C

Guilherme P. Telles

IC

22 de março de 2023

Entrada e saída

- Em C, operações de entrada e saída são implementadas como macros e funções de bibliotecas.
- A biblioteca `stdio` possui as funções e macros de entrada e saída.

stdio

- As funções escrevem e lêem de *streams*.
  - Uma stream representa uma sequência de objetos (bytes, registros etc.) que podem ser acessados sequencialmente.
- Uma stream é uma variável associada com um arquivo e do tipo `FILE*`.
- Quando um programa é iniciado, três streams são definidas e abertas automaticamente: `stdin`, `stdout` e `stderr`.
- Estas streams não estão associadas com arquivos, mas com o teclado e com o monitor.

Saída formatada

- `int printf(const char *fmt, ...)`

  Imprime os parâmetros na lista `...` em stdout usando a cadeia de formato `fmt` para determinar a forma como cada um será impresso.

  Retorna o número de caracters impressos ou um número negativo se houver erro.

Cadeia de formato

- A cadeia de formato é composta de
  - zero ou mais caracteres comuns que serão copiados na saída e
  - *especificações de conversão* que se aplicam a zero ou mais parâmetros na lista.

Especificação de conversão

- Cada especificação de conversão tem a forma

```
%[parâmetro][flags][largura][.precisão][tamanho]conversão
```

`%[parâmetro][flags][largura][.precisão][tamanho]conversão`

Espec.	Parâmetro	Conversão
d,i	int	inteiro decimal com sinal
u	unsigned	decimal sem sinal
o	unsigned	octal sem sinal
x,X	unsigned	hexadecimal sem sinal
f,F	float	arredondamento e notação [-]ddd.ddd
e,E	float	arredondamento e notação científica
g,G	float	e se o expoente é menor que -4 ou maior que a precisão, senão f
a,A	float	formato hexadecimal
c	char	unsigned char
s	const char*	nenhuma
p	void*	hexadecimal
n	int*	o número de caracteres impressos até aquele ponto é armazenado na variável

exm-printf-1.c

```c
#include <stdio.h>

int main() {

  int i = -10;
  unsigned u = 13;
  char c = 119;
  float f = 3.1416;

  printf("Uma pequena frase\n");

  printf("Um inteiro com valor %d\n", i);

  printf("Dois inteiros com valores %d e %u\n", i, u);

  printf("Um char com valor %d e simbolo %c\n", c, c);

  printf("Um fracionario com valor %f\n", f);
}
```

- O tamanho especifica modificações da representação, p.ex.:

char	%c, %hhi
unsigned char	%c, %hhu
short	%hi
unsigned short	%hu
int	%i, %d
unsigned	%u
long	%li
unsigned long	%lu
long long	%lli
unsigned long long	%llu
float	%f, %F, %g, %G, %e, %E, %a, %A
double	%lf, %lF, %lg, %lG, %le, %lE, %la, %lA
long double	%Lf, %LF, %Lg, %LG, %Le, %LE, %La, %LA
size_t	%zu

exm-printf-2.c

```c
#include <stdio.h>

int main() {

  long l = 50;
  double d = 3.1416;

  printf("Um inteiro com valor %ld.\n", l);
  printf("Um fracionario com valor %lf.\n", d);
}
```

`%[parâmetro][flags][largura][.precisão][tamanho]conversão`

- Especifica o número mínimo de dígitos que deve ser mostrado nas conversões `d i o u x X`.
- Especifica o número mínimo de dígitos à direita da vírgula nas conversões `a A e E f F`.
- Especifica o número máximo de dígitos significativos nas conversões `g G`.
- Especifica o número máximo de caracteres impressos nas conversões `s S`.

```c
#include <stdio.h>

int main() {

  int i = 7;
  printf("%.1d \n", i);
  printf("%.2d \n", i);
  printf("%.3d \n\n", i);

  i = -7;
  printf("%.1d \n", i);
  printf("%.2d \n", i);
  printf("%.3d \n\n", i);

  float f = 3.1416;
  printf("%.1f \n", f);
  printf("%.2f \n", f);
  printf("%.3f \n\n", f);

  char s[] = "abcdef";
  printf("%.1s \n", s);
  printf("%.2s \n", s);
  printf("%.3s \n", s);
}
```

