FUNÇÕES

PROF. FÁBIO KRAVETZ

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

2º SEMESTRE/2024

Universidade POSITIVO

FUNÇÕES

- **Funções** são **utilizadas** quando um **bloco** de **código** se **repete** na **resolução** de um **problema**, pois uma sub-rotina é escrita apenas uma vez e pode ser chamada sempre que necessário;
- **Problemas complexos** exigem **algoritmos complexos** - mas **sempre** é **possível dividir** um problema grande em **problemas menores;**
- Esses problemas menores, ou sub-rotinas, podem efetuar diversas operações e facilitam a resolução de um problema por serem bastante específicos.

FUNÇÕES

- Uma função é um subprograma que auxilia o programa principal através da realização de uma determinada subtarefa;

- As funções são chamadas dentro do corpo do programa principal como se fossem comandos. Após seu término, a execução continua a partir do ponto onde foi chamado;

- É essencial compreender que a chamada de uma função simplesmente **gera um desvio provisório no fluxo de execução**.

FUNÇÕES

- Uma função, além de executar uma determinada tarefa **pode retornar um valor para quem a chamou**, que é o resultado da sua execução;

- Por este motivo, a chamada de uma função que retorna algum valor, aparece no corpo do programa principal como uma expressão e não como um comando.

FUNÇÕES

- Uma chamada de função, pode ser realizada sem a passagem de nenhum valor (parâmetro);

- Ao se chamar uma função, também é possível enviar determinadas informações que recebem o nome de parâmetros (valores que, na linha de chamada, ficam entre parênteses e que estão separados por vírgulas);

- A quantidade de parâmetros, sua ordem e seus respectivos tipos não podem mudar: devem estar de acordo com o que foi especificado na sua correspondente declaração (chamada).

FUNÇÕES

- Função main():
  - Ponto de partida de um programa desenvolvido em linguagem C;
  - A função main pode receber parâmetros direto do terminal;
  - Os parâmetros da função main são utilizados na execução do programa através da linha de comando;
  - **int argc:** indica o número de argumentos passados para a função main. O primeiro argumento é o nome do programa, por isso argc é sempre no mínimo 1;
  - **char *argv[]:** é um ponteiro para um vetor de strings (cadeia de caracteres), onde cada string é um dos parâmetros da linha de comando.

Exemplo

```c
int main(int argc, char *argv[])
{

    setlocale(LC_ALL, "Portuguese");

    int codigo;
    float area;

    printf("Cálculo de áreas: \n");
    printf("1 - Retângulo \n");
    printf("2 - Circunferência \n");
    scanf("%i", &codigo);

    if(codigo == 1)
    {
        area = Retangulo();
        printf("\nA área do Retângulo é: %.2f", area);
    }

    if(codigo == 2)
    {
        area = Circunferencia();
        printf("\nA área da Circunferência é: %.2f", area);
    }

}
```

A execução do programa sempre começa na função main.

A execução é desviada para a respectiva função, conforme a escolha do usuário

```c
float Retangulo()
{
    float base, altura, area;
    printf("Informe a base: ");
    scanf("%f",&base);
    printf("Informe a altura: ");
    scanf("%f",&altura);
    area = base * altura;
    return area;
}
float Circunferencia()
{
    float raio, area;
    const float pi = 3.1415;
    printf("Informe o valor do raio: ");
    scanf("%f",&raio);
    area = (4 * pi * pow(raio,2));
    return area;
}

int main(int argc, char *argv[])
{
    setlocale(LC_ALL, "Portuguese");
    int codigo;
    float area;
    printf("Cálculo de áreas: \n");
    printf("1 - Retângulo \n");
    printf("2 - Circunferência \n");
    scanf("%i", &codigo);

    if(codigo == 1) {
        area = Retangulo();
        printf("\nA área do Retângulo é: %.2f", area);
    }
    if(codigo == 2) {
        area = Circunferencia();
        printf("\nA área da Circunferência é: %.2f", area);
    }
}
```

Definição de uma Função válida para o cálculo da área.

