

Lógica de Programação: notas de aula

Prof. Dr. Jonatha Costa

2024

Organização

1 Bibliotecas

Biblioteca `<stdio.h>`

Bibliotecas próprias

2 Exemplos-bibliotecas

3 Comandos especiais

Objetivo da aula

- Apresentar comandos de biblioteca `<stdio.h>`;
- Apresentar comandos especiais;
- Apresentar bibliotecas da linguagem C.

Organização

1 Bibliotecas

Biblioteca <stdio.h>

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca math

Biblioteca string.h

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Biblioteca <stdio.h>

A biblioteca padrão **stdio.h** do C é responsável por funções de entrada e saída (I/O), como leitura e escrita de dados, dentre outras funções (comandos).

Principais funções da Biblioteca<stdio.h>

Principais funções de entrada e saída padrão presentes em **stdio.h**:

- `printf()`: Escreve dados formatados na saída padrão (geralmente a tela).
- `scanf()`: Lê dados formatados da entrada padrão (geralmente o teclado).
- `putchar()`: Escreve um único caractere na saída padrão.
- `getchar()`: Lê um único caractere da entrada padrão.
- `puts()`: Escreve uma string na saída padrão seguida por uma nova linha.
- `gets()`: Lê uma linha de texto da entrada padrão (descontinuada nas versões mais recentes de C devido a problemas de segurança).
- `fgets()`: Lê uma string de um fluxo de entrada (utilizado como alternativa segura a `gets()`).
- `fputs()`: Escreve uma string em um fluxo de saída.

Principais funções da Biblioteca<stdio.h>

Principais funções de manipulação de arquivo presentes em **stdio.h**:

- `fopen()`: Abre um arquivo e retorna um ponteiro para o arquivo.
- `fclose()`: Fecha um arquivo aberto.
- `fread()`: Lê dados de um arquivo para a memória.
- `fwrite()`: Escreve dados da memória para um arquivo.
- `fseek()`: Move o ponteiro do arquivo para uma posição específica.
- `ftell()`: Retorna a posição atual do ponteiro do arquivo.
- `rewind()`: Move o ponteiro do arquivo para o início do arquivo.
- `fprintf()`: Escreve dados formatados em um fluxo de saída (arquivo ou outro).
- `fscanf()`: Lê dados formatados de um fluxo de entrada.
- `feof()`: Verifica se o final do arquivo foi alcançado.
- `ferror()`: Verifica se houve um erro no arquivo.
- `fflush()`: Limpa (flush) o buffer de saída de um arquivo.

Principais funções da Biblioteca<stdio.h>

- Principais funções de manipulação de caractere presentes em **stdio.h**:
 - `ungetc()`: Devolve um caractere lido de volta ao fluxo de entrada;
 - `putc()`: Escreve um caractere em um fluxo de saída;
 - `getc()`: Lê um caractere de um fluxo de entrada.
- Principais funções de erro presentes em **stdio.h**:
 - `perror()`: Imprime uma mensagem de erro para a saída padrão com base no código de erro fornecido.
 - `clearerr()`: Limpa os indicadores de erro e fim de arquivo (EOF) associados ao fluxo.
- Principal função de redirecionamento presente em **stdio.h**:
 - `freopen()`: Redireciona um fluxo de entrada ou saída (útil para redirecionar a saída padrão para um arquivo).

Organização

1 Bibliotecas

Biblioteca `<stdio.h>`

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca `math`

Biblioteca `string.h`

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Bibliotecas próprias da linguagem

A linguagem C possui várias bibliotecas padrão que fornecem funcionalidades essenciais para diversas operações, como manipulação de strings, operações matemáticas, controle de tempo, dentre outras.

