Programação Imperativa e Funcional

Funções e procedimentos

Diego Bezerra

dfb2@cesar.school

C.E.S.A.R school

Na aula anterior...

- Discussão sobre Arrays em C
- **Lista avaliativa usando o Beecrowd**

Objetivos da aula

- Funções
- Procedimentos

Funções e Procedimentos: Definições

- A **modularização** define estruturas que agrupam um conjunto de comandos que executam tarefas menores (**subprograma**)
  - Evita que os blocos do programa fiquem grandes demais e, consequentemente, mais difíceis de entender
  - Separa o programa em partes que possam ser **reutilizáveis (reaproveitamento)** e logicamente compreendidos de forma isolada
  - Melhora a **legibilidade e a manutenabilidade do código,** permitindo alterar apenas um trecho

Funções e Procedimentos: Definições

- Blocos de código que podem ser nomeados e chamados/invocados em um programa
- Programas grandes e complexos devem ser construídos definindo funções e procedimentos
- Os comandos são executados quando a função ou procedimento são chamadas na função **main**

```
#include <stdio.h>
int main() {
    int x;
    scanf("%d", &x); // chamada do procedimento scanf
    return 0;
}
```

Funções

- São subprogramas que **retornam um valor** ao final de sua execução

```c
#include <stdio.h>
#include <math.h> // necessária para funções matemáticas

int main() {

    int x = sqrt(4);
    printf("%d\n", x); // chamada do procedimento scanf

    return 0;
}
```

Função: Declaração

- **Toda função deve ter um tipo** e determina qual o tipo de **valor de retorno**
- Os parâmetros de uma função determinam qual será o seu comportamento. Ex.: função matemática
- Pode não ter parâmetros, basta não informá-los
- Funções devem ser declaradas antes de seu uso e fora do programa principal (**main**)

```
tipo nome (<lista de parametros>) {
    //comandos
    return valor_retorno;
}
```

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

Função: Invocando

- Podemos invocar funções atribuindo seu valor a uma variável
- Na verdade, podemos invocar uma função em “qualquer lugar”
- Os argumentos passados para a função necessariamente possuem o mesmo tipo
- Os valores permanecerão intocados durante a execução da função

```
int main() {
    int a = 10;
    int b = 20;

    int resultado = soma(a, b);
    soma(a, b);
}
```

Procedimentos

- O tipo **void** é um tipo especial, utilizado principalmente em procedimentos
- Representa tipo nenhum, armazena conteúdo indeterminado
- Em C, **procedimentos são funções do tipo void** e necessariamente não retornam nada

```
void imprime(int numero) {printf("Numero: %d\n", numero);}

int main() {
    imprime(10);
    return 0;
}
```

Procedimento: Declaração

- **Possui tipo de retorno como void**
- Os parâmetros de uma procedimento determinam qual será o seu comportamento.
- Pode não ter parâmetros, basta não informá-los
- Funções devem ser declaradas antes de seu uso e fora do programa principal (**main**)

```
void nome (<lista de parametros>) {
    //comandos
}
```

```
void imprimir (int a, int b) {
    printf("%d ", a + b);
}
```

A função main

- Função especial que é invocada automaticamente pelo programa quando inicia sua execução
- O comando return informa ao sistema operacional se o programa funcionou corretamente ou não
- O padrão é que retorne o valor zero caso tenha funcionado corretamente
- Uma boa prática é declarar as demais funções antes da função main

Variáveis globais e locais

- Uma variável é chamada **local se foi declarada dentro de um escopo específico** (como o de uma função)
  - Só podem ser manipuladas dentro do subprograma
  - **Invisíveis em qualquer outra parte do programa**
- Uma variável **global** é chamada global se foi declarada fora de qualquer função e **está visível em todo o programa**
  - **Qualquer alteração feita irá refletir em todas as funções do programa**
  - Não pode ser redeclarada em outra função, o que geraria um erro de compilação

Variáveis globais e locais

```c
#include <stdio.h>

int a; // variável global

int main() {
    int quadrado = 0; // variável local
    a = 0;
    return 0;
}
```

Escopo de variáveis

- Se temos escopos diferentes, variáveis de mesmo nome podem existir ao longo do código

```c
#include <stdio.h>

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

int main() {
    int resultado = 0;
    return 0;
}
```

Exercícios

- **Defina uma função em que dado o nome de um aluno e suas 3 notas, calcule a média e informe se ele foi aprovado (média >= 6.0)**
  - **Uma função calcular média deve ser definida**
  - **Um procedimento deve exibir o resultado**
- Defina uma função que calcula e retorna o fatorial de um número
- Defina uma função que busca e retorna o menor elemento de um vetor
- Defina uma função que ordena todos os elementos de um vetor
- **Crie uma função que verifica se uma palavra é um palíndromo**

Referências

- Rangel Netto, J. L. M., Cerqueira, R. D. G., & Celes Filho, W. (2004). **Introdução a estrutura de dados: com técnicas de programação em C**.