

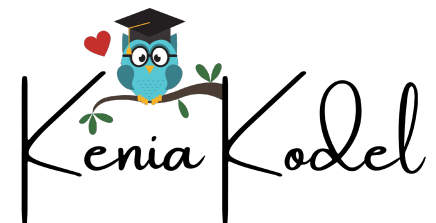


UNIVERSIDADE
FEDERAL DE
SERGIPE



Construção de Funções

PROGRAMAÇÃO IMPERATIVA



Construção de Funções em C

DEFINIÇÃO

Segundo DEITEL e DEITEL (2011), página 114, “[...] a melhor maneira de desenvolver e manter um programa grande é construí-lo a partir de partes menores ou de **módulos**, cada um mais facilmente administrável que o programa original. Essa técnica é chamada de **dividir e conquistar**. [...] Os módulos em C são chamados de **funções**. Os programas em C normalmente são escritos combinando-se novas funções com funções pré-definidas, disponíveis na **biblioteca-padrão de C**. ”

```
1. /*
2.  Quadrado de um número dado N.
3.  */
4.
5. #include <stdio.h>
6.
7. int quadrado (int x) {
8.  //retorna o quadrado de x passado como parâmetro
9.  return x * x; }
10.
11. int main()
12. {
13.     printf("Numero: ");
14.     int N;
15.     scanf("%d", &N);
16.     printf("Quadrado: %d", quadrado(N));
17.     return 0;
18. }
```



Como funciona?

```

1  /*
2   Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23  }

```



Como funciona?

```

1  /*
2   Cálculo da hipotenuza, dados os catetos. Sabendo que hipotenusa ao
3   quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23  }

```

Os programas em C normalmente são escritos combinando-se novas funções com funções pré-definidas, disponíveis na **biblioteca-padrão de C**. [DEITEL e DEITEL, 2011, p. 114]

Construção de Funções em C

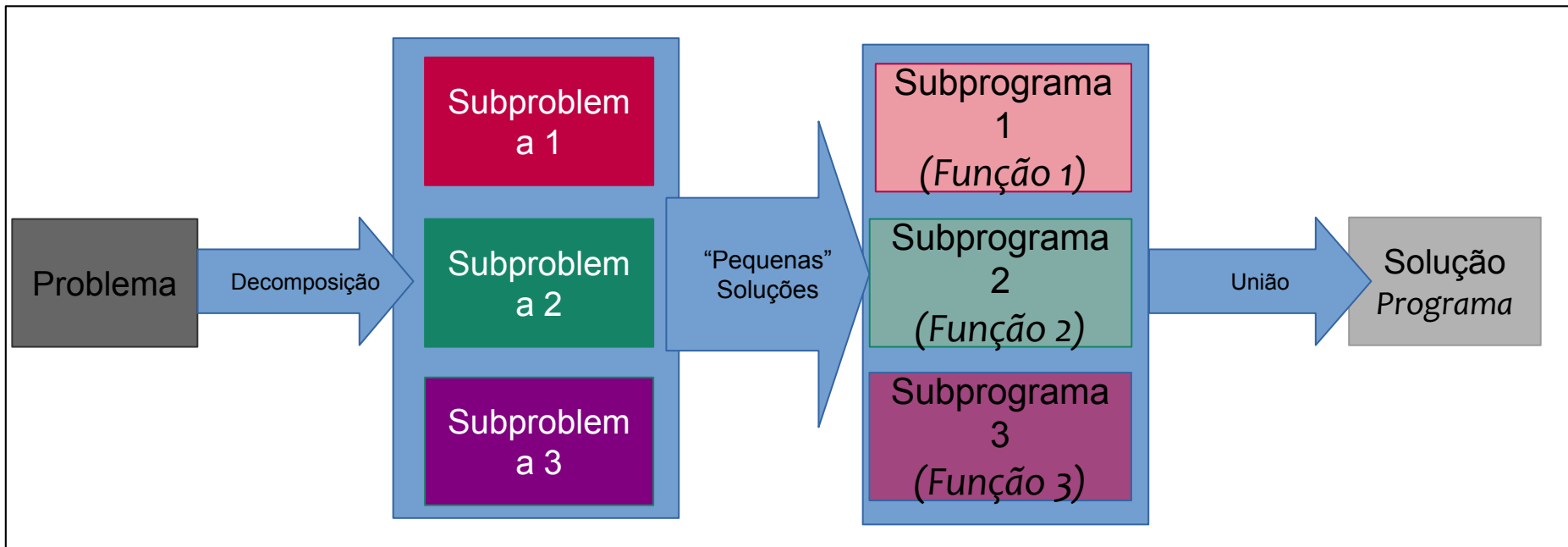
DEFINIÇÃO

Em outras palavras, no processo de construção de soluções computacionais algorítmicas, as **situações problemas podem ser decompostas em subproblemas** cujas lógicas de **resolução**, por sua vez, são mais fáceis de serem compreendidas e **implementadas por subprograma ou módulos, ou ainda funções.**

Construção de Funções em C

DEFINIÇÃO

Em geral, no processo de construção de soluções computacionais algorítmicas, as **situações problemas** podem ser **decompostas em subproblemas** cujas lógicas de **resolução**, por sua vez, são mais fáceis de serem compreendidas e implementadas por subprograma ou módulos, ou ainda funções.



