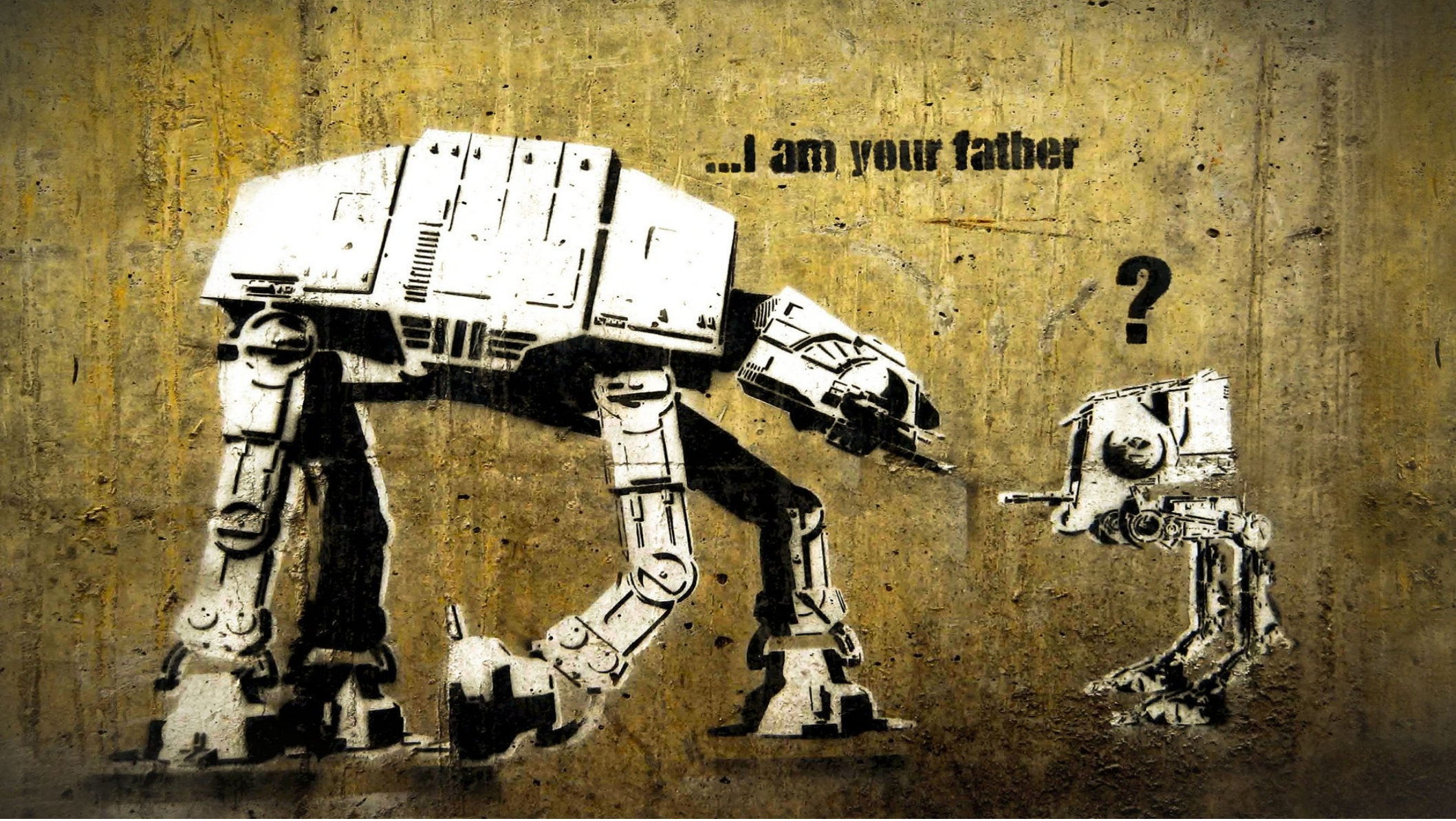
