

Estrutura de Dados

Revisão: Fundamentos da Linguagem C

Wolney Henrique Queiroz Freitas
wolney.freitas@professores.unifavip.edu.br

Roteiro de Aula

- Estrutura de um programa em C;
- Variáveis;
- Comandos de Entrada e Saída.
- Estruturas de Decisão
- Estruturas de Repetição

Ambiente de Desenvolvimento

- CLion
 - <https://www.jetbrains.com/pt-br/clion/>
- Dev-C++
 - <https://www.bloodshed.net/>
- Online
 - https://www.onlinegdb.com/online_c_compiler

Estrutura de um Programa em C

- Vamos implementar nossos programas na linguagem de programação C;
- A estrutura básica de um arquivo em C para escrever algoritmos consiste do seguinte:
 - Importação de bibliotecas;
 - Função main.

Estrutura de um Programa em C

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

```
main(){
```

```
}
```

Bibliotecas, contém funções e recursos que vamos precisar para escrever nossos algoritmos, a biblioteca stdio.h por exemplo, é responsável pelos comandos de entrada e saída.

Nossos algoritmos
são escritos aqui

Tipos de Variáveis em C

- Em C, vamos trabalhar com os seguintes tipos de variáveis / dados:
 - Números inteiros (`int`);
 - Números reais (`float`);
 - Caracteres (`char`);
 - Cadeia de caracteres, também conhecido como Strings (`char[tamanho]`).

Declarando Variáveis

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

```
main(){  
    float peso;  
    int idade = 20;  
}
```

Nome da variável

Tipo

Todo comando em C se encerra com um ponto e vírgula

Declarando variável e inicializando com um valor

Declarando Variáveis

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

```
main(){
```

```
// Número inteiro
```

```
int idade = 20;
```

```
// Número real
```

```
float peso = 58.7;
```

```
// Caractere
```

```
char sexo = 'F';
```

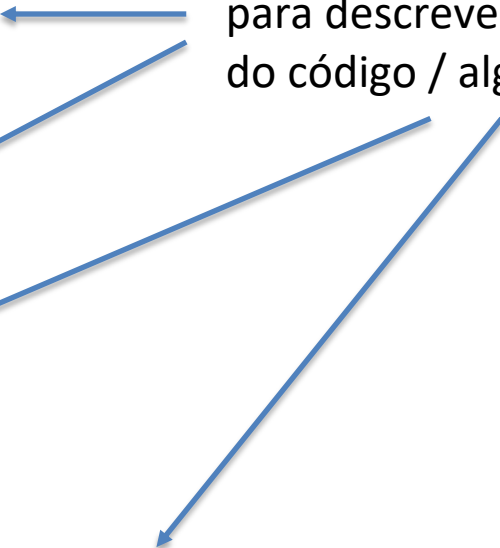
```
// Cadeia de caracteres, ou String
```

```
// Serve para guardar nomes, frases, etc...
```

```
char nome[100] = "Maria";
```

```
}
```

Isso são comentários, servem para descrever ou explicar partes do código / algoritmo



Comandos de Entrada e Saída

- Entradas e saídas são a comunicação do programa com o mundo real, a forma com que o programa recebe os dados a serem processados do mundo real e devolve a este a resposta.
- A entrada pode ser feita pelo **teclado**, mouse, arquivos de texto e de outros dispositivos físicos de Entrada;
- A saída pode ser feita no **vídeo**, na impressora, arquivos texto e de outros dispositivos de Saída.

Comandos de Entrada e Saída

- Na linguagem C, utilizamos as funções `scanf` e `printf` para realizar a entrada e a saída de dados, respectivamente;
- Com `scanf`, a entrada é realizada pelo teclado, e com `printf`, a saída é realizada no vídeo.

Comando de Saída

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

```
main(){
```

```
    printf("Oi mundo.");
```

```
}
```

Função para mostrar
algo no vídeo / tela do
computador

Escreve a mensagem "Oi
mundo." na tela do
computador

Comando de Saída

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

```
main(){
```

```
    float nota = 8.0;
```

```
    printf("A nota do aluno foi: %f", nota);
```

```
}
```

Exibindo o valor de uma variável na saída



Comando de Saída

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

```
main(){
```

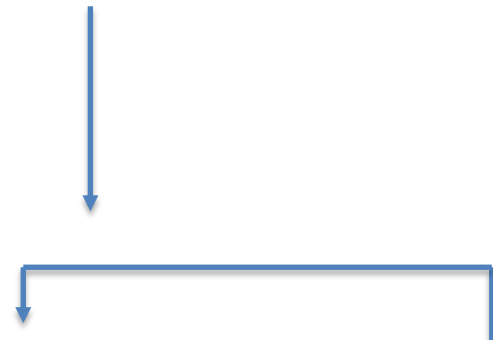
```
    int idade = 20;
```

```
    float peso = 58.7;
```

```
    printf("Idade %d e Peso %f", idade, peso);
```

```
}
```

