

Programação de software básico

Aula 5: Entrada e saída por console e arquivo

Apresentação

Enviar e receber dados por console (teclado e monitor) ou arquivos é essencial para a comunicação (interface) com usuário e o armazenamento de informações básicas de um programa. Mesmo que você já tenha sido apresentado a funções básicas de entrada e saída, como `scanf()` e `printf()`, a possibilidade de uso de várias outras funções facilita a criação de interfaces.

Nesta aula, abordaremos várias funções que permitirão utilizar diferentes modos de entrada e saída, um aprendizado importante para a criação de interfaces de softwares robustas e confiáveis para aplicações de software básico.

Objetivos

- Reconhecer os parâmetros e códigos especiais das funções de entrada e saída por console e arquivo;
- Identificar as funções de entrada e saída para console e arquivos;
- Desenvolver programas em C com entradas e saídas para console e arquivos.

Entrada e saída em C

Quando Dennis Ritchie (1941-2011) desenvolveu a linguagem C, ele não quis dispensar uma característica importante, a compactidade. Para obter uma linguagem compacta, ele deliberadamente não incluiu tudo relacionado a saída e entrada na definição da linguagem. Assim, a linguagem C não contém nenhum código para receber dados do teclado e enviá-lo na tela, por exemplo.

Então, como estamos usando as funções `scanf()` e `printf()` em C? Dennis Ritchie usou as funções de entrada e saída do sistema operacional (SO) as vinculou à linguagem C. Isso significa que as funções `printf()` e `scanf()` funcionarão de acordo com o SO que se estiver usando. O programador não precisa se preocupar com o funcionamento dessas funções.

Várias funções de entrada e saída estão disponíveis na linguagem C e podem ser classificadas em duas grandes categorias:

1

Funções de entrada e saída do console

Essas funções recebem entrada do teclado e as gravam no monitor de vídeo.

2

Funções de entrada e saída de arquivos

Essas funções realizam operações de entrada e saída em disco rígido ou pen-drive.

Funções de entrada e saída pelo console em C

Teclado e tela (monitor de vídeo) juntos são identificados como console. Funções de entrada e saída pelo console podem ser classificadas em:

- Funções de entrada e saída formatadas.
- Funções de entrada e saída não formatadas.

Funções de entrada e saída formatadas em C

As funções `printf()` e `scanf()`, da biblioteca `stdio.h`, estão nessa categoria. Elas oferecem a flexibilidade para receber a entrada em algum formato fixo e fornecer a saída no formato desejado.

Função `printf()`

Essa função tem por finalidade imprimir dados na tela. Isso é feito por meio da sintaxe:

```
printf("expressão de controle", lista de argumentos);
```

Comentário

Na “expressão de controle”, são inseridos todos os caracteres a serem exibidos na tela e/ou códigos de formatação, responsáveis por indicar o formato em que os argumentos devem ser impressos. O argumento deve estar incluído na lista de argumentos, e, caso exista mais de um, eles devem ser separados por vírgula.

Veja um exemplo:

Exemplo

```
printf(“A resposta é %d”, c);
```

Onde c é uma variável do tipo inteiro.

A tabela 1 mostra os códigos de formatação permitidos na linguagem C.

Código de formatação	Descrição
%c	Caracteres simples
%d	Inteiros decimais com sinal
%l	Inteiros decimais com sinal
%e	Notação científica (e minúsculo)
%E	Notação científica (E maiúsculo)
%f	Ponto flutuante decimal
%g	Usa %e ou %f (qual for mais curto)
%G	Usa %E ou %F (qual for mais curto)
%o	Octal sem sinal
%s	Cadeia de caracteres
%u	Inteiros decimais sem sinal
%x	Hexadecimal sem sinal (letras minúsculas)
%X	Hexadecimal sem sinal (letras maiúsculas)
%%	Escreve o símbolo de porcentagem (%)

 Tabela 1– Códigos de formatação na linguagem C.

