

# SSC 0301 – ICC para Engenharia Ambiental

## *Definição e uso de funções*

Prof. Marcio E. Delamaro

[delamaro@icmc.usp.br](mailto:delamaro@icmc.usp.br)

# Recordando – comando if

- `if C1; else C2;`
- `if C1;`
- `if { C1; C2; } else {C3; C4;}`
- `if { C1; C2; }`

# Recordando – operadores lógicos

- AND (&&)
- OR (||)
- NOT (!)

# Funções

- Como vimos, a linguagem C tem diversas funções que são pré-definidas
  - Funções de entrada e saída (scanf e printf)
  - Funções matemáticas (sin e sqrt)
- Essas funções têm alguns componentes
  - Nome – é a forma de identificar qual função deve ser chamada (sin, sqrt, scanf, printf)
  - Parâmetros – são os valores que as funções precisam para fazerem o que deve ser feito (sin(3.14), sqrt(2.0), printf("O valor de x é: %d", x);)
  - O valor de retorno – é o resultado produzido pela chamada da função ( $x = \sin(3.14)$ )
  - Tipo – é o tipo do valor retornado (double sin, int scanf)

# Funções – nome

- Valem as mesmas regras aplicadas para variáveis
- Não podem existir duas funções com o mesmo nome

# Funções – parâmetros

- Colocados entre parênteses
- Separados por vírgula `scanf("%d %d", &i, &j)`
- Maioria tem um número fixo de parâmetros (`sin`, `sqrt`)
- Algumas aceitam um número qualquer de parâmetros (`scanf`, `printf`)
- Em geral, parâmetros são de tipos determinados

# Funções – valor de retorno

- Uma função produz um valor que pode (ou não) ser usado no retorno da sua chamada
- Por exemplo `x = sin(2*y*y)`
- Por exemplo `k = scanf("%d %d", &i, &j);`
- Em alguns casos o valor não precisa ser usado
  - `scanf("%d %d", &i, &j);`
  - `sin(x)`
- Os retornos têm tipos específicos

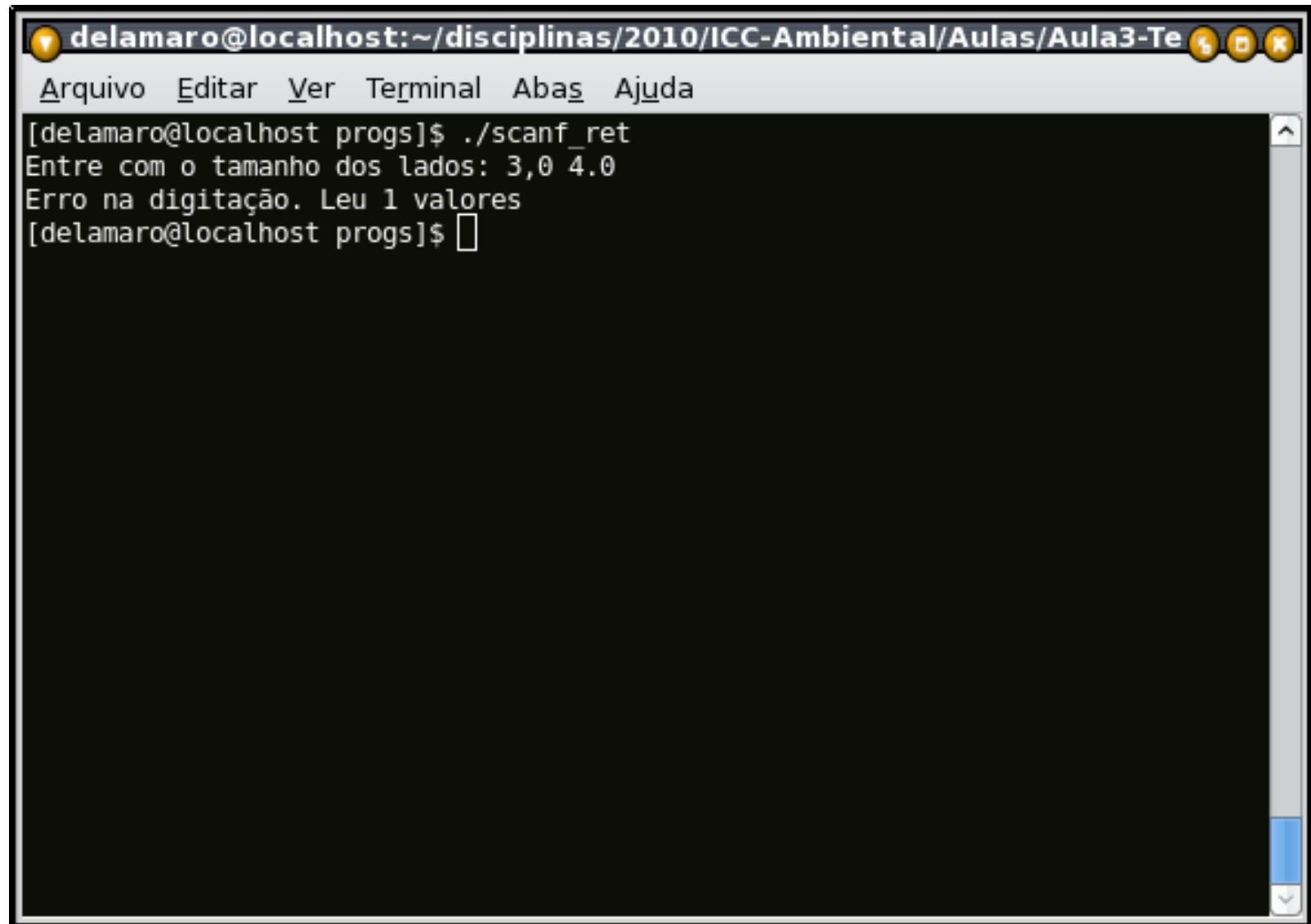
# Valor de retorno – exemplo

```
int main()
{
double cateto1, cateto2;
double hipo, cat1_2, cat2_2;
int k;

// le o primeiro valor
printf("Entre com o tamanho dos lados: ");
k = scanf("%lf %lf", &cateto1, &cateto2);
if ( k != 2 )
{
    printf("Erro na digitação. Leu %d valores\n", k);
    return 0;
}
cat1_2 = cateto1 * cateto1;
cat2_2 = cateto2 * cateto2;
hipo = cat1_2 + cat2_2;
hipo = sqrt(hipo);
printf("O valor da hipotenusa é: %-5.2lf\n", hipo);
}
```



