

Notas de Aula Algoritmos e Técnicas de Programação

(Baseado na Resolução IFF n.º 37, de 22 de novembro de 2018)

OBJETIVOS DA APRENDIZAGEM:

Conhecer as principais estruturas na construção de soluções computacionais voltadas para dispositivos programáveis.
Aplicar o raciocínio lógico-dedutivo na criação de programas computacionais em Linguagem de Programação C.

CONTEÚDOS:

80h/a

Algoritmos

☛ Solução de problemas computacionais.

1. Descrição narrativa (linguagem natural).
2. Pseudocódigo.
3. Fluxogramas.

Técnicas de Programação

☛ Programação estruturada ou top-down no desenvolvimento de programas com Linguagem C.

1. Conceitos básicos:
 - 1.1. Tipos primitivos de dados.
 - 1.2. Variáveis.
 - 1.3. Comando de atribuição.
 - 1.4. Entrada e saída de dados.
2. Estruturas de Seleção:
 - 2.1. Conceito de estruturas de seleção.
 - 2.2. Operadores aritméticos, relacionais e lógicos.
 - 2.3. Seleção simples (IF).
 - 2.4. Seleção composta (IF-ELSE).
 - 2.5. Seleção encadeada (IF's encadeados).
 - 2.6. Seleção de múltipla escolha (SWITCH-CASE).
3. Estruturas de Repetição:
 - 3.1. Conceito de estruturas de repetição.
 - 3.2. Repetição com teste (DO-WHILE).
 - 3.3. Repetição com variável de controle (FOR).
 - 3.4. Lógica dos somadores e contadores.
4. Estruturas de Dados:
 - 4.1. Variáveis compostas homogêneas unidimensionais.
 - 4.2. Variáveis compostas homogêneas bidimensionais.
 - 4.3. Variáveis compostas heterogêneas.

Técnicas de Programação

☛ Modularização no desenvolvimento de programas com Linguagem de Programação C.

1. Estrutura básica.
2. Retorno de dados.
3. Parâmetros e escopo de variáveis.

Para compilar os programas utilize a ferramenta de acesso online

<https://www.programiz.com/c-programming/online-compiler/>

2022

☛



Algoritmos

▼ Solução de problemas computacionais.

1. Descrição narrativa (linguagem natural).
2. Pseudocódigo.
3. Fluxogramas.



Considere um problema para somar dois números.

Qualquer problema de computação pode ser resolvido executando uma série de ações, em uma sequência específica.

Algoritmo pode ser definido como o passo a passo para a solução de um problema.

A solução pode ser apresentada de várias formas. Observe a imagem.

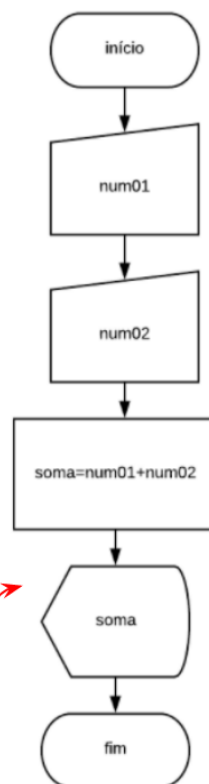
1. Definir o número 1.
2. Definir o número 2.
3. Somar os números.
4. Exibir o resultado.

Descrição narrativa consiste em analisar o enunciado do problema e escrever, utilizando linguagem natural, os passos a serem seguidos para sua resolução.

```
início
    inteiro num01, num02, soma;
    leia(num01);
    leia(num02);
    soma=num01+num02;
    escreva(soma);
fim.
```

Pseudocódigo é uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples e minimamente formal.

Fluxograma é a representação, por meio de símbolos gráficos, de um pseudocódigo.



Dado um problema, uma vez proposta a solução inicial, através de algoritmos (descrição narrativa ou pseudocódigo) e/ou fluxogramas, o próximo passo é o desenvolvimento do **Programa Computacional**.

Técnicas de Programação compreendem os conceitos e boas práticas para a criação de programas computacionais.

A conversão dos algoritmos e/ou fluxogramas em programas computacionais é feita através das **Linguagens de Programação**.

A linguagem de programação é uma metodologia, composta por um conjunto de regras, de implementação de um programa computacional.

Exemplos de linguagens de programação na imagem ao lado.



Na atividade letiva de Algoritmos e Técnicas de Programação, programa-se em linguagem de programação C e aprende-se a fazer programas que realizam funções básicas.

A linguagem de programação C é muito usada nas Engenharias. Com ela podemos desenvolver **aplicações que integram software e hardware**, permitindo o acesso ao nível quase de máquina. O que se destaca é a possibilidade do Engenheiro criar seu próprio software para uma ação específica.

Além disso, a linguagem de programação C é utilizada como base para praticamente todas as aplicações de **eletrônica embarcada que utilizam microcontroladores**. Pois permite a criação de programas rápidos e com pouco uso de memória.

Como exemplo, podemos citar a programação do **Arduíno**, um microcontrolador que permite implementar automação industrial e residencial, a um custo relativamente baixo. O Arduíno é programado na linguagem de programação C. Um exemplo de automação simples, com o uso do Arduíno, é o controle de relés permitindo identificar quando um aparelho elétrico está ligado ou desligado. Esse controle pode ser feito de maneira local, por ondas de rádio, por mensagens SMS, por aplicativos de celular, entre outros.

Técnicas de Programação

Programação estruturada ou top-down no desenvolvimento de programas com Linguagem C.

1. Conceitos básicos:
 - 1.1. Tipos primitivos de dados.
 - 1.2. Variáveis.
 - 1.3. Comando de atribuição.
 - 1.4. Entrada e saída de dados.



Programação estruturada é um padrão para desenvolvimento de programas onde os códigos formam um único bloco com ênfase em uma sequência lógica. A **linguagem de programação C** permite o desenvolvimento com base na programação estruturada.

Utilizando a linguagem de programação C podemos criar sistemas operacionais, aplicativos de todos os tipos, drivers e outros controladores de dispositivos, programar microcontroladores, etc. A linguagem C influenciou de forma direta muitas outras linguagens de programação como C++, Java, C#, Objective C, entre outras.

É extremamente portátil, ou seja, um programa escrito em linguagem C pode ser facilmente usado em qualquer plataforma. Além de toda essa flexibilidade, a linguagem C é capaz de gerar programas extremamente rápidos em tempo de execução, possui uma sintaxe simples e poderosa, com instruções de alto nível.