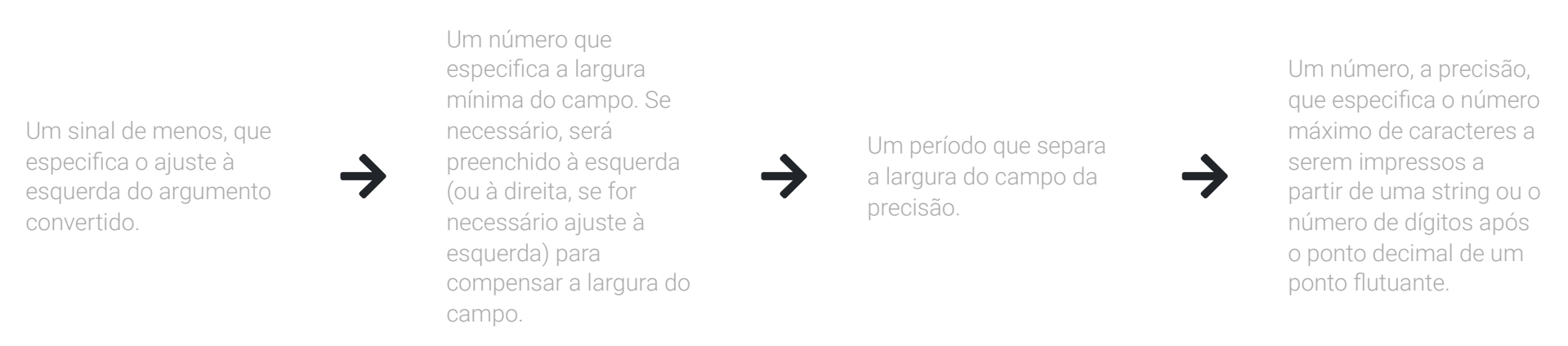
Os códigos especiais (barra invertida) são apresentados na tabela 2. Eles também podem ser inseridos na “expressão de controle”.

Código especial	Descrição
\n	Nova linha
\t	Tab
\b	Retrocesso
\r	Retorno ao início
\"	Aspas
\\	Barra
\f	Salta formulário
\0	Nulo

 Tabela 2 – Códigos especiais.

Especificações de conversão causam a conversão e a impressão do argumento correspondente pela função printf. Cada especificação de conversão começa com um símbolo % e termina com um caractere de conversão.

Entre o% e o caractere de conversão, pode haver poucos números, sinais, pontos etc., e eles ocorrem na seguinte ordem:



Exemplo

```
int num = 12;
int num2 = 12345;
printf("%d\n",num2);//Imprime 12345
printf("%5d\n",num);//Imprime12, com 3 espaços antes do número 12

loatfnum = 0.9388;
printf("%.2f", fnum);//Imprime 0.94
```

Função scanf()

A função scanf permite que se aceitem as entradas do padrão, que geralmente é o teclado.

Exemplo

```
printf("Entre com valor inteiro: ");  
scanf("%d", &a);
```

O programa lerá um valor inteiro que o usuário digitar no teclado (%d é para números inteiros, como em printf; portanto, **a** deve ser declarado como int) e botará esse valor em **a**.

A função scanf usa os mesmos códigos de formatação de printf. Você deve colocar & na frente da variável usada no scanf. Esse operador indica o endereço da variável.

Funções sprintf() e sscanf ()

Essas funções de entrada e saída formatadas funcionam de um modo um pouco diferentes das funções printf() e scanf(). A função sprintf () é bastante semelhante à função printf(), mas, em vez de imprimir a saída na tela, ela é armazenada em um vetor de caracteres.

Segue um exemplo:

Exemplo

```
#include<stdio.h>  
  
intmain()  
{  
int j=32;  
char cha='m';  
float a=123.2;  
char str[20];  
sprintf(str,"%d %c %3.1f",j,cha,a);  
printf("%s",str);  
}
```


No programa do exemplo, str irá armazenar os valores de “j”, “cha” e “a”. Então, é necessário imprimir o valor de str usando printf().

A função sscanf() é o equivalente na função scanf(). Ela permite que o programador armazene os caracteres de string em alguma outra variável.

