ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite

Funções

INSTITUTO FEDERAL
Pernambuco

- Uma função é um bloco nomeado de instruções que executa uma operação específica.
- Ela é definida a partir de um identificador (nome) e de um conjunto de instruções que a compõem, podendo ser reutilizada posteriormente sempre que necessário.
- As funções permitem dividir o código em partes menores e mais organizadas, o que aumenta a legibilidade, facilita a manutenção e contribui para o reaproveitamento do código

Funções

INSTITUTO FEDERAL
Pernambuco

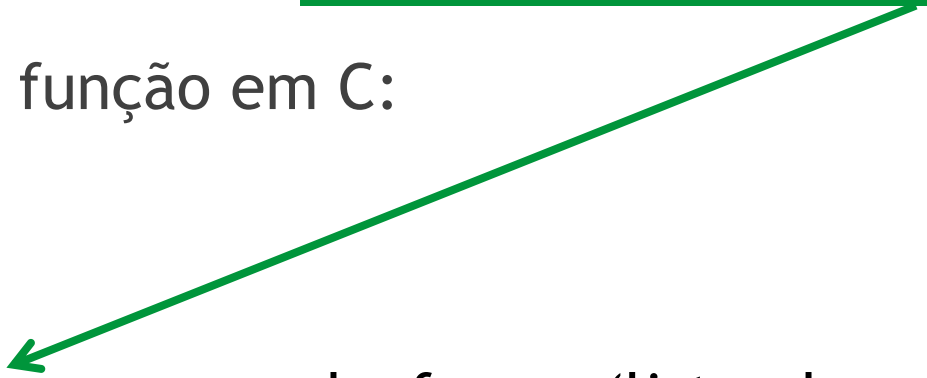
- Declaração de uma função em C:

```
tipo_retorno nome_da_funcao(lista_de_parametros) {
    // corpo da função
}
```

Funções

O tipo de retorno é o tipo de dado que a função devolve ao final da execução.

- Declaração de uma função em C:



```
tipo_retorno nome_da_funcao(lista_de_parametros) {
    // corpo da função
}
```

INSTITUTO FEDERAL
Pernambuco

Funções

O tipo de retorno é o tipo de dado que a função devolve ao final da execução.

O nome da função é o identificador da função

- Declaração de uma função em C:

```
tipo_retorno nome_da_funcao(lista_de_parametros) {
    // corpo da função
}
```

Funções

- Declaração de uma função em C:

O tipo de retorno é o tipo de dado que a função devolve ao final da execução.

O nome da função é o identificador da função

```
tipo_retorno nome_da_funcao(lista_de_parametros) {
    // corpo da função
}
```

A lista de parâmetros é opcional e define as variáveis que a função recebe.

INSTITUTO FEDERAL
Pernambuco

Funções

- Declaração de uma função em C:

O tipo de retorno é o tipo de dado que a função devolve ao final da execução.

O nome da função é o identificador da função

```
tipo_retorno nome_da_funcao(lista_de_parametros) {
    // corpo da função
}
```

A lista de parâmetros é opcional e define as variáveis que a função recebe.

O corpo da função é o bloco de código que contém as instruções executadas quando a função é chamada.

Funções

INSTITUTO FEDERAL
Pernambuco

- Exemplo de função:

```
void saudacao() {
    printf("Olá! Seja bem-vindo ao programa!\n");
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Exemplo de função com parâmetro:

```c
void soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Exemplo de função com parâmetro:

```
void soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}
```

função soma possui dois parâmetros: o primeiro é do tipo float e o segundo é do tipo int.

Funções

INSTITUTO FEDERAL
Pernambuco

- Chamando uma função:

```c
void soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}

int main(){
    float num1 = 10;
    int num2 = 15;
    soma(num1, num2);
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Chamando uma função:

```c
void soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}

int main(){
    float num1 = 10;
    int num2 = 15;
    soma(num1, num2);
}
```

Os parâmetros de uma função são passados de acordo com a ordem em que aparecem na chamada.

No caso da função soma, ela é chamada com as variáveis num1 e num2, exatamente nessa sequência.

Funções

INSTITUTO FEDERAL
Pernambuco

- Retornando um valor em uma função:

```c
float soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    return resultado;
}

int main(){
    float num1 = 10;
    int num2 = 15;
    float resultado = soma(num1, num2);
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Retornando um valor em uma função:

```c
float soma(float num1, int num2){
    float resultado;
    resultado = num1 + num2;
    return resultado;
}

int main(){
    float num1 = 10;
    int num2 = 15;
    float resultado = soma(num1, num2);
    printf("A soma de %.2f com %d é %.2f", num1, num2, resultado);
}
```

A função soma agora retorna um valor do tipo float. Esse valor retornado pode ser utilizado diretamente na própria linha em que a função é chamada.