Considerando o problema para somar dois números inteiros, observe o pseudocódigo a seguir:

```
1  inicio
2  int a,b,soma;
3  a=1;
4  b=2;
5  soma=a+b;
6  escreva(soma);
7  fim.
```

Esse pseudocódigo, utilizando a linguagem de programação C, será convertido no seguinte programa:

```
1  #include <stdio.h>
2
3  main() {
4      int a,b,soma;
5      a=1;
6      b=2;
7      soma=a+b;
8      printf("%i",soma);
9  }
```

Linha 1: inclusão da biblioteca `stdio.h` para que o comando `printf` funcione corretamente.

Linha 3: `main()` { corresponde ao início do bloco de comandos.

Linha 4: criação de três variáveis inteiras que serão utilizadas para armazenar os dados manipulados pelo programa.

Linha 5: atribuição do valor inteiro 1 à variável `a`.

Linha 6: atribuição do valor inteiro 2 à variável `b`.

Linha 7: atribuição do resultado de uma soma inteira à variável `soma`.

Linha 8: exibição, na tela do dispositivo do usuário, do número 3 resultante da soma entre 1 e 2.

- **stdio.h** é uma biblioteca que permite a utilização de comandos específicos da linguagem C. Entre eles, o comando `printf` da linha 8. Então, para que o comando `printf` funcione corretamente, o programador deve incluir os recursos dessa biblioteca, ao seu programa.
- **Variáveis** são espaços reservados na memória do dispositivo para armazenamento de informações que serão manipuladas nos programas.
- **printf** é o comando em linguagem C para exibição de um dado na tela do dispositivo.

O programa deverá ser digitado em alguma ferramenta que permita a sua **compilação**. A compilação de um programa é o processo pelo qual cada linha digitada é verificada. Nessa verificação, o compilador identifica possíveis erros. Os erros dizem respeito ao que se espera da codificação, considerando a linguagem de programação escolhida. Cada linguagem de programação tem seus padrões de escrita, e um compilador específico.

O programa apresentado anteriormente tem um problema, sempre será exibido o número 3, pois fixamos os valores 1 e 3 às variáveis *a* e *b*. Resolvemos esse problema utilizando a leitura de dados, via teclado do dispositivo. Dessa forma, o usuário, a cada execução do programa, informa os números desejados para a soma.

main.c	Output
<pre> 1 #include <stdio.h> 2 //Programa para somar dois números 3 main() { 4 int a,b,soma; 5 printf("Informe um número:"); 6 scanf("%i",&a); 7 printf("Informe outro número:"); 8 scanf("%i",&b); 9 soma=a+b; 10 printf("A soma: %i",soma); 11 }</pre>	<pre> /tmp/0hlo9kIotN.o Informe um número:30 Informe outro número:24 A soma: 54</pre>

→ **scanf** é o comando em linguagem C para leitura de um dado, via teclado, e posterior armazenamento na variável reservada na memória do dispositivo. Para cada variável, faz-se um comando **scanf** indicando o tipo de dado que será lido ("%i") e a variável presente na memória, que receberá o dado (&a). O símbolo "&" significa o armazenamento no endereço específico da memória.

Observe o mesmo programa com algumas modificações.

main.c	Output
<pre> 1 #include <stdio.h> 2 //Programa para somar dois números 3 main() { 4 float a,b; 5 printf("Informe um número:"); 6 scanf("%f",&a); 7 printf("Informe outro número:"); 8 scanf("%f",&b); 9 printf("A soma: %.1f",a+b); 10 }</pre>	<pre> /tmp/0hlo9kIotN.o Informe um número:4.6 Informe outro número:3.2 A soma: 7.8</pre>

Linha 2: inclusão de um comentário com //. Os comentários são apenas informativos e serão ignorados pelo compilador.

Linha 4: criação de três variáveis reais (float), permitindo a leitura de números com casas decimais.

Linha 6: %f indicando que o tipo de dado que será lido é real.

Linha 9: exibição do resultado da variável diretamente na tela. Evitando o uso da variável *soma*, conseguimos otimizar o uso da memória. Quanto mais memória livre disponível para o dispositivo, mais rápida será a execução do programa.


Linha 9: limitação do número de casas decimais na exibição do valor resultante real (".1" corresponde a uma casa decimal).

Observe que, nos números digitados durante a execução do programa, utilizamos o ponto final para indicar casas decimais e não a vírgula.

Com relação às variáveis e seus tipos de dados, podemos dividir as informações em três **tipos primitivos de dados**. Como exemplo, considere as declarações de variáveis a seguir:

<code>int idade;</code>	Conjunto de números inteiros. (1043, -19, 0, ...)
<code>float media;</code>	Conjunto de números reais. (2.3, 12, -2.76, ...)
<code>char nome[10];</code>	Conjunto de caracteres composto por, no máximo, 10 caracteres. (@3, Ana, ...)

Com relação à exibição de dados na tela do dispositivo, através do comando `printf`, podemos fazer uso de **caracteres de controle**. Como exemplo, considere o mesmo programa alterado a seguir:

main.c	  	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 float a,b; 5 printf("Programa para somar dois números\n"); 6 printf("-----\n\n"); 7 printf("Informe um número:"); 8 scanf("%f",&a); 9 printf("Informe outro número:"); 10 scanf("%f",&b); 11 printf("A soma: %.2f",a+b); 12 }</pre>		<pre> /tmp/0hlo9kIotN.o Programa para somar dois números ----- Informe um número:3.65 Informe outro número:9.32 A soma: 12.97</pre>

Linha 6: uso do caracter de controle `"\n"` orientando o compilador a “pular uma linha” na posição indicada.

Linha 11: limitação do número de casas decimais na exibição do valor resultante, sendo duas casas decimais.

Você também pode usar o caracter de controle `"\t"` para orientar o compilador a executar uma tabulação (tecla TAB).

▶ ▶ ▶ Exercício de fixação:

1. Elabore um programa para solicitar uma base e uma altura. Em seguida, calcule a área do triângulo: $base \cdot altura / 2$.
2. Elabore um programa para solicitar R e U. Em seguida, calcule $i = U/R$ e $P = U \cdot i$. Ao final, informe os resultados.
3. Elabore um programa para executar o cálculo do consumo de energia $E = Potencia \cdot \Delta t$.
4. Elabore um programa que pergunta um valor em metros e mostre o correspondente em decímetros, centímetros e milímetros.
5. Elabore um programa que solicite a quantidade e o preço unitário de um determinado tipo de componente eletrônico vendido. Deve-se informar o total da compra (preço*quantidade). Em seguida, calcule e informe a taxa de entrega (10% do total da compra).
6. Faça um programa que leia o número de horas trabalhadas por um gerador elétrico, o valor em dinheiro que o seu consumo representa por hora e o número de manutenções realizadas. O programa deve calcular o gasto total com o gerador, sabendo que para cada manutenção incide um adicional de 3% do valor calculado.

