Guia das Principais Funções de Entrada e Saída Padrão da Linguagem C <stdio.h>

De Aluno Para Aluno

Compartilhe! Distribuição Gratuita :)
Link para download-> http://goo.gl/XsgeC

Versão 1.0

12/05/2013

Índice

1. Considerações para um bom entendimento desse guia

- Por mais que esse guia tenha sido escrito com um vocabulário não técnico e com português coloquial, para uma compreensão completa assume-se que o leitor já possua conhecimento no mínimo básico da Linguagem C.

- Se esse é seu primeiro contato com a Linguagem C ou você sinta que é necessário revisar algum assunto da linguagem é extremamente recomendado checar as vídeo aulas(gratuitas) sobre C no YouTube e fazer parte do fórum de programação do De Aluno Para Aluno.

Link para as vídeo aulas -> http://goo.gl/72Cu2
Link para o fórum -> http://goo.gl/CaUnq

- Por comodidade vamos considerar que:
char *c, char c[] e string; São a mesma coisa.
Tecnicamente falando alguém poderia argumentar que em C não existem strings mas sim, vetores de caracteres. E que um ponteiro char não é necessariamente um vetor de caracteres. Mas por comodidade de explicação do conteúdo e bom senso podemos considerar que quando a palavra string for utilizada estamos falando de um vetor de caracteres e vice-versa.

- Por último, você verá que esse guia foi escrito com palavras de diversas cores. Isso não foi uma escolha apenas por motivos estéticos mas funcional. Observe o exemplo abaixo:

```
int umaFuncao(float valor1, float valor2);

A função tem valor1 e valor2 como parâmetros.
```

Perceba que a coloração das palavras valor1 e valor2 é utilizada para referenciar as variáveis na função “umaFuncao”.

2. Vocabulário necessário para entender esse guia:

stream - Fluxo de dados em um sistema computacional(Wikipedia).

stream de input - Fluxo de entrada de dados.

stream de output - Fluxo de saída de dados.

buffer - Espaço temporário na memória para armazenamento de informação.

NULL - Uma constante que contém o valor de 0. Normalmente é utilizado para NÂO referenciar um ponteiro à um endereço de memória no computador.

EOF - END OF FILE (Final do Arquivo). Informa a condição em que dados não podem mais ser lidos de uma fonte específica(Wikipedia).

\n - Caractere "quebra-de-linha" ou "nova linha". Corresponde ao caractere da tecla Enter.

\0 - Caractere nulo. Um caractere de controle com valor 0. Em C é utilizado para informar o final de uma string.

S.O - Sistema Operacional.

Padrões C - C89 / C99 / C11

input padrão - Refere-se a entrada padrão de dados o que para nós, será considerado como sendo o teclado do computador.

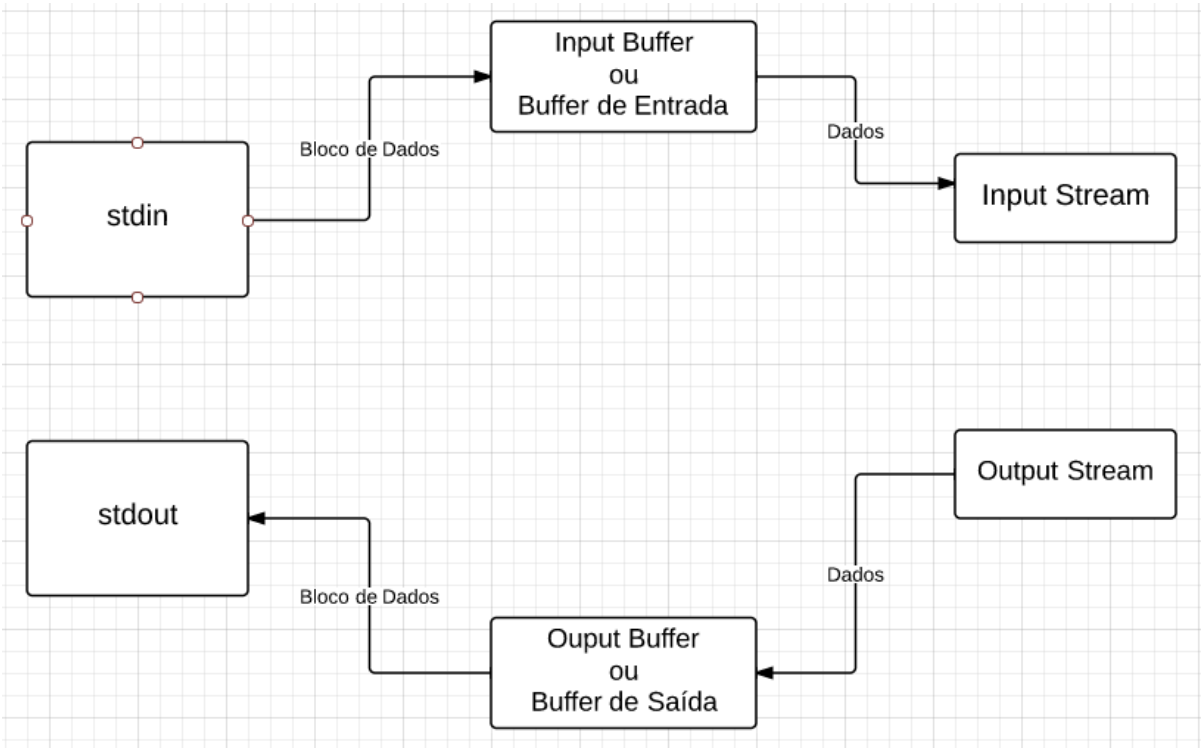
output padrão - Refere-se a saída padrão de dados o que para nós, será considerado como sendo o console no monitor do computador.