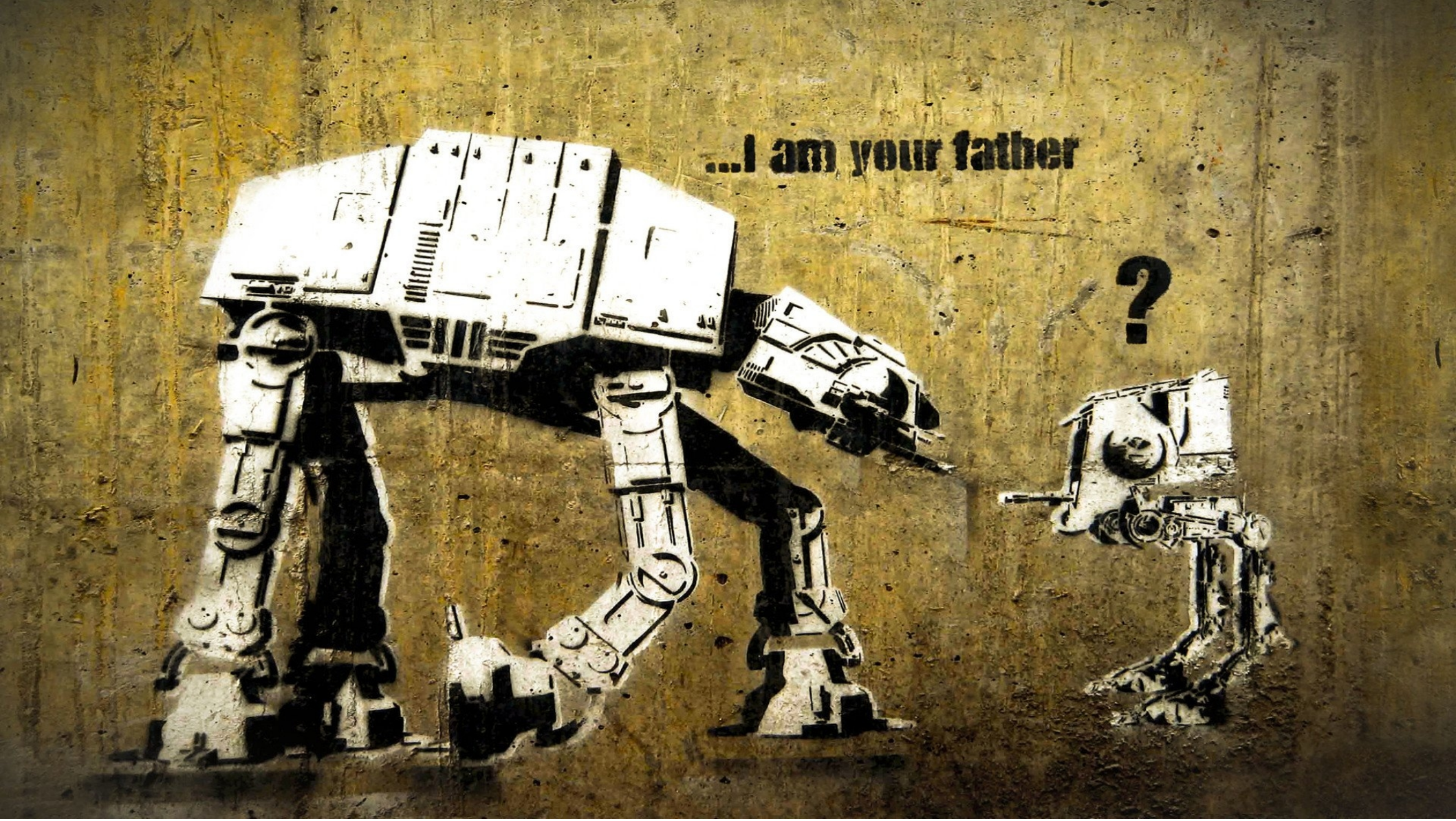