O resultado gerado pela execução da Função é devolvido para quem usa a Função.

A execução é desviada para função.

Existe alguma coisa a melhorar neste código?

DEFINIÇÃO FUNÇÃO

- Formato Geral de definição:

```
cabeçalho
{
    declarações de variáveis;
    comandos ;
    return valor_retorno;
}
```

- Bloco de commandos (entre {}) – Código que é executado pela função;
- Para funções simples, que apenas devolve um resultado:
  - **cabeçalho:** **tipo_funcao nomeFuncao ( )**;
  - **tipo_funcao:** indica o tipo de valor_retorno;
  - tipo de resultado que a função deve retornar ao final de sua execução;
  - Comando **return** indica o final da execução da função e o valor a ser devolvido por ela – valor de retorno;
  - Uma função definida nesta forma simples não recebe dados, ou seja, não há parâmetros dentro dos parênteses.

DEFINIÇÃO FUNÇÃO

- Uma função pode ser definida antes ou depois da main()
  - Caso seja definida antes, não há passo adicional;
  - Caso seja definida depois, é preciso declarar o protótipo da função antes da main();
  - O protótipo consiste em informar o mesmo cabeçalho de definição da função seguido de ponto-e-vírgula;
    - **Exemplo: float Media();**
  - O importante é que ou a definição ou protótipo de uma função seja informada (escrita) no código-fonte **ANTES** da **PRIMEIRA VEZ** que ela é referenciada no código fonte de todo o programa.

FUNÇÕES – COMANDO RETURN

- **O comando RETURN tem alguns objetivos específicos:**
  - Sinalizar que a execução da função terminará:
    - **Nenhum commando informado após um return será executado;**
    - A execução do programa continua, a partir, do ponto onde a função foi utilizada/chamada.
  - Devolver um valor a quem chamou a função
    - Valor deve ter o mesmo tipo da função definido no cabeçalho da função;
    - Somente é possível devolver **UM ÚNICO VALOR**.

FUNÇÕES – CHAMADA

- Como o programa usa ou invoca a execução de uma função?
  - Através da chamada
  - **Exemplo: nomeFuncao ( );**

- A chamada da função pode ser feita em qualquer ponto de um programa onde seu valor de retorno será utilizado.
  - media = calculoMedia( );
  - O **valor retornado** pela função calculoMedia( ) será atribuído à variável media.

Escopo de Variáveis

- As variáveis possuem um tempo de vida que é o mesmo tempo de vida do programa e/ou da funcão/procedimento onde ela foi criada;
- As variáveis de uma função não se misturam com as variáveis de outra, ou seja:
  - É possível ter variáveis com mesmo nome em diferentes funções, ele sempre vai se referir à variável que foi criada no local.

Variáveis Globais: São variáveis definidas fora de qualquer função. Todas as funções podem acessar ou alterar o seu valor.

Variáveis Locais: São variáveis definidas dentro do escopo de uma função (dentro do corpo). Somente a função pode acessar a variável ou alterar o seu valor.

FUNÇÃO MAIN()

- O programa main() é uma função especial onde são realizadas chamadas para outras funções ou se é escrito somente o código estruturado;
  - Possui um tipo fixo (**int**);
  - É invocada automaticamente quando se inicia a execução do programa.
- O comando **return** que se coloca ao final do bloco main() informa ao Sistema Operacional se o programa terminou corretamente ou não;
- Por convenção:
  - **return 0** indica que tudo correu bem;
  - Qualquer valor diferente de zero indica ao SO que ocorreu erro no programa

FUNÇÕES – PASSO A PASSO

- O código escrito dentro da main() é executado até encontrar uma chamada de função;
- O programa é **“interrompido”** e o fluxo do programa passa para a função chamada;
- Caso haja parâmetros na função, os valores da chamada da função são copiados para os parâmetros no Código da função;
- Os comandos (código) da função são executados;
- Quando a função termina (seus comandos acabaram ou o comando return foi encontrado) o programa volta ao ponto em que foi interrompido para continuar sua execução normal;
- Se houver um comando return, o valor dele será copiado para a variável que foi escolhida para receber o retorno da função.

