UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

Estrutura de Dados I / Projeto de Algoritmos I

Prof. Denis Rosário
Email: denis@ufpa.br

Agenda

Unidade 2: Cadeia de Caracteres

1. Operações
2. Implementações

Caracteres

- Um caractere é considerado um tipo de dado primitivo na maioria dos computadores
- Um tipo de dado é primitivo se o computador possui instruções em linguagem de máquina que permitem a manipulação deste tipo
- Desde que uma cadeia é uma sequência ordenada de caracteres, o caractere é a entidade fundamental de manipulação de uma cadeia

Caracteres

- Um caractere pertence a um conjunto finito de caracteres: um alfabeto
- Um exemplo de alfabeto é o conjunto de letras utilizado na língua portuguesa
- Um caractere pode ser representado na memória como uma sequência de bits (uma sequência de zeros e uns), sendo que a cada caractere do conjunto é atribuída uma sequência distinta de bits, segundo uma convenção escolhida

Caracteres

- Geralmente, adota-se uma codifcação do conjunto de caracteres em sequência de bits de comprimento fixo
- Uma sequência de bits de tamanho n consegue representar 2^n caracteres
  - Cada caractere do conjunto é representado pelo mesmo número de bits
- Por exemplo, com n=7 podemos codifcar até =128; com n=8, podemos representar até =256

Caracteres

- Caracteres são representados internamente por códigos numéricos
- Tipo char (inteiro “pequeno”)
  - 1 byte (8 bits)
  - 256 caracteres possíveis
- Códigos são associados em uma tabela de códigos
  - Tabela ASCII
  - código EBCDIC (8 bits) - Extended Binary Coded Decimal Interchange
  - Code código UNICODE (8 bits)

Tabela ASCII

- Cada byte armazena um caractere: algarismo, letra, símbolo ou caractere de controle
- Possibilidade de representações diversas (128 caracteres)
  - caracteres decimais numéricos (10)
  - alfabeto inglês em letras minúsculas e maiúsculas (52)
  - caracteres de controle (33)
  - caracteres especiais e de operação (33)

Tabela ASCII

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]

Codificação de Caracteres

- A representação de caracteres utilizando a codifcação ASCII
- A representação do caractere ’a’ é a sequência de bits: 0110 0001

0	1	1	0	0	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- onde o peso de cada dígito binário é dado abaixo daquele dígito
- O valor decimal equivalente da representação para o caractere ’a’ é

$$0\times 2^7 + 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$= 64 + 32 + 1 = 97$$

Representação de Caracteres em C

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main() {
6     char c = 97;
7     cout << " caracter " << c << ", em decimal " << (int) c;
8   }
```

- cout imprime o conteúdo da variável c em dois formatos diferentes:
  - (int) c: imprime o valor do código numérico (97)
  - c : imprime o caracter associado ao código na tabela (‘a’)

Representação de Caracteres em C

- Devemos evitar o uso explícito dos códigos
  - Muda de acordo com a arquitetura usada
  - Em C, usamos “constantes de caractere”
- Constantes de caractere
  - Caractere com aspas simples

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6    char c = 'a';
7    cout << " caracter " << c << ", em decimal " << (int) c;
8  }
```

Representação de Caracteres em C

- Os dígitos são codifcados sequencialmente na tabela ASCII
  - ‘0’ (48), ‘1’ (49), etc...
- O código a seguir verifca se um caracter é um dígito (de 0 a 9)

```
int digito (char c) {
  if ((c>='0') && (c<='9'))
    return 1;
  else
    return 0;
}
```

Representação de Caracteres em C

```
1   #include <iostream>
2
3   using namespace std;
4
5   int digito (char c){
6     if ((c>='0') && (c<='9'))
7       return 1;
8     else
9       return 0;
10  }
11
12  int main() {
13    char c = 'a';
14    int d = digito(c);
15    if (digito(c))
16      cout << "caracter " << c << " é um digito";
17    else
18      cout << "caracter " << c << " não é um digito";
19  }
```

Representação de Caracteres em C

- Para converter de letra minúscula para maiúscula (usando codifcação sequencial)

```
char maiuscula (char c) {
  if (c>='a' && c<='z')
    c = c - 'a' + 'A';
  return c;
}
```

Representação de Caracteres em C

```
1   #include <iostream>
2
3   using namespace std;
4
5   char maiuscula (char c){
6     if (c>='a' && c <= 'z')
7       c = c - 'a' + 'A';
8     return c;
9   }
10  int main() {
11    char c = 'a';
12    cout << "caracter " << c << " em maiuscula " << maiuscula
      (c);
13  }
```

Cadeia de Caracteres (Strings)

- Uma cadeia de caracteres (string em inglês) é uma sequência de caracteres, ou seja, um conjunto de símbolos, que fazem parte do conjunto de caracteres, defnido pelo código ASCII.
- Outra defnição: Uma cadeia de caracteres, mais conhecida como string, é uma sequência de letras e símbolos, onde os símbolos podem ser espaços em branco, dígitos e vários outros como pontos de exclamação e interrogação, símbolos matemáticos, etc.

