COMANDOS
DA
LINGUAGEM C++
PARA A
MANIPULAÇÃO DE ARQUIVOS

ARQUIVOS

Usados para:

- Armazenar dados fora da memória principal do computador;
- Armazenar dados permanentemente.

ARQUIVOS

- A palavra **stream** é usada para indicar fluxo de bytes. Assim, todo objeto que tem a capacidade de receber ou transferir *bytes* de ou para a memória do computador é chamado de objeto stream.

MANIPULAÇÃO DE ARQUIVOS

Para um arquivo ser criado e manipulado por um programa é necessário a introdução da Biblioteca fstream da linguagem C++.

Forma geral: **#include<fstream>**

MANIPULAÇÃO DE ARQUIVOS

- **o**fstream: **output**, para escrever em arquivos;
- **i**fstream: **input**, para ler de arquivos;
- fstream: para ler e/ou escrever em arquivos.

ARQUIVO DE SAÍDA

CRIANDO UM ARQUIVO DE SAÍDA:

```
std::ofstream nomeDoArquivo;
```

ABRINDO UM ARQUIVO DE SAÍDA:

```
nomeDoArquivo.open(nomeDoArquivoNoDisco);
```

obs.: Arquivo aberto com o padrão de saída do ofstream - *std::ios::out*;

ARQUIVO DE SAÍDA

VERIFICANDO ERRO DE ABERTURA NO ARQUIVO DE SAÍDA:

```
nomeDoArquivo.is_open();
```

ESCREVENDO NO ARQUIVO DE SAÍDA:

```
nomeDoArquivo<< entrada;
```

FECHANDO UM ARQUIVO DE SAÍDA:

```
nomeDoArquivo.close();
```

ARQUIVO DE SAÍDA

VERIFICANDO ERRO DE ABERTURA NO ARQUIVO DE SAÍDA:

nomeDoArquivo.is_open();

Indica que o arquivo foi aberto com sucesso.

Indica que o arquivo não pôde ser aberto.

ARQUIVO DE ENTRADA

CRIANDO UM ARQUIVO DE ENTRADA:

```
std::ifstream nomeDoArquivo;
```

ABRINDO UM ARQUIVO DE ENTRADA:

```
nomeDoArquivo.open(nomeDoArquivoNoDisco);
```

VERIFICANDO FIM DO ARQUIVO DE ENTRADA:

```
nomeDoArquivo.eof();
```

obs.: Arquivo aberto com o padrão de entrada do ofstream - *std::ios::in*;

ARQUIVO DE ENTRADA

VERIFICANDO FIM DO ARQUIVO DE ENTRADA:

nomeDoArquivo.eof();

Indica que o marcador de arquivo está no fim do arquivo.

Não há dados a serem lidos.

Indica que o marcador de arquivo não está no fim do arquivo.

Há dados a serem lidos.

ARQUIVO DE ENTRADA

VERIFICANDO ERRO DE ABERTURA NO ARQUIVO DE ENTRADA:

```
nomeDoArquivo.is_open();
```

LENDO DO ARQUIVO DE ENTRADA:

```
nomeDoArquivo>> entrada;
getline(nomeDoArquivo,entrada);
```

FECHANDO UM ARQUIVO DE ENTRADA:

```
nomeDoArquivo.close();
```

OBSERVAÇÃO 1

O operador de extração de fluxo (>>) pula os caracteres de espaço em branco como espaços, tabulações e nova linha no fluxo de entrada

OBSERVAÇÃO 2

ifstream e **ofstream** possuem modos de abertura de arquivo *default.* Logo, o segundo parâmetro da função open() é **opcional**.

OBSERVAÇÃO 2

ofstream e **ifstream** possuem modos de abertura de arquivo *default.* Logo, o segundo parâmetro da função open() é **opcional**.

ofstream

std::ios::out – Abre um arquivo para **Escrita(S)**. Se o arquivo não existir será criado. Se existir, será regravado, com a perda do conteúdo original.

OBSERVAÇÃO 2

ifstream e **ofstream** possuem modos de abertura de arquivo *default.* Logo, o segundo parâmetro da função open() é **opcional**.

ifstream

std::ios::in – Abre um arquivo para **Leitura(E)**.

A abertura falha caso o arquivo não exista.

