + 100;  
    printf("O valor em Soma  
    é: %d", soma);  
}
```



Variáveis na Memória

■ Exemplo

```
int main() {  
    int produto;  
    produto = 2;  
    produto = produto * produto;  
    produto = produto * produto;  
    produto = produto * produto;  
    printf("O valor final  
é:%d", produto);  
}
```

Constantes

- Quando usamos um mesmo valor várias vezes.

```
/* Programa que calcula e exibe o tamanho da circunferência de um
círculo de raio 3,5,7 */
#include <stdio.h>
int main() {
    printf ("Tamanho da circunferencia de um circulo de raio 3: ");
    printf ("%f\n", 2*3.1415926536*3);
    printf ("Tamanho da circunferencia de um circulo de raio 5: ");
    printf ("%f\n", 2*3.1415926536*5);
    printf ("Tamanho da circunferencia de um circulo de raio 7: ");
    printf ("%f\n", 2*3.1415926536*7);
}
```

- Saída:

Tamanho da circunferencia de um circulo de raio 3: 18.849556
Tamanho da circunferencia de um circulo de raio 5: 31.415927
Tamanho da circunferencia de um circulo de raio 7: 43.982297

Constantes

- Podemos definir uma constante
 - Seu conteúdo não pode ser modificado durante a execução de um programa.

```
/* Programa que calcula e exibe o tamanho da circunferência de um
círculo de raio 3,5,7 */
#include <stdio.h>
#define PI 3.1415926536 /* PI tem o valor 3.1415926536 */
int main() {
    printf ("Tamanho da circunferencia de um circulo de raio 3: ");
    printf ("%f\n", 2*PI*3);
    printf ("Tamanho da circunferencia de um circulo de raio 5: ");
    printf ("%f\n", 2*PI*5);
    printf ("Tamanho da circunferencia de um circulo de raio 7: ");
    printf ("%f\n", 2*PI*7);
}
```


Constantes

- Outras constantes:

- Número inteiro:

```
#define V 1
```

```
/* V tem valor 1 */
```

- Caractere:

```
#define UNIDADE 'm'
```

```
/* UNIDADE tem valor 'm' */
```

- Texto (string ou cadeia de caracteres):

```
#define MSG "Informe um número inteiro positivo: "
```

```
/* um texto é o valor de MSG */
```

Tipos de Dados

- Tipos de dados básicos:
 - **char**: Guarda um caractere;
 - **int**: Guarda um número inteiro;
 - **float**: Guarda um número real com certa precisão;
 - **double**: Guarda um número real com precisão maior que float;
 - **void**: Tipo vazio.
 - Modificadores de tipos de dados:
 - **unsigned** = sem sinal;
 - **long** = domínio estendido;
 - **short** = domínio reduzido;
- ✓ Ao **float** não se pode aplicar nenhum e ao **double** pode-se aplicar apenas o **long**.

Tipos de Dados

Tipo	Num de bits	Formato para leitura com scanf	Intervalo	
			Início	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%i	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3.4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

Entrada de Dados

scanf("expressão de controle", argumento);

- É uma função de I/O, que permite ler dados formatados da entrada padrão (teclado).
- Expressão de controle
 - Cadeia de caracteres que informam como será a entrada;
 - Sempre contém o especificador para a variável, indicado pelo caractere '%';
- Argumento
 - Depende da lista de formato;
 - Sempre é uma variável, precedida por '&' (importante!);

Entrada de Dados

```
int mat;  
float media;  
printf("Entre com matrícula e media:");  
scanf("%d ", &mat);  
scanf("%f", &media);
```

```
int x,y;  
printf ("Entre com o par coordenado:");  
scanf ("%d", &x);  
scanf ("%d", &y);  
printf ("x=%d y=%d", x, y);
```

Entrada de Dados:

Operador de Endereço '&'

- A lista de argumentos deve consistir nos endereços das variáveis.
- C oferece um operador para tipos básicos chamado operador de endereço e referenciado pelo símbolo "&" que retorna o endereço do operando.