Na construção de uma ponte, a ser efetuada em etapas, tem-se gastos com material de construção - GMaterial e com mão de obra - GMaoObra, bem como uma previsão dos gastos para finalizar todas as etapas - GTotal. Construir programa em C, para dados GMaterial, GMaoObra e GTotal, exibir o percentual da obra que falta construir.

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}
int main(){
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f",&GMaterial);
    printf("Gasto parcial com mao de obra? ");
    scanf("%f",&GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f",&GTotal);
    PFalta = RestaFazer(GMaterial,GMaoObra,GTotal);
    printf("Para concluir falta %.1f%%.",PFalta);
    return 0; }
```



Como funciona?

```
Qual o gasto com material ate este momento? 50
Qual o gasto com mao de obra ate este momento? 25
Qual a previsao de gasto total com a ponte? 1000
Para concluir a obra falta 92.5%.

-----
(program exited with code: 0)

Pressione qualquer tecla para continuar...
```


Na construção de uma ponte, a ser efetuada em etapas, tem-se gastos com material de construção - GMaterial e com mão de obra - GMaoObra, bem como uma previsão dos gastos para finalizar todas as etapas - GTotal. Construir programa em C, para dados GMaterial, GMaoObra e GTotal, exibir o percentual da obra que falta construir.

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main(){
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f", &GMaterial);
    printf("Gasto parcial com mao de obra? ");
    scanf("%f", &GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f", &GTotal);
    PFalta = RestaFazer(GMaterial, GMaoObra, GTotal);
    printf("Para concluir falta %.1f%%.", PFalta);
    return 0; }
```

Há subproblemas?



Na construção de uma ponte, a ser efetuada em etapas, tem-se gastos com material de construção - GMaterial e com mão de obra - GMaoObra, bem como uma previsão dos gastos para finalizar todas as etapas - GTotal. Construir programa em C, para dados GMaterial, GMaoObra e GTotal, exibir o percentual da obra que falta construir.

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main() {
    //Entrada de dados.
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f",&GMaterial);
    printf("Gasto parcial com mao de obra? ");
    scanf("%f",&GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f",&GTotal);
    //Processamento. Cômputo. Cálculo.
    PFalta = RestaFazer(GMaterial, GmaoObra, Gtotal);
    //Saída de dados.
    printf("Para concluir falta %.1f%%.", PFalta);
    return 0; }
```

Construção de Funções em C

DEFINIÇÃO

Considerando o programa de construção de ponte foram identificados como subproblemas:

- (1) entrada de dados;
- (2) cálculo do percentual que falta construir;
- (3) saída de dados.

Na construção de uma ponte, a ser efetuada em etapas, tem-se gastos com material de construção - GMaterial e com mão de obra - GMaoObra, bem como uma previsão dos gastos para finalizar todas as etapas - GTotal. Construir programa em C, para dados GMaterial, GMaoObra e GTotal, exibir o percentual da obra que falta construir.

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main() {
    //Entrada de dados.
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f",&GMaterial);
    printf("Gasto parcial com mao de obra? ");
    scanf("%f",&GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f",&GTotal);
    //Processamento. Cômputo. Cálculo.
    PFalta = RestaFazer(GMaterial, GmaoObra, Gtotal);
    //Saída de dados.
    printf("Para concluir falta %.1f%%.", PFalta);
    return 0; }
```

Há subprogramas /
módulos / funções?



Na construção de uma ponte, a ser efetuada em etapas, tem-se gastos com material de construção - GMaterial e com mão de obra - GMaoObra, bem como uma previsão dos gastos para finalizar todas as etapas - GTotal. Construir programa em C, para dados GMaterial, GMaoObra e GTotal, exibir o percentual da obra que falta construir.

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main(){
    //Entrada de dados.
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f",&GMaterial);
    printf("Gasto parcial com mao de obra? ");
    scanf("%f",&GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f",&GTotal);
    //Processamento. Cômputo. Cálculo.
    PFalta = RestaFazer(GMaterial, GmaoObra, Gtotal);
    //Saída de dados.
    printf("Para concluir falta %.1f%%.", PFalta);
    return 0; }
```

RestaFazer e main
são subprogramas.

Para efetuar compra online é preciso dispor de valor para cobrir o valor do produto, bem como do valor do frete. Para tanto é preciso poupar e contar com cupons de desconto sobre o frete. Construir programa em C para ler: (a) valor final (produto e frete), (b) valor do cupom (desconto) para frete, e (c) quanto foi economizado; e retornar quanto falta em percentual para efetuar a compra.

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main(){
    float VFinal, Cupom, Economia, PFalta;
    printf("Qual o valor final (produto + frete)? ");
    scanf("%f",&VFinal);
    printf("Valor do cupom de desconto? ");
    scanf("%f",&Cupom);
    printf("Qual o valor economizado? ");
    scanf("%f",&Economia);
    PFalta = RestaFazer(Economia,Cupom,VFinal);
    printf("Para concluir compra falta %.1f%%.",PFalta);
    return 0; }
```

```
Qual o valor final (produto + frete)? 2000
Qual o valor do cupom de desconto sobre o frete? 25
Qual o valor economizado? 175
Para concluir a compra falta 90.0%.

-----
(program exited with code: 0)

Pressione qualquer tecla para continuar. . .
```

Para efetuar compra online é preciso dispor de valor para cobrir o

o valor do produto, bem como do valor do frete. Para tanto é preciso

poupar e contar com cupons de desconto sobre o frete.

