centro universitário FEI

CCP130 Desenvolvimento de Algoritmos

Prof. Danilo H. Perico

Strings

O que é o tipo *char* ?

- Representa **1 byte** (8 bits)
  - Ou seja: é um valor entre **0** e **255** (2^8)

- Normalmente utilizado para representar caracteres

O que é o tipo *char* ?

- A representação de um caractere é dada através de um número inteiro

- Esse número segue um **padrão conhecido** entre diversos sistemas computacionais:
  - **ASCII** - *American Standard Code for Information Interchange*
  - **UTF** - *Unicode Transformation Format*

Tabela ASCII

- **Tabela ASCII:**
  - 7 bits (números de 0 a 127)
- **Tabela ASCII Estendida**
  - 8 bits
    - Igual a tabela ASCII, porém contém mais caracteres:
      - Além do 0 ao 127,
      - Contém do 128 até o 255 (inclui os caracteres com acentos)

Tabela ASCII

- Afinal, como é a **Tabela ASCII** ?

- Vamos ver!

Tabela ASCII - 7 bits

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

Tabela ASCII

- Como consultar a tabela em código (Linguagem C)?

```c
int i = 97;
printf("%d: %c\n", i, i);
```

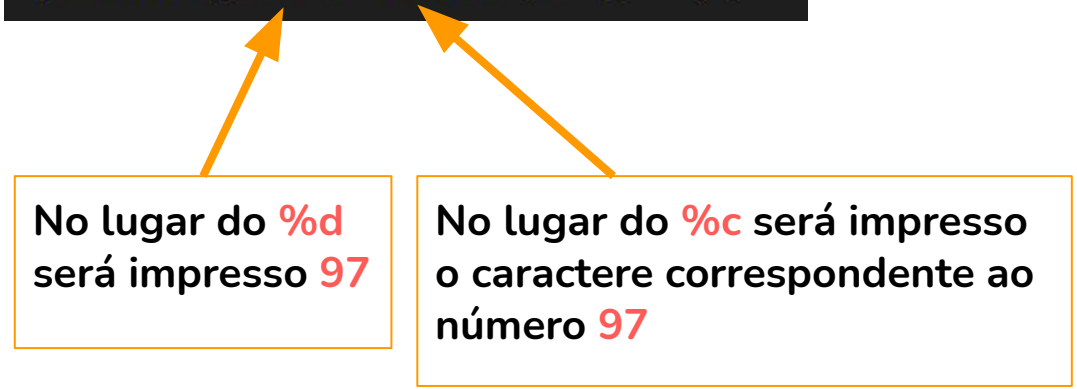
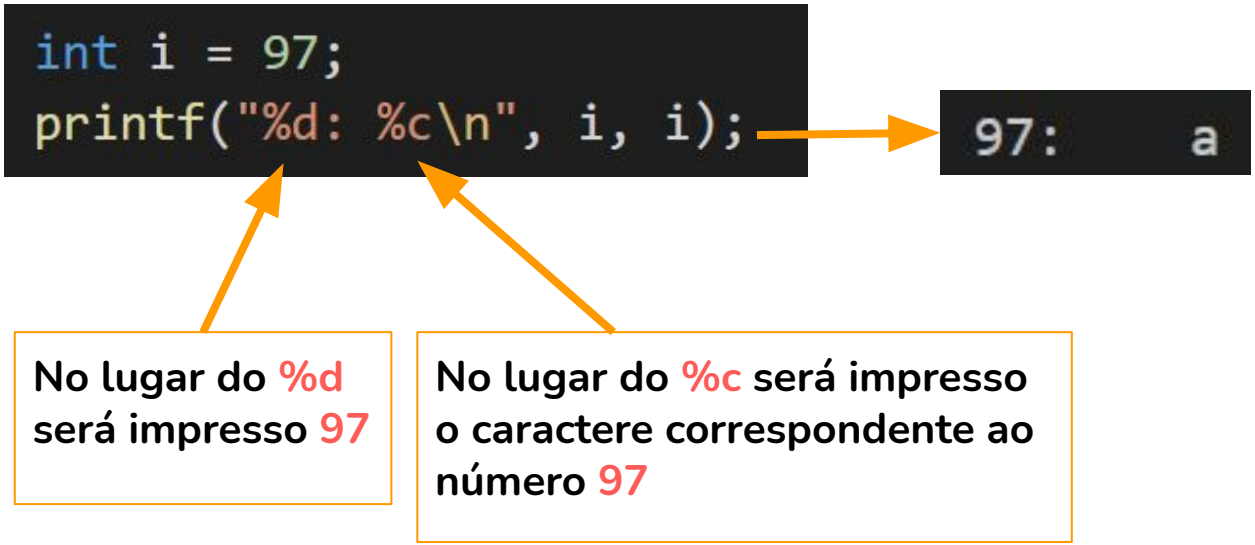


Tabela ASCII

- Como consultar a tabela em código (Linguagem C)?