Funções de entrada e saída pelo console em C

Funções como `getch()`, `getche()` e `getchar()` estão nessa categoria. Essas funções armazenam apenas um caractere. Até agora, usamos a função `scanf()` para armazenar valores, tendo que pressionar a tecla Enter para armazenar os valores na memória. Em uma condição em que temos que armazenar apenas um caractere, as funções não formatadas são úteis.

O arquivo de cabeçalho usado para essas três funções é `conio.h`. Geralmente, essas funções não estão disponíveis em ambiente Unix, pois não fazem parte do padrão American National Standards Institute (ANSI).

 Clique nos botões para ver as informações.

[Função getch\(\).](#)



Essa função é usada para armazenar apenas um caractere na memória. Ela não exibe esse caractere na tela durante a execução do programa.

[Função getche\(\).](#)



Essa função funciona de maneira semelhante à função `getch`. No entanto, ela exibe o caractere na tela.

[Função getchar\(\).](#)



Essa função tem funcionamento muito semelhante à função `getche`. Ela armazena um caractere e o exibe na tela. A diferença é que temos que pressionar a tecla Enter para armazenar um caractere enquanto estivermos usando essa função.

Considere o exemplo abaixo:

```
#include<stdio.h>
#include<conio.h>
int main()
{
char ch1,ch2,ch3;
ch1=getch(); //Não exibe o caractere na tela
ch2=getche(); //Exibe o caractere na tela
ch3=getchar(); //Exibe o caractere na tela e aguarda Enter para armazenar na variável ch3
printf("%c %c %c",ch1,ch2,ch3);
}
```

[Função putchar\(\).](#)



Escreve um caractere na tela. Recebe um inteiro com parâmetro ou um caractere que será convertido.

O programa a seguir imprime 0123456789 na tela.

```
#include <stdio.h>
int main() {
unsigned char c;
for(c = '0'; c <= '9'; c++) {
putchar(c);
}
return 0;
}
```

Trabalhando com strings – funções gets(), puts() e fgets

Função gets()

Lê da entrada-padrão (teclado) até encontrar uma nova linha (Enter) ou o fim de arquivo (EOF). A nova linha não é incluída na string lida. O caractere NULL ('\0') é automaticamente adicionado ao fim da string.

Essa função não é segura! Como não é possível especificar o número máximo de caracteres a serem lidos, é possível ler caracteres além do tamanho da string passada como parâmetro, causando uma falha de segurança conhecida como buffer overflow.

Exemplo

```
#include <stdio.h>
int main() {
    char nome[128];
    puts("Digite o seu nome: ");
    gets(nome); //Inseguro
    printf("Seu nome é %s.\n", nome);
    return 0;
}
```

Função puts()

A função puts imprime na tela uma string especificada, incluindo uma nova linha ('\n') ao fim da impressão.

Exemplo

```
#include <stdio.h>
int main() {
    puts("Olá mundo!"); //Equivalente àprintf("Olá mundo!\n");
    return 0;
}
```

Função fgets()

A função fgets lê uma linha do fluxo especificado e a armazena na string apontada por str. Realiza o procedimento até que os caracteres (n-1) sejam lidos, o caractere de nova linha seja lido ou o fim do arquivo seja alcançado, o que ocorrer primeiro. É normalmente usada com arquivos. A leitura pelo teclado é feita com stdin.

Sintaxe:

```
char * fgets (char * string, int tamanho, FILE * fluxo);
```

Veja que é possível digitar strings com espaço e obter a impressão correta.

Exemplo

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

```
int main() {  
    char string[50];  
    printf("Nome: ");  
    fgets(string, 50, stdin);  
    printf("Seu nome eh %s\n", string);  
    return 0;  
}
```

Função fputs()

Protótipo:

```
int fputs (const char * string, FILE * fluxo);
```

Escreve a cadeia de caracteres string no fluxo. A função fputs não tem a mesma funcionalidade que a função puts. Além de ser possível especificar o fluxo ao utilizar a fputs, ela não adiciona uma nova linha (\n) na saída.

