Entrada/saída padrão

Este módulo apresenta as funções básicas para entrada e saída de dados em programas C usando os arquivos de entrada e saída padrão, `stdin` e `stdout`.

A função printf

A função `printf`, definida no arquivo `stdio.h`, permite escrever no arquivo de saída padrão (normalmente a tela do terminal).

Sua sintaxe básica é:

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

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

O parâmetro `formato` define o que será impresso. É basicamente uma string com caracteres fixos e campos reservados para a impressão dos demais argumentos da chamada. Por exemplo, a linha abaixo irá escrever 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 formato `%d` indica que o argumento deve ser escrito como um decimal. Há dezenas de formatos disponíveis, sendo os principais:

`\n`	nova linha
`\t`	tabulação
`\b`	retorno
`\“`	aspas
`\\`	barra
`%%`	porcento %
`%c`	caractere simples
`%s`	string
`%d` ou `%i`	decimal
`%u`	decimal sem sinal
`%l`	decimal (long)
`%ll`	decimal (long long)
`%Nd`	decimal com N dígitos, com espaços à esquerda
`%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
`%x`	hexadecimal
`%p`	ponteiro (endereço)

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

A função 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 e formato dos dados a serem lidos.

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

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

O segundo argumento da função não é a variável a, mas **o endereço** da variável a (descrito como &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 depositar nela o valor lido, por isso é necessário informar o endereço de a.

Eis um exemplo de uso:

media.c

```c
// 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);

  // cálculo da média simples
  c = (a + b) / 2.0 ;

  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') ;
```

Exemplo de uso:

chars.c

```c
#include <stdio.h>

int main ()
{
  char c ;

  c = getchar() ;       // lê um caractere de STDIN
  while (c != EOF)      // enquanto a entrada não terminar
  {
    if (c == ' ')       // troca espaço por '_'
      c = '_' ;

    putchar (c) ;       // escreve o caractere em STDOUT

    c = getchar () ;    // lê o próximo caractere
  }
  return (0) ;
}
```

Entrada/saída de strings

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

- `gets (char *s)` : lê uma string da entrada padrão (**não usar**: tem problemas de segurança)
- `puts (char *s)` : escreve uma string na saída padrão

A função `gets` não limita o número de bytes lidos e pode provocar **estouro de buffer**, por isso não deve ser usada! Use a função `fgets` ao invés dela.

Exercícios

Escrever programas em C para:

1. Ler um inteiro N e uma sequência de N inteiros, gerando na saída o valor de N, os valores máximo e mínimo observados (inteiros), a média (real) e o desvio padrão (real) dos N valores lidos.
2. Imprimir a seguinte sequência de números, até N=10000:

```
    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
...
```

3. Ler um texto da entrada padrão e produzir o mesmo texto na saída padrão, mas com as letras convertidas em maiúsculas. Sugestão: usar a função `getchar()` para ler caracteres da entrada (até encontrar um EOF), a função `putchar()` para escrever caracteres na saída e uma estrutura `switch` (ou uma tabela) para converter os caracteres.
   Sugestão: em C, caracteres são tratados como números inteiros; consulte uma tabela ASCII para ver seus valores respectivos.
4. 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 seguinte forma:

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