Dica para acentuação correta no DEV C++:

```

#include <stdio.h>
#include <locale.h>
main(){
    setlocale(LC_ALL, "Portuguese");
```

2. Estruturas de Seleção:

- 2.1. Conceito de estruturas de seleção.
- 2.2. Operadores aritméticos, relacionais e lógicos.
- 2.3. Seleção simples (IF).
- 2.4. Seleção composta (IF-ELSE).
- 2.5. Seleção encadeada (IF's encadeados).
- 2.6. Seleção de múltipla escolha (SWITCH-CASE).

A estrutura de seleção é utilizada para alterar o fluxo de execução de um programa baseado no valor, verdadeiro ou falso, de uma expressão lógica.

Considere um programa para ler a idade do usuário e informar se o mesmo é maior ou menor de idade.

main.c	Output
<pre>1 #include <stdio.h> 2 3 main() { 4 int idade; 5 printf("Informe uma idade:"); 6 scanf("%i",&idade); 7 if(idade>=18) 8 printf("Maior de idade."); 9 }</pre>	<pre>/tmp/0h1o9kIotN.o Informe uma idade:18 Maior de idade.</pre>

O **comando IF** (se) é uma estrutura de seleção que faz o teste lógico “idade>=18” (linha 7) e, caso o teste resulte em “verdadeiro”, a frase “Maior de idade.” será exibida na tela do dispositivo.

O teste lógico faz uso de **operadores relacionais**. Os operadores relacionais são:

== igualdade
!= diferença
> maior que
< menor que
>= maior ou igual a
<= menor ou igual a

Observação importante: devemos sempre atentar para um erro bastante comum que é confundir o operador == que verifica a igualdade, com o operador de atribuição = que é usado para atribuir valor a uma variável.

if (a == b) verifica se o valor de a é igual ao valor de b, resultando em verdadeiro ou falso.
a = 3; atribui o valor 3 à variável a.

Você já fez uso dos **operadores aritméticos**, sendo:

+ Adição
- Subtração
* Multiplicação
/ Divisão
% Resto da divisão inteira

Existem ainda os **operadores lógicos**, sendo:

&& E
|| OU
! Negação

Continuando, vamos considerar que se deseja informar o caso “falso”, que corresponde à frase “Menor de idade.”.

main.c	Output
<pre>1 #include <stdio.h> 2 3 main() { 4 int idade; 5 printf("Informe uma idade:"); 6 scanf("%i",&idade); 7 if(idade>=18) 8 printf("Maior de idade."); 9 else 10 printf("Menor de idade."); 11 }</pre>	<pre>/tmp/0hlo9kIotN.o Informe uma idade:12 Menor de idade.</pre>

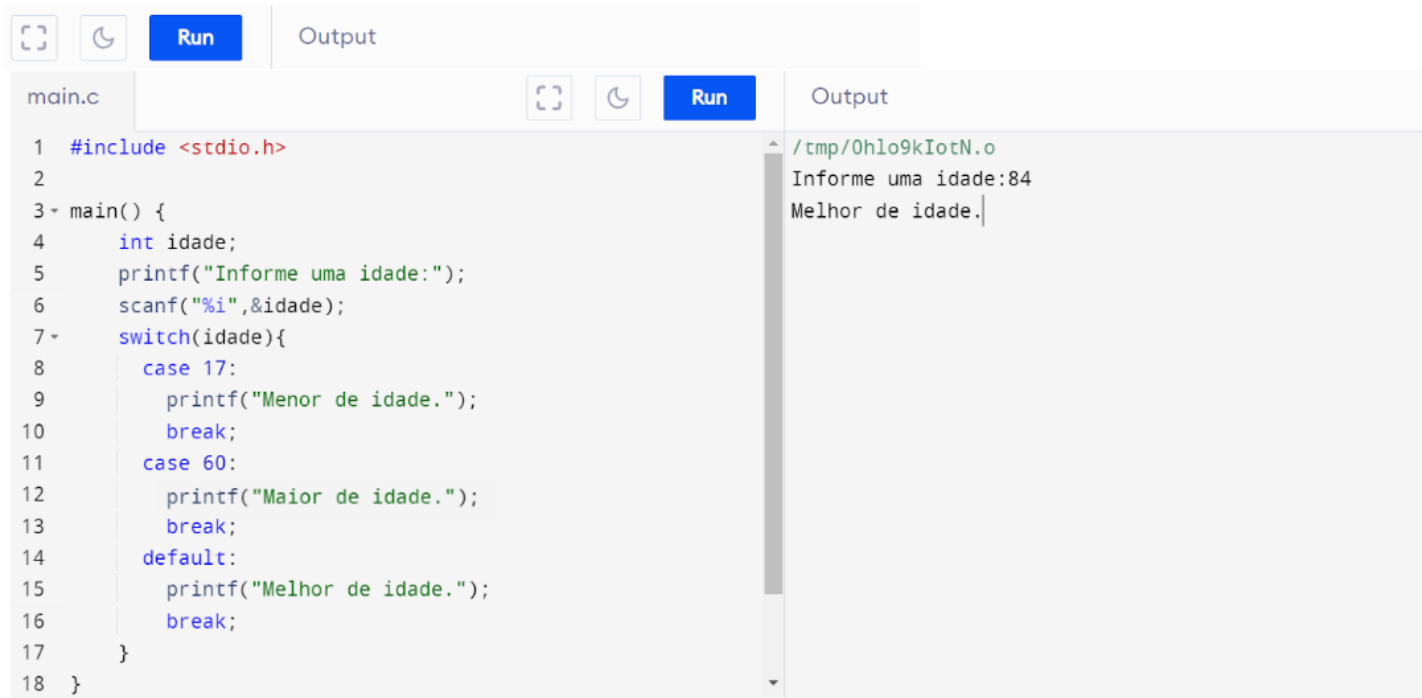
Para o teste de três ou mais condições, utiliza-se o **IF encadeado**.

main.c	Output
<pre>1 #include <stdio.h> 2 3 main() { 4 int idade; 5 printf("Informe uma idade:"); 6 scanf("%i",&idade); 7 if(idade<18) 8 printf("Menor de idade."); 9 else if(idade>=18 && idade<=60) 10 printf("Maior de idade."); 11 else if(idade>60) 12 printf("Melhor de idade."); 13 }</pre>	<pre>/tmp/0hlo9kIotN.o Informe uma idade:70 Melhor de idade.</pre>

Podemos ainda, utilizar mais de um comando para cada caso (verdadeiro ou falso). Dessa forma, utilizam-se chaves (“{ ”) para delimitar o bloco de comando para cada caso. Observe as linhas 11 e 14.