# Valor de retorno – exemplo



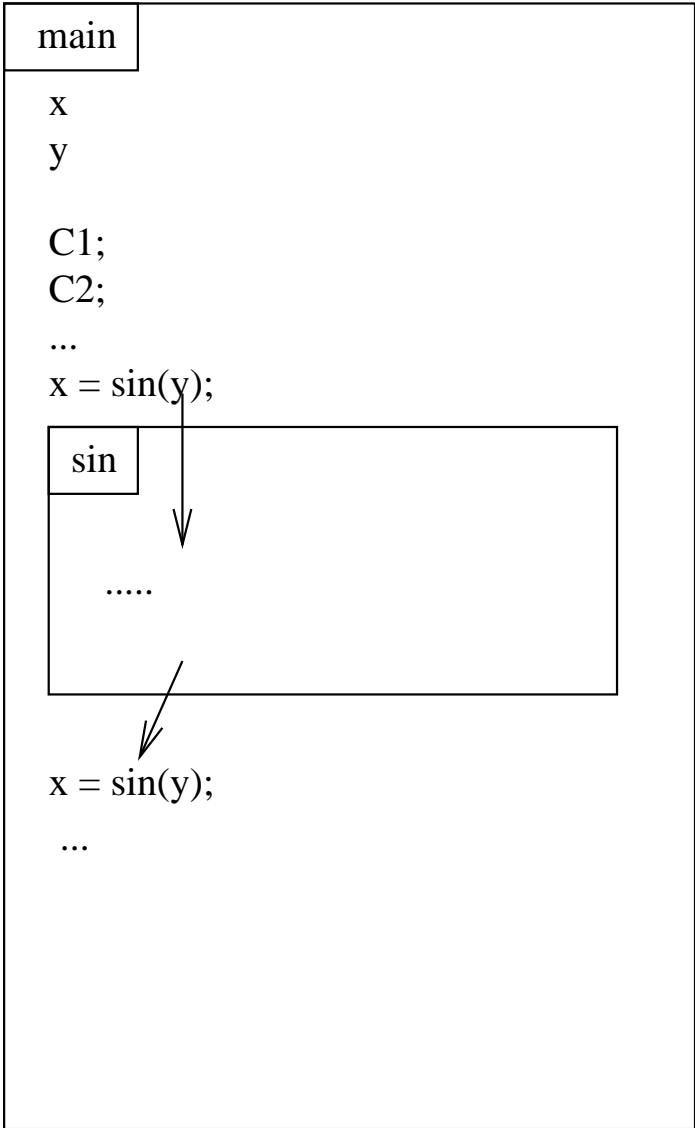
```
delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
[delamaro@localhost progs]$ ./scanf_ret
Entre com o tamanho dos lados: 3,0 4.0
Erro na digitação. Leu 1 valores
[delamaro@localhost progs]$
```

