Entrada e saída padrão

Printf

A função "printf", definida no arquivo de cabeçalhos "stdio.h", permite escrever no arquivo de saída padrão, que normalmente é a tela do terminal.

Sua sintaxe é:

```
int printf (formato, argumento, argumento, ...) ;
```

O valor de retorno (que pode ser ignorado) indica o número de bytes que foram escritos na saída.

O parâmetro "formato" define o que será impresso, e é uma *string* com caracteres fixos e campos reservados para a impressão dos demais argumentos da chamada. Por exemplo, a linha abaixo escreve o valor de duas variáveis inteiras "i" e "j" em decimal:

```
printf ("i vale %d e j vale %d\n", i, j) ;
```

A saída será:

```
i vale 10 e j vale -37
```

O primeiro %d corresponde à primeira variável (da esq. para a dir.) e o segundo %d à segunda.

O formato "%d" indica que o argumento deve ser escrito como um decimal. Há dezenas de formatos disponíveis, sendo os principais:

form	**função**
\n	nova linha
\t	tabulação
\b	retorno
\"	aspas
\\	barra
%%	porcento %
%c	caractere simples
%s	string
%d	decimal
%i	decimal
%u	decimal sem sinal (unsigned)
%l	decimal (long)
%ll	decimal (long long)
%Nd	decimal com N dígitos
%0Nd	decimal com N dígitos, com zeros à esquerda
%f	real
%g	real (double)
%e	real (em notação científica)
%M.Nf	real com M dígitos, sendo N após a vírgula
%o	octal

form	função
%x	hexadecimal

A *string* abaixo causa a impressão de um inteiro sem formatação (%d), de um inteiro num campo de largura 6 (%6d), de um inteiro representado em hexadecimal num campo de largura 8 (%08d) e completado com zeros (0x44 seria representado como 0x00000044), e de um inteiro num campo de largura 10 (%-10d) e alinhado à esquerda. Cada número está cercado' por *underscores* para facilitar a visualização.

```
"_%d_  _%6d_  _%08x_  _%-10d_\n"
```

Uma descrição detalhada do formato usado no comando "printf" pode ser obtida nesta página ou nesta.

Scanf

A função "scanf" permite ler dados da entrada padrão (normalmente o teclado do terminal). Ela opera de forma similar à função "printf", usando uma string de formato para indicar os tipos dos dados a serem lidos.

Por exemplo, a operação abaixo permite ler um valor inteiro e depositá-lo na variável "a":

```
  scanf ("%d", &a) ;
```

Deve-se observar que o segundo argumento da função não é a variável "a", mas o endereço da variável "a" (denotado por "&a"). Isso é necessário porque nas funções em C a passagem de parâmetros é feita por valor. A função "scanf" precisa saber "onde fica" a variável "a" para poder escrever nela o valor lido, por isso é necessário informar o endereço de "a".

Exemplo de uso

O programa abaixo lê dois inteiros do teclado, calcula a sua média aritmética, e a imprime na tela. Os dois valores de entrada são lidos com dois "scanf" e armazenados nas variáveis "a" e "b".

O exemplo abaixo contém um pequeno problema que será enfrentado e discutido em uma próxima aula. A primeira invocação de "scanf" lê do teclado um inteiro *mas **não** lê o caractere '\n' (ENTER) após o número*. Este caractere permanece na fila do teclado e atrapalha na leitura do segundo número. A solução *ad hoc* é colocar um espaço em branco antes do **%d** do segundo "scanf"; este espaço força a leitura do '\n', e então a leitura do segundo número.

Stackoverflow indica esta solução em duas perguntas/respostas.

```
// Cálculo da média simples de dois valores
#include <stdio.h>

int main ()
{
  int a, b;
  float c;

  printf("entre com o valor de a: ");
  scanf("%d", &a);

  printf("entre com o valor de b: ");
  scanf(" %d", &b);                  // note o espaço ANTES do %d

  // cálculo da média simples (ERRADO)
  c = (a + b) / 2;
```

```
  printf("a = %d, b = %d\n", a, b);
  printf("a media simples de a e b eh %f\n", c);

  return (0);
}
```

Por default, o compilador C avalia expressões matemáticas usando aritmética inteira, *salvo se* houverem valores reais envolvidos.