UTF - Unicode

- Um dos padrões mais utilizados da atualidade é o **UTF-8 (8-bit Unicode Transformation Format)**
- **UTF-8** pode representar qualquer caractere universal padrão do Unicode, sendo também compatível com o **ASCII**
- Tabela Unicode:
  - https://www.utf8-chartable.de/unicode-utf8-table.pl

E o que é *string* ?

- ***string*** é um vetor de ***char*** !

```
// declarando uma string de tamanho 10
char  minha_string[10];
```

- Uma particularidade de ***string*** é que seu último caractere é ***NULO***:
  - Último caractere: **‘\0’** → indica o final de uma ***string***
  - Pode ser chamado de **terminador** da ***string***

char vs. string

- Deve-se ter muita atenção ao saber com qual tipo de dados está se trabalhando

- Em linguagem C:
  - as ***strings*** são representadas por **aspas duplas “ “**
  - enquanto ***char*** usam **aspas simples ‘ ‘**

char vs. string

- Exemplos de ***char***:
  - ‘a’
  - ‘A’
  - ‘!’
  - ‘C’
  - ‘1’
- Exemplos de ***string***:
  - “Uma string”
  - “A”
  - “Linguagem C”
  - “Palavra”
  - “olá mundo!”

char vs. string

- Exemplos de ***char***:
  - ‘a’
  - ‘A’
  - ‘!’
  - ‘C’
  - ‘1’
- Exemplos de ***string***:
  - “Uma string”
  - “A”
  - “Linguagem C”
  - “Palavra”
  - “olá mundo!”

char vs. string

- São diferentes:

‘A’ != “A”

char vs. string

- São diferentes:

‘A’ != “A”

Caractere **A**
ou
Número (ASCII): **65**

Vetor de caracteres:

‘A’	‘\0’
0	1

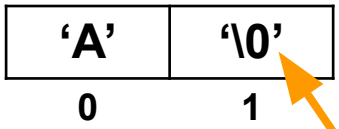
char vs. string

- São diferentes:

‘A’ != “A”

Caractere **A**
ou
Número (ASCII): **65**

Vetor de caracteres:



Caractere nulo: indica o final da string

Strings - impressão

- Para imprimir uma ***string*** em um dispositivo de saída, usa-se a função ***printf()***:
  - argumento de formatação da string: **“%s”**

Strings - impressão

- Exemplos:

```c
// imprime "minha string" na tela
printf("%s", "minha string"); // ou
printf("minha string");

// imprime o conteúdo da variável string nome
printf("%s", nome);

// imprime o conteúdo das variáveis strings dia, mes e ano
printf("%s de %s de %s", dia, mes, ano);
```

strings - impressão

- Para imprimir uma ***string*** podemos usar também a função ***puts()***
- Exemplo:

```
// imprime "minha string" na tela
puts("minha string");
// imprime o conteúdo da variável string nome
puts(nome)
```

strings - impressão

- Diferenças entre ***printf()*** e ***puts()***:
  - Ao final da ***string*** impressa, ***puts()*** adiciona uma mudança automática de linha
  - Não é possível imprimir mais de uma ***string*** utilizando o ***puts***

strings - leitura

- A leitura de ***strings*** pode ser feita por meio dos seguintes comandos:
  - ***scanf()***
  - ***gets()***
  - ***fgets()***

strings - leitura

- Para ler uma ***string*** em um dispositivo de entrada, usa-se a função ***scanf()***:
  - argumento de formatação da string: **“%s”**
- Exemplo:

```
// leitura de valor para a variável nome
scanf("%s", nome);
```

strings - leitura

- Problema do uso do ***scanf()***:
  - O ***scanf*** entende que a ***string*** acaba no primeiro **espaço em branco**
  - Assim, frases não serão lidas por esta função

strings - leitura

- Leitura de *strings* com *gets()*:
- Exemplo:

```
// leitura de valor para a variável nome
gets(nome);
```

strings - leitura

- Problema do uso do ***gets()***:
  - Com o uso da função ***gets***, se uma cadeia maior que o tamanho de criação da ***string*** for digitada, os caracteres excedentes serão transportados para a sequência de memória que não pertence à ***string***, o que pode gerar comportamentos inesperados do programa.