# A chamada da função

- A função é um programa igual àquele que estamos escrevendo
- Quando uma chamada de função acontece dentro do nosso programa, ele é interrompido
- A execução da função chamada acontece e ela é executada até que o resultado seja produzido
- O nosso programa “pega” (ou não) o valor e continua executando normalmente

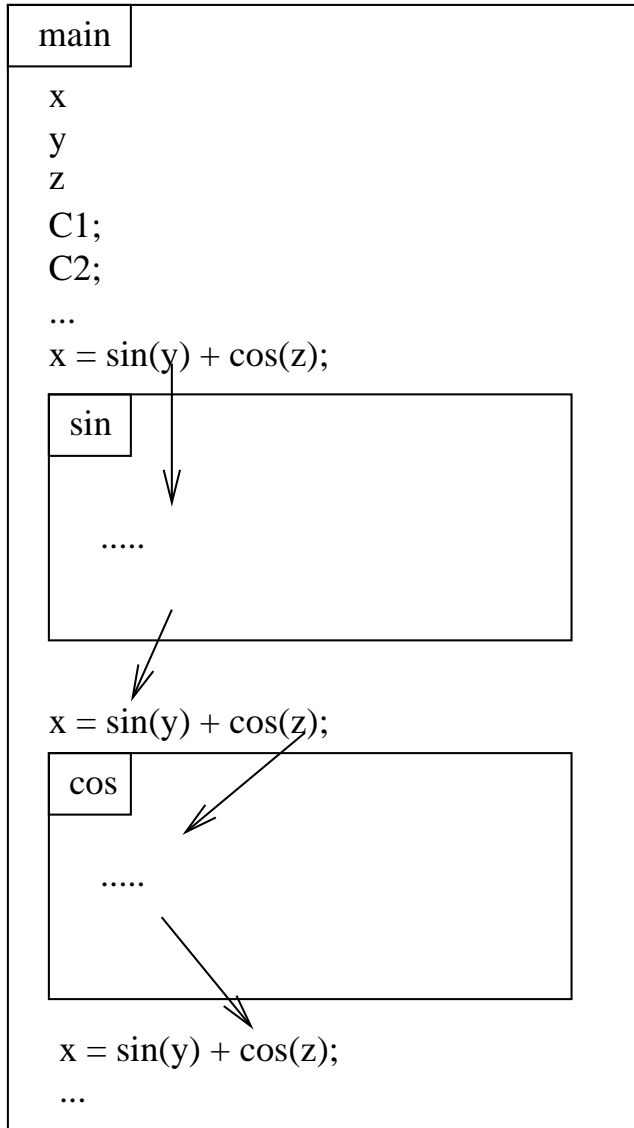
# Chamada – exemplo

•  $x = \sin(y)$



# Chamada – exemplo 2

•  $x = \sin(y) + \sin(z)$



# As nossas funções

- Se reparar a própria função `main` tem todos esses atributos
- Tipo `int`
- Nenhum parâmetro (por enquanto)
- Podemos definir outras funções além da principal
- O esquema é (quase) igual ao da função `main`

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?
  - `le_int`



# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?
  - `le_int`
- Qual é o tipo dessa função ?

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?
  - `le_int`
- Qual é o tipo dessa função ?
  - `int` (é o tipo do valor que ela produz)

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?
  - `le_int`
- Qual é o tipo dessa função ?
  - `int` (é o tipo do valor que ela produz)
- Como essa função vai ser usada?