Funções

INSTITUTO FEDERAL
Pernambuco

- O escopo de uma variável determina em quais partes do código ela pode ser acessada ou modificada.
- Escopo Local:
  - Variáveis declaradas dentro de uma função pertencem ao seu escopo local.
  - Só podem ser utilizadas dentro da própria função onde foram criadas.
- Escopo Global:
  - Variáveis declaradas fora de funções, no corpo principal do programa.
  - Podem ser acessadas em qualquer parte do código, tanto no escopo global quanto dentro de funções.

Funções

INSTITUTO FEDERAL
Pernambuco

```c
#include <stdio.h>

char var_global[] = "Sou uma variável global \\o/";

void showMeTheCode(){
    char var_local[] = "Sou uma variável local o/";
    printf("%s\n", var_local);
    printf("%s\n", var_global);
}

int main(){
    showMeTheCode();
    printf("%s\n", var_global);
    printf("%s\n", var_local);
    return 0;
}
```

Funções

- Escopo local:
  - Criado dentro de uma função.
  - Só pode ser utilizado na função onde foi declarado.

```c
#include <stdio.h>

void minhaFuncao(){
    int x = 10;
    printf("Dentro da função %d \n", x);
}

int main(){
    minhaFuncao();
    print(x);
    return 0;
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Escopo global:
  - Criado fora de funções, no corpo principal do programa.
  - Pode ser acessado tanto globalmente quanto dentro de funções.

```c
#include <stdio.h>

int y = 20;

void outraFuncao(){
    printf("Dentro da função: %d \n", y);
}

int main(){
    outraFuncao();
    printf("Fora da função: %d \n", y);
    return 0;
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

```c
#include <stdio.h>

int x = 10;

void teste() {
    int x = 20;
    printf("%d\n", x);
}

int main() {
    teste();
    printf("%d\n", x);
}
```

Qual a saída do programa?

Funções

INSTITUTO FEDERAL
Pernambuco

```c
#include <stdio.h>

int x = 10;

void teste() {
    int x = 20;
    printf("%d\n", x);
}

int main() {
    teste();
    printf("%d\n", x);
}
```

```
20
10
```

Funções

```c
#include <stdio.h>

int contador = 0;

void incrementaContador() {
    contador++;
    printf("Contador dentro da função: %d\n", contador);
}

int main() {
    printf("Valor inicial do contador: %d\n", contador);

    incrementaContador();
    incrementaContador();
    incrementaContador();

    printf("Valor final do contador no main: %d\n", contador);

    return 0;
}
```

Qual a saída do programa?

Funções

```c
#include <stdio.h>

int contador = 0;

void incrementaContador() {
    contador++;
    printf("Contador dentro da função: %d\n", contador);
}

int main() {
    printf("Valor inicial do contador: %d\n", contador);

    incrementaContador();
    incrementaContador();
    incrementaContador();

    printf("Valor final do contador no main: %d\n", contador);

    return 0;
}
```

```
Valor inicial do contador: 0
Contador dentro da função: 1
Contador dentro da função: 2
Contador dentro da função: 3
Valor final do contador no main: 3
```

INSTITUTO FEDERAL
Pernambuco

Funções

INSTITUTO FEDERAL
Pernambuco

- Passagem de parâmetro por valor:
  - Na passagem por valor, a função recebe apenas uma cópia do valor da variável usada na chamada.
  - Por isso, qualquer alteração feita dentro da função afeta somente a cópia, e não modifica a variável original que está no código chamador.

Funções

INSTITUTO FEDERAL
Pernambuco

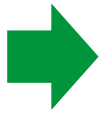
```c
#include <stdio.h>

void alterar(int n) {
    n = 50;
}

int main() {
    int x = 10;

    printf("Antes da função: x = %d\n", x);

    alterar(x);

    printf("Depois da função: x = %d\n", x);

    return 0;
}
```

Qual a saída do programa?