ARQUIVO DE ENTRADA/SAÍDA

CRIANDO UM ARQUIVO DE ENTRADA/SAÍDA:

```
std::fstream nomeDoArquivo;
```

ABRINDO UM ARQUIVO DE ENTRADA/SAÍDA:

```
nomeDoArquivo.open(nomeDoArquivoNoDisco, modo de abertura);
```

ARQUIVO DE ENTRADA/SAÍDA

MODOS DE ABERTURA DE UM ARQUIVO

std::ios::in – Abre um arquivo para **Leitura(E)**.

std::ios::out – Abre um arquivo para **Escrita(S)**. Se o arquivo não existir será criado. Se existir, será regravado, com a perda de conteúdo original.

std::ios::app – Abre um arquivo para **Escrita(S)**, para adição de novas linhas no final. Se o arquivo não existir será criado.

ARQUIVO DE ENTRADA/SAÍDA

MODOS DE ABERTURA DE UM ARQUIVO

std::ios::ate – Abre um arquivo para **Escrita(S),** posicionando no seu final. Se o arquivo não existir será criado, se existir, o conteúdo original será perdido.

std::ios::trunc – Elimina o conteúdo de um arquivo. O arquivo fica com tamanho Zero.

std::ios::binary – Abre um arquivo no modo binário.

ARQUIVO DE ENTRADA/SAÍDA

O método open() aceita a inclusão de um segundo argumento indicando o modo de abertura do arquivo. Exemplos:

- **Declaração**:

  fstream nomeDoArquivo;

- **Abrir arquivo só para leitura**:

  nomeDoArquivo.open(nomeDoArquivoNoDisco, **ios::in**);

- **Abrir arquivo só para escrita**:

  nomeDoArquivo.open(nomeDoArquivoNoDisco, **ios::out**);

- **Abrir arquivo para leitura e escrita:**

  nomeDoArquivo.open(nomeDoArquivoNoDisco, **ios::in | ios::out**);

ARQUIVO DE ENTRADA/SAÍDA

VERIFICANDO ERRO DE ABERTURA NO ARQUIVO DE ENTRADA/SAÍDA:

```
nomeDoArquivo.is_open();
```

LENDO DO ARQUIVO DE ENTRADA/SAÍDA:

```
nomeDoArquivo>> entrada;
getline(nomeDoArquivo,entrada);
```

ARQUIVO DE ENTRADA/SAÍDA

ESCREVENDO NO ARQUIVO DE ENTRADA/SAÍDA:

```
nomeDoArquivo<< entrada;
```

FECHANDO UM ARQUIVO DE ENTRADA:

```
nomeDoArquivo.close();
```

MANIPULAÇÃO DE CARACTERES

Manipulação de caracteres

A manipulação dos caracteres depende da tabela ASCII, nela é feita uma vinculação entre os caracteres existentes no teclado e um número decimal.

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]

Manipulação de caracteres

Os caracteres de A a Z maiúsculos são representados na tabela ACSII como:

‘A’ 65

‘B’ 66

‘C’ 67

...

‘Z’ 90

Manipulação de caracteres

Os caracteres de a a z minúsculos são representados na tabela ACSII como:

‘a’ 97

‘b’ 98

‘c’ 99

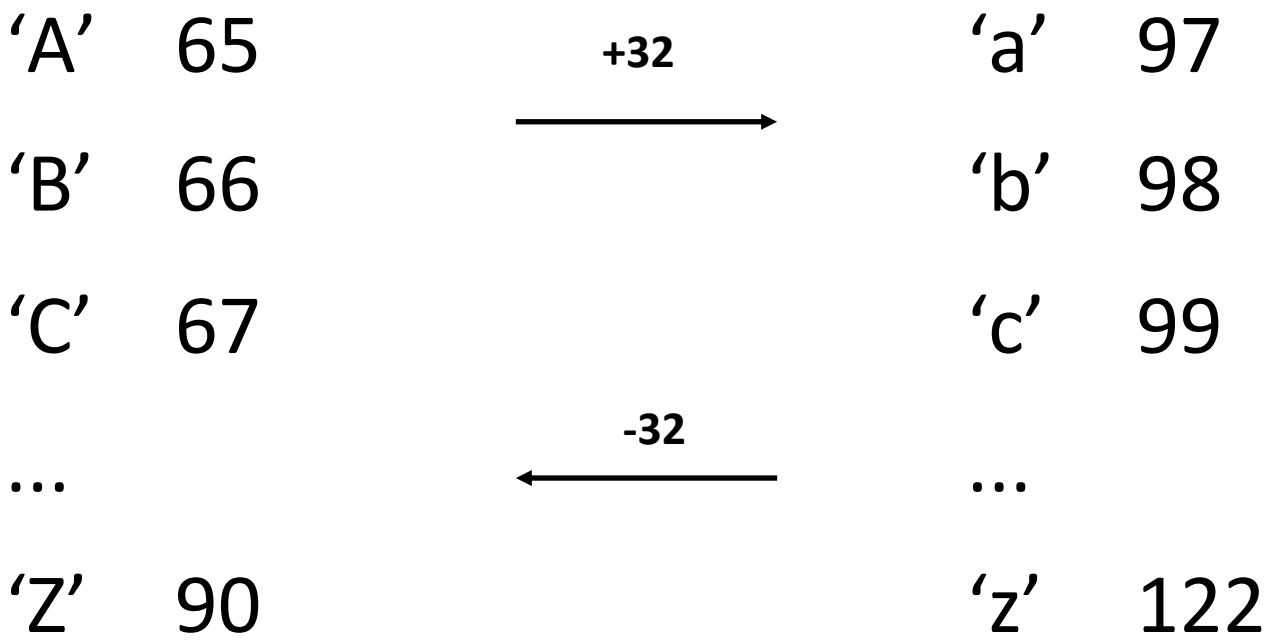
...