# Nossas funções – Exemplo

- Vamos escrever uma função que a mensagem “Digite um inteiro ==>”, leia um número inteiro e retorne o seu valor
- Qual vai ser o nome dessa função?
  - `le_int`
- Qual é o tipo dessa função ?
  - `int` (é o tipo do valor que ela produz)
- Como essa função vai ser usada?
  - `k = le_int()`

# Antes de mais nada

- Declarar um **protótipo** da função
- O protótipo deve identificar o nome, o tipo e os parâmetros da função
- deve ser colocado no início do programa, antes da função `main`

# Antes de mais nada

- Declarar um **protótipo** da função
- O protótipo deve identificar o nome, o tipo e os parâmetros da função
- deve ser colocado no início do programa, antes da função `main`

```
int le_int();
```

# Declarar a função

```
#include <stdio.h>

int le_int(); // protótipo

// Declaração da função principal
int main()
{
    ...
}

// Declaração da função le_int
int le_int()
{
    ...
}
```

# Implementar a função

```
int le_int()  
{  
    int u;  
  
    printf("Digite um inteiro ==> ");  
    scanf("%d", &u);  
    return u;  
}
```



# Usar a função

```
#include <stdio.h>

int le_int();

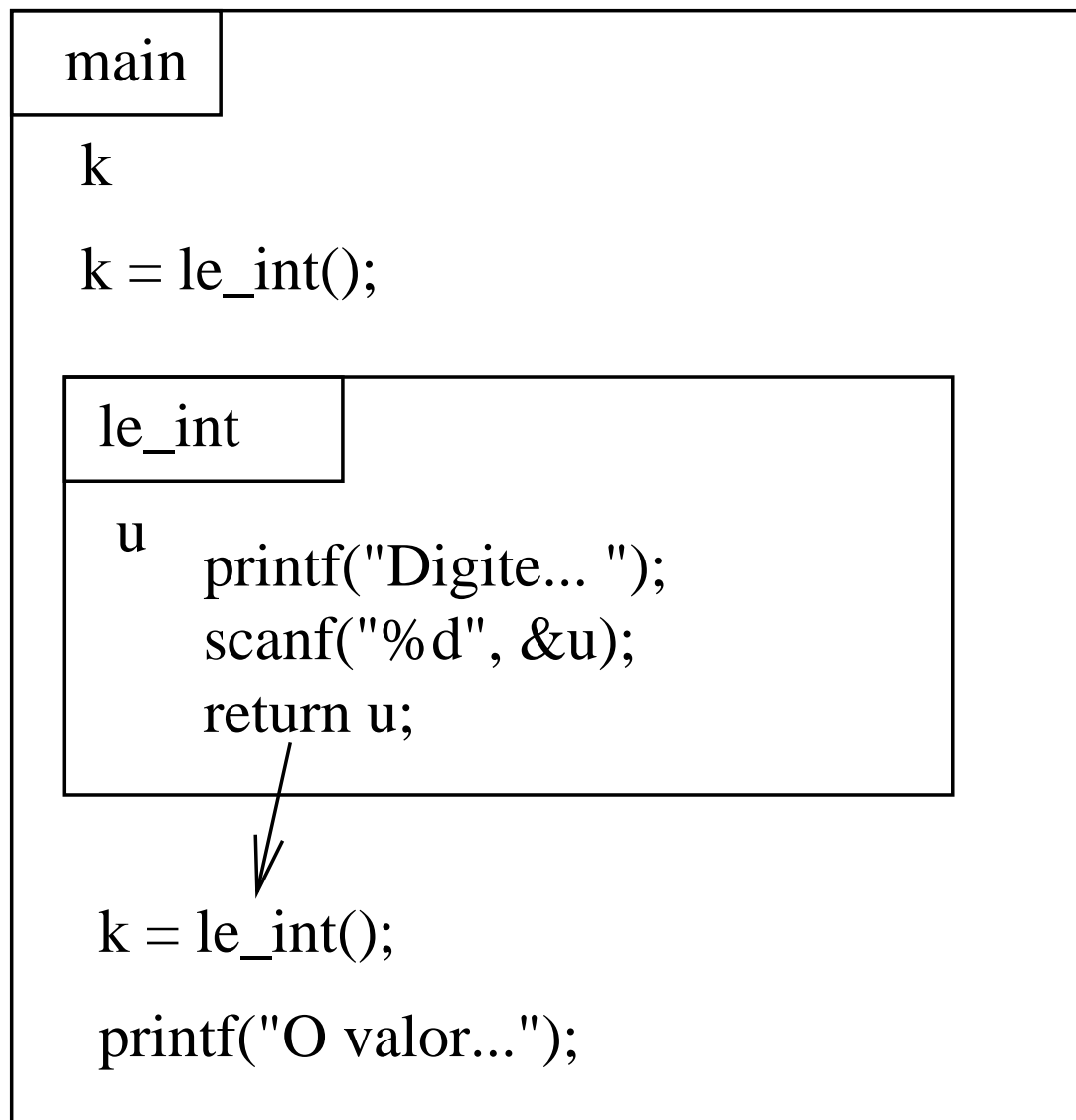
int main()
{
    int k;

    k = le_int();
    printf("Valor lido: %d\n", k);
}

int le_int()
{
    int u;

    printf("Digite um inteiro ==> ");
    scanf("%d", &u);
    return u;
}
```

