Strings: Cadeia de Caracteres

Prof. Rafael Alves Bonfim de Queiroz
rafael.queiroz@ufop.edu.br

Conteúdo

Introdução

- **Strings de texto** são estruturas de dados de importância crescente
- **Mecanismos de busca na Internet**, como o Google, pesquisam bilhões de documentos quase instantaneamente
- O **sequenciamento do genoma humano** nos deu três bilhões de caracteres de texto descrevendo todas as proteínas das quais somos construídos

Códigos de Caracteres

- **Códigos de caracteres** são mapeamentos entre números e os símbolos que compõem um alfabeto específico
- Computadores são fundamentalmente projetados para trabalhar com dados numéricos
- O **ASCII** (*American Standard Code for Information Interchange*) é um código de caractere de byte único, onde $2^7 = 128$ caracteres são especificados
  - Bytes são entidades de oito bits; então isso significa que o bit de ordem mais alta é deixado como zero
- Projetos de código de caracteres internacionais mais modernos, como **Unicode**, usam dois ou até três bytes por símbolo e podem representar praticamente qualquer símbolo em todos os idiomas do planeta

Tabela ASCII

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	”	35	#	36	$	37	%	38	&	39	’
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	/	93	]	94	^	95	_
96	‘	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	—	125	}	126	~	127	DEL

Tabela ASCII

- Os caracteres não imprimíveis têm os três primeiros bits como zero ou todos os sete bits mais baixos como um
- As letras maiúsculas e minúsculas e os dígitos numéricos aparecem sequencialmente
  - Podemos percorrer todas as letras simplesmente fazendo um loop do valor do primeiro símbolo (“a”) para o valor do último símbolo (“z”)
  - Podemos converter um caractere (digamos, “l”) para sua classificação na sequência de agrupamento (oitavo, se “A” for o caractere zero) simplesmente subtraindo do primeiro símbolo (“A”)

Tabela ASCII

- Podemos converter um caractere (digamos "C") de maiúsculo para minúsculo adicionando a diferença do caractere inicial maiúsculo e minúsculo ("C" - "A" + "a")
- Um caractere x é maiúsculo se e somente se estiver entre "A" e "Z"
- Podemos converter um caractere (digamos "c") de minúsculo para maiúsculo adicionando a diferença do caractere inicial maiúsculo e minúsculo ("c" - "a" + "A")

Representando Strings

- Strings são **sequências de caracteres**, onde a ordem claramente importa
- Representação de strings:
  - **Vetores com terminação nula**: $C/C++$ trata strings como vetores de caracteres
    - A string termina no instante em que atinge o caractere nulo ("\0")
    - A vantagem desta representação é que todos os caracteres individuais são acessíveis pelo índice do vetor

Representando Strings

- **Vetor mais comprimento**: a primeira localização do vetor armazena o comprimento da string, evitando assim a necessidade de qualquer caractere nulo final
- **Lista encadeadas de caracteres**: strings de texto podem ser representadas usando listas encadeada
  - Geralmente é evitado devido à alta sobrecarga de espaço associada a um ponteiro de vários bytes para cada caractere de byte único
  - Tal representação pode ser útil se você inserir ou deletar substrings frequentemente dentro do corpo de uma string

Procurando por Padrões

- O algoritmo mais simples para procurar a presença da string padrão p no texto t sobrepõe a string padrão em todas as posições do texto e verifica se cada caractere padrão corresponde ao caractere de texto correspondente

Manipulando Strings

- Manipular strings requer saber exatamente qual a representação de string
- Representação de strings por vetores e suas operações:
  - Calculando o **comprimento de uma string**: percorre os caracteres da string, adicionando um à contagem a cada vez até atingirmos o caractere nulo
  - **Copiando uma string**: a menos que sua linguagem de programação suporte a cópia de vetores de uma só vez, você deve fazer um loop explícito por cada caractere da string
    - Lembre-se de alocar memória suficiente para a nova cópia e não esqueça de colocar o caractere nulo no final

Manipulando Strings

- **Invertendo uma string**: a maneira mais fácil de fazer isso é copiar a string da direita para a esquerda em um segundo vetor
  - A extremidade direita é encontrada calculando o comprimento da string
  - Lembre-se de terminar a nova string com o caractere nulo (null)
  - A reversão de strings também pode ser feita trocando caracteres se for possível alterar o vetor da string original

Funções da Biblioteca de Strings

- Se você trabalha em C, C++ ou Java, esteja ciente do suporte fornecido para caracteres e strings por meio de bibliotecas ou classes
- C contém bibliotecas de caracteres e strings:
  - A biblioteca de caracteres da linguagem C "**ctype.h**"
  - As seguintes funções aparecem na biblioteca de strings da linguagem C: "**string.h**"

Biblioteca de manipulação de caracteres do C: "ctype.h"

```
#include <ctype.h>      /* include the character library */

int isalpha(int c);     /* true if c is either upper or lower case */
int isupper(int c);     /* true if c is upper case */
int islower(int c);     /* true if c is lower case */
int isdigit(int c);     /* true if c is a numerical digit (0-9) */
int ispunct(int c);     /* true if c is a punctuation symbol */
int isxdigit(int c);    /* true if c is a hexadecimal digit (0-9,A-F) */
int isprint(int c);     /* true if c is any printable character */

int toupper(int c);     /* convert c to upper case -- no error checking */
int tolower(int c);     /* convert c to lower case -- no error checking */
```

Biblioteca de manipulação de caracteres do C: "string.h"

```
#include <string.h>      /* include the string library */

char *strcat(char *dst, const char *src);     /* concatenation */
int strcmp(const char *s1, const char *s2);   /* is s1 == s2? */
char *strcpy(char *dst, const char *src);     /* copy src to dist */
size_t strlen(const char *s);                 /* length of string */
char *strstr(const char *s1, const char *s2); /* search for s2 in s1 */
char *strtok(char *s1, const char *s2);       /* iterate words in s1 */
```

C++: string.h

- Além de suportar strings no estilo C/C++ tem uma classe string que contém métodos para essas operações e mais:

```
string::size()          /* string length */
string::empty()         /* is it empty */
string::c_str()         /* return a pointer to a C style string */

string::operator [](size_type i)        /* access the ith character */

string::append(s)       /* append to string */
string::erase(n,m)      /* delete a run of characters */
string::insert(size_type n, const string&s) /* insert string s at n */

string::find(s)
string::rfind(s)        /* search left or right for the given string */

string::first()
string::last()          /* get characters, also there are iterators */
```

Objetos de String em Java

- Strings em Java são objetos de primeira classe derivados da **classe String** ou da **classe StringBuffer**
- A classe String é para strings estáticas que não mudam, enquanto StringBuffer é projetado para strings dinâmicas
- Java foi projetado para suportar **Unicode**, então seus caracteres são entidades de 16 bits
- O **pacote java.text** contém operações mais avançadas em strings, incluindo rotinas para analisar dados e outros textos estruturados