CENTRO UNIVERSITÁRIO UNIFAVIP | WYDEN

Estrutura de Dados
Bibliotecas em C

Wolney Henrique Queiroz Freitas

Roteiro de Aula

- Funções e Procedimentos
- Strings
- Bibliotecas

Funções

- Conjunto de comandos agrupados em um bloco, que recebe um nome e através deste pode ser evocado;

- Modularização e Reuso.

Funções

- Propriedades de uma função:
  - Nome;
  - Parâmetros;
  - Retorno.

Formato Geral das Funções

```
Tipo_Retorno_Da_Funcao  Nome_Da_Funcao (Lista_de_Parametros)
{
// corpo da função
}
```

Funções

- Exemplo de função:

```c
#include <stdio.h>

int somar(int numero1, int numero2){
    int resultado = numero1 + numero2;
    return resultado;
}

main(){
    int soma = somar(2, 4);
    printf("Resultado da soma: %d", soma);
}
```

Procedimentos

- Exemplo de procedimento, que é uma função que não retorna nenhum valor:

```c
#include <stdio.h>

void somarExibirResultado(float n1, float n2){
    printf("%2.f + %2.f = %2.f", n1, n2, n1 + n2);
}

main() {
    somarExibirResultado(10, 5);
}
```

Protótipos de Funções

- Por padrão e para a maioria dos compiladores, as funções devem ser definidas antes de serem utilizadas:

```c
#include <stdio.h>

void somarExibirResultado(float n1, float n2){
    printf("%2.f + %2.f = %2.f", n1, n2, n1 + n2);
}

main() {
    somarExibirResultado(10, 5);
}
```

Protótipos de Funções

- Se quisermos implementar as funções após a função principal do programa, podemos utilizar protótipos:

```c
#include <stdio.h>

// Protótipo da função:
void somarExibirResultado(float n1, float n2);

main() {
    somarExibirResultado(10, 5);
}

void somarExibirResultado(float n1, float n2){
    printf("%2.f + %2.f = %2.f", n1, n2, n1 + n2);
}
```

Escopo de Variáveis

- Por escopo de uma variável entende-se o bloco de código onde esta variável é válida. Com base nisto, temos as seguintes afirmações:

  - As variáveis valem no bloco que são definidas;
  - As variáveis definidas dentro de uma função recebem o nome de variáveis locais;
  - Os parâmetros de uma função valem também somente dentro da função;
  - Uma variável definida dentro de uma função não é acessível em outras funções, MESMO ESTAS VARIÁVEIS TENHAM NOME IDÊNTICOS.

Escopo de Variáveis

```c
#include <stdio.h>

void multiplicar(float a, float b);

main() {
    float n = 3;
    multiplicar(n, 4);
    printf("\nn = %f", n);
}

void multiplicar(float a, float b){
    float n = a * b;
    printf("%f * %f = %f", a, b, n);
}
```

Vamos praticar

1) Escreva um procedimento que receba um número inteiro e imprima o mês correspondente ao número.

2) Escreva uma função que receba dois números inteiros x e y. Essa função deve verificar se x é divisível por y. No caso positivo, a função deve retornar 1 (um), caso contrário a função deve retornar 0 (zero).

3) Crie uma função que realize a conversão de Polegadas (pol) para Centímetros (cm), onde pol, é uma variável, e deve ser passada como parâmetro e cm, é uma variável, e deve ser retornado da função. Sabe-se que 1 (uma) polegada equivale a 2.54 centímetros.

4) Crie uma função que receba um valor e um percentual de aumento e realize o cálculo e retorne o valor reajustado.

Strings em C

- O termo string serve para identificar uma sequência de caracteres;
- Na prática, as strings são usadas para representar textos;
- Em linguagem C, ao contrário de outras linguagens, não existe um tipo de dados string nativo;
- Para representar uma string em C, devemos criar um array de caracteres, ou seja um vetor do tipo char.

Strings em C

```c
#include <stdio.h>

main() {
    // Declarando Strings
    char nome[61];
    char nome2[61] = "Fulano";
    char nome3[61] = {'F','u','l','a','n','o'};
    char nome4[] = "Fulano";

    printf("%s", nome);
    printf("\n%s", nome2);
    printf("\n%s", nome3);
    printf("\n%s", nome4);

}
```

Lendo Strings

- Usando scanf
  - A função scanf permite fazer leitura de strings usando %s;
  - Em relação ao uso de scanf para armazenar string devemos observar duas coisas:
    - A função scanf realiza a leitura até encontrar um espaço, depois encerra a leitura e coloca o caracter terminador \0;
    - A variável que vai armazenar a string não necessita ser precedida por &.