Construir

programa em C para ler: (a) valor final (produto e frete), (b) valor do

cupom (desconto) para frete, e (c) quanto foi economizado; e retornar

quanto falta em percentual para

```
1. #include <stdio.h>
2.
3. float RestaFazer(float Feito1, float Feito2, float Total)
4. {
5.     float Parcial;
6.     Parcial = Feito1 + Feito2;
7.     float Resta;
8.     Resta = Total - Parcial;
9.     //Retorno em percentual.
10.    return Resta/Total*100;
11. }
12.
13. int main()
14. {
15.     float VFinal, Cupom, Economia, PFalta;
16.     printf("Qual o valor final (produto + frete)? ");
17.     scanf("%f",&VFinal);
18.     printf("Valor do cupom de desconto? ");
19.     scanf("%f",&Cupom);
20.     printf("Qual o valor economizado? ");
21.     scanf("%f",&Economia);
22.     PFalta = RestaFazer(Economia,Cupom,VFinal);
23.     printf("Para concluir compra falta %.1f%%.",PFalta);
24.     return 0;
25. }
```

Há subproblemas?

Há subprogramas?



Construção de Funções em C

DEFINIÇÃO

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main(){
    float VFinal, Cupom, Economia, PFalta;
    printf("Qual o valor final (produto + frete)? ");
    scanf("%f",&VFinal);
    printf("Valor do cupom de desconto? ");
    scanf("%f",&Cupom);
    printf("Qual o valor economizado? ");
    scanf("%f",&Economia);
    PFalta = RestaFazer(Economia,Cupom,VFin
    printf("Para concluir compra falta %.1f
    return 0; }
```

```
#include <stdio.h>

float RestaFazer(float Feito1, float Feito2, float Total){
    float Parcial;
    Parcial = Feito1 + Feito2;
    float Resta;
    Resta = Total - Parcial;
    //Retorno em percentual.
    return Resta/Total*100;}

int main(){
    //Entrada de dados.
    float GMaterial, GMaoObra, GTotal, PFalta;
    printf("Gasto parcial com material: ");
    scanf("%f",&GMaterial);
    printf("Gasto parcial com mão de obra? ");
    scanf("%f",&GMaoObra);
    printf("Gasto total com a ponte: ");
    scanf("%f",&GTotal);
    //Processamento. Cômputo. Cálculo.
    PFalta = RestaFazer(GMaterial,GMaoObra,Gtotal);
    //Saída de dados.
    printf("Para concluir falta %.1f%%",PFalta);
    return 0; }
```

Aqui observa-se o **reuso** de um módulo, ou subprograma; evidenciando outra vantagem de uso da modularização.

Construção de Funções em C

DEFINIÇÃO

Em C os módulos são construídos por recurso originariamente destinado a construção de função:

```
<tipo do retorno> <nome>([parâmetros...])  
{  
    <instruções>;  
    [return [<valor de retorno>]];  
}
```

```
float soma(float x, float y)  
{  
    float res;  
    res = x + y;  
    return res;  
}
```

Função

Construção de Funções em C

DECLARAÇÃO

```
<tipo do retorno> <nome>([parâmetros...])  
{  
    <instruções>;  
    [return [<valor de retorno>]];  
}
```

Onde:

- `<Nome>` é o nome do módulo;
- `parâmetros...` lista opcional de itens denominados argumentos, ou parâmetros; de comunicação entre programa e módulo.
- `<instruções>;` comando ou bloco de comandos correspondentes à resoluções de um subproblema.
- `<retorno>` conterá o valor esperado pelo programa na resolução do subproblema. O tipo do retorno (int, ou float, ou char, dentre outros) é definido no cabeçalho do módulo `<tipo do retorno>` e é efetuado por meio da instrução `<return>`.

Construção de Funções em C

DECLARAÇÃO

- `<Nome>` é o nome do módulo;
- `parâmetros...` lista opcional de itens denominados argumentos, ou parâmetros; de comunicação entre programa e módulo.
- `<instruções>;` comando ou bloco de comandos correspondentes à resoluções de um subproblema.
- `<retorno>` conterá o valor esperado pelo programa na resolução do subproblema. O tipo do retorno (int, ou float, ou char, dentre outros) é definido no cabeçalho do módulo `<tipo do retorno>` e é efetuado por meio da instrução `<return>`.

```
float soma(float x, float y)
{
    float res;
    res = x + y;
    return res;
}
```

Identificar elementos de
composição de funções.



Construção de Funções em C

DECLARAÇÃO

- `<Nome>` é o nome do módulo;
- `parâmetros...` lista opcional de itens denominados argumentos, ou parâmetros; de comunicação entre programa e módulo.
- `<instruções>;` comando ou bloco de comandos correspondentes à resoluções de um subproblema.
- `<retorno>` conterá o valor esperado pelo programa na resolução do subproblema. O tipo do retorno (int, ou float, ou char, dentre outros) é definido no cabeçalho do módulo `<tipo do retorno>` e é efetuado por meio da instrução `<return>`.