`%[parâmetro][flags][largura][.precisão][tamanho]conversão`

- Especifica largura mínima do campo onde o valor vai ser impresso.
- O valor impresso será alinhado à direita ou à esquerda e o campo será preenchido com espaços ou zeros.
- Se o campo for menor que o valor a ser impresso, a largura é ignorada.

```c
#include <stdio.h>

int main() {

  int i = 13;
  printf("%1d \n", i);
  printf("%6d \n", i);
  printf("%12d \n\n", i);

  i = -13;
  printf("%1d \n", i);
  printf("%6d \n", i);
  printf("%12d \n\n", i);

  float f = 3.1416;
  printf("%1f \n", f);
  printf("%6f \n", f);
  printf("%12f \n\n", f);

  char s[] = "abcdef";
  printf("%1s \n", s);
  printf("%6s \n", s);
  printf("%12s \n", s);
}
```

exm-printf-5.c

```c
#include <stdio.h>

int main() {

  printf("%7.4d \n", 13);

  printf("%10.5d \n", -13);

  printf("%12.2f \n", 3.1416);

  printf("%7.3s \n", "abcdef");
}
```

%[parâmetro][flags][largura][.precisão][tamanho]conversão

- A largura do campo e a precisão podem ser especificadas pelos próximos parâmetros para printf, usando `*`.
- Também podem ser especificadas pelo parâmetro de ordem `m`, usando `*m$`.

exm-printf-6.c

```c
#include <stdio.h>

int main() {

  int i = 100;
  float f = 3.1416;

  int largura = 7, precisao = 3;

  printf("%*d \n", largura, i);
  printf("%*.*f \n", largura, precisao, f);
}
```

`%[parâmetro][flags][largura][.precisão][tamanho]conversão`

- Alteram o formato do campo.

0	Usa zeros ao invés de espaços para alinhar à direita
-	Alinha à esquerda
+	Sempre acrescenta um sinal + ou -
espaço	Acrescenta um espaço antes de número positivo
#	Forma alternativa para g,G,f,F,e,E,o,x,X

- Mais detalhes no manual.

exm-printf-7.c

```c
#include <stdio.h>

int main() {

  int i = 51;

  printf("0: %010d \n",i);
  printf("-: %-10d \n",i);
  printf("+: %+10d \n",i);

  return 0;
}
```

`%[parâmetro][flags][largura][.precisão][tamanho]conversão`

- Tem a forma `n$` indicando que o n-ésimo parâmetro deve ser usado na próxima conversão.
- Permite o reuso de um parâmetro múltiplas vezes e em ordem arbitrária.
- Se `$` for usado, ele deve ser usado em todas as conversões.

exm-printf-8.c

```c
#include <stdio.h>

int main() {

  printf("%2$d %2$c, %1$d %1$c \n", 90, 65);

  return 0;
}
```

Entrada formatada

- `int scanf(const char *fmt, ...)`

  Lê valores de `stdin` usando a cadeia de formato `fmt` para determinar a conversão e armazena nos endereços de memória na lista `...`

- Retorna o número de variáveis convertidas, ou EOF se houver uma condição que impediu qualquer conversão, como por exemplo, o fim da entrada. Atribui valor à variável `errno` em caso de erro.

Cadeia de formato

- A cadeia de formato é composta de
  - *Especificações de conversão* que se aplicam a zero ou mais parâmetros na lista. Opcionalmente podem ter flags.
  - Zero ou mais caracters comuns que são casados com a entrada.
  - Espaço, tabulação e fim-de-linha na cadeia de formato casa com zero ou mais espaços, tabulações e fins-de-linha na entrada.

Especificadores de conversão

	Casa com	Parâmetro
%	%	
d	inteiro decimal com sinal	int*
i	dec., hexa (0x... 0X...) ou octal (0...) com sinal	int*
u	decimal sem sinal	unsigned*
o	octal sem sinal	unsigned*
x, X	hexadecimal sem sinal	unsigned*
e,E,f,g	número em ponto flutuante com sinal	float*
s	uma cadeia sem brancos	char*
c	uma cadeia de tamanho w	char*
[	uma cadeia apenas com os caracteres especificados	char*
[^	uma cadeia sem os cararacteres especificados	char*
p	void*	hexadecimal
n	Nada. Armazena o número de caracteres já lidos.	int*

exm-scanf-1.c

```c
#include <stdio.h>

int main() {

  int x, y;

  scanf("x=%d y=%d", &x, &y);
  printf("lidos: %d %d\n", x, y);
}

// x=20 y=20
// s=10 z=20
```