Funções

INSTITUTO FEDERAL
Pernambuco

```c
#include <stdio.h>

void alterar(int n) {
    n = 50;
}

int main() {
    int x = 10;

    printf("Antes da função: x = %d\n", x);

    alterar(x);

    printf("Depois da função: x = %d\n", x);

    return 0;
}
```



```
Antes da função: x = 10
Depois da função: x = 10
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Passagem de parâmetro por referência:
  - Na passagem por referência, a função recebe o endereço da variável, permitindo alterar diretamente o valor armazenado na variável original.
  - Para isso, devem ser observados três pontos fundamentais:
    - Na chamada da função, deve-se usar o operador & antes do nome da variável, enviando seu endereço de memória.
    - No cabeçalho da função, o parâmetro deve ser declarado como um ponteiro, indicando que receberá um endereço.
    - Dentro da função, utiliza-se o operador de indireção * para acessar ou modificar o conteúdo apontado pelo ponteiro. Ou seja, o valor real da variável.
  - Essa abordagem permite que a função modifique efetivamente a variável usada na chamada.

Funções

```c
#include <stdio.h>

void alterar(int *ptr) {
    *ptr = 50;
}

int main() {
    int x = 10;

    printf("Antes da função: x = %d\n", x);

    alterar(&x);

    printf("Depois da função: x = %d\n", x);

    return 0;
}
```

Qual a saída do programa?

Funções

```c
#include <stdio.h>

void alterar(int *ptr) {
    *ptr = 50;
}

int main() {
    int x = 10;

    printf("Antes da função: x = %d\n", x);

    alterar(&x);

    printf("Depois da função: x = %d\n", x);

    return 0;
}
```

```
Antes da função: x = 10
Depois da função: x = 50
```

INSTITUTO FEDERAL
Pernambuco

Funções

INSTITUTO FEDERAL
Pernambuco

- Passagem de parâmetro por referência:
  - Em C, a passagem de vetores para funções é sempre por referência.
  - Isso acontece porque, ao enviar um vetor como argumento, o que realmente é passado para a função é o endereço do primeiro elemento e não uma cópia de todo o vetor.
  - Assim, qualquer alteração feita dentro da função afeta diretamente o vetor original.
  - No cabeçalho da função, existem duas formas principais de declarar o parâmetro que representa o vetor:

```c
void funcao(int v[]){

}
```

Notação de vetor

```c
void funcao(int *v){

}
```

Notação de ponteiro

Funções

INSTITUTO FEDERAL
Pernambuco

```c
void dobrarVetor(int v[], int tamanho) {
    for (int i = 0; i < tamanho; i++) {
        v[i] = v[i] * 2;
    }
}
```

```c
void dobrarComPonteiro(int *p, int tamanho) {
    for (int i = 0; i < tamanho; i++) {
        p[i] = p[i] * 2;
    }
}
```

Funções

INSTITUTO FEDERAL
Pernambuco

- Um protótipo “prototype” é uma declaração antecipada de uma função.
- Ele informa ao compilador:
  - nome da função;
  - tipo de retorno;
  - quantidade de parâmetros;
  - tipos dos parâmetros.
- Sintaxe: tipo nome(parametros);
- Exemplo:

```
int soma(int a, int b);
```

- O compilador lê o código de cima para baixo. Se uma função for chamada antes de ser definida, o compilador precisa saber previamente que ela existe, o que ela retorna e quais os parâmetros recebe.

Funções

- Problema sem protótipo:

```c
#include <stdio.h>

int main() {

    int r = soma(2, 3);

    printf("%d\n", r);

    return 0;
}

int soma(int a, int b) {
    return a + b;
}
```

Isso pode gerar erro ou warning.

INSTITUTO FEDERAL
Pernambuco

Funções

- Problema sem protótipo:

```c
#include <stdio.h>

int main() {

    int r = soma(2, 3);

    printf("%d\n", r);

    return 0;
}

int soma(int a, int b) {
    return a + b;
}
```

```c
#include <stdio.h>

int soma(int a, int b);

int main() {

    int r = soma(2, 3);

    printf("%d\n", r);

    return 0;
}

int soma(int a, int b) {
    return a + b;
}
```

INSTITUTO FEDERAL
Pernambuco

Exercícios

INSTITUTO FEDERAL
Pernambuco

1. Elabore uma função que receba três notas de um aluno como parâmetro e uma letra. Se a letra for ‘A’, a função deve calcular a média aritmética das notas do aluno; se a letra for ‘P’, deverá calcular a média ponderada, com pesos 5, 3 e 2. Retorne a média calculada para o programa principal.
2. Escreva uma função que, dado um número real passado como parâmetro, retorne a parte inteira e a parte fracionária desse número por referência.
3. Faça um programa que implemente uma função com três parâmetros, utilizando passagem por referência, para armazenar a soma dos três valores na variável do primeiro parâmetro.

ALGORITMOS E ESTRUTURA DE DADOS

Curso de Engenharia de Software
Lucas Sampaio Leite