main.c	Output
<pre>1 #include <stdio.h> 2 3 main() { 4 int idade; 5 printf("Informe uma idade:"); 6 scanf("%i",&idade); 7 if(idade<18) 8 printf("Menor de idade."); 9 else if(idade>=18 && idade<=60) 10 printf("Maior de idade."); 11 else if(idade>60){ 12 printf("Melhor de idade."); 13 printf("\nVocê possui gratuidade."); 14 } 15 }</pre>	<pre>/tmp/0hlo9kIotN.o Informe uma idade:70 Melhor de idade. Você possui gratuidade.</pre>

Para **testes de igualdade**, uma alternativa ao uso de IFs encadeados, é o comando **SWITCH**.



```
main.c
1 #include <stdio.h>
2
3 main() {
4     int idade;
5     printf("Informe uma idade:");
6     scanf("%i",&idade);
7     switch(idade){
8         case 17:
9             printf("Menor de idade.");
10            break;
11        case 60:
12            printf("Maior de idade.");
13            break;
14        default:
15            printf("Melhor de idade.");
16            break;
17    }
18 }
```

Output

```
/tmp/0hlo9kIotN.o
Informe uma idade:84
Melhor de idade.
```

Para cada caso, utilize o comando `break` para finalizar as ações. O caso `default` é acionado quando nenhuma igualdade é encontrada.

▶▶▶ Exercício IF:

1. Elabore um programa que, com base em duas notas do aluno, informe a média e se o mesmo está aprovado ou reprovado.
2. Elabore um programa que, com base em dois números inteiros, informe se são diferentes ou iguais.
3. Elabore um programa para, a partir de um número qualquer, informar se o mesmo é positivo, nulo ou negativo.
4. Elabore um programa que leia um número inteiro e informe se o mesmo é par ou ímpar. Pesquise como definir par ou ímpar em C.
5. Escreva um programa, com o comando IF, para ler o número de lados de um polígono regular, a medida do lado e verificar:
 - Se o número de lados for igual a 3 escrever "Triângulo" e o valor do seu perímetro, em `printfs` diferentes.
 - Se o número de lados for igual a 4 escrever "Quadrado" e o valor da sua área, em `printfs` diferentes.
 - Se o número de lados for igual a 5 escrever "Pentágono".
 - Em qualquer outra situação escrever "Polígono não identificado".
6. Elabore um programa que informe o fator de demanda, de acordo com o número de aparelhos de ar condicionado instalados. Use `if` encadeado e considere:
 - Número de aparelhos 1 a 10, Fator de demanda (%%) 100.
 - Número de aparelhos 11 a 20, Fator de demanda (%%) 86.
 - Número de aparelhos 21 a 30, Fator de demanda (%%) 80.
 - Número de aparelhos 31 a 40, Fator de demanda (%%) 75.
 - Número de aparelhos maior que 40, Fator de demanda (%%) 63.
7. Para uma empresa de aluguel de geradores, faça um programa com o comando IF, que leia o número de dias no mês para aluguel e o tipo do aparelho (1=importado ou 2=nacional). Em seguida, calcule e informe o valor do aluguel.
 - Nacional: número de dias no mês para aluguel * 1250. Ainda nesse caso, deve-se informar, em um `printf` separado, que o aparelho deve ser devolvido em até 3 dias úteis após o término do período de uso.
 - Importado: número de dias no mês para aluguel * 1550 + 1500. Ainda nesse caso, deve-se informar, em um `printf` separado, que existe a possibilidade de pagamento de um seguro para o aparelho.
8. Faça um programa que leia um número e mostre uma das mensagens: "Maior do que 20", "Igual a 20" ou "Menor do que 20".

▶▶▶ Exercício SWITCH:

1. Faça um programa que leia um número entre 1 e 7. Em seguida, com o comando SWITCH, e exiba o dia correspondente da semana. (1- Domingo , 2- Segunda, etc.). Caso o usuário digite outro valor, deve-se exibir a mensagem "Valor inválido."
 2. Elabore um programa que leia dois valores (float) do usuário e a operação (int) que ele deseja executar com esses dois números. Considere as operações de soma, subtração, divisão, multiplicação. Em seguida, com o uso do comando SWITCH, execute a operação desejada e exiba o resultado na tela.
- Observe o código a seguir. Ele deverá ser utilizado após a leitura dos dois valores. Copie e cole em seu programa antes do comando SWITCH.

```
printf("\nOperações possíveis:\n");  
printf("Digite 1: soma\n");  
printf("Digite 2: subtração\n");  
printf("Digite 3: divisão\n");  
printf("Digite 4: multiplicação\n");  
printf("\nQual operação deseja fazer:");  
scanf("%i",&operacao);
```

▶▶▶ Exercício de fixação:

1. Escreva um programa que pergunte o raio de uma circunferência, e em seguida mostre o diâmetro, comprimento e área da circunferência.
2. Para doar sangue é necessário ter entre 18 e 67 anos. Faça um programa na linguagem C que pergunte a idade de uma pessoa e diga se ela pode doar sangue ou não.
3. Crie um programa em C que recebe uma nota de um aluno e checa se o mesmo passou direto, ficou de recuperação ou foi reprovado na matéria. Use IF encadeado. A regra é a seguinte:
Nota 6 ou mais: aprovado
Entre 4 e 6: tem direito de fazer uma prova de recuperação
Abaixo de 4: reprovado
4. Faça um programa que leia o salário de um colaborador, calcule e exiba o novo salário acrescido de uma gratificação, segundo o seguinte critério:
Salários até R\$ 1280,00: aumento de 20%;
Salários entre R\$ 1280,00 e R\$1700,00: aumento de 15%;
Salários entre R\$ 1700,00 e R\$ 11500,00: aumento de 10%;
Salários de R\$ 11500,00 em diante: aumento de 5%

3. Estruturas de Repetição:

- 3.1. Conceito de estruturas de repetição.
- 3.2. Repetição com teste (DO-WHILE).
- 3.3. Repetição com variável de controle (FOR).
- 3.4. Lógica dos somadores e contadores.



A estrutura de repetição é utilizada para repetir um trecho do programa baseado no valor, verdadeiro ou falso, de uma expressão lógica.

Considere um programa para ler várias idades do usuário. Um critério de parada deve ser definido. Nesse caso, vamos definir que, para encerrar a leitura de idades, o número 0 deverá ser digitado.

main.c	Run	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 int idade; 5 printf("Leitura de idades. Para parar, digite 0."); 6 do{ 7 printf("\nInforme uma idade:"); 8 scanf("%i",&idade); 9 }while(idade!=0); 10 printf("Fim da leitura."); 11 }</pre>		<pre> /tmp/0hlo9kIotN.o Leitura de idades. Para parar, digite 0. Informe uma idade:10 Informe uma idade:2 Informe uma idade:0 Fim da leitura.</pre>

O comando **DO...WHILE** (faça...enquanto) é uma estrutura de repetição que repete o trecho entre as linhas 6 e 9, quantas vezes o usuário desejar. Ou seja, repete o trecho por um número **indeterminado** de vezes. A estrutura faz o teste lógico "idade!=0" (linha 9) e, caso o teste resulte em "verdadeiro", repete o trecho delimitado.