EXERCÍCIOS

1. Escreva um programa que solicite dois números do tipo inteiro distintos ao usuário e que apresente na tela o maior deles. Caso o usuário digite números iguais deve ser implementada uma lógica para que o segundo número seja solicitado novamente. Esse programa deve possuir uma função para verificar qual é o maior número.

2. Escreva um programa onde o usuário tenha a possibilidade de escolher em um menu uma das quatro operações matemáticas básicas (soma, subtração, multiplicação e divisão). A operação a ser realizada deve considerar somente dois números a serem informados pelo usuário. Além disso, deve-se ter uma opção de escolha em tal menu caso o usuário deseje sair do programa, sendo que, enquanto o usuário não opte por tal opção ele terá o menu a sua disposição para realizar as operações desejadas. Para a operação relativa a divisão, tem-se que o denominado precisa ser diferente de zero.

FUNÇÕES – PARÂMETROS

- Os parâmetros de uma função são o que o programador utiliza para **passer a informação** de um trecho do código para dentro da função;

- Basicamente, os parâmetros de uma função são uma **lista de variáveis**, separadas por vírgula, em que são especificados o tipo e o nome de cada variável passada para a função.

FUNÇÕES – PARÂMETROS

```c
#include <stdio.h>
float area_quadrado (float);
float area_triang_ret (float, float);

int main()
{   int opcao;
    float lado, lado1, lado2, area = 0;

    printf ("Cálculo de áreas. \n");
    printf ("0 - Área de um quadrado. \n");
    printf ("1 - Área de um triângulo ret. \n");
    printf ("Informe sua opção: \n");
    scanf ("%d", &opcao);

    switch(opcao)
    {
      case 0:  { printf("Informe o valor do lado \n");
                 scanf("%f", &lado);
                 area = area_quadrado (lado);
               } break;

      case 1:  { printf ("Informe o valor do primeiro lado \n");
                 scanf ("%f", &lado1);
                 printf ("Informe o valor do segundo lado \n");
                 scanf ("%f", &lado2);
                 area = area_triang_ret (lado1, lado2);
               } break;
 default:   {
                 printf ("Opcao inválida! \n");
            }
    }
    printf("O cálculo da área é: %f \n", area);
    return 0;
}

float area_quadrado (float lado)
{
    float result = 0.0;
    result = (lado * lado);
    return result;
}

float area_triang_ret (float lado1, float lado2)
{
    float result = 0.0;
    result = (lado1 * lado2) / 2;
    return result;
}
```

FUNÇÕES – PARÂMETROS

- Formato geral:

```
tipo_funcao nomeFuncao ( lista_parâmetros )
{
    declarações de variáveis;
    comandos;
    return valor_retorno;
}
```

- **tipo_funcao:** indica o tipo de valor_retorno
  - Resultado que a função deve retornar ao final de sua execução
  - **lista_parâmetros:** lista de declarações de variáveis usadas para passagem de valores para a função.
  - Variáveis separadas por vírgula, cada variável com seu tipo.

EXERCÍCIOS

3. Crie um programa onde o usuário informe três notas e imprima na tela a média destes valores. Utilize o conceito de função com seus respectivos parâmetros.