Cadeia de Caracteres (Strings)

- Todas as funções que manipulam cadeias de caracteres (e a biblioteca padrão de C oferece várias delas) recebem como parâmetro um vetor de char, isto é, um ponteiro para o primeiro elemento do vetor que representa a cadeia, e processam caractere por caractere, até encontrarem o caractere nulo, que sinaliza o final da cadeia.

Cadeia de Caracteres (Strings)

- Representação de cadeias de caracteres
  - Vetor do tipo char, terminando com o caractere nulo '\0'

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main() {
6     char str[4];
7     str[0] = 'o';
8     str[1] = 'l';
9     str[2] = 'a';
10    str[3] = '\0';
11    cout << str;
12  }
```

Cadeia de Caracteres (Strings)

- Outras Representações

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main() {
6     char str2 [] = "ola";
7     cout << str2 << endl;
8
9     char str3 [] = {'o', 'l', 'a', '\0'};
10    cout << str3 << endl;
11  }
```

Cadeia de Caracteres (Strings)

- Exercício:
- Faça um código em C++ que inicie com a String: rio e converta para a String: RIO

Cadeia de Caracteres (Strings)

- Exercíci
- Faça ur ie com a String: ng: RIO

```
1   #include <iostream>
2
3   using namespace std;
4
5   char maiuscula (char c){
6     if (c>='a' && c <= 'z')
7       c = c - 'a' + 'A';
8     return c;
9   }
10
11  int main() {
12    char cidade [] = "rio";
13    cout << "cidade " << cidade << endl;
14
15    cout << "cidade em maiuscula: ";
16    for (int i=0; i<4; i++){
17      cout << maiuscula(cidade[i]);
18    }
19  }
```

Cadeia de Caracteres (Strings)

- Para ilustrar a declaração e inicialização de cadeias de caracteres, consideremos as declarações abaixo:
- char s1[] = "";
- char s2[] = "Rio de Janeiro";
- char s3[81];
- char s4[81] = "Rio";

Cadeia de Caracteres (Strings)

- char s1[] = "";
  - Nestas declarações, a variável s1 armazena uma cadeia de caracteres vazia, representada por um vetor com um único elemento, o caractere '\0' .
- char s2[] = "Rio de Janeiro";
  - A variável s2 representa um vetor com 15 elementos.
- char s3[81];
  - A variável s3 representa uma cadeia de caracteres capaz e representar cadeias com até 80 caracteres, já que foi dimensionada com 81 elementos. Esta variável, no entanto, não foi inicializada e eu conteúdo é desconhecido.
- char s4[81] = "Rio";
  - A variável s4 também foi dimensionada para armazenar cadeias até 80 caracteres, mas seus primeiros quatro elementos foram atribuídos na declaração

Leitura de Strings

- Usando cin
- Com especifcação do formato char a;

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6    char a;
7    cin >> a;
8    cout << "caracter " << a;
9  }
```

Leitura de Strings

- Usando cin
- Com especifcação do formato string;

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6    char a[81];
7    cin >> a;
8    cout << "cidade " << a;
9  }
```

  - Cidades com nomes duplos não seriam capturadas corretamente

Leitura de Strings

- Para capturar nomes compostos:
  - cin.get(nome, 100)
    - lê até o 99º caractere ou <ENTER>
    - garante que o tamanho é sufciente

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6    char city[81];
7    cin.get( city, 81);
8    cout << "cidade " << city;
9  }
```

Leitura de Strings

- Para capturar nomes compostos:
  - A função getline() é capaz de ler o dado de entrada até que uma nova linha seja detectada

```
1   #include <iostream>
2   #include <string>
3
4   using namespace std;
5
6   int main() {
7     string city;
8     getline( cin, city );
9     cout << "cidade " << city;
10  }
```

Funções de Cadeia de Caracteres

- Função que calcula comprimento de uma cadeia de caracteres

```
int comprimento (char *c){
  int n =0;
  for (int i =0; c[i]!='\0'; i++)
    n++;
  return n;
}
```

- Função análoga da biblioteca padrão (string.h): strlen

Fun

```
1   #include <iostream>
2   #include <string.h>
3
4   using namespace std;
5
6   int comprimento (char *c){
7     int n =0;
8     for (int i =0; c[i]!='\0'; i++)
9       n++;
10    return n;
11  }
12
13  int main() {
14    char city[81];
15    cin.get(city, 81);
16    cout << "palavra " << city << " contendo " <<
      comprimento(city) << " caracteres" << endl;
17    cout << "palavra " << city << " contendo " <<
      strlen(city) << " caracteres" << endl;
18  }
```