‘z’ 122

Manipulação de caracteres

A conversão de uma letra de maiúsculo para minúsculo e de minúsculo para maiúsculo é feita través da adição ou subtração do valor 32.

Manipulação de caracteres



Manipulação de caracteres

Os caracteres de 0 a 9 são representados na tabela ACSII como:

‘0’ 48

‘1’ 49

‘2’ 50

…

‘9’ 57

Exercício

Fazer um projeto, pode ser no modo texto, que utilize arquivos, e que converta os caracteres fornecidos pelo usuário de maiúscula para minúscula, ou vice versa.

t -> T

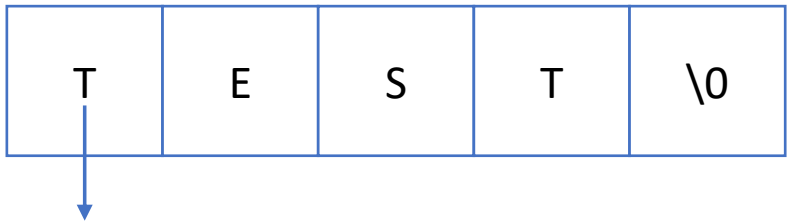
A -> a

cmp -> CMP

nOmE -> NoMe

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]

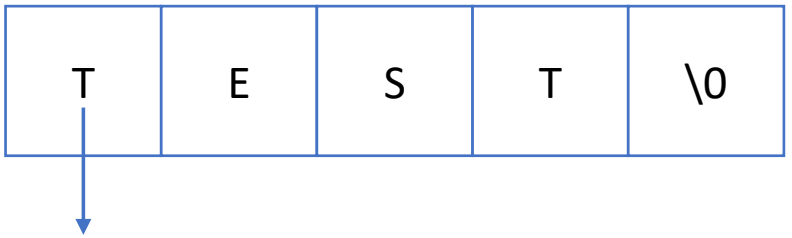
QString



QChar (16 bits)

UNICODE

string



char (8 bits)

ASCII

UNICODE

ASCII

Graphic character symbol	Hexadecimal character value

0020	0 0030	@ 0040	P 0050	` 0060	p 0070	00A0	° 00B0	À 00C0	Ð 00D0	à 00E0	ð 00F0
! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071	¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
" 0022	2 0032	B 0042	R 0052	b 0062	r 0072	¢ 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
# 0023	3 0033	C 0043	S 0053	c 0063	s 0073	£ 00A3	³ 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074	¤ 00A4	´ 00B4	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
% 0025	5 0035	E 0045	U 0055	e 0065	u 0075	¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	õ 00F5
& 0026	6 0036	F 0046	V 0056	f 0066	v 0076	¦ 00A6	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
' 0027	7 0037	G 0047	W 0057	g 0067	w 0077	§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
(0028	8 0038	H 0048	X 0058	h 0068	x 0078	¨ 00A8	¸ 00B8	È 00C8	Ø 00D8	è 00E8	ø 00F8
) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079	© 00A9	¹ 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A	ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B	« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C	¬ 00AC	¼ 00BC	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
- 002D	= 003D	M 004D	] 005D	m 006D	} 007D	00AD	½ 00BD	Í 00CD	Ý 00DD	í 00ED	ý 00FD
. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E	® 00AE	¾ 00BE	Î 00CE	Þ 00DE	î 00EE	þ 00FE
/ 002F	? 003F	O 004F	_ 005F	o 006F	007F	¯ 00AF	¿ 00BF	Ï 00CF	ß 00DF	ï 00EF	ÿ 00FF

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]