▶▶▶ Exercício:

1. Com o comando `do`, elabore um programa para ler números reais e informar o dobro. Sugestão de critério de parada: 0.
2. Com o comando `do`, elabore um programa para ler notas do aluno. Sugestão de critério de parada: nota negativa.
3. Altere o programa anterior para informar se o aluno está aprovado ou não.
4. Faça um programa em C que calcule a conversão entre graus centígrados e Fahrenheit. Para isso, leia o valor em centígrados e calcule com base na fórmula a seguir. Após calcular o programa deve mostrar o resultado da conversão. A leitura da temperatura deve ser realizada até o usuário digitar um temperatura negativa. Quando essa temperatura negativa for digitada, o cálculo não deve ser exibido.

Em que:

$$F = \frac{9 * C + 160}{5}$$

- F = Graus em Fahrenheit
- C = Graus centígrados

5. Com o comando `do`, elabore um programa para ler a base e a altura de um retângulo, quantas vezes o usuário desejar. Deve-se calcular a área do quadrado. Sugestão de critério de parada: base negativa.
6. Uma empresa estipulou o preço para o aluguel de um equipamento em R\$30,00 e mais uma taxa de serviços diários de: R\$15,00, se o número de dias for menor que 10; R\$8,00, se o número de dias for maior ou igual a 10. O programa deverá ler o número de dias até que o usuário digite 0 (zero). Para cada número de dias lido, deve-se informar o valor a ser pago pelo serviço de aluguel.

Uma outra estrutura de repetição, é definida a seguir.

main.c	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 int idade, i; 5 printf("Leitura de 3 idades."); 6 for(i=1;i<=3;i++){ 7 printf("\nInforme uma idade:"); 8 scanf("%i",&idade); 9 } 10 printf("Fim da leitura."); 11 }</pre>	<pre> /tmp/0hlo9kIotN.o Leitura de 3 idades. Informe uma idade:6 Informe uma idade:5 Informe uma idade:3 Fim da leitura.</pre>

O **comando FOR** (para) é uma estrutura de repetição que repete o trecho entre as linhas 6 e 9, por três vezes. Ou seja, repete o trecho por um número **determinado** de vezes. É necessária a criação de um variável específica para “contar” as rodadas do trecho que se repete. Considere a seguinte lógica:

1. Cria-se a variável `int i` na linha 4;
2. A variável `i` se inicia com o valor 1, na primeira rodada (linha 6);
3. A cada rodada, a estrutura de repetição faz o teste lógico “`i<=3`” (linha 6) e
4. Incrementa em +1 o valor da variável `i` (“`i++`”).
5. Caso o teste “`i<=3`” resulte em “verdadeiro”, repete-se o trecho delimitado entre as linhas 6 e 9.

▶▶▶ Exercício:

1. Faça um programa que leia os valores para aplicar a 2ª lei de Ohm. O cálculo deve ser realizado 3 vezes. Sendo: $R = \rho \cdot L \div A$. Onde: R =Resistência elétrica em Ω , ρ =Resistividade elétrica, L =Comprimento do condutor e A =Área de seção transversal.

2. Elabore um programa para, com o comando `for`, ler 4 notas de um aluno e informar se o mesmo está com a nota acima ou abaixo da média.

3. Uma empresa de fornecimento de energia elétrica faz a leitura mensal dos medidores de consumo de um terminal qualquer com 5 pontos de consumo. Para cada ponto, são digitados os seguintes dados:

- quantidade de kWh consumidos durante o mês
- tipo (código) do consumidor
 - 1-residencial, preço em reais por kWh = 0,3
 - 2-comercial, preço em reais por kWh = 0,5
 - 3-industrial, preço em reais por kWh = 0,7

O programa deve calcular e mostrar o custo total para cada ponto de consumo lido.

4. Faça um programa que permita entrar com o salário bruto de 5 pessoas. Em seguida deve-se mostrar o valor da alíquota do imposto de renda calculado conforme a tabela a seguir:

Salário	IRRF
Salário menor que R\$1300,00	Isento
Salário maior ou igual a R\$1300,00 e menor que R\$2300,00	10% do salário bruto
Salário maior ou igual a R\$2300,00	15% do salário bruto

5. Elabore um programa para exibir todos os números entre 100 e 500.
6. Elabore um programa para exibir todos os números ímpares entre 100 e 500.
7. Elabore um programa que monte a tabuada de multiplicação de 1 a 10 por 5 (Exemplo: $5 \times 1 = 5$).
8. Altere o programa anterior para montar a tabuada de multiplicação de 1 a 10 por qualquer número.

Uma aplicação comum à programação é a lógica dos **contadores**.

Considere um programa para ler 3 números inteiros e informar a quantidade de zeros digitados.

main.c	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 int n, i, cont=0; 5 for(i=1;i<=3;i++){ 6 printf("\nInforme um número:"); 7 scanf("%i",&n); 8 if(n==0) 9 cont++; 10 } 11 printf("Número de zeros digitados:%i", cont); 12 }</pre>	<pre> /tmp/0hlo9kIotN.o Informe um número:2 Informe um número:7 Informe um número:0 Número de zeros digitados:1</pre>

Outra aplicação comum à programação é a lógica dos **somadores**.

Considere um programa para ler 3 números inteiros e informar a soma dos números digitados.

main.c	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 int n, i, soma=0; 5 for(i=1;i<=3;i++){ 6 printf("\nInforme um número:"); 7 scanf("%i",&n); 8 soma=soma+n; 9 } 10 printf("Soma dos números digitados:%i", soma); 11 }</pre>	<pre> /tmp/0hlo9kIotN.o Informe um número:5 Informe um número:4 Informe um número:3 Soma dos números digitados:12</pre>

As variáveis contadoras e somadoras devem sempre ser criadas atribuindo o valor 0. Deve-se agir dessa forma porque a variável, quando é criada, assume um endereço na memória do dispositivo e, esse endereço pode estar vazio ou com algum dado de execuções anteriores. Dessa forma, assegura-se que a contagem, ou a soma, se inicie sempre do valor zero.

O comando de **incremento** (`cont++`) a cada rodadas do comando `for`, acrescenta o número 1 ao valor da variável `cont`. Logo, na primeira rodada `cont=0+1`, na segunda rodada, `cont=1+1`, e assim por diante.

A lógica dos **somadores** (`soma=soma+n`) a cada rodadas do comando `for`, acrescenta o número lido ao valor da variável `soma`. Logo, na primeira rodada `soma=0+5`, na segunda rodada, `soma=5+4`, na terceira rodada, `soma=9+3`, resultando em 12.