Exibindo o valor de mais de uma
variável



Comando de Saída

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

```
main(){
```

```
    int idade = 20;
```

```
    float peso = 58.7;
```

```
    printf("Idade %d e Peso %f", idade, peso);
```

```
}
```

int: %d

float: %f

char: %c

string: %s

Comando de Entrada

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

```
main(){
```

```
    int idade;
```

```
    printf("Informe uma idade: ");
```

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

```
}
```

Comando de
entrada, ler
dados
informados pelo
teclado

Tipo de dados
que será lido

Endereço de memória da variável
que guardará o valor lido (& + nome
da variável)

Comando de Entrada

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

```
main(){
```

```
    int idade;
```

```
    printf("Informe uma idade: ");
```

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

```
    printf("Idade informada: %d", idade);
```

```
}
```


Comando de Entrada

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

```
main(){
```

```
    char nome[80];
```

```
    int idade;
```

```
    printf("Informe seu nome: ");
```

```
    scanf("%s", &nome);
```

```
    fflush(stdin);
```

```
    printf("Informe sua idade: ");
```

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

```
    fflush(stdin);
```

```
    printf("Nome: %s \nIdade: %d", nome, idade);
```

```
}
```

Limpa o “lixo” do teclado para evitar erros.
Recomendado sempre utilizar após um comando de entrada.

↑
/n é utilizada para “pular”
linha na saída de dados.

Comando de Entrada

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

```
main(){
```

```
    char nome[80];
```

```
    int idade;
```

```
    printf("Informe seu nome e sua idade: ");
```

```
    scanf("%s%d", &nome, &idade);
```

← Ler mais de um valor no mesmo scanf

```
    fflush(stdin);
```

```
    printf("Nome: %s \nIdade: %d", nome, idade);
```

```
}
```

Comando de Entrada

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

```
main(){
```

```
    char nome[80];
```

```
    printf("Informe seu nome: ");
```

```
    scanf("%s", &nome);    Ler "Ana Maria"
```

```
    fflush(stdin);
```

```
    printf("Nome informado: %s", nome);    Mostra apenas "Ana"
```

```
    printf("\nInforme seu nome: ");
```

```
    gets(nome);    Ler "Ana Maria"
```

```
    fflush(stdin);
```

```
    printf("Nome informado: %s", nome);    Mostra "Ana Maria"
```

```
}
```

Programa de Cálculo de Média

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

```
main(){
```

```
    float nota1, nota2, media;
```

```
    printf("Informe duas notas: ");
```

```
    scanf("%f%f", &nota1, &nota2);
```

```
    media = (nota1 + nota2) / 2;
```

```
    printf("Media: %f", media);
```

```
}
```

Estrutura de Decisão Simples

- Utiliza a seguinte sintaxe:

```
if (condição) {  
    // comandos executados se a condição for  
    verdadeira  
}
```

- Se a condição for verdadeira, serão executados todos os comandos dentro do **if**;
- Se a condição for falsa, os comandos dentro do **if** serão ignorados.

Estrutura de Decisão Simples

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

```
main(){  
    float nota1, nota2, media;  
    printf("Informe a primeira nota: ");  
    scanf("%f", &nota1);  
    printf("Informe a segunda nota: ");  
    scanf("%f", &nota2);  
    media = (nota1 + nota2) / 2;  
    printf("Media do aluno: %f", media);  
  
    if(media >= 7){  
        printf("\nAluno aprovado.");  
    }  
}
```

Estrutura de Decisão Simples

- Vamos praticar
 - Ler dois valores, efetuar a adição e apresentar seu resultado apenas se o valor somado for maior que 10.

Estrutura de Decisão Simples

- Vamos praticar
 - Faça um algoritmo que leia 4 números;
 - Informe quantos desses números são maiores que 50.

Estrutura de Decisão Simples

- Vamos praticar
 - Ler 4 números inteiros;
 - Calcular e mostrar a soma dos que forem par.

Estrutura de Decisão Composta

- Utiliza a seguinte sintaxe:

```
if (condição) {  
    // comandos executados se a condição for verdadeira  
} else {  
    // comandos executados se a condição for falsa  
}
```

- Se a condição for verdadeira, serão executados todos os comandos dentro do **if**;
- Se a condição for falsa, serão executados todos os comandos dentro do **else**.