stdin - Do inglês, Standard Input, refere-se a entrada padrão de dados o que para nós, será considerado como sendo o teclado do computador.

stdout - Do inglês, Standard Output, refere-se a saída padrão de dados o que para nós, será considerado como sendo o console no monitor do computador.

3. Introdução

Sem mais formalidades, vamos ao que interessa! As funções que você irá aprender nas linhas abaixo tem como objetivo permitir a entrada ou saída de dados.



É importante notar que em C, e em muitas outras linguagens, esse processo de input e output é feito com a ajuda de um buffer(ver imagem acima). O buffer nada mais é que uma região temporária da memória para escritura e leitura de dados.

Se você já utilizou a função scanf() ou fgets() em C deve ter notado que o programa apenas “pega” o que você digitou quando você aperta a tecla Enter. Antes de apertar a tecla Enter os dados informados por você estão sendo armazenados no buffer de entrada.

Agora que você já sabe o que irá aprender vamos dividir as funções apresentadas nesse guia em três categorias:

1- Funções para acesso de arquivos.
2- Funções de Input e Output não formatados.
3- Funções de Input e Output formatados.

Obs. Você NÂO irá ver TODAS as funções de Entrada e Saída padrão da Linguagem C mas as que ao meu ver são essenciais. Expliquei as funções com português simples e nada técnico propositadamente! Se você tem alguma sugestão para melhorar esse guia entre em contato :)

4. Funções para Acesso de Arquivo

4.1. Função fopen

Como é definida em stdio.h?

```
FILE *fopen(const char *restrict filename, const char *restrict mode);
```

O que faz?

A função fopen abre o arquivo cujo nome é a informado pela string filename e associa um stream à ele.

Que parâmetros recebe?

Recebe 2 parâmetros:

filename: Uma string que informa o nome do arquivo ou diretório + nome do arquivo.
mode: Uma string que informa a modalidade de abertura.

obs. Para ver quais valores mode pode assumir faça o download do arquivo "Modo de Abertura fopen" clicando aqui.

O que retorna?

Se sucedido retorna um ponteiro tipo FILE para o objeto controlando o stream do arquivo. Se a operação falhar retorna NULL.

Vídeo que utiliza esta função:

http://goo.gl/CgVAF

4.2. Função freopen

Como é definida em stdio.h?

```
FILE *freopen(const char *restrict filename,
              const char *restrict mode, FILE *restrict stream);
```

O que faz?

A função freopen abre o arquivo cujo nome é informado pelo parâmetro filename e associa o stream apontado por ele à stream.

Que parâmetros recebe?

Recebe 3 parâmetros:

filename: Uma string que informa o nome do arquivo ou diretório + nome do arquivo.
mode: Uma string que informa a modalidade de abertura.
stream: Um ponteiro tipo FILE.

obs. Para ver quais valores mode pode assumir faça o download do arquivo "Modo de Abertura fopen" clicando aqui.