▶▶▶ Exercício:

1. Faça um programa em C que permita entrar com a potência de 5 aparelhos elétricos e exiba, o total de aparelhos com potência igual a 1000 watts e o total de aparelhos com mais de 1000 watts de potência.
2. Elabore um programa para solicitar 10 números. Deve-se exibir a quantidade de números negativos, positivos e zeros digitados.
3. Elabore um programa para solicitar 10 números. No final deve-se exibir a soma e a média dos números digitados.
4. Elabore um programa para ler temperaturas positivas por quantas vezes o usuário desejar (comando `do`). Deve-se somar todas as temperaturas digitadas e contar as temperaturas entre 100°C e 200°C.
5. Elabore um programa para solicitar a vida útil (em anos) e a potência de aparelhos elétricos. No final deve-se exibir a quantidade de aparelhos com vida útil maior que 2 anos e a média das potências digitadas. Critério de parada sugerido: vida útil = 0, mas atenção, quando a vida útil = 0 for digitada, não se deve solicitar a potência e realizar os cálculos.

A **identação**, uma boa prática de programação, é a técnica referente ao espaçamento ao iniciar uma linha de comando.

Programa 1
Com idenação correta.

```
exit(1);
proximo_ponto = ini_ponto;

while(1)
{
    printf("Digite x: ");
    scanf("%d", &proximo_ponto->x);
    printf("Digite y: ");
    scanf("%d", &proximo_ponto->y);
    printf("Deseja continua? <1> SIM <out
    scanf("%d", &resp);
    if(resp == 1) {
        proximo_ponto->proximo = (t_ponto
    }
}
return 0;
```

Programa 2
Com idenação incorreta.

```
exit(1);
proximo_ponto = ini_ponto;

while(1)
{
    printf("Digite x: ");
    scanf("%d", &proximo_ponto->x);
    printf("Digite y: ");
    scanf("%d", &proximo_ponto->y);
    printf("Deseja continua? <1> SIM <out
    scanf("%d", &resp);
    if(resp == 1) {
        proximo_ponto->proximo = (t_ponto
    }
}
return 0;
```

Observem esses dois programas. O programador 2 digita o texto com muitas tabulações, ficando em um formato de "escadinha". Isso não interfere na execução do programa, mas não é uma boa prática na programação. Só se avança na tabulação quando se cria um conjunto de comandos "dentro" de outro comando. A tabulação correta também facilita a leitura do programa.

▶▶▶ Exercício de Revisão:

1. Faça um programa que leia três notas de um aluno, calcule e escreva a média final deste aluno. Considerar que a média é ponderada e que os pesos das notas são 2, 3 e 5. Fórmula para o cálculo da média final é:

$$\text{mediafinal} = \frac{n1 * 2 + n2 * 3 + n3 * 5}{10}$$

2. O custo de um equipamento novo qualquer ao consumidor é a soma do custo de fábrica com a porcentagem do distribuidor e a dos impostos (aplicados ao custo de fábrica). Supondo que o percentual do distribuidor seja de 28% e os impostos de 45%, escrever um algoritmo para ler o custo de fábrica de um carro, calcular e escrever o custo final ao consumidor.

3. Escreva um programa para ler uma temperatura em graus Fahrenheit, calcular e escrever o valor correspondente em graus Celsius (baseado na fórmula abaixo):

$$\frac{C}{5} = \frac{F - 32}{9}$$

Observação: Para testar se a sua resposta está correta saiba que **100°C = 212°F**

Se a temperatura calculada for maior que 100°C, deve-se exibir a mensagem "Temperatura fora do limite aceitável".

4. Escreva um algoritmo para ler 10 números. Todos os números lidos devem ser somados. Ao final, escreva o valor da soma efetuada e calcule a média.

5. Uma loja está levantando o valor total de todas as mercadorias em estoque. Escreva um programa que permita a entrada das seguintes informações para produtos:

- o número total de itens no estoque;
- o valor para venda.

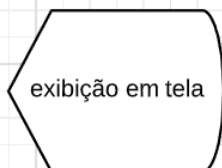
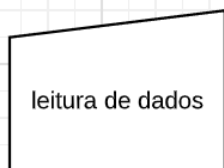
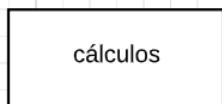
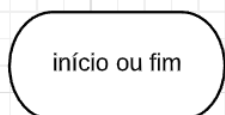
O usuário deve informar quantos dados desejar. Para cada mercadoria deve-se informar o total da mercadoria em estoque (o número total de itens no estoque x o valor para venda). Ao final, exibir o valor total em estoque (somatório dos totais de cada mercadoria em estoque).

6. Elabore um programa para ler um valor N e exibir todos os valores inteiros entre 1 e N (inclusive). Considere que o N será sempre maior que ZERO.

7. Altere o programa anterior para contar os divisíveis por 3 e os nulos.

▶ ▶ ▶ **Exercício Fluxogramas:**

**Representação
em Fluxogramas**



entre outros ...

Pesquise para saber mais sobre fluxogramas.

1. Apresente, em forma de fluxogramas, as soluções apresentadas nas questões 1, 4 e 6 acima.
2. Considere a representação básica ao lado.
3. Utilize o aplicativo "Lucidchart" disponível na loja de aplicativos para celular, ou sua versão online.

4. Estruturas de Dados:

- 4.1. Variáveis compostas homogêneas unidimensionais.
- 4.2. Variáveis compostas homogêneas bidimensionais.
- 4.3. Variáveis compostas heterogêneas.



Variáveis compostas homogêneas unidimensionais são denominadas **vetores**. O vetor armazena sempre dados de um mesmo tipo. Observe o programa para armazenar, em uma única variável, as 3 notas de um aluno.

main.c	Output
<pre> 1 #include <stdio.h> 2 3 main() { 4 float notas[3]; 5 int i; 6 for(i=0;i<3;i++){ 7 printf("\nInforme a nota:"); 8 scanf("%f",&notas[i]); 9 } 10 printf("Fim da leitura."); 11 }</pre>	<pre> /tmp/0hlo9kIoTn.o Informe a nota:7 Informe a nota:6 Informe a nota:6 Fim da leitura.</pre>

Linha 4: criação do vetor `notas` com 3 posições reservadas na memória do dispositivo.

Linha 8: leitura do vetor `notas` nas posições controladas pela variável `i` do comando `for`.