```
float RestaFazer(float Feito1, float Feito2, float Total){  
    float Parcial;  
    Parcial = Feito1 + Feito2;  
    float Resta;  
    Resta = Total - Parcial;  
    //Retorno em percentual.  
    return Resta/Total*100;}  
}
```

Identificar elementos de composição de funções.



Construção de Funções em C

CHAMADA / INVOCAÇÃO

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2,
float Parcial;
Parcial = Feito1 + Feito2;
float Resta;
Resta = Total - Parcial;
//Retorno em percentual.
return Resta/Total*100;}

int main() {
float VFinal, Cupom, Economia, PFalta;
printf("Qual o valor final (produto + f
scanf("%f",&VFinal);
printf("Valor do cupom de desconto? ");
scanf("%f",&Cupom);
printf("Qual o valor economizado? ");
scanf("%f",&Economia);
PFalta = RestaFazer(Economia,Cupom,VFinal);
printf("Para concluir compra falta %.1f%%",PFalta);
return 0; }
```

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2, float Total) {
float Parcial;
Parcial = Feito1 + Feito2;
float Resta;
Resta = Total - Parcial;
//Retorno em percentual.
return Resta/Total*100;}

int main() {
//Entrada de dados.
float GMaterial, GMaoObra, GTotal, PFalta;
printf("Gasto parcial com material: ");
scanf("%f",&GMaterial);
printf("Gasto parcial com mão de obra? ");
scanf("%f",&GMaoObra);
printf("Gasto total com a ponte: ");
scanf("%f",&GTotal);
//Processamento. Cômputo. Cálculo.
PFalta = RestaFazer(GMaterial,GMaoObra,Gtotal);
//Saída de dados.
printf("Para concluir falta %.1f%%",PFalta);
return 0; }
```

Construção de Funções em C

CHAMADA / INVOCACÃO

```
float soma(float x, float y)
{
    float res;
    res = x + y;
    return res;
}
```

Chamadas ou invocações – ocorrem quando um subprograma, depois de declarado, é usado, provocando sua execução.

```
PrecoFinal=soma (PrecoProduto,Frete);
```

```
NotaFinal=soma (Prova,Atividades);
```

```
PesoFinal=soma (PesoInicial,PesoObtido)
```

```
printf("Total: %.2f", soma(Valor,Imposto));
```

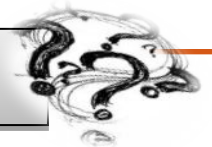
Construção de Funções em C

CHAMADA / INVOCAÇÃO

```
PrecoFinal=soma (PrecoProduto, Frete);  
NotaFinal=soma (Prova, Atividades);  
PesoFinal=soma (PesoInicial, PesoObtido);  
printf("Total: %.2f", soma (Valor, Imposto));
```

```
float soma(float x, float y)  
{  
    float res;  
    res = x + y;  
    return res;  
}
```

Como funciona?



Construção de Funções em C

COMPOSIÇÃO DE UMA FUNÇÃO

Função Soma

```
float soma(float x, float y)
{
    float res;
    res = x + y;
    return res;
}
```

Nas funções são identificadas:

- a) assinaturas, protótipo ou cabeçalho;
- b) corpo;
- c) return conforme o retorno da assinatura.

Outra Resolução

```
float soma(float x, float y)
{
    return x + y;
}
```


Revisando

Para efetuar compra online é preciso dispor de valor para cobrir o valor do produto, bem como do valor do frete. Para tanto é preciso poupar e contar com cupons de desconto sobre o frete. Construir programa em C para ler: (a) valor final (produto e frete), (b) valor do cupom (desconto) para frete, e (c) quanto foi economizado; e retornar quanto falta em percentual para efetuar a compra.