O que retorna?
Se sucedido retorna o stream do arquivo. Se a operação falhar retorna NULL.

Vídeo que utiliza esta função:
http://goo.gl/5byjP

4.3. Função fclose

Como é definida em stdio.h?
```
int fclose(FILE *stream);
```

O que faz?
Fechar o fluxo de stream. Qualquer dado "preso" no buffer de saída é descartado para o S.O. Qualquer dado "preso" no buffer de entrada é descartado.

Que parâmetros recebe?
Recebe 1 parâmetro:
- stream: Um ponteiro tipo FILE.

O que retorna?
Se sucedido retorna 0. Se a operação falhar retorna EOF.

Vídeo que utiliza esta função:
http://goo.gl/CgVAF

4.4. Função fflush

Como é definida em stdio.h?
```
int fflush(FILE *stream);
```

O que faz?
Sincroniza o stream de output com o próprio stream.

Obs. Se o argumento recebido for do tipo input a função tem comportamento indeterminado.

Que parâmetros recebe?
Recebe 1 parâmetro:
- stream: Um ponteiro tipo FILE de output.

O que retorna?
Se sucedido retorna 0. Se a operação falhar retorna EOF.

5. Funções de Entrada e Saída Não Formatadas

5.1. Funções fgetc e getc()

Como são definidas em stdio.h?

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

O que fazem?

Ambas funções obtém o próximo caractere do fluxo informado por stream.

Que parâmetros recebem?

Ambas funções recebem 1 parâmetro:

stream: Um ponteiro tipo FILE de input.

O que retornam?

Se sucedido retornam o caractere escrito. Se um erro ocorrer retornam EOF. Se e a função alcançar o final do fluxo retorna EOF.

Observação:

AS FUNÇÕES getc e fgetc são exatamente iguais mas getc também pode ser implementada como um macro. Se getc for implementada como uma macro o argumento não devem nunca ser uma variável com efeito colateral.
Não se preocupe tanto com essa particularidade, não é todo dia que você fará um programa que caiará nessa exceção. O importante é ao menos saber que em C getc pode ser uma função ou uma macro.

Vídeo que utiliza esta função:

http://goo.gl/QWDxy

5.2. Funções fputc e putc()

Como são definidas em stdio.h?

```
int fputc(int ch, FILE *stream);
int putc(int ch, FILE *stream);
```

O que fazem?

Ambas funções escrevem um caractere no fluxo informado por stream.

Que parâmetros recebem?

Ambas funções recebem 2 parâmetros:

ch: Um valor int que é o valor do caractere a ser escrito.

stream: Um ponteiro tipo FILE de output.

O que retornam?

Se sucedido retornam o caractere lido. Se um erro ocorrer retornam EOF.

Observação:
AS FUNÇÕES putc e fputc são exatamente iguais mas putc também pode ser implementada como um macro. Se putc for implementada como uma macro os argumentos não devem nunca ser variáveis com efeito colateral.
Não se preocupe tanto com essa particularidade, não é todo dia que você fará um programa que caiará nessa exceção. O importante é ao menos saber que em C putc pode ser uma função ou uma macro.

Vídeo que utiliza esta função:
http://goo.gl/OuWMC

5.3. Função fgets

Como é definida em stdio.h?
```
char *fgets(char *restrict s, int n, FILE *restrict stream);
```

O que faz?
Lê uma string de stream até que a leitura encontre o caractere \n(incluso na leitura) ou a leitura alcance EOF ou a quantidade máxima de caracteres especificada por n-1 seja atingida.

Que parâmetros recebe?
Recebe 3 parâmetros:
- s: Uma string para armazenar os caracteres lidos.
- n: Um valor int que determina a quantidade máxima de caracteres a serem lidos - 1.
- stream: Um ponteiro tipo FILE de input.

O que retorna?
Se sucedido retorna a string lida. Se um erro ocorrer retorna NULL.

Observação:
fgets irá automaticamente incluir o caractere \0 na última posição da string.