Comentário

A função continua escrevendo até encontrar o indicador de fim da cadeia de caracteres, o caractere NULL ('\0'). O NULL não é incluído no fluxo.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Padrão ANSI C

- 1** No início, C e o SO Unix estavam bem atrelados, pois cada nova implementação de Unix para um tipo de máquina requeria um novo compilador C específico para essa máquina.
- 2** Nos anos 80, a linguagem C tornou-se popular também fora do ambiente Unix. Nessa época, surgiram novos compiladores comerciais de C, e a linguagem passou a ser reconhecida como linguagem de propósito geral.
- 3** Com o desenvolvimento de diversos compiladores, tornou-se necessário padronizar a linguagem, a fim de garantir a compatibilidade e a portabilidade da linguagem.
- 4** A ANSI foi a entidade encarregada de realizar a padronização da linguagem. Depois da laboração do padrão, este foi denominado de C ANSI. Esse padrão foi revisto diversas vezes ao longo do tempo e, posteriormente, foi também reconhecido pela ISO, dando origem ao que chamamos de padrão ANSI/ISO C.

Sistema de arquivos

O tratamento de informações, ou seja, o armazenamento e a recuperação de informações em um programa, muitas vezes se depara com o problema de capacidade do processo em reter maior quantidade de dados. Em algum momento, também será preciso armazenar a informação para uso futuro, de forma persistente, ou disponibilizar a informação para outra aplicação.

Em todos esses casos, a manipulação dessas unidades lógicas de informação, chamadas de arquivos, será necessária.

O gerenciamento de arquivos é feito pelo SO, que cria um mecanismo de abstração para definir como eles são estruturados, nomeados, acessados, usados e protegidos em discos rígidos, óticos, pen-drives etc.

Depois que salvamos um arquivo em um determinado diretório (pasta), informações como o início do cluster do arquivo, o tamanho do arquivo, a hora de criação, entre outras, serão gravadas pelo sistema de arquivos.

Informações no diretório



início do cluster do arquivo



tamanho do arquivo



hora de criação etc.

Comentário

Quando fizermos alterações nesse arquivo, todas as informações registradas serão atualizadas simultaneamente.

O sistema de arquivos é gerado quando estamos criando partições e pode ser modificado pelo comando Formatar no Windows Explorer ou no utilitário de gerenciamento de disco. Também podemos alterá-lo usando ferramentas de particionamento de terceiros.

Existem vários tipos de sistemas de arquivos, incluindo: FAT12, FAT16, FAT32 e NTFS, para dispositivos no SO Windows; Ext2, Ext3 e Ext4, para dispositivos no Linux; HFS/HFS +, para mídia de armazenamento no Mac OS; eISO-9660, Universal Disc Format (UDF) e Compact Disc File System (CDFS), para discos ópticos.

Se uma partição for formatada sem um sistema de arquivos ou o sistema de arquivos estiver danificado, todos os arquivos salvos nessa partição ficarão inacessíveis para sistemas operacionais que não sabem onde esses arquivos foram salvos.

Operações básicas com arquivos em C

Existem quatro operações básicas que podem ser executadas em qualquer arquivo na linguagem C. Elas são:

Abrir/criar um arquivo

Fechar um arquivo

Ler um arquivo

Escrever em um arquivo

Saiba mais

Leia o texto [Funções básicas em C <galeria/aula5/anexo/PDF_Programacao_Aula_05.pdf>](#).

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Atividades

1. Escrever para console e arquivo na maneira adequada ao que necessitamos é uma questão de usar a função correta, passando os parâmetros de forma adequada. Tente reconhecer como os parâmetros abaixo formatam as saídas.

a. #include <stdio.h>

```
int main()
{
printf( "<%d>\n", 123 );
printf( "<%2d>\n", 123 );
printf( "<%10d>\n", 123 );
printf( "<%-10d>\n", 123);
return 0;
}
```

b. #include <stdio.h>

```
int main()
{
printf( "<%f>\n", 1234.56 );
printf( "<%e>\n", 1234.56 );
printf( "<%4.2f>\n", 1234.56 );
printf( "<%3.1f>\n", 1234.56 );
printf( "<%10.3f>\n", 1234.56 );
printf( "<%10.3e>\n", 1234.56 );
return 0;
}
```

c. #include <stdio.h>

#include <stdlib.h>

int main()

```
{
float num;
FILE *fptr;
```

```
fptr = fopen("programa.txt","w");
```

```
if(fptr == NULL)
```

```
{
printf("Erro ao abrir arquivo!");
exit(1);
}
```

```
printf("Entre com um numero fracionário: ");
```

```
scanf("%f",&num);
```

```
fprintf(fptr,"%2f",num);
```

```
fclose(fptr);
```

```
return 0;
```

```
}
```