EXERCÍCIOS

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
float media(float n1, float n2, float n3)
{
    float media;
    media = (n1 + n2 + n3)/3;
    return media;
}
int main(int argc, char *argv[]) {

    setlocale(LC_ALL, "Portuguese");
    float nota1, nota2, nota3, mediaAluno;
    printf("Informe a primeira nota: \n");
    scanf("%f", &nota1);
    printf("Informe a segunda nota: \n");
    scanf("%f", &nota2);
    printf("Informe a terceira nota: \n");
    scanf("%f", &nota3);

    mediaAluno = media(nota1, nota2, nota3);

    printf("O valor da média é %.2f \n", mediaAluno);
}
```

FUNÇÕES – PARÂMETROS

- float media(float n1, float n2, float n3)
  - Ao ser chamada, esta função espera receber 3 valores nas três variáveis;
  - Três variáveis com valores do tipo float;
  - Estas variáveis serão usadas dentro do bloco de comandos da função.
- Os parâmetros são variáveis locais da função;
  - Recebem um valor inicial quando a função é chamada durante a execução do programa.

FUNÇÕES – CHAMADAS

- Ao chamar uma função, cada argumento deve ser uma variável do mesmo tipo do respectivo parâmetro na definição da função;
- Associação argumento-parâmetro é realizada pelaposição:
  - Valor do 1º argumento é copiado para o 1º parâmetro;
  - Valor do 2º argumento é copiado para o 2º parâmetro;
  - Valor do 3º argumento é copiado para o 3º parâmetro;

```
Float media(float n1, float n2, float n3)…{}

int main(){

float nota1

…

mediaAluno = media(nota1, nota2, nota3);

}
```

FUNÇÕES – VOID

- Formato geral:

```
tipo_funcao nomeFuncao ( lista_parâmetros )
{
    declarações de variáveis;
    comandos;
}
```

- **O tipo void é um tipo especial**
  - Uma variável desse tipo armazena um conteúdo indeterminado;
  - **Em geral, utiliza-se void para informar quando uma função não retorna nenhum valor;**
  - Variáveis separadas por vírgula, cada variável com seu tipo.

FUNÇÕES – VOID

- O código abaixo imprime um número que foi informado pelo usuário na função principal e passado como parâmetro para uma função do tipo VOID.

```
void imprime(int numero)
{
    printf("\nO número informada na função principal é %d \n", numero);
}
int main()
{
   setlocale(LC_ALL,"Portuguese");
   int numero;
   printf("Informe um número qualquer \n");
   scanf("%d", &numero);
   imprime(numero);
   return 0;
}
```

FUNÇÕES – VOID

```c
void calculaDivisao()
{
    float numero1, numero2, result;
    printf("Informe o primeiro número \n");
    scanf("%f", &numero1);

    printf("Informe o segundo número \n");
    scanf("%f", &numero2);

    if(numero2 != 0)
    {
        result = (numero1 / numero2);
        printf("O valor da divisão é %.2f \n", result);
    }
    else
    {
        printf("O denominador informado foi %.2f, logo a divisão não pode ser realizada", numero2);
    }
}
int main()
{
   setlocale(LC_ALL,"Portuguese");
   calculaDivisao();
   return 0;
}
```

FUNÇÕES – ERROS COMUNS

```
int somaValor ( int qt, int p ) {
        int  res, i;
        i=0;    res = p;
        while (i < qt)   res = res + i;
        return res;
}

int main() {
      int quant, x, n;
      float y;
      .....
      x = somaValor ( quant, n );    // Forma de chamada da função → OK.
      .......
```

```
  x = somaValor ( y , n );
```

// ERRO !!! Tipo do 1º argumento (float)
// não é o mesmo do 1º parâmetro da função (int)