Vídeo que utiliza esta função:
http://goo.gl/5byjP

5.4. Função gets(Não recomendável em C99 e inexistente em C11)

Como é definida em stdio.h?
```
char *gets(char *restrict s);
```

O que faz?
Lê uma string de stdio até que a leitura encontre o caractere \n(incluso na leitura).

Que parâmetros recebe?
Recebe 1 parâmetro:
s: Uma string para armazenar os caracteres lidos.

O que retorna?
Se sucedido retorna a string lida. Se um erro ocorrer retorna NULL.

Observação:
Não utilize essa função! Se o seu professor insistir que a função é segura e pode ser usada não tem motivo pra criar encrenca e receber uma nota ruim. Mas saiba que fgets e gets_s(inclusa apenas em C11) são funções que fazem o mesmo serviço com o nível de segurança adequado. Assita o vídeo do link abaixo para entender porque a função é insegura.

Vídeo que utiliza esta função:
http://goo.gl/yxFJw

5.5. Função puts

Como é definida em stdio.h?
```
int puts(const char *s);
```

O que faz?
Escreve a string s em stdout e automaticamente adiciona \n ao final da string escrita.

Que parâmetros recebe?
Recebe 1 parâmetro:
s: Uma string que será escrita em stdout.

O que retorna?
Se sucedido retorna um número positivo. Se um erro ocorrer retorna EOF.

Vídeo que utiliza esta função:
http://goo.gl/Z42f5

5.6. Função fputs

Como é definida em stdio.h?
```
int fputs(const char *restrict s, FILE *restrict stream);
```

O que faz?
Escreve a string s no stream informado e automaticamente adiciona o caractere \n ao final da string escrita.

Que parâmetros recebe?
Recebe 2 parâmetros:
s: Uma string que será escrita no fluxo apontado por stream.
stream: Um ponteiro tipo FILE de output.

O que retorna?
Se sucedido retorna um número positivo. Se um erro ocorrer retorna EOF.

Vídeo que utiliza esta função:
http://goo.gl/Z42f5
* Igual a puts exceto que te permite especificar onde escrever os dados.

5.7.Função getchar

Como é definida em stdio.h?
int getchar(void);

O que faz?
Lê o próximo caractere de stdin.

Que parâmetros recebe?
Não recebe nenhum parâmetro.

O que retorna?
Se sucedido retorna o valor do caractere obtido. Se um erro ocorrer retorna EOF.

Vídeo que utiliza esta função:
http://goo.gl/C3iMQ

5.8. Função putchar

Como é definida em stdio.h?
int putchar(int ch);

O que faz?
Escreve o valor do caractere ch em stdout.

Que parâmetros recebe?
Recebe 1 parâmetro:
ch: Um valor int correspondente ao caractere que será escrito em stdout.

O que retorna?
Se sucedido retorna o valor do caractere escrito. Se um erro ocorrer retorna EOF.

Vídeo que utiliza esta função:
http://goo.gl/C3iMQ

6. Funções de Entrada e Saída Formatadas

6.1. Função scanf

Como é definida em stdio.h?

```
int scanf(const char * restrict format, ...);
```

O que faz?

Lê dados de stdout, sob o controle da string apontada por format que informa a ordem e o modo como os dados são convertidos para dentro do programa.

Que parâmetros recebe?

Recebe 2 parâmetros:

format: Uma string que pode ou não conter um ou mais "Especificadores de Conversão de Entrada".

...: Reticências. Informa que a função recebe uma quantidade variável de parâmetros. Nesta função em específico você deve informar os endereços de memória de variáveis que correspondam aos "Especificadores de Conversão de Entrada".

obs. Para ver quais "Especificadores de Conversão de Input" format pode receber faça o download do arquivo "Especificadores de Conversão de Entrada" clicando aqui.

O que retorna?

O número de argumentos que foram corretamente relacionados à um Especificador de Conversão de Entrada.

6.2. Função fscanf

Como é definida em stdio.h?

```
int fscanf(FILE *restrict stream, const char *restrict format, ...);
```

O que faz?

Lê dados de um stream apontado por stream, sob o controle da string apontada por format que informa a ordem e o modo como os dados são convertidos para dentro do programa.

Que parâmetros recebe?

Recebe 3 parâmetros:

stream: Um ponteiro tipo FILE de input.

format: Uma string que pode ou não conter um ou mais "Especificadores de Conversão de Entrada".