```
1. #include <stdio.h>
2.
3. float RestaFazer(float Feito1, float Feito2, float Total)
4. {
5.     float Parcial;
6.     Parcial = Feito1 + Feito2;
7.     float Resta;
8.     Resta = Total - Parcial;
9.     //Retorno em percentual.
10.    return Resta/Total*100;
11. }
12.
13. int main()
14. {
15.     float VFinal, Cupom, Economia, PFalta;
16.     printf("Qual o valor final (produto + frete)? ");
17.     scanf("%f",&VFinal);
18.     printf("Valor do cupom de desconto? ");
19.     scanf("%f",&Cupom);
20.     printf("Qual o valor economizado? ");
21.     scanf("%f",&Economia);
22.     PFalta = RestaFazer(Economia,Cupom,VFinal);
23.     printf("Para concluir compra falta %.1f%%.",PFalta);
24.     return 0;
25. }
```

(1) Qual o nome da função do exemplo? (2) Em que linha consta sua assinatura? (3) Em que linhas constam seu corpo? (4) Em que linha consta seu retorno? (5) Em que linha consta sua invocação?



Construção de Funções em C

APLICAÇÃO DE FUNÇÃO

Há pessoas que precisam fazer tratamento de saúde contínuo (em comprimidos), a exemplo de reposição hormonal. Sabendo que uma pessoa deve fazer uso de X comprimidos ao mês e que em cada caixa desse há N unidades (comprimidos); escrever um programa em C onde se lê X e N e é exibido o número de caixas a serem adquiridas.

No pré-escolar “Balão Mágico” o número máximo de alunos de uma turma é definido anualmente, após as reformas de fim de ano e evolução curricular. Escrever programa em C onde se lê o número total de alunos matriculados, o número máximo de alunos por turma e exibir o número de turmas a serem formadas.

Em C há a função de

double ceil(double x);

da biblioteca `math.h`, a qual retorna o menor inteiro maior ou igual ao valor passado como parâmetro.

Se `A = ceil(2.8);` A passa a conter 3.0.

Se `B = ceil(33.2);` A passa a conter 34.0.

Se `C = ceil(12.0);` B contém 12.0



É possível propor uma função que atenda às duas situações?

Construção de Funções em C

APLICAÇÃO DE FUNÇÃO

Uma pessoa deve fazer uso de X comprimidos ao mês e que em cada caixa desse há N unidades (comprimidos); escrever um programa em C onde se lê X e N e é exibido o número de caixas a serem adquiridas.

Escrever programa em C onde se lê o número total de alunos matriculados, o número máximo de alunos por turma e exibir o número de turmas a serem formadas.

Em C há a função de

double ceil(double x);

da biblioteca `math.h`, a qual retorna o menor inteiro maior ou igual ao valor passado como parâmetro.

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

int QuocienteExato(int Dividendo, int Divisor)
//Quociente arredondado, matematicamente.
{
    float Divisao;
    Divisao = (float) Dividendo / Divisor;
    Divisao = ceil(Divisao);
    return (int) Divisao;
}
```

Construção de Funções em C

APLICAÇÃO DE FUNÇÃO

Uma pessoa deve fazer uso de X comprimidos ao mês e que em cada caixa desse há N unidades (comprimidos); escrever um programa em C onde se lê X e N e é exibido o número de caixas a serem adquiridas.

```
1  /*
2   Entradas: Total de comprimidos ao mês e Número de comprimidos em
3   caixas. Saída: Caixas a comprar por mês.
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int QuocienteExato(int Dividendo, int Divisor)
10 {
11     float Divisao;
12     Divisao = (float) Dividendo / Divisor;
13     Divisao = ceil(Divisao);
14     return (int) Divisao;
15 }
16
17 int main()
18 {
19     int TMensal, TCaixa;
20     printf("Numeros de comprimidos ao mes: ");
21     scanf("%d",&TMensal);
22     printf("Numeros de comprimidos numa caixa: ");
23     scanf("%d",&TCaixa);
24     printf("Compre %d caixas.",QuocienteExato(TMensal,TCaixa));
25     return 0;
26 }
```

C:\Windows\SYSTEM32\cmd.exe

```
Numeros de comprimidos ao mes: 90
Numeros de comprimidos numa caixa: 20
Compre 5 caixas.
-----
```

Construção de Funções em C

APLICAÇÃO DE FUNÇÃO

Escrever programa em C onde se lê o número total de alunos matriculados, o número máximo de alunos por turma e exibir o número de turmas a serem formadas.

Inspiração

```
1  /*
2   Entradas: Total de comprimidos ao mês e Número de comprimidos em
3   caixas. Saída: Caixas a comprar por mês.
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int QuocienteExato(int Dividendo, int Divisor)
10 {
11     float Divisao;
12     Divisao = (float) Dividendo / Divisor;
13     Divisao = ceil(Divisao);
14     return (int) Divisao;
15 }
16
17 int main()
18 {
19     int TMensal, TCaixa;
20     printf("Numeros de comprimidos ao mes: ");
21     scanf("%d",&TMensal);
22     printf("Numeros de comprimidos numa caixa: ");
23     scanf("%d",&TCaixa);
24     printf("Compre %d caixas.",Quo
25     return 0;
26 }
```



Como resolver esta
outra situação
problema?