Ao final, o vetor `notas` irá possuir uma aparência semelhante a:

posição	0	1	2
notas	7	6	6

▶ ▶ ▶ Exercício:

1. Faça um programa que leia 4 números inteiros em um vetor e, para cada número lido, exiba o dobro.
2. Faça um programa que receba do usuário dois vetores, A e B, com 5 números inteiros cada. Crie um novo vetor denominado C calculando $C = A - B$. Mostre na tela os dados do vetor C.
3. Leia um vetor de 5 posições e atribua valor 0 para todos os elementos lidos que possuírem valores negativos.
4. Faça um programa com um vetor de tamanho 50, preenchido automaticamente, com o seguinte valor: $(i + 5 * i) / (i + 0.1)$, sendo `i` a posição do elemento no vetor. Em seguida mostre o vetor na tela. Lembre-se, o usuário não irá digitar valores, você irá calculá-los.

Variáveis compostas homogêneas bidimensionais são denominadas **matrizes**. A matriz armazena sempre dados de um mesmo tipo. Observe o programa para armazenar, em uma única variável, as 3 notas de 30 alunos.

main.c	Output
1 #include <stdio.h>	/tmp/0hlo9kIotN.o
2	Aluno:0
3 main() {	Informe a nota:6
4 float notas[30][3];	
5 int l,c;	Aluno:0
6 for(l=0;l<30;l++){	Informe a nota:5
7 for(c=0;c<3;c++){	Aluno:0
8 printf("\nAluno:%i",l);	Informe a nota:7
9 printf("\nInforme a nota:");	
10 scanf("%f",¬as[l][c]);	Aluno:1
11 }	Informe a nota:5
12 }	
13 printf("Fim da leitura.");	Aluno:1
14 }	Informe a nota:7

A execução continua ↗

Linha 4: criação da matriz `notas` com 30x3 posições reservadas na memória do dispositivo.

Linha 8: leitura da matriz `notas` nas posições controladas pela variável `l` e `c` do comando `for`.

Ao final, a matriz `notas` irá possuir uma aparência semelhante a:

posição	0	1	2
notas 0	6	5	7
1	5	7	...
...			

▶▶▶ Exercício:

1. Faça um programa para ler uma matriz de 3 x 3 elementos, multiplicar cada elemento por 5 e exibir o resultado.
2. Dadas duas matrizes reais 2 x 2, A e B, calcular a soma de A + B.
3. Dada uma matriz A 2x2 exibir a quantidade de elementos iguais a zero lidos.
4. Faça um programa para ler 3 notas de 5 alunos em uma matriz. Para cada nota lida, verificar se está abaixo de 6. Caso verdadeiro, informar a situação.

Variáveis compostas heterogêneas são denominadas **estruturas**. A estrutura pode armazenar dados de diferentes tipos. Observe o programa para armazenar, em uma única variável denominada `aluno`, os dados `media` (`float`) e `idade` (`int`).

main.c	Run	Output
<pre>1 #include <stdio.h> 2 3 main() { 4 struct{ 5 float media; 6 int idade; 7 }aluno; 8 printf("\nInforme a média:"); 9 scanf("%f",&aluno.media); 10 printf("\nInforme a idade:"); 11 scanf("%i",&aluno.idade); 12 }</pre>		<pre>/tmp/0h1o9kIotN.o Informe a média:9 Informe a idade:20</pre>

▶▶▶ Exercício:

1. Crie uma estrutura chamada `retângulo`. Essa estrutura deverá conter o valor da base e o valor da altura do retângulo. Faça um programa que declare e leia uma estrutura `retângulo` e exiba a área e o perímetro desse retângulo.
2. Altere o programa anterior para contar a quantidade de áreas maiores que 10 cm.
3. É possível criar um vetor de estruturas. Esse é um desafio para você. Pesquise e altere o programa anterior. O mesmo deverá ser capaz de ler dados de 5 retângulos. Dica: a variável da estrutura deverá ser um vetor de 5 posições.
4. Crie uma estrutura para representar as coordenadas de um ponto no plano (posições X e Y). Em seguida, declare a variável. O programa deve ler um ponto e exibir a distância dele até a origem das coordenadas. Para realizar o cálculo, utilize a fórmula a seguir. Pesquise como realizar o cálculo de potência e raiz quadrada.

$$d = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$$

Técnicas de Programação

❗ Modularização no desenvolvimento de programas com Linguagem de Programação C.

1. Estrutura básica.
2. Retorno de dados.
3. Parâmetros e escopo de variáveis.



A **modularização** é uma metodologia de desenvolvimento diferente da qual nós estávamos utilizando até o momento.

Utilizamos a programação estruturada, que é um padrão para desenvolvimento de programas onde os códigos formam um único bloco com ênfase em uma sequência lógica.

A modularização consiste na programação em **funções**. Uma função é um bloco de código que faz alguma tarefa específica e pode ser chamado de qualquer parte do programa, quantas vezes desejarmos.

Utilizamos funções para obter:

1. **Clareza** do código, pois o programa fica melhor estruturado,
2. **Reutilização** do código, pois a função pode ser chamada várias vezes e
3. **Divisão** de tarefas, pois em grandes programas, a programação é feita em equipe. Dessa forma, cada programador ficará responsável por um conjunto de funções.

Para entender a **estrutura básica** de uma função, observe o programa abaixo com uma função para somar dois números.

main.c	Run	Output
<pre>1 #include <stdio.h> 2 int n1,n2; 3 soma(){ 4 printf("Soma: %i", n1+n2); 5 } 6 main() { 7 printf("Informe um número:"); 8 scanf("%i",&n1); 9 printf("Informe outro número:"); 10 scanf("%i",&n2); 11 soma(); 12 }</pre>		<pre>/tmp/0h1o9kIotN.o Informe um número:2 Informe outro número:3 Soma: 5</pre>

Linha 2: observe que as variáveis não estão mais alocadas no `main`. Elas agora são **variáveis globais**, e podem ser usadas no `main` e dentro da função `soma`.

Linhas 3 a 5: implementação da função `soma`.

Linha 11: chamada da função `soma`.

▶▶▶ Exercício:

1. Faça um programa que leia os valores para aplicar a 2ª lei de Ohm. O cálculo deve ser realizado em uma função. Sendo: $R = \rho \cdot L \div A$.
2. Elabore um programa que leia dois números `float` e implemente 4 funções: adição, subtração, multiplicação e divisão.
3. Elabore um programa com 2 funções, sendo: uma para informar se um número é par ou ímpar (`if n%2==0`) e outra

para informar se o número é múltiplo de 3 (`if n%3==0`). Leia o número no `main`.

4. Elabore um programa que, no `main`, abra um comando `for` e dentro desse comando leia 5 temperaturas, faça as chamadas das funções descritas a seguir, exiba a contagem e a soma.

a) Crie uma função com um `if` para contar as temperaturas acima de 100°C.

b) Crie outra função para somar as temperaturas lidas.

5. Faça um programa que leia os valores para aplicar a 2ª lei de Ohm. O cálculo deve ser realizado em uma função com retorno e com parâmetro. Sendo: $R = \rho \cdot L \div A$.