- **<stdio.h>**
 - Propósito: Funções de entrada e saída padrão;
 - Principais Funções: 'printf()', 'scanf()', 'fopen()', 'fclose()', 'fread()', 'fwrite()', 'getchar()', 'putchar()';
- **<math.h>**
 - Propósito: Funções matemáticas básicas e avançadas;
 - Principais Funções: 'pow()', 'sqrt()', 'sin()', 'cos()', 'tan()', 'log()', 'exp()';
- **<string.h>**
 - Propósito: Manipulação de strings e arrays de caracteres;
 - Principais Funções: 'strlen()', 'strcpy()', 'strcat()', 'strcmp()', 'memcpy()', 'memset()';

Bibliotecas próprias da linguagem

- **<stdlib.h>**
 - Propósito: Funções utilitárias, alocação de memória, controle de processos, conversões de variáveis e geração de números aleatórios;
 - Principais Funções: 'malloc()', 'free()', 'exit()', 'atoi()', 'rand()', 'srand()', 'system()';
- **<ctype.h>**
 - Propósito: Manipulação de caracteres (como verificação de tipos de caracteres e conversões);
 - Principais Funções: 'isalpha()', 'isdigit()', 'isspace()', 'toupper()', 'tolower()';
- **<time.h>**
 - Propósito: Manipulação de tempo e data;
 - Principais Funções: ** 'time()', 'clock()', 'difftime()', 'strftime()', 'mktime()';
- **<limits.h>**
 - Propósito: Define constantes relacionadas aos limites dos tipos de dados primitivos;
 - Exemplos: 'INT_MAX', 'CHAR_MIN', 'LONG_MAX';

Bibliotecas próprias da linguagem

- **<float.h>**
 - Propósito: Define constantes relacionadas aos limites dos tipos de dados de ponto flutuante;
 - Exemplos: 'FLT_MAX', 'DBL_MIN', 'LDBL_EPSILON';
- **<stdbool.h>**
 - Propósito: Define o tipo 'bool' para representar valores booleanos ('true' e 'false');
 - Principais Macros: 'true', 'false';
- **<stddef.h>**
 - Propósito: Define tipos e macros comuns, como 'size_t', 'ptrdiff_t', 'NULL';
 - Principais Definições: ** 'NULL', 'offsetof()', 'size_t';
- **<stdint.h>**
 - Propósito: Define tipos inteiros de tamanho fixo, como 'int8_t', 'int16_t', 'uint32_t';
 - Exemplos: 'int8_t', 'uint16_t', 'int32_t', 'uint64_t';

Bibliotecas próprias da linguagem

- **<errno.h>**
 - Propósito: Manipulação de códigos de erro retornados por funções do sistema;
 - Principais Definições: 'errno', 'EDOM', 'ERANGE', 'EFAULT';
- **<assert.h>**
 - Propósito: Fornece a macro 'assert()' para fazer verificações em tempo de execução, normalmente usada para depuração;
 - Principal Função: 'assert()';
- **<signal.h>**
 - Propósito: Manipulação de sinais, que são notificações que um processo pode receber de outras partes do sistema operacional.
 - Principais Funções: 'signal()', 'raise()', 'sigaction()'.
- **<locale.h>**
 - Propósito: Manipulação de localidade, permitindo ajustar o comportamento de funções para diferentes regiões geográficas;
 - Principais Funções: 'setlocale()', 'localeconv()';

Bibliotecas próprias da linguagem

- **<setjmp.h>**
 - Propósito: Fornece suporte para saltos não locais no código, permitindo pular entre diferentes partes do código (normalmente usado em tratamentos de erro);
 - Principais Funções: 'setjmp()', 'longjmp()';
- **<stdarg.h>**
 - Propósito: Manipulação de listas de argumentos de tamanho variável em funções;
 - Principais Funções: 'va_start()', 'va_arg()', 'va_end()';
- **<complex.h>**
 - Propósito: Fornece suporte para operações com números complexos (adicionado no padrão C99);
 - Principais Funções: 'cabs()', 'creal()', 'cimag()', 'cexp()';
- **<tgmath.h>**
 - Propósito: Proporciona macros genéricas que funcionam com números inteiros, de ponto flutuante e complexos (adicionado no padrão C99);
 - Principais Funções: Macros genéricas para operações matemáticas, como 'tgamma()'.