Cadeia de formato

- Alguns especificadores consomem brancos que precedem o valor, como d, i, f, s.
- Outros não, como c, [.

exm-scanf-2.c

```c
#include <stdio.h>

int main() {

  int x, y;

  scanf("%d %d", &x, &y);
  printf("lidos: %d %d\n", x, y);
}

/*
10

   20    [enter]
*/
```

exm-scanf-3.c

```c
#include <stdio.h>

int main() {

  int x, y, cont;

  scanf("%d %d %n", &x, &y, &cont);
  printf("lidos: %d %d.\n", x, y);
  printf("Foram lidos %d caracteres.\n", cont);

  scanf("%d", &x);
  printf("depois: %d.\n", x);

  return 0;
}

//10  20  30  40
```

Flags

*	O campo deve ser lido mas não deve ser armazenado em alguma variável.
número	Limita o número de dígitos lidos com i ou f, ou o número de símbolos com s.
’	Com números, especifica que há separadores de milhar.
m	Com cadeias, faz com que memória suficiente para a cadeia seja alocada.

exm-scanf-4.c

```c
#include <stdio.h>

int main() {

  int x, y;

  scanf("%d %d %*d ", &x, &y);
  printf("lidos: %d %d.\n", x, y);

  scanf("%d", &x);
  printf("depois: %d.\n", x);
}

//10 20 30 40
```

Flags

- `scanf` também admite conversões com `%n$`.
- Mais detalhes no manual.

Retorno

- Retorna o número de variáveis convertidas, ou EOF se houver uma condição que impediu qualquer conversão, como por exemplo, o fim da entrada. Atribui valor à variável `errno` em caso de erro.

exm-scanf-5.c

```c
#include <stdio.h>
#include <errno.h>

int main() {

  int k;
  int st = scanf("%d", &k);

  if (st == 1)
    printf("read: %d\n", k);
  else {
    if (errno == 0)
      printf("No matching integer\n");
    else
      perror("scanf");
  }
}
```

exm-scanf-6.c

```c
#include <stdio.h>
#include <errno.h>

int main() {

  int k;
  int occ = 0;

  while (scanf("%d ",&k) == 1)
    occ += 1;

  if (errno != 0)
    perror("scanf");

  printf("%d\n",occ);
}
```

Saída não-formatada

- `int putc(int c, FILE* stream)`

  Imprime o inteiro `c` convertido para `unsigned char` na stream indicada.
  Retorna `c` convertido para `unsigned char` ou EOF se houver erro.

- `int putchar(int c)`

  Equivalente a `putc(c,stdout)`.

- `int puts(const char* s)`

  Escreve a cadeia `*s` no monitor, seguida de um `\n`.
  Retorna um número não-negativo se for bem-sucedida ou EOF se houver erro.

exm-put-1.c

```c
#include <stdio.h>

int main() {

  puts("Tres caracteres: ");
  putchar('Z');
  putchar('\n');
  putchar(48);
  puts("Uma pequena frase.\n");
}
```

Entrada não-formatada

- `int getc(FILE *stream)`
  `int fgetc(FILE *stream)`

  Lê o próximo caractere da stream e retorna como um `unsigned int` convertido para `int`.
  Retorna EOF no fim-de-arquivo ou em caso de erro.

- `int getchar(void)`

  Equivalente a `getc(stdin)`.

- `char* fgets(char* s, int n, FILE* stream)`
  Lê no máximo `n`-1 caracteres ou até encontrar `\n` ou EOF e armazena em `s`.
  Se o `\n` for lido, ele é colocado em `s`. Adiciona `\0`.
  Retorna `s` ou `NULL` se houver um erro.

exm-get-1.c

```c
#include <stdio.h>

int main() {

  int i, j;

  i = getchar();
  j = getc(stdin);

  char frase[30];
  fgets(frase,30,stdin);

  putchar(i);
  putchar(' ');
  puts(frase);
}

// Nao funciona bem no terminal.  Redirecinar a entrada funciona.
// exm <in
```