No código acima, a expressão "(a + b) / 2" somente envolve valores inteiros, por isso sua avaliação será feita com aritmética inteira, gerando resultado incorreto em algumas situações. O código abaixo corrige esse erro ao colocar um valor real na fórmula e assim forçar o compilador da usar aritmética real.

```
// Cálculo da média simples de dois valores
#include <stdio.h>

int main ()
{
  int a, b;
  float c;

  printf("entre com o valor de a: ");
  scanf("%d", &a);

  printf("entre com o valor de b: ");
  scanf(" %d", &b);                    // note o espaço ANTES do %d

  // cálculo da média simples (CERTO)
  c = (a + b) / 2.0;                   // usa um float no divisor

  printf("a = %d, b = %d\n", a, b);
  printf("a media simples de a e b eh %f\n", c);

  return (0);
}
```

Entrada/saída de caracteres

Algumas funções estão disponíveis para a leitura e escrita de caracteres isolados. A função "getchar" lê um caractere da entrada padrão (normalmente o teclado):

```
  c = getchar () ;
```

Caso a entrada não tenha mais dados a serem lidos, essa função devolve um valor específico "EOF" (*end-of-file*), para indicar que a entrada encerrou.

A função "putchar" escreve um caractere na saída padrão (normalmente a tela do terminal):

```
  putchar ('x') ;
```

Entrada/saída de strings

Algumas funções estão disponíveis para a leitura e escrita de strings:

- "gets (char *s)" : lê uma string da entrada padrão (problema de segurança)
- "puts (char *s)" : escreve uma string na saída padrão

A função "gets()" é problemática porque o usuário pode escrever uma sequência de caracteres maior do

que o espaço reservado pelo programador, assim sobre escrevendo outras variáveis do programa.

Exercícios

- Escreva um programa para imprimir a seguinte sequência de números, até N=100, no formato indicado abaixo.

```
    1     2     3     4     5     6     7     8
00001 00002 00003 00004 00005 00006 00007 00008
    9    10    11    12    13    14    15    16
00009 00010 00011 00012 00013 00014 00015 00016
...
```

- Escreva um programa em C que lê um inteiro N e uma sequência de N inteiros. O programa deve gerar uma saída contendo N, os valores máximo e mínimo observados (inteiros), e a média (float) dos valores lidos.

- Escreva um programa que leia um texto da entrada padrão e produza o mesmo texto na saída padrão, mas com as letras convertidas em maiúsculas. Sugestão: use a função "getchar()" para ler caracteres da entrada (até encontrar um "EOF"), a função "putchar()" para escrever caracteres na saída e código para converter os caracteres.

  Seu programa, chamado **toUpper** pode ser testado com a linha de comando

  ```
  cat texto | toUpper
  ```

  sendo "texto" um arquivo com texto, que pode ser o próprio código fonte do programa.

  Pistas: o caractere 'a' é codificado como 0x61 enquanto que o caractere 'A' é codificado como 0x41. Diga "man ascii". O trecho abaixo imprime na tela o que é digitado no teclado, e pode ser chamado de "eco".

```
#define EOF -1

char c;

while ( (c = getchar()) != EOF )
{
   // altere o caractere recém-lido aqui
   putchar(c);
}
```

- Escreva um programa que imprima as raízes quadradas e os logaritmos (base 10) de todos os números inteiros entre 1 e 1000. Os valores devem ser impressos com 4 casas decimais, da forma indicada. Além de incluir a biblioteca de E/S padrão (**#include <stdio.h>**), seu programa deve incluir o arquivo de cabeçalho da biblioteca de matemática (**#include <math.h>**), e ser compilado com o comando abaixo. O **-lm** indica ao GCC que a biblioteca de matemática (**libmath.so**) deve ser ligada ao seu programa.

```
gcc -Wall tab.c -lm

saída:

    1   1.0000  0.0000
  ...
  500  22.3606  2.6989
  ...
  999  31.6069  2.9999
 1000  31.6227  3.0000
```