Organização

1 Bibliotecas

2 Exemplos-bibliotecas

- Biblioteca math

- Biblioteca string.h

- Códigos diversos

- Exemplos de códigos executáveis

3 Comandos especiais

Organização

1 Bibliotecas

Biblioteca `<stdio.h>`

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca `math`

Biblioteca `string.h`

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Exemplo de aplicação de `<math.h>`

```
1  #include <stdio.h>
2  #include <math.h>
3  main() {
4  int num;
5  printf("Digite um número: "); scanf("%d",&num);
6  // Função pow(): Calcula base elevada ao expoente
7  double potencia = pow(base, expoente);
8  printf("%.2f elevado a %.2f é: %.2f/n", "base, expoente, potencia");
9  // Função sqrt(): Calcula a raiz quadrada de um número double
10 raiz = sqrt(numero);
11 printf("A raiz quadrada de %.2f é: %.2fn", numero, raiz);
12 }
```

Organização

1 Bibliotecas

Biblioteca <stdio.h>

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca math

Biblioteca string.h

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Exemplo de aplicação de `<string.h>`

```
1  #include <stdio.h>
2  #include <string.h>

3  struct Estudante {
4      char nome[50];
5      int idade;
6      float nota; };
7  int main() {
8      struct Estudante aluno;
9      strcpy(aluno.nome, "Ana"); //“strcpy” escreve caracteres no struct
10     aluno.idade = 20; aluno.nota = 8.5;
11     printf("Nome: %sn", aluno.nome);
12     printf("Nome: %dn", aluno.idade);
13     printf("Nome: %.2fn", aluno.nota);
14 }
```

Organização

1 Bibliotecas

Biblioteca `<stdio.h>`

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca `math`

Biblioteca `string.h`

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Organização

1 Bibliotecas

Biblioteca `<stdio.h>`

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca `math`

Biblioteca `string.h`

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Exemplos de códigos executáveis

- Veja a lista de códigos em: <https://github.com/jonathacosta/PL>

Organização

- ① Bibliotecas
- ② Exemplos-bibliotecas
- ③ Comandos especiais
 - Comandos Especiais

Organização

1 Bibliotecas

Biblioteca <stdio.h>

Bibliotecas próprias

2 Exemplos-bibliotecas

Biblioteca math

Biblioteca string.h

Códigos diversos

Exemplos de códigos executáveis

3 Comandos especiais

Comandos Especiais

Comandos especiais para compactação de códigos em C

- **#define** - A diretiva #define é utilizada para criar macros, que são substituições de texto, podendo serem utilizadas para definir constantes ou expressões;
- **#undef** - A diretiva #undef é utilizada para desfazer a definição de uma macro feita com #define;
- **operador ternário** - (condição ? expressão1 : expressão2;)
condição: Uma expressão que será avaliada como verdadeira (não-zero) ou falsa (zero).
expressão1: Será executada se a condição for verdadeira.
expressão2: Será executada se a condição for falsa.

Código Conciso:

Versão 01

```
1  #include <stdio.h>
2  main() {
3  int x=3, y;
4  if (x<5){y=11;}
5  else{y=12;}
6  printf("%d",y);
7  }
```

Versão 02

```
1  #include <stdio.h>
2  main() {
3  int x=3, y;
4  y=(x<5)?11:10;
5  printf("%d",y);
6  }
```

Versão 03

```
1  #include <stdio.h>
2  main() {
3  int x=3, y;
4  printf("%d",(x<5)?10:11);
5  }
```

Preferências?

Exemplos de códigos executáveis

- Veja a lista de códigos em: <https://github.com/jonathacosta/PL>

Exercícios

- Veja material auxiliar e lista de códigos em:
<https://github.com/jonathacosta/PL>