...: Reticências. Informa que a função recebe uma quantidade variável de parâmetros. Nesta função em específico você deve informar os endereços de memória de variáveis que correspondam aos "Especificadores de Conversão de Entrada".

obs. Para ver quais "Especificadores de Conversão de Entrada" format pode receber faça o download do arquivo "Especificadores de Conversão de Entrada" clicando aqui.

O que retorna?
O número de argumentos que foram corretamente relacionados à um Especificador de Conversão de Entrada.

6.3. Função printf

Como é definida em stdio.h?
```
int fprintf(const char *restrict format, ...);
```

O que faz?
Escreve uma string em stdout que pode ou não conter um ou mais parâmetros recebidos por ... convertidos em caracteres de string

Que parâmetros recebe?
Recebe 2 parâmetros:

format: Uma string que pode ou não conter um ou mais "Especificadores de Conversão de Saída".

...: Reticências. Informa que a função recebe uma quantidade variável de parâmetros. Nesta função em específico você deve informar os nomes variáveis ou valores literais que correspondam aos "Especificadores de Conversão de Saída".

obs. Para ver quais "Especificadores de Conversão de Output" pode format pode receber faça o download do arquivo "Especificadores de Conversão de Saída" clicando aqui.

O que retorna?
Um valor int que corresponde ao número de caracteres que foram corretamente escritos em stdout.

6.4. Função fprintf

Como é definida em stdio.h?
```
int fprintf(FILE *restrict stream, const char *restrict format, ...);
```

O que faz?
Escreve uma string em stream que pode ou não conter um ou mais parâmetros recebidos por ... convertidos em caracteres de string

Que parâmetros recebe?

Recebe 3 parâmetros:

stream: Um ponteiro tipo FILE de output.

format: Uma string que pode ou não conter um ou mais "Especificadores de Conversão de Saída".

...: Reticências. Informa que a função recebe uma quantidade variável de parâmetros. Nesta função em específico você deve informar os nomes variáveis ou valores literais que correspondam aos "Especificadores de Conversão de Saída".

obs. Para ver quais "Especificadores de Conversão de Output" pode format pode receber faça o download do arquivo "Especificadores de Conversão de Saída" clicando aqui.

O que retorna?

Um valor int que corresponde ao número de caracteres que foram corretamente escritos em stream.

7. Limpeza de Buffer

Para finalizar te deixo o link para o vídeo "Limpeza de Buffer". Esse é um problema muito comum quando utilizamos as funções apresentadas por esse guia.
No vídeo você irá aprender o que é um "buffer sujo" e possíveis soluções para o problema.

Vídeo:

http://goo.gl/k8oGL

FIM

Espere só mais um minuto!

Para obter a versão mais recente desse guia basta acessar o link http://goo.gl/XsgeC e verificar se a versão que você possui corresponde a versão disponível no site(a versão do guia está escrita na primeira página no canto inferior direito).

Eu torço para que esse guia tenha sido útil e esclarecido qualquer dúvida referente as funções de input e output padrão de C. No entanto se você ainda tem dúvidas cheque nosso fórum de perguntas clicando nesse link-> http://goo.gl/CaUnq

O canal De Aluno Para Aluno também possui vídeo aulas de outros assuntos e outras linguagens de programação. A melhor maneira de acompanhar as vídeo aulas e inscrevendo-se no canal clicando nesse link -> http://goo.gl/olNWt

Se você acha que esse guia pode ser útil para outras pessoas eu peço que você o compartilhe. Essa é uma ótima maneira de retribuir o empenho colocado em criar materiais 100% gratuitos desse tipo.
Para compartilhar esse guia apenas envie o link abaixo para seus amigos -> http://goo.gl/XsgeC

Se você possui um blog, site, canal no YouTube ou qualquer meio de divulgação e gostaria de utilizar partes ou todo o conteúdo escrito nessa guia fique a vontade mas por favor não deixe de adicionar os créditos.

E por último, você pode se conectar com o De Aluno Para Aluno de várias maneiras:

 www.facebook.com/DeAlunoParaAluno

 http://goo.gl/8jlJZ

 www.twitter.com/AlunoParaAluno

 http://goo.gl/rUJzN

Ajude o Canal -> http://goo.gl/fiYBa