Lendo Strings

```c
#include <stdio.h>

main() {

    char nome[61];

    printf("Informe seu nome: ");
    fflush(stdin);
    scanf("%s", nome);
    printf("O nome armazenado foi: %s", nome);

}
```

Lendo Strings

- Usando gets
  - Esta função armazena tudo que foi digitado, inclusive os espaços, até que a tecla ENTER seja pressionada.

Lendo Strings

```c
#include <stdio.h>

main() {

    char nome[61];

    printf("Informe seu nome: ");
    fflush(stdin);
    gets(nome);
    printf("O nome armazenado foi: %s", nome);

}
```

Vamos Praticar

- Escreva um programa que leia o nome e o cargo de um funcionário (cargo é um código representado por número inteiro, onde: 1- gerente e 2 – administrativo). Se o funcionário for um gerente, ler também o nome do departamento que ele gerencia. No fim, o programa deve todas as informações lidas.

Bibliotecas em C

- As bibliotecas em C são rotinas padronizadas da linguagem que contém operações comuns como tratamento de entrada/saída e cadeia de caracteres.
  - Biblioteca padrão ANSI C.

Bibliotecas em C

- O nome e as características de cada função estão em um arquivo chamado cabeçalho, mas a implementação das funções está em um arquivo separado;

- Cada compilador C possui sua implementação da biblioteca padrão C.

Bibliotecas em C

- A biblioteca padrão ANSI C consiste de 24 cabeçalhos, cada um contendo uma ou mais declarações de funções, tipos de dados e macros;

- Ela fornece um conjunto básico de operações matemáticas, manipulação de cadeias de caracteres, conversão de tipos de dados e entrada e saída de arquivo e da tela.

Bibliotecas em C

- Exemplos de bibliotecas:
  - <stdio.h> - Manipulação de entrada/saída;
  - <math.h> - Funções matemáticas comuns em computação;
  - <string.h> - Tratamento de cadeia de caracteres;
  - <stdlib.h> - Diversas operações, incluindo conversão, geração de números pseudo-aleatórios, alocação de memória, controle de processo, sinais, busca e ordenação.

Bibliotecas de Strings

- A biblioteca string.h contém uma série de funções para manipular strings.

- Operações que conseguimos fazer utilizando funções da biblioteca de Strings:

  - Copiar strings;
  - Concatenar strings;
  - Descobrir o tamanho de uma string;
  - Comparar strings;

Bibliotecas de Strings

- Copiar Strings:

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

main() {

    char nome[15];
    char teste[15] = "Teste";

    // Copia o valor de uma String em outra
    //strcpy(string_destino, string_origem);
    strcpy(nome, "Fulano de Tal");


    printf("Nome = %s", nome);
}
```

Bibliotecas de Strings

- Copiar Strings especificando o tamanho:

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

main() {

    char nome[15] = "Fulano de Tal";
    char nome2[7];


    strncpy(nome2, nome, 6);


    printf("Nome2 = %s", nome2);
}
```

Bibliotecas de Strings

- Concatenar Strings:

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

main() {

    char msg[10] = "Hello";

    //Concatena a string " World!" com o conteúdo da string msg
    strcat(msg, " World!");

    //Será exibido Hello World!
    printf("str = %s\n", msg);

}
```

Bibliotecas de Strings

- Concatenar Strings especificando o tamanho:

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

main() {

    char msg[25] = "Disciplina";
    char msg2[25] = " de Programacao em C";

    strncat(msg, msg2, 15);

    printf("str = %s\n", msg);

}
```

Bibliotecas de Strings

- Verificar tamanho da String:

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

main() {

    char str[20] = "Linguagem C";
    int tamanho;

    // Retorna um int que representa
    // o tamanho da String
    tamanho = strlen(str);

    // O tamanho exibido é o tamanho do conteúdo da String
    printf("O tamanho da string %s vale %d\n", str, tamanho);