Estrutura de Decisão Composta

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

```
main(){  
    float nota1, nota2, media;  
    printf("Informe a primeira nota: ");  
    scanf("%f", &nota1);  
    printf("Informe a segunda nota: ");  
    scanf("%f", &nota2);  
    media = (nota1 + nota2) / 2;  
    printf("Media do aluno: %f", media);  
  
    if(media >= 7){  
        printf("\nAluno aprovado.");  
    }else{  
        printf("\nAluno reprovado.");  
    }  
}
```

Estrutura de Decisão Composta

- Vamos praticar
 - Ler um número inteiro;
 - Informar se é par ou ímpar.

Estrutura de Decisão Composta

- Vamos praticar
 - Ler dois valores, efetuar a adição;
 - Se o valor somado for menor ou igual a 10, deverá ser apresentado ao usuário o resultado da adição mais 5;
 - Se o valor somando não for menor ou igual a 10, deverá ser apresentado ao usuário o resultado da adição menos 7.

Estrutura de Decisão Encadeada

- Utiliza a seguinte sintaxe:

```
if (condição) {  
    // comandos executados se a primeira condição for verdadeira  
} else if (condição) {  
    // comandos executados se a primeira condição for falsa  
    // e a segunda condição for verdadeira  
} else {  
    // comandos executados se todas as condições anteriores forem  
    falsas  
}
```

- Uma determinada ação não poderá ser executada se uma condição anterior for satisfeita.

Estrutura de Decisão Encadeada

```
#include <stdio.h>

main(){
    float nota1, nota2, media;
    printf("Informe a primeira nota: ");
    scanf("%f", &nota1);
    printf("Informe a segunda nota: ");
    scanf("%f", &nota2);
    media = (nota1 + nota2) / 2;
    printf("Media do aluno: %f", media);

    if(media >= 7){
        printf("\nAluno aprovado.");
    }else if(media >= 3 && media < 7){
        printf("\nAluno na final.");
    }else{
        printf("\nAluno reprovado.");
    }
}
```

Estrutura de Decisão Encadeada

- Vamos praticar
 - Ler dois números inteiros, A e B;
 - Se A for igual a B, imprimir OS NÚMEROS (VALOR DE A) E (VALOR DE B) SÃO IGUAIS;
 - Se A for menor que B, imprimir O NÚMERO (VALOR DE A) É MENOR QUE (VALOR DE B);
 - Se A for maior que B, imprimir O NÚMERO (VALOR DE A) É MAIOR QUE (VALOR DE B).

Estrutura de Decisão Encadeada

- Também é possível colocar instruções de decisão dentro de outras instruções de decisão:

```
if (condição) {  
    if (condição) {  
        // comandos executados se a primeira  
        // condição e a condição interna foram verdadeiras  
    } else {  
        // comandos executados se a primeira  
        // condição for verdadeira e a condição interna for falsa  
    }  
}
```

Estrutura de Decisão de Múltipla Escolha

- Uma decisão de múltipla escolha pode ser construída com os comandos **if...else if...else** encadeados, porém, torna-se de difícil leitura;
- Há um comando, **switch...case**, que oferece uma melhoria na leitura do algoritmo.

Estrutura de Decisão de Múltipla Escolha

- Utiliza a seguinte sintaxe:

```
switch (expressão de seleção){  
    case opcao1:  
        // comandos a serem executado  
    case opcao2:  
        // comandos a serem executado  
    case opcao3:  
        // comandos a serem executado  
    default:  
        // comandos a serem executado  
}
```

Estrutura de Decisão de Múltipla Escolha

```
int faixa_idade;
printf("Escolha sua faixa de idade: ");
printf("\n1 - Crianca ou adolescente (0 a 20 anos).");
printf("\n2 - Jovem ou meia idade (21 a 59 anos).");
printf("\n3 - Idoso (60 a 80 anos).\nEscolha: ");
scanf("%d", &faixa_idade);

switch(faixa_idade){
    case 1:
        printf("Primeira idade.");
        break;
    case 2:
        printf("Segunda idade.");
        break;
    case 3:
        printf("Terceira idade.");
        break;
    default:
        printf("Escolha invalida.");
}
```

Estrutura de Decisão de Múltipla Escolha

- Vamos praticar
 - Escreva um algoritmo para ler um valor inteiro de 1 a 7 e escrever a descrição do dia da semana correspondente. Se for 1, imprimir DOMINGO, e assim sucessivamente;
 - Caso o valor digitado esteja fora desse intervalo o usuário deverá ser alertado que a escolha é inválida.