Construção de Funções em C

PROCEDIMENTO \times FUNÇÃO

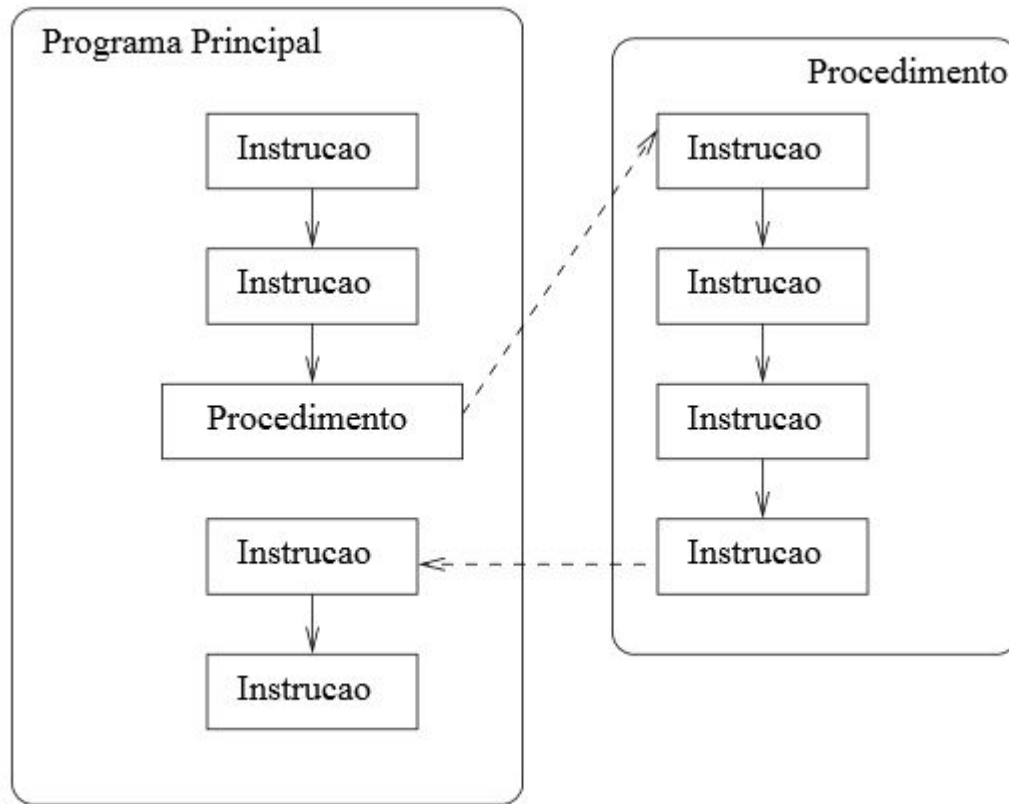
Um **módulo** ou subprograma corresponde a trecho de código executados quando o programa principal o invoca, interrompendo o fluxo de execução do programa e desviando-o para a execução das instruções deste.

Existem dois tipos de módulos:

- a) Procedimento** - ao final deste, o fluxo de execução retorna ao comando seguinte à chamada do procedimento no programa principal.
- b) Função** - semelhante ao procedimento, exceto que retorna com um valor para o programa principal.

Construção de Funções em C

EXECUÇÃO DE FUNÇÕES EM PROGRAMAS



Esquema evidencia funcionamento de procedimento.

As **funções** se comportam da mesma forma, mas retornam valor.

As funções são usadas em expressões, e os procedimentos, como comandos (isolados).

Construção de Funções em C

PAPEL DO RETURN EM FUNÇÕES

Exemplo: finalizando a função com return

```
01  int maior(int x, int y){  
02      if(x > y)  
03          return x;  
04      else  
05          return y;  
06      printf("Fim da funcao\n");  
07  }
```

Exemplo de Backes.

O **return** provoca a interrupção da função.


No exemplo, o **printf** não é executado nunca.



Construção de Funções em C

DECLARAÇÃO DE FUNÇÕES

```
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```



Recomenda-se que a definição dos subprogramas seja efetuada fora e antes do módulo principal: `main`. Quando após, deve haver uma assinatura (cabeçalho) do módulo antes do `main`.

Construção de Funções em C

DECLARAÇÃO DE FUNÇÕES

Exemplo: função declarada *depois* da cláusula main.

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  //protótipo da função
04  int Square (int a);
05
06  int main(){
07      int n1,n2;
08      printf("Entre com um numero: ");
09      scanf("%d", &n1);
10      n2 = Square(n1);
11      printf("O seu quadrado vale: %d\n", n2);
12      system("pause");
13      return 0;
14  }
15
16  int Square (int a){
17      return (a*a);
18  }
```

Quando opta-se por declarar o módulo após o **main**, deve haver uma assinatura (cabeçalho ou protótipo) deste antes do **main**.

E no protótipo não são necessários os nomes dos parâmetros.

Exemplo de Backes.

Construção de Funções em C

CHAMADA OU INVOCAÇÃO DE FUNÇÕES

```
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```

Efetuada a declaração de um módulo, este pode ser usado (“chamado”, acionado, executado, invocado) em qualquer local do programa, inclusive dentro de outros módulos; e tantas vezes quantas forem necessárias.

Construção de Funções em C

CHAMADA OU INVOCAÇÃO DE FUNÇÕES

```
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```

Um módulo é acionado através de seu nome. Para invocar o módulo **quadrado(x)**, por exemplo, basta usar dentre as instruções: **quadrado(x)** substituindo x por uma variável, ou valor, do tipo inteiro.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

Usando módulo, evita-se redundância de código.