}
```

Bibliotecas de Strings

- Comparar o conteúdo de duas strings:

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

main() {

    char nome1[20] = "Fulano";
    char nome2[20] = "Beltrano";

    // Retorna um inteiro
    int retorno = strcmp(nome1, nome2);

    // 0: conteúdo das strings são iguais
    // < 0: conteúdo da string1 é menor do que string2
    //> 0: conteúdo da string1 é maior do que string2
    if(retorno == 0){
        printf("Nomes iguals.");
    }else{
        printf("Nomes diferentes.");
    }

}
```

Bibliotecas de Strings

- Comparar o conteúdo de duas strings especificando o tamanho:

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

main() {

    char nome1[20] = "Fulano";
    char nome2[20] = "Fulano Beltrano";

    int retorno = strncmp(nome1, nome2, 6);

    if(retorno == 0){
        printf("Nomes iguals.");
    }else{
        printf("Nomes diferentes.");
    }

}
```

Vamos Praticar

1) Escreva um programa que leia o primeiro nome, o nome do meio e o ultimo nome de uma pessoa, concatene e apresente o nome das seguintes formas:
   - Nome completo;
   - Primeiro nome + ultimo nome.

2) Leia uma palavra e informe se ela é uma palavra pequena (menos de 5 caracteres) ou uma palavra grande (mais de 20 caracteres).

3) Escreva um programa que peça o login e a senha de um usuário, e informe se ele foi autenticado ou se o login ou a senha estão incorretos. Considere os valores de acesso do usuário como:
   - Login: user123
   - Senha: abc123

Biblioteca Math

- Fornece um conjunto de funções para operações matemáticas, tais como funções trigonométricas, hiperbólicas, logaritmos, potência e arredondamentos;

- Todas as funções da biblioteca math.h retornam um valor de ponto flutuante.

Biblioteca Math

- Arredondamento:

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

main() {

    float numero = 1.67;

    float arredondaParaCima = ceil(numero);
    float arredondaParaBaixo = floor(numero);

    printf("%2.f", arredondaParaCima); // imprime 2
    printf("%2.f", arredondaParaBaixo); // imprime 1

}
```

Biblioteca Math

- Calcular raiz quadrada:

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

main() {

    float numero = 36;

    float raizQuadrada = sqrt(numero);

    printf("%2.f", raizQuadrada); // imprime 6

}
```

Biblioteca Math

- Potenciação:

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

main() {

    float numero = 4;

    // (Número, Expoente)
    float potenciacao = pow(numero, 2);

    printf("%2.f", potenciacao); // imprime 16

}
```

Vamos Praticar

1) Escreva um programa para ler 3 notas, calcular a média e arredonda-la para cima, e informe se o aluno foi aprovado ou reprovado (aprovado: media >=5).

2) Escreva um programa para ler 4 números, calcular o cubo de cada um, somar os resultados, e exibir a raiz quadrada do resultado da soma.

Criar Bibliotecas

- Quando desenvolvemos diversas funções utilitárias e reutilizáveis para nossos projetos em C, podemos criar nossas próprias bibliotecas para agrupar essas funções;

- A utilidade de criar bibliotecas e headers é possuir arquivos que contenham diversas funções específicas, separadas e organizadas por assuntos.

Criando uma Biblioteca

- Crie um arquivo vazio;
- Salve esse arquivo com qualquer nome válido e com extensão .h
  - Exemplo: biblioteca.h
- Salve na mesma pasta em que guarda seu código-fonte de C;
- Escreva funções no arquivo, como uma que mostre uma mensagem na tela, com um simples printf e salve o arquivo.

Criando uma Biblioteca

- Arquivo fonte de uma biblioteca:

biblioteca.h

```
void helloWorld(){
    printf("Hello World!");
}
```

Criando uma Biblioteca

- Agora volte no seu projeto, que tem extensão .c

- Inclua sua biblioteca junto das demais:

#include “biblioteca.h"

Criando uma Biblioteca

- Incluindo a biblioteca criada no código fonte C:

```c
#include <stdio.h>
#include "biblioteca.h"

main() {

    helloWorld();

}
```

Vamos Praticar

1) Crie, inclua e utilize uma biblioteca com as seguintes funções:

- float somar(float n1, float n2);
- float subtrair(float n1, float n2);
- float multiplicar(float n1, float n2);
- float dividir(float n1, float n2);
- float numeroAoQuadrado(float numero);

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