Programação Orientada a Objetos

Prof. Alexandre Krohn



Aula 0 – Revisão sobre Funções em C

Prof. Alexandre Krohn

Roteiro

- Revisão do Conceito de Funções em linguagem C
- Exercícios

Revisão: Uso de Funções em Linguagem C

- As funções são blocos de código reutilizáveis que facilitam a organização do programa.

- Melhoram a modularização e permitem a reutilização de código.

Modularização

- Divide um programa grande em partes menores e mais gerenciáveis.

- Melhora a legibilidade e facilita a manutenção do código.

Um exemplo em C

```c
#include <stdio.h>

void saudacao() {
    printf("Ola! Bem-vindo ao programa.\n");
}

int main() {
    saudacao();
    return 0;
}
```

Declaração e Definição de Funções

- Declaração: Define o nome, tipo de retorno e parâmetros da função.

- Definição: Implementa a lógica da função.

Exemplo

```c
#include <stdio.h>

// Declaração
int soma(int a, int b);

int main() {
    int resultado = soma(3, 4);
    printf("Soma: %d\n", resultado);
    return 0;
}

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

Protótipos de Funções

- Um protótipo de função é uma declaração antecipada de uma função antes de sua definição.

- Ele informa ao compilador sobre o nome, tipo de retorno e parâmetros da função.

- É útil para modularização e quando a função é definida após sua utilização no código.

Exemplo sem protótipo (erro possível)

```c
#include <stdio.h>

int main() {
    int resultado = soma(3, 4);
    // Erro: soma ainda não foi declarada
    printf("Soma: %d\n", resultado);
    return 0;
}

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

A função **soma** é chamada aqui

Mas só é declarada algumas linhas depois

Exemplo com protótipo (correto)

```c
#include <stdio.h>

int soma(int a, int b); // Protótipo da função

int main() {
    int resultado = soma(3, 4);
    printf("Soma: %d\n", resultado);
    return 0;
}

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

O uso do protótipo evita erros de compilação e melhora a legibilidade do código.

Parâmetros em Funções

Os parâmetros são valores passados para uma função no momento de sua chamada. Eles permitem que a função receba diferentes entradas e, assim, produza diferentes saídas.

- Cada parâmetro é declarado com um tipo e um nome dentro dos parênteses na declaração da função.
- Eles atuam como variáveis locais dentro da função.
- Os parâmetros podem ser usados para receber valores do programa principal e realizar operações específicas dentro da função.

Exemplo de parâmetros

```c
#include <stdio.h>

void exibirMensagem(char nome[]) {
    printf("Olá, %s! Seja bem-vindo.\n", nome);
}

int main() {
    exibirMensagem("Carlos");
    return 0;
}
```

No exemplo acima, a função exibirMensagem recebe um parâmetro do tipo char nome[], que representa uma string.
Ao chamarmos a função no main(), passamos o nome "Carlos" como argumento, que é utilizado dentro da função para exibir a mensagem.

Passagem de Parâmetros

Parâmetros de funções podem ser passados de duas formas:

- Por valor
- Por referência

Passagem por Valor

- Os parâmetros são copiados para variáveis locais dentro da função.

- Mudanças feitas não afetam o valor original.

Passagem por Valor : Exemplo

```c
#include <stdio.h>

void dobrar(int x) {
    x = x * 2;
}

int main() {
    int num = 5;
    dobrar(num);
    printf("Numero: %d\n", num); // Continua 5
    return 0;
}
```

Passagem por Referência (Ponteiros)

- Os parâmetros são passados por endereço, permitindo alterações permanentes.
- Mudanças feitas no endereço apontado afetam o valor original.

Passagem por Referência : Exemplo

```c
#include <stdio.h>

void dobrar(int *x) {
    *x = *x * 2;
}

int main() {
    int num = 5;
    dobrar(&num);
    printf("Numero dobrado: %d\n", num); // Agora será 10
    return 0;
}
```

Tipos de retorno

- Uma função em C pode retornar somente um valor, de qualquer tipo que seja indicado
- Precisando retornar mais do que um valor, pode-se utilizar structs

Exemplo de função retornando um float

```c
#include <stdio.h>

float dividir(int a, int b) {
    return (float)a / b;
}

int main() {
    float resultado = dividir(5, 2);
    printf("Divisao: %.2f\n", resultado);
    return 0;
}
```

Funções sem retorno

- Uma função pode não retornar nada para o programa que a invocou. Nesse caso declaramos o tipo de retorno como void

Exemplo: Funções sem retorno

```c
#include <stdio.h>

void mostra(int n) {
    printf("Número : %d\n", n);
}

void main() {

    int a;

    printf("Informe um número\n");
    scanf("%d", &a);

    mostra(a);
}
```

Dúvidas?

Atividades

- Execute as atividades presentes no documento

  00.Lista.de.Exercícios.Revisão.pdf

Próximos passos

- Princípios da Orientação a Objetos

Referências

- Victorine Viviane Mizhari, **Treinamento em Linguagem C**