```
.....
x = somaValor ( n , quant );
```

// ATENÇÃO !!! Os dois argumentos são do tipo certo
// mas a ordem pode estar errada.
// O programa terá erro ao executar !!

FUNÇÕES – PASSAGEM POR VALOR

- Ao chamar uma função, apenas o valor dos argumentos é passado (atribuído) aos respectivos parâmetros da função;

  - É comum dizer que a passagem de parâmetros é por valor;
  - O parâmetro neste caso é dito ser um parâmetro de entrada;

- Os parâmetros podem ter o mesmo nome de variáveis usadas como argumentos na chamada da função;
- Existe também a passagem de parâmetros por referência, todavia tal assunto não será nosso objeto de estudo.

ATIVIDADES

1. Desenvolva um programa que calcule e compare a área de dois retângulos A e B. O programa deverá ter a capacidade de distinguir qual retângulo possui a maior área ou se eles possuem tamanhos iguais. Esse programa deve possuir uma função para calcular a área do retângulo;
2. Escreva um programa que solicite a temperatura em graus Celsius ao usuário e seja apresentado em tela o resultado da conversão dessa temperatura em Fahrenheit.
   - Dados: F = C * 1,8 + 32.
3. Desenvolva um programa que solicite ao usuário a idade de três pessoas e apresente em tela a maior idade. Esse programa deve possuir uma função para verificar qual é a maior idade.

ATIVIDADES

4. Implemente um programa que receba os dados de um funcionário e aplique um aumento sobre o seu salário, sendo que a empresa definiu um aumento de 10% para quem possuir a partir de 5 anos de casa e for casado, para os demais funcionários o aumento é de 8%. O aumento, em questão, deve ser calculado por uma função. Deve ser informado o nome do respectivo funcionário(a);
5. Implemente um código em linguagem C que calcule e retorne a distância entre dois pontos (x1, y1) e (x2, y2). Todos os números e valores de retorno devem ser do tipo float;
6. Desenvolva um programa que receba 5 números inteiros positivos (caso seja digitado um número negativo o código deve apresentar alguma solução). Para cada número informado escrever a soma de seus divisors (exceto ele mesmo) e os respectivos números que compõem tal soma.

ATIVIDADES

7. O governo estadual acaba de liberar 10 milhões de dólares para a construção de casas populares. Uma casa custa o equivalente a 150 salários mínimos. Implemente um algoritmo onde seja informado o valor do salário mínimo e o valor do dólar. Deve ser apresentado em tela a quantidade de casas construídas e o valor de cada casa. Devem ser implementadas todas as validações necessárias. Faça o uso de funções para deixar o código principal o mais enxuto possível;
8. Desenvolva um programa que possua uma função que recebe, por parâmetro, a altura e o sexo de uma pessoa e retorna o peso ideal. Para homens calcular usando a formula peso ideal = 72,7 * altura – 58 e, para mulheres, peso ideal = 62,1 * altura – 44,7. Apresentar o resultado final em tela para visualização do usuário.

ATIVIDADES

9. Elabore um programa contendo uma ou mais função(ões) que recebam 3 números inteiros, representando horas, minutos e segundos e, os converta em segundos. Implemente todas as validações necessárias para que números inválidos não sejam considerados na conversão;

10. Implemente um programa onde se tenha uma função que receba, como parâmetro, um número N (inteiro e maior ou igual a 1). Determine o valor da sequência Z, descrita abaixo:

    Z = 1 + 1/2 + 1/3 + 1/4 + ...

    A quantidade de termos que compõe Z é igual a N.

ATIVIDADES

11. Escreva um programa que solicite dois números ao usuário e apresente em tela o resultado da soma, a soma dos dois números elevado ao cubo e por fim a raiz quadrada da soma de dois números. Implemente funções para o cálculo da soma, raiz quadrada e potenciação;

12. Em alguns estados do Brasil está ocorrendo crises hídricas, logo diversas famílias construíram reservatórios para armazenar água em suas residências. Implemente um programa, em linguagem C, que ajude os usuários dos reservatórios a controlarem seu consumo. Obtenha as dimensões de um reservatório (altura, largura e comprimento, em centímetros) e o consumo médio diário das famílias que utilizam o reservatório (em litros/dia).

    a) A capacidade total do reservatório em litros;

    b) A autonomia do reservatório em dias;

    c) A classificação do consumo: Consumo elevado se a autonomia for menor que 2 dias; Consumo moderado se a autonomia estiver entre 2 e 7 dias; Consumo reduzido se a autonomia maior que 7 dias.