strings - leitura

- Leitura de *strings* com *fgets()*:
- Exemplo:

```
// leitura de valor para a variável nome
fgets(nome, 10, stdin);
```

strings - leitura

- Problema do uso do ***fgets()***:
  - Com o uso da função ***fgets***, se uma ***string*** de tamanho maior que o segundo argumento for digitada, apenas aquela quantidade de caracteres será transferida para a ***string***, sendo os demais desprezados.

strings

- Exemplo de utiização:

```c
#include <stdio.h>

int main(void) {

    char msg[] = {'O', 'l', 'a', ' ',
        'M', 'u', 'n', 'd', 'o', '!', '\n' ,'\0'};

    printf("%s", msg);
    return 0;
}
```

```
PS C:\Users\danil\Documents> .\string1
Ola Mundo!
PS C:\Users\danil\Documents>
```

strings

- Exemplo de utiização:

```c
#include <stdio.h>

int main(void){
    char *str = "declaracao como ponteiro para char";
    printf("%s", str);
   return 0;
}
```

```
PS C:\Users\danil\Documents> .\string2
declaracao como ponteiro para char
PS C:\Users\danil\Documents>
```

string.h - biblioteca

- ***string.h***: é uma biblioteca padrão da linguagem C com objetivo de manipular strings.
- Principais funções:
  - *int strlen (char*);*
    - Retorna o comprimento de uma string sem contar seu terminador (‘\0’)
  - *char * strcpy (char *, char *);*
    - Copia a segunda string na primeira
  - *char * strcat (char *, char*);*
    - Concatena a segunda string na primeira

string.h - biblioteca

- Principais funções:
  - *int strcmp (char *, char *);*
    - Compara strings. Retorna 0, negativo, ou positivo se forem iguais, se a primeira for menor (alfabeticamente) que a segunda ou a primeira for maior (alfabeticamente) que a segunda, respectivamente.
  - *char *strupr (char*);*
    - Converte e retorna a string recebida em maiúsculos
  - *char * strlwr (char *);*
    - Converte e retorna a string recebida em minúsculos.

string.h - biblioteca

- Principais funções:
  - *int strcmp (char *, char *);*
    - Compara strings. Retorna 0, negativo, ou positivo se forem iguais, se a primeira for menor (alfabeticamente) que a segunda ou a primeira for maior (alfabeticamente) que a segunda, respectivamente.
  - *char *strupr (char*);*
    - Converte e retorna a string recebida em maiúsculos
  - *char * strlwr (char *);*
    - Converte e retorna a string recebida em minúsculos.

Não funcionam em todos os lugares.

No Windows, com o gcc funciona

string.h - biblioteca

- Principais funções de **manipulação**:
  - *int sprintf (char * str, const char * format, ...);*
    - Imprime o mesmo texto que seria impresso com printf porém na string str. Retorna o número total de caracteres escritos em str.
  - *int sscanf (const char * str, const char * format, ...);*
    - Lê a string str assim como o scanf faria com a entrada padrão. Retorna o número de items que foram lidos.

strings

- E se eu quiser fazer um vetor de ***strings***?
- Posso fazer uma matriz de ***char***?
- Como?

strings

- E se eu quiser fazer um vetor de ***strings***?
- Posso fazer uma matriz de ***char***?
- Como?

```
char meses[12][10];
strcpy(meses[0], "Janeiro");
strcpy(meses[1], "Fevereiro");
...
strcpy(meses[11], "Dezembro");
```

strings

- E se eu quiser fazer um vetor de ***strings***?
- Há outra maneira:
  - Vetores de ponteiros para ***char***

strings

- E se eu quiser fazer um vetor de ***strings***?
- Há outra maneira:
  - Vetores de ponteiros para ***char***

```
char *strings[12];
strings[0] = "Janeiro";
strings[1] = "Fevereiro";
...
strings[11] = "Dezembro";
```

Exercícios

Exercícios

1. Escreva um programa que recebe quatro strings (*sem utilizar o scanf()*) que representam números inteiros, converte as strings em inteiros, soma os valores e imprime o total dos quatro valores.
2. Escreva um programa que recebe quatro strings (*sem utilizar o scanf()*) que representam números reais (float), converte as strings em reais (float), soma os valores e imprime o total dos quatro valores.
3. Escreva um programa que recebe uma linha de texto no vetor de char *s[100]*. Produza a linha em letras maiúsculas e em letras minúsculas.