Repetição com teste inicial: while

- A instrução **while** repete uma determinada instrução **enquanto** uma determinada condição for **verdadeira**.
- Sintaxe:
while (<condição>) {
 <comando>
 <comando>
}

Repetição com teste inicial: while

- A <condição> é testada, se for verdadeira, os comandos dentro do **while** são executados;
- Quando todos os comandos dentro do **while** são executados, é testada novamente a <condição>;
- Se a <condição> for verdadeira, ele repete tudo novamente;
- Ele só irá sair do bloco de repetição quando a <condição> for falsa.

Repetição com teste inicial: while

```
1  #include <stdio.h>
2
3  main(){
4      int numero = 1;
5
6      while(numero <= 5){
7          printf("\n%d", numero);
8          numero++;
9      }
10 }
```


Repetição com teste inicial: while

- Algoritmo para ler 100 números e informar a média deles:

```
1  #include <stdio.h>
2
3  main(){
4      int contador = 1;
5      float soma = 0, numero, media;
6
7      while(contador <= 100){
8          printf("\nInforme %d numero: ", contador);
9          scanf("%f", &numero);
10         soma += numero;
11         contador++;
12     }
13     media = soma / 100;
14     printf("Media: %f", media);
15 }
```

Vamos Praticar

- Um banco tem 10 clientes;
- O algoritmo irá ler o nome e o saldo de cada cliente;
- Se o saldo estiver negativo, o algoritmo informará **CLIENTE <nome> ESTÁ COM A CONTA ESTOURADA!**
- Se o saldo estiver positivo, o algoritmo informará **CLIENTE <nome> ESTÁ COM A CONTA NORMAL!**

Vamos Praticar

- A empresa Engenheiros Felizes possui alguns funcionários;
- O algoritmo deverá ler a quantidade de funcionários da empresa;
- O algoritmo deverá ler o valor do salário de cada um deles e informar a média salarial da empresa.

Vamos Praticar

- Imprima a soma de todos os números ímpares de 0 a 200.

Vamos Praticar

- Faça um algoritmo que:
 - Leia vários números;
 - Finalize quando o valor 9999 for digitado;
 - Imprima o maior valor informado (exceto 9999).

Repetição com teste final: do-while

- A instrução **do-while** repete uma determinada instrução **enquanto** uma determinada condição for **verdadeira**;
- A diferença entre o **do-while** e o **while** é que o bloco de repetição é executado uma vez e só depois a condição é testada;
- Sintaxe:
 do {
 <comando>
 <comando>
 } while (<condição>);

Repetição com teste final: do-while

- O bloco é executado a primeira vez;
- Se a <condição> for verdadeira, ele repete tudo novamente;
- Ele só irá sair do bloco de repetição quando a **<condição> for falsa.**

Repetição com teste final: do-while

```
1  #include <stdio.h>
2
3  main(){
4      // Imprimir todos os pares 0 e 100
5      int numero = 0;
6      do{
7          if(numero % 2 == 0){
8              printf("\n%d", numero);
9          }
10         numero++;
11     }while(numero <= 100);
12 }
```


Vamos Praticar

- Faça um algoritmo que peça para o usuário informar apenas números ímpares;
- Quando o usuário informar um número par, a leitura deve ser encerrada, e a soma dos números ímpares deve ser exibida.

Vamos Praticar

- O algoritmo deverá pedir para o usuário digitar números menores que 10 e maiores que 20;
- Se o usuário digitar algum entre 10 e 20, o algoritmo imprimirá a palavra FIM e encerrará.

Repetição controlada: for

- A instrução **for** tem a função de repetir um determinado bloco de comandos até um determinado número;
- Sintaxe:
for(<valor inicial> ; <condição de parada> ;
 <incremento / decremento>){
 <comando>
 <comando>
}

Repetição controlada: for

```
1  #include <stdio.h>
2
3  main(){
4      // Imprimir todos os pares 0 e 100
5      int numero;
6      for(numero = 0; numero <= 100; numero++){
7          if(numero % 2 == 0){
8              printf("\n%d", numero);
9          }
10     }
11 }
```

Vamos Praticar

- Faça um algoritmo para ler a nota de 5 alunos;
- Exiba qual foi a maior e a menor nota.

Vamos Praticar

- Faça um algoritmo para o usuário digitar um número inteiro e o algoritmo imprimir sua contagem regressiva;

Vamos Praticar

- O algoritmo deverá pedir para o usuário digitar um número par;
- O algoritmo imprimirá uma contagem em ordem crescente, a partir de 0, de 2 em 2, até o número digitado pelo usuário.



Faci facid FACIMP FBV fmf Presidência
Martha Falcão ISL UNIFAVIP UNI
METROCAMP RUY
BARBOSA | AREA1 UniFBV UniFanor