2. Crie um código para ler o número do arquivo do item c e apresente na tela com uma casa decimal.

3. Crie um programa em C para ler o nome e a nota de n alunos(dadospassados pelo usuário),armazenando em um arquivo alunos.txt. Se o arquivo já existir, o programa deve adicionar as informações ao arquivo.

4. Qual será a saída do programa?

```
#include <stdio.h>
int main()
{
    printf("\nVAI");
    printf("\bRRO");
    printf("\rCA");
    return 0;
}
```

5. A função scanf () pode ser usada para ler mais de uma variável. Escreva um código para ler um número inteiro para a variável X e um número float para a variável Y. Imprima os resultados, com duas casas para o número fracionário.

Notas

Título modal ¹

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Título modal ¹

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Referências

SÁ, Marcela Rocha Tortureli de. **Apostila de Introdução à Linguagem C**. Disponível em: [//www.ufjf.br/petcivil/files/2009/02/Apostila-de-Introdução-à-Linguagem-C.pdf](http://www.ufjf.br/petcivil/files/2009/02/Apostila-de-Introdução-à-Linguagem-C.pdf) <[//www.ufjf.br/petcivil/files/2009/02/Apostila-de-Introdução-à-Linguagem-C.pdf](http://www.ufjf.br/petcivil/files/2009/02/Apostila-de-Introdução-à-Linguagem-C.pdf)>. Acesso em: 27 dez. 2019.

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas operacionais modernos**. 4. ed. São Paulo: Pearson, 2015.

WIKIBOOKS. **PROGRAMAR em C/Entrada e saída em arquivos**. Disponível em: https://pt.wikibooks.org/wiki/Programar_em_C/Entrada_e_sa%C3%ADda_em_arquivos <https://pt.wikibooks.org/wiki/Programar_em_C/Entrada_e_sa%C3%ADda_em_arquivos>. Acesso em: 27 dez. 2019.

Próxima aula

- Porta serial do computador, muito útil para o entendimento da interface com diferentes hardwares e diferentes protocolos;
- Controle de dispositivos com microcontroladores;
- Aquisição de dados com microcontroladores.

Explore mais

- Não é fácil se acostumar com funções que têm parâmetros com muitas codificações de formatação, como as que foram vistas nesta aula. É natural uma confusão inicial para saber escrever os parâmetros com os códigos de formatação corretos e nos lugares certos. Uma boa forma de gravar é observar a escrita de código por alguém, para, em seguida, usar as funções. Assista aos vídeos:
 - [Variáveis, entrada e saída de dados, operadores aritméticos em Linguagem C <https://www.youtube.com/watch?v=ELQPwusHzrk>](https://www.youtube.com/watch?v=ELQPwusHzrk).
 - [Programação em C/C++ – Aula 18 – Manipulação de arquivos <https://www.youtube.com/watch?v=6h2ja9MzBkc>](https://www.youtube.com/watch?v=6h2ja9MzBkc).
 - [Programar em C – Manipulação de Arquivos txt em C / Escrever Dados – Aula 81 <https://www.youtube.com/watch?v=eriDnpkh5kA>](https://www.youtube.com/watch?v=eriDnpkh5kA).
 - [Programar em C – Funções freopen e fgets / stdin – Aula 86 <https://www.youtube.com/watch?v=7mKC07_IAg4>](https://www.youtube.com/watch?v=7mKC07_IAg4).
 - [Programar em C – Funções freopen e fgets / stdin \[Parte 2\] – Aula 87 <https://www.youtube.com/watch?v=08LkxBAz4LU>](https://www.youtube.com/watch?v=08LkxBAz4LU).