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2,
float Parcial;
Parcial = Feito1 + Feito2;
float Resta;
Resta = Total - Parcial;
//Retorno em percentual.
return Resta/Total*100;}

int main() {
float VFinal, Cupom, Economia, PFalta;
printf("Qual o valor final (produto + f
scanf("%f",&VFinal);
printf("Valor do cupom de desconto? ");
scanf("%f",&Cupom);
printf("Qual o valor economizado? ");
scanf("%f",&Economia);
PFalta = RestaFazer(Economia,Cupom,VFinal);
printf("Para concluir compra falta %.1f%%",PFalta);
return 0; }
```

```
#include <stdio.h>
float RestaFazer(float Feito1, float Feito2, float Total) {
float Parcial;
Parcial = Feito1 + Feito2;
float Resta;
Resta = Total - Parcial;
//Retorno em percentual.
return Resta/Total*100;}

int main() {
//Entrada de dados.
float GMaterial, GMaoObra, GTotal, PFalta;
printf("Gasto parcial com material: ");
scanf("%f",&GMaterial);
printf("Gasto parcial com mão de obra? ");
scanf("%f",&GMaoObra);
printf("Gasto total com a ponte: ");
scanf("%f",&GTotal);
//Processamento. Cômputo. Cálculo.
PFalta = RestaFazer(GMaterial,GMaoObra,Gtotal);
//Saída de dados.
printf("Para concluir falta %.1f%%",PFalta);
return 0; }
```

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

Usando módulo, evita-se redundância de código.

```
1  /*
2   * Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   * quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

```
1  /*
2   * Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   * quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```



Se se resolvesse ajustar o tipo do parâmetro e o tipo do retorno da função quadrado para float. Como ajustar?

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

```
1  /*
2   | Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   | quadrado é a soma do quadrado dos catetos.
4   | */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10 | //retorna o quadrado de x passado como parâmetro
11 |     return x * x;}
12
13  int main()
14  {
15 |     printf("Cateto 1: ");
16 |     int C1;
17 |     scanf("%d",&C1);
18 |     printf("Cateto 2: ");
19 |     int C2;
20 |     scanf("%d",&C2);
21 |     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22 |     return 0;
23 | }
```

A aplicação de módulos / funções facilita a manutenção do código.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

```
1  /*
2   * Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   * quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```

Entendendo modularização, é possível observar que `main` é um módulo responsável por iniciar o programa e executar as instruções contidas neste; e é obrigatório.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

```
1  /*
2   * Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   * quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: %.1f",sqrt(quadrado(C1)+quadrado(C2)));
22     return 0;
23 }
```

Também entendendo modularização, é possível observar que já há módulos pré-definidos, a exemplo de: `printf()`, `scanf()`, `ceil()`, `sqrt()`, e outros.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

```
1  /*
2   * Cálculo da hipotenusa, dados os catetos. Sabendo que hipotenusa ao
3   * quadrado é a soma do quadrado dos catetos.
4   */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int quadrado(int x){
10     //retorna o quadrado de x passado como parâmetro
11     return x * x;}
12
13  int main()
14  {
15     printf("Cateto 1: ");
16     int C1;
17     scanf("%d",&C1);
18     printf("Cateto 2: ");
19     int C2;
20     scanf("%d",&C2);
21     printf("Hipotenusa: ");
22     return 0;
23 }
```

Observar que para aplicação/uso de uma função, a exemplo de: `printf()`, `scanf()`, `sqrt()`, `ceil()`, o programador não precisa conhecer seu código. É suficiente saber seu nome, parâmetros e objetivo.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

- Os módulos, então, podem ser entendidos como uma forma de se adicionar novos comandos à linguagem. Exemplos: `printf`, `scanf`, `quadrado`.
- Estes novos comandos aumentam o poder da linguagem podendo atender a necessidades específicas de cada problema.
- Também podem ser entendidos como uma forma de organizar melhor os programas favorecendo a legibilidade e manutenção destes.

Construção de Funções em C

FUNÇÕES: APLICAÇÃO E VANTAGENS

- Permitir o reaproveitamento de código já construído. Seja de própria autoria ou não.
- Evitar redundância de trechos de código.
- Facilitar a alteração de trechos de código; seja por evitar a redundância, seja por promover a legibilidade.
- Promover a legibilidade (fáceis de ler) e inteligibilidade (fáceis de compreender) do código.

Construção de Funções em C

EXERCÍCIO

Escrever programa em C para ler um número N e retornar N^3 (N ao cubo), aplicando função.

Construção de Funções em C

EXERCÍCIO

Uma pessoa deve fazer uso de X comprimidos ao mês e que em cada caixa desse há N unidades (comprimidos); escrever um programa em C, onde se lê X e N e é exibido o número de caixas a serem adquiridas, aplicando a função `QuocienteExato` dada abaixo, ajustando seu nome para `QuocienteInteiro` :

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

int QuocienteExato(int Dividendo, int Divisor)
//Quociente arredondado, matematicamente.
{
    float Divisao;
    Divisao = (float) Dividendo / Divisor;
    Divisao = ceil(Divisao);
    return (int) Divisao;
}
```

A função dada há a função de `double ceil(double x)`; da biblioteca `math.h`, a qual retorna o menor inteiro maior ou igual ao valor passado como parâmetro.

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}

int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d",&CoefA,&CoefB,&CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA,CoefB,CoefC,&Raiz1,&Raiz2);
    printf("As raizes sao %.1f e %.1f.\n",Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5,6,1,&Raiz1,&Raiz2,&EhReal);
    printf("As raizes sao %.1f e %.1f.\n",Raiz1, Raiz2);
    return 0;}
```


Construção de Funções em C

ESCOPO

- Com o uso de módulos, passamos a identificar dois tipos de variáveis.
 1. **Locais** – declaradas dentro dos módulos, inclusive dentro do principal
 2. **Globais** – declaradas fora dos módulos
- Esta característica é denominada escopo.
- Quando um módulo é executado, os itens locais são criados em memória e existem somente durante a vida (execução) deste subprograma.
- Já os globais existem durante a execução de todo programa.