A mesma função pode ser escrita com **retorno**.

main.c	Run	Output
<pre> 1 #include <stdio.h> 2 int n1,n2; 3 int soma(){ 4 return n1+n2; 5 } 6 main() { 7 printf("Informe um número:"); 8 scanf("%i",&n1); 9 printf("Informe outro número:"); 10 scanf("%i",&n2); 11 printf("Soma: %i", soma()); 12 }</pre>		<pre> /tmp/0hlo9kIotN.o Informe um número:3 Informe outro número:6 Soma: 9</pre>

Linha 2: observe que as variáveis são **variáveis globais**.

Linhas 3 a 5: implementação da função `soma`. Observe que antes do nome da função `soma`, coloca-se o tipo de retorno `int` que será gerado através do cálculo da linha 4.

Linha 4: retorno do valor calculado na função `soma`. Observe que o comando `printf` não pertence mais à função, e foi transferido para o `main`.

Linha 11: uma função com retorno deve ser chamada como parte de outro comando, no caso, o comando `printf`.

A utilização de retorno permite que a função fique mais clara e objetiva, executando somente a sua função primitiva. No caso, a função `soma` deveria somente calcular a soma, e não direcionar a saída do resultado para a tela do dispositivo. Esse direcionamento do resultado da soma deve ficar a cargo do `main`. Essa é a situação desejada e deve ser implementada na medida do possível.

Quando uma função não tem retorno, pode-se escrever `void` (vazio) antes do nome da função.

▶▶▶ Exercício:

1. Faça um programa que leia os valores para aplicar a 2ª lei de Ohm. O cálculo deve ser realizado em uma função com retorno. Sendo: $R = \rho \cdot L \div A$.

2. Elabore um programa que leia dois números `float` e implemente 4 funções com retorno: adição, subtração, multiplicação e divisão.

3. Elabore um programa com uma função chamada `parimpar` com retorno, para retornar 0, se um número é par, ou retornar 1, se um número é ímpar. No `main`, utilize o comando `if` para chamar a função e identificar o retorno. Se o retorno for 0, escreva a frase "é par", senão, escreva a frase "é ímpar".

Observe a dica:

```
if (parimpar() == 0)
    printf "é par";
else printf "é ímpar";
```

A mesma função pode ser escrita com retorno e **parâmetros**.

main.c	Run	Output
<pre>1 #include <stdio.h> 2 int soma(int n1, int n2){ 3 return n1+n2; 4 } 5 main() { 6 int n1,n2; 7 printf("Informe um número:"); 8 scanf("%i",&n1); 9 printf("Informe outro número:"); 10 scanf("%i",&n2); 11 printf("Soma: %i", soma(n1,n2)); 12 }</pre>		<pre>/tmp/0h1o9kIotN.o Informe um número:7 Informe outro número:1 Soma: 8</pre>

Linhas 2 a 4: implementação da função `soma`. Observe que após o nome da função `soma`, colocam-se os parâmetros `n1` e `n2`. Parâmetros são os dados que serão manipulados pela função.

Linha 6: observe que as variáveis estão no bloco do `main`. São **variáveis locais**.

Linha 11: uma função com parâmetros deve ser chamada como os dados que serão manipulados entre parênteses.

A utilização de parâmetros permite que as variáveis globais possam ser convertidas em variáveis locais. Essa é a situação desejada e deve ser implementada na medida do possível.

A variável local só existe dentro da função. Ela é criada durante a execução da função, e após o término desta, ela deixa de existir. Assim, não é possível acessar a variável local fora da função, a menos que se use a passagem de parâmetros junto à chamada da mesma.

É a melhor abordagem, pois otimiza o uso da memória do dispositivo, alocando espaço somente quando necessário, já que a variável global reserva espaço na memória durante todo o tempo de execução do programa, mesmo não estando sendo utilizada por um determinado momento.

Define-se como **escopo de variáveis** o fato das mesmas serem variáveis locais ou globais.

▶▶▶ Exercício:

1. Faça um programa que leia os valores para aplicar a 2ª lei de Ohm. O cálculo deve ser realizado em uma função com parâmetros. Sendo: $R = \rho \cdot L \div A$.
2. Elabore um programa que leia dois números `float` e implemente 4 funções com retorno e parâmetros: adição, subtração, multiplicação e divisão.
3. Elabore um programa de acordo com as instruções:
 - a) No `main`, solicite o preço e a quantidade de um produto comprado, faça a chamada da função descrita a seguir.
 - b) Crie uma função com parâmetro e retorno para retornar o total da compra (preço x quantidade). Os parâmetros são o preço e a quantidade.
4. Altere o `main` do programa anterior para solicitar 5 vezes o preço e a quantidade de um produto comprado e executar a função para cada solicitação.

▶▶▶ Exercício de revisão:

1. Elabore um programa com 2 funções, sem retorno e sem parâmetro para calcular a área do triângulo e do retângulo. As variáveis são base e altura.
2. Elabore um programa com 2 funções, com retorno e sem parâmetro para calcular a área do triângulo e do retângulo. As variáveis são base e altura.
3. Elabore um programa com 2 funções, com retorno e com parâmetro para calcular a área do triângulo e do retângulo. As variáveis são base e altura.
4. Elabore um programa que, no main, pergunta um valor em metros e faça 3 funções, sem retorno e sem parâmetro para mostrar o valor correspondente em decímetros, centímetros e milímetros.
5. Elabore um programa que, no main, pergunta um valor em metros e faça 3 funções, com retorno e sem parâmetro para mostrar o valor correspondente em decímetros, centímetros e milímetros.
6. Elabore um programa que, no main, pergunta um valor em metros e faça 3 funções, com retorno e com parâmetro para mostrar o valor correspondente em decímetros, centímetros e milímetros.
7. Elabore um programa com uma função com retorno e sem parâmetro, que calcule a distância entre dois pontos de acordo com a fórmula a seguir.

$$d = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$$

8. Elabore um programa que solicite o salário da pessoa. Em uma função com retorno e com parâmetro, deve-se retornar o imposto devido: 5% do salário.
9. Elabore um programa que com base em duas notas do aluno (P1 e P2), informe a média em uma função com retorno e com parâmetro. Em uma outra função sem retorno e com parâmetro, sendo a P2 o parâmetro, faça o seguinte teste: se $P2 < 4$ exibir na tela a mensagem “Aluno em Recuperação final por nota”.
10. Dadas duas matrizes reais 2×2 , A e B, elabore um programa com uma função sem retorno e sem parâmetro que leia as matrizes. Faça uma função sem retorno e sem parâmetro calcular $A + B$. Faça uma função sem retorno e sem parâmetro calcular $A - B$. As variáveis são globais e o main somente chamará as funções.