UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Fundamentos de Programação

CP41F

Funções de entrada e saída pelo console.

Aula 5

Prof. Daniel Cavalcanti Jeronymo

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação – 1º Período

Plano de Aula

- Ponteiros para dispositivos
- Sequências de escape e identificadores de formato
- Funções de saída
  - printf
  - puts
  - putchar
- Funções de entrada
  - scanf
  - gets
  - getchar

Ponteiros para dispositivos

- Herança do Unix – no C todo dispositivo é um arquivo:

Standard File	File Pointer	Device
Standard input	stdin	Keyboard
Standard output	stdout	Screen
Standard error	stderr	Your screen

- As funções de entrada e saída utilizam os ponteiros stdin e stdout
- Algumas funções permitem especificar o ponteiro

Sequências de escape e identificadores de formato

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\ooo	Octal number of one to three digits
\xhh . . .	Hexadecimal number of one or more digits

Sequências de escape e identificadores de formato

```
%[flags][width][.precision][length]specifier
```

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

http://www.cplusplus.com/reference/cstdio/printf/

Saída

- int putchar(int c);
- Retorno: o caractere escrito como um inteiro ou EOF em caso de erro

- int puts(const char *s);
- Retorno: um número não-negativo em caso de sucesso ou EOF em caso de erro

Saída

- int printf(const char *format, ...);

- Retorno: número de caracteres escritos ou, no caso de erro, um valor negativo

Entrada

- int getchar(void);
- Retorno: caractere obtido ou EOF
- A variável para retorno deve ser um inteiro!
- Equivalente a getc(stdin)

- char *gets(char *s);
- Retorno: s no caso de sucesso, nulo em caso de EOF ou erro

Entrada

- int scanf(const char *format, ...);

- Retorno: número de termos lidos ou EOF no caso de erro

scanf - problemas

- **Misturando scanf com gets. Qual o problema do código abaixo?**

```
int n = 0;
char str[80] = {0};

printf("entre com um número: ");
scanf("%d", &n);
printf("entre com uma string: ");
gets(str);
printf("você digitou %d e \"%s\"\n", n, str);
```

scanf - problemas

- **Agora teste o seguinte código e veja o que acontece:**

```
int c=0;
printf("Primeira linha\n");
scanf("%d",&c);
while((c = getchar()) != '\n' && c != EOF);
printf("Segunda linha %X\n", c);
```

scanf - problemas

- **Moral da história: não misture scanf com outras funções de entrada!**
  Leia:

  http://c-faq.com/stdio/scanfinterlace.html
  http://c-faq.com/stdio/gets_flush2.html

- **Alternativas:**

  - **Toda chamada para scanf deve acompanhar uma linha para limpar o buffer de entrada:**

    ```
    while((c = getchar()) != '\n' && c != EOF);
    ```

    Leia: http://c-faq.com/stdio/getcharc.html

  - **Considere usar fgets + sscanf. Porque?**

  - **NÃO USE fflush(stdin)! Porque?**