Estrutura de Dados 1

Prof. Igor Calebe Zadi
igor.zadi@ifsp.edu.br

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

1. Programação modular

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

1. Módulos

- Partes independentes de um programa que podem ser **desenvolvidas, testadas e mantidas** separadamente.
- Facilitam a reutilização e organização do código.

- **Vantagens da modularização:**
  - Organização lógica do código.
  - Redução da complexidade de programas grandes.
  - Facilidade na manutenção e correção de erros.
  - Reutilização de código em diferentes projetos.
- **Exemplos práticos:** Comparação com tarefas do dia a dia, como dividir um projeto em etapas.

Funções

- Bloco de código que executa uma tarefa específica e retorna um valor.
- Estrutura básica de uma função em C: tipo de retorno, nome, parâmetros e corpo.
- Exemplo de função com retorno de valor:

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

Procedimentos

- Bloco de código que executa uma tarefa, mas não retorna um valor.
- A palavra reservada ***void*** indica o módulo que não retornará um valor.

```
void imprimirMensagem() {
    printf("Olá, mundo!");
}
```

Passagem de parâmetros

- **Por Valor:**
  - Cópia do valor original é passada.
  - Alterações dentro do módulo não afetam o valor original.

```
void incrementar(int x) {
    x++;
}
```

Passagem de parâmetros

- **Por Referência (uso de ponteiros):**
  - O endereço da variável é passado, permitindo alteração direta do valor original.

```
void incrementar(int *x) {
    (*x)++;
}
```

Passagem de parâmetros

Vetores como parâmetro

- sempre passados por referência.

```
void imprimirVetor(int vetor[], int tamanho) {
    for (int i = 0; i < tamanho; i++) {
        printf("%d ", vetor[i]);
    }
}
```

Passagem de parâmetros

Matrizes como parâmetro

- é necessário informar todas as dimensões, exceto a primeira.

```
void imprimirMatriz(int matriz[][3], int linhas) {
    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }
}
```

Passagem de parâmetros

Ponteiros como parâmetro

```
void trocarValores(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Retorno do tipo ponteiro genérico

- O retorno **void *** permite retornar ponteiros para qualquer tipo de dado, sendo amplamente utilizado para estruturas de dados genéricas.
- É necessário realizar ***cast*** (conversão de tipo) para utilizar o dado corretamente após o retorno.

```
void * retornaPonteiro(int *valor) {
    return (void *)valor;
}
```

```
int main() {
    int numero = 42;
    int *ptr = (int *)retornaPonteiro(&numero);
    printf("Valor apontado: %d", *ptr);
    return 0;
}
```

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

Recursividade

INSTITUTO FEDERAL
São Paulo
Campus Catanduva

Recursividade - conceitos

- Fundamental em Matemática e Ciência da Computação
  - Um programa recursivo é um programa que chama a si mesmo
  - Uma função recursiva é definida em termos dela mesma
    - Exemplos
      - Função fatorial, Árvore
  - Conceito poderoso
    - Define conjuntos infinitos com comandos finitos

Recursividade - conceitos

- A recursividade é uma estratégia que pode ser utilizada sempre que o cálculo de uma função para o valor ***n***, pode ser descrita a partir do cálculo desta mesma função para o termo anterior *(n-1)*.

Recursividade - conceitos

- **Exemplo – Função Fatorial:**
  O fatorial de um número natural ***n***, indicado por ***n!***, é o produto de todos os inteiros positivos de ***n*** até 1:
  - $n! = n \times (n-1) \times (n-2) \times \cdots \times 1$
- Podemos observar que o fatorial de **n−1** é:
  - $(n-1)! = (n-1) \times (n-2) \times \cdots \times 1$
- Assim, podemos reescrever a fórmula do fatorial de **n** de forma recursiva:
  - $n! = n \times (n-1)!$

Recursividade - conceitos

Exemplo – Função Fatorial:

Essa definição recursiva possui dois componentes essenciais:

1. **Caso base:** quando **n=0**, definimos 0!=1.
2. **Passo recursivo:** quando **n>0**, o cálculo depende do fatorial de um número menor (n−1).

- A recursividade facilita a implementação da função fatorial, tornando o código mais elegante e próximo da definição matemática.

Recursividade - definição

- Definição: dentro do corpo de uma função, chamar novamente a própria função
  - **recursão direta**: a função A chama a própria função A
  - **recursão indireta**: a função A chama uma função B que, por sua vez, chama A

Recursividade

- Nenhum programa nem função pode ser exclusivamente definido por si
  - Um programa seria um *loop* infinito
  - Uma função teria definição circular
- Condição de parada
- Permite que a função pare de se executar
  - $F(x) > 0$ onde x é decrescente

Recursividade

- Para **cada chamada** de uma função, recursiva ou não, os **parâmetros e as variáveis locais** são empilhados na **pilha** de execução.

Recursividade - execução

- Internamente, quando qualquer chamada de função é feita dentro de um programa, é criado um **Registro de Ativação** na Pilha de Execução do programa
- O registro de ativação armazena os parâmetros e variáveis locais da função bem como o "**ponto de retorno**" no programa ou subprograma que chamou essa função.
- Ao final da execução dessa função, o registro é desempilhado e a execução volta ao programa ou subprograma que chamou a função

Recursividade - exemplo

```
int fat(int n) {
    if (n<=0)
        return 1;
    else
        return n * fat(n-1);
}
```

Recursividade - exemplo

Série de Fibonacci:

$F_n = F_{n-1} + F_{n-2}$ $n > 2$,

$F_0 = 0$ $F_1 = 1$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

Recursividade - exemplo

```
int fib(int n) {
    if (n<=0)
        return 0;
    else if(n == 1)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}
```

Recursividade - execução

- Implemente uma função recursiva em C para inverter uma string.
  - Exemplo:
    - Entrada: "ABCDE"
    - Saída: ”EDCBA"
- Além da implementação, responda às seguintes perguntas:
- Qual é o caso base e o caso recursivo da função?
- Explique como a pilha de chamadas recursivas é utilizada para alcançar a inversão da string.

Observações sobre o material eletrônico

- O material ficará disponível na pasta compartilhada que é acessada sob convite
- O material foi elaborado a partir de diversas fontes (livros, internet, colegas, alunos etc.)
- Alguns trechos podem ter sido inteiramente transcritos a partir dessas fontes
- Outros trechos são de autoria própria
- Esta observação deve estar presente em qualquer utilização do material fora do ambiente de aulas do IFSP - Catanduva

INSTITUTO FEDERAL
São Paulo
Campus Catanduva