```
main()
{
    int num;
    printf("Digite um número: ");
    scanf("%d",&num);
    printf("\no número é %d",num);
    printf("\no endereço é %u",&num);
}
```

Saída de Dados

printf("expressão de controle",argumentos);

- É uma função de I/O, que permite escrever no dispositivo padrão (tela).
- A expressão de controle pode conter caracteres que serão exibidos na tela e os códigos de formatação que indicam o formato em que os argumentos devem ser impressos.
- Cada argumento deve ser separado por vírgula.

Saída de Dados

■ Códigos de Formatação:

%c	caractere simples
%d	decimal
%e	notação científica
%f	ponto flutuante
%o	octal
%s	cadeia de caracteres
%u	decimal sem sinal
%x	hexadecimal

■ Caracteres de Escape:

\n	nova linha
\t	tab
\b	retrocesso
\"	aspas
\\	barra

■ Exemplo:

```
main()
{
    printf("Este é o numero dois: %d",2);
    printf("%s está a  %d milhões de milhas\ndo sol","Vênus",67);
}
```


Saída de Dados

- Imprimindo caracteres:

```
main(){  
    printf("%d %c %x %o\n",'A','A','A','A');  
    printf("%c %c %c %c\n",'A',65,0x41,0101);  
}
```

- A tabela ASCII possui 256 códigos de 0 a 255;
- Se imprimirmos em formato caractere um número maior que 255, será impresso o resto da divisão do número por 256;
- Se o número for 3393 será impresso A pois o resto de 3393 por 256 é 65.

Saída de Dados: Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Saída de Dados: Tabela ASCII Extendida

128	Ç	144	É	160	á	176	░	192	Ł	208	⌌	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	⌍	209	⌎	225	β	241	±
130	é	146	Æ	162	ó	178	▓	194	⌏	210	⌐	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	⌑	211	⌒	227	π	243	≤
132	ä	148	ö	164	ñ	180	⌔	196	—	212	⌓	228	Σ	244	∫
133	à	149	ò	165	Ñ	181	⌕	197	+	213	⌔	229	σ	245	∫
134	å	150	û	166	²	182	⌖	198	⌕	214	⌕	230	μ	246	÷
135	ç	151	ù	167	°	183	⌗	199	⌖	215	⌕	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	⌘	200	⌗	216	⌕	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	⌙	201	⌘	217	⌕	233	⊖	249	·
138	è	154	Ü	170	⌑	186	⌚	202	⌙	218	⌐	234	Ω	250	·
139	ï	155	÷	171	½	187	⌛	203	⌚	219	■	235	δ	251	√
140	î	156	£	172	¼	188	⌜	204	⌛	220	■	236	∞	252	∞
141	ì	157	¥	173	¡	189	⌝	205	=	221	■	237	φ	253	²
142	Ä	158	£	174	«	190	⌞	206	⌛	222	■	238	ε	254	■
143	Å	159	ƒ	175	»	191	⌟	207	⌞	223	■	239	∩	255	

Source: www.LookupTables.com

Programa Exemplo

```
#include <stdio.h>
int main() {
    int a, b, soma;
    scanf ("%d", &a);
    scanf ("%d", &b);
    soma = a + b;
    if (soma > 5) {
        printf ("%d é maior que 5.", soma);
    }
    else {
        printf ("%d é menor que 5.", soma);
    }
}
```

IDE

Integrated Development Environment

- Ambientes de Desenvolvimento Integrado são softwares ou pacotes de softwares que facilitam a tarefa de programação.
- Geralmente contam com um **editor de texto** (com recursos de ressaltar a sintaxe por meio de cores, identificação de erros, identificação automática, autocompletar, etc.) e um **depurador compilador**.

IDE: DevC++

- IDE livre voltado para a linguagem C/C++ para a plataforma Microsoft Windows.
- Download: <http://www.bloodshed.net/devcpp.html>