Construção de Funções em C

EXEMPLO



Qual o escopo da variável Delta?

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}

int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d",&CoefA,&CoefB,&CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA,CoefB,CoefC,&Raiz1,&Raiz2);
    printf("As raizes sao %.1f e %.1f.\n",Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5,6,1,&Raiz1,&Raiz2);
    printf("As raizes sao %.1f e %.1f.\n",Raiz1, Raiz2);
    return 0;}
```

Construção de Funções em C

EXEMPLO



Qual o escopo das variáveis CoefA, CoefB e CoefC? Raiz1 e Raiz2?

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Neste exemplo observa-se o uso das variáveis locais Delta, CoefA, CoefB, CoefC, Raiz1 e Raiz2.

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

Neste exemplo não se observa o uso da variáveis globais.

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Os **itens locais** são reconhecidos somente dentro dos módulos onde são criados. Já os **globais**, em todo o programa; dentro e fora dos módulos.

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

As **variáveis globais** devem ser evitadas com vista à economia de memória e legibilidade de código.

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

Como dito, opcionalmente ao lado do nome do procedimento, pode ser listado, entre parênteses, os argumentos ou parâmetros. Os parâmetros servem como elo de comunicação entre módulos.

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```


Construção de Funções em C

TIPOS DE PARÂMETROS

- a) Os parâmetros são definidos, entre parênteses, ao lado do nome dos módulos:

`<tipo> <nome>;`

- a) Quando o parâmetro for de saída, ou seja, sofrer ajuste dentro do módulo, deve ser precedido por `*` na definição deste; e na invocação, por `&`; exceto se do tipo vetor.

Construção de Funções em C

TIPOS DE PARÂMETROS

Os parâmetros classificam-se como:

- a) de entrada** – aqueles que fornecerão dados úteis ao processamento do módulo. Dado sua funcionalidade, não devem ser alterados pelos subprogramas.
- b) de saída** – aqueles que conterão resultados obtidos pelo processamento dos módulos. Assim, durante a execução do subprograma, os valores destes parâmetros devem ser definidos.

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```



Quais os parâmetros
contidos no subprograma
calcularaizes?

Construção de Funções em C

EXEMPLO

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

void calcularaizes(int A, int B,
int C, float *R1, float *R2){
    int Delta;
    Delta = B*B-4*A*C;
    //Delta não menor que zero
    *R1=(-B+sqrt(Delta))/2*A;
    *R2=(-B-sqrt(Delta))/2*A;}
return;}
```

Neste caso tem-se os parâmetros **A**, **B** e **C** como de **entrada**; já que subsidiam o cálculo das raízes. E **R1** e **R2** como de **saída**; pois armazenam os resultados do processamento.

```
int main() {
    printf(">>> Equacao do 2o Grau <<<\n\n");
    printf("Quais os coeficientes? ");
    int CoefA, CoefB, CoefC;
    scanf("%d%d%d", &CoefA, &CoefB, &CoefC);
    float Raiz1, Raiz2;
    calcularaizes(CoefA, CoefB, CoefC, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    //outra chamada
    calcularaizes(5, 6, 1, &Raiz1, &Raiz2);
    printf("As raizes sao %.1f e %.1f.\n", Raiz1, Raiz2);
    return 0;}
```

Construção de Funções em C

EXERCÍCIO

Criar função para retornar o maior valor de três números passados como parâmetros. *A função criada deve ser testada num programa.*

Construção de Funções em C

EXERCÍCIO

Criar subprograma para receber três números (parâmetros) e retornar os três números ordenados. *A função criada deve ser testada num programa.*

Construção de Funções em C

EXERCÍCIO

Criar função para receber dois números(parâmetros): base e expoente e retornar a potência da base elevada ao expoente. *A função criada deve ser testada num programa.*

Programação Imperativa

COMPLEMENTAR AULA...

Fundamentos da Programação de Computadores

Ana Fernanda Gomes Ascencio
Edilene Aparecida Veneruchi de Campos

Capítulos
SubRotinas



Programação Imperativa

Curso de Linguagem C
UFMG

COMPLEMENTAR AULA...

*linux.ime.usp.br/~lu
casmmg/livecd/doc
umentacao/docume
ntos/curso_de_c/w
ww.ppgia.pucpr.br/
_maziero/ensino/so/
projetos/curso-c/c.h
tml*

Capítulos
Funções

Aula 1: Introdução e Sumário

Aula 2 - Primeiros Passos

Aula 3 - Variáveis, Constantes, Operadores e Expressões

Aula 4 - Estruturas de Controle de Fluxo

Aula 5 - Matrizes e Strings

Aula 6 - Ponteiros

Aula 7 - Funções



Programação Imperativa

PRÓXIMO PASSO



Estruturas de Decisão