# Execução da função



# Função com parâmetros

- Vamos modificar a nossa função de modo que ela possa ser usada para ler um inteiro, mas que a mensagem possa mudar
- Para isso, vamos passar a mensagem como um parâmetro
- Assim poderemos usar:

```
d = le_int( "Digite o dia" );  
m = le_int( "Digite o mes" );  
a = le_int( "Digite o ano" );
```

# Protótipo com parâmetro

```
#include <stdio.h>

int le_int(char[]); // protótipo

// Declaração da função principal
int main()
{
    ...
}

// Declaração da função le_int
int le_int(char msg[])
{
    ...
}
```

# Implementação com parâmetro

```
int le_int(char msg[])  
{  
    int u;  
  
    printf(msg);  
    scanf("%d", &u);  
    return u;  
}
```

# Implementação com parâmetro

```
#include <stdio.h>

int le_int(char[]);

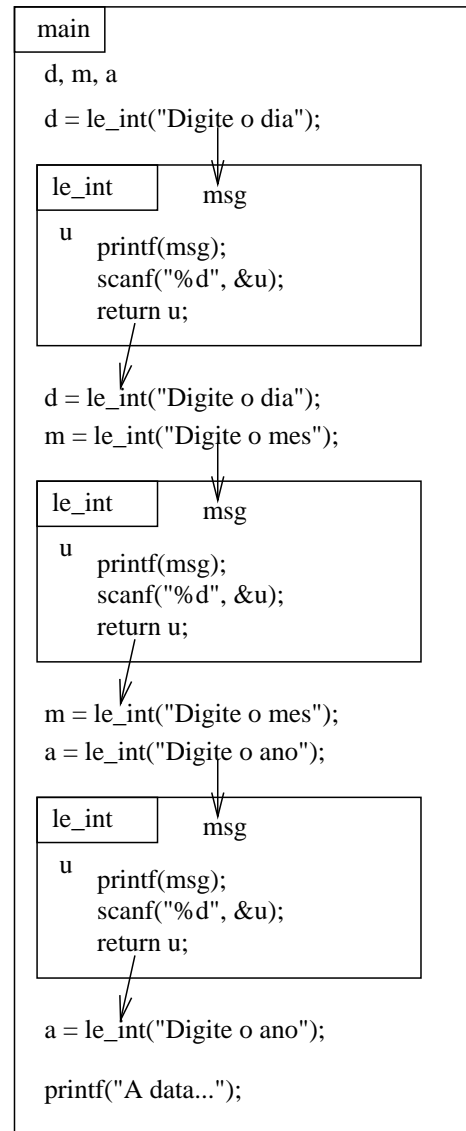
int main()
{
    int d, m, a;

    d = le_int("Digite o dia: ");
    m = le_int("Digite o mes: ");
    a = le_int("Digite o ano: ");
    printf("Data: %d/%d/%d\n", d, m, a);
}

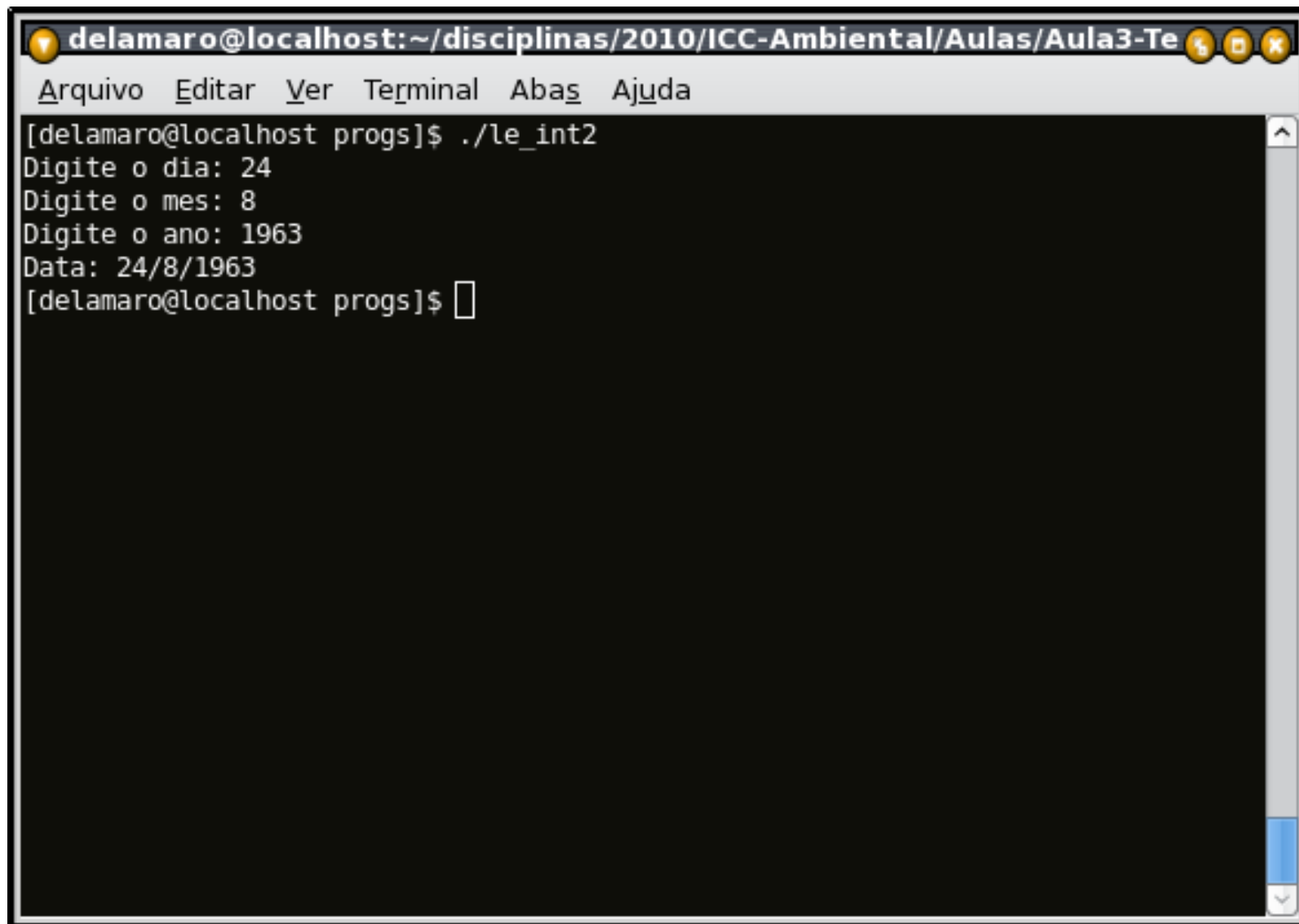
int le_int(char msg[])
{
    int u;

    printf(msg);
    scanf("%d", &u);
    return u;
}
```

# Execução com parâmetro



# Execução com parâmetro



A terminal window titled "delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te" with standard window controls. The menu bar includes "Arquivo", "Editar", "Ver", "Terminal", "Abas", and "Ajuda". The terminal content shows the execution of a program named "le\_int2" from the "progs" directory. The program prompts for day, month, and year, and outputs the date "24/8/1963".

```
delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
[delamaro@localhost progs]$ ./le_int2
Digite o dia: 24
Digite o mes: 8
Digite o ano: 1963
Data: 24/8/1963
[delamaro@localhost progs]$
```



# Aproveitando – string

- Um string é uma sequência de caracteres legíveis
- O primeiro parâmetro do `scanf` e do `printf` são sempre um string
- Podemos declarar variáveis e guardar strings nelas
- Para isso usá-se o tipo `char [ ]`
- Por exemplo `char nome[50];`
- O número representa o tamanho máximo do string que se pode colocar na variável

# Entrada e saída

- Para ler e escrever um string usa-se o formato "`\%s`"
- `char nome[50];`  
`scanf("%s", nome); // sem o & !!!!`
- `printf("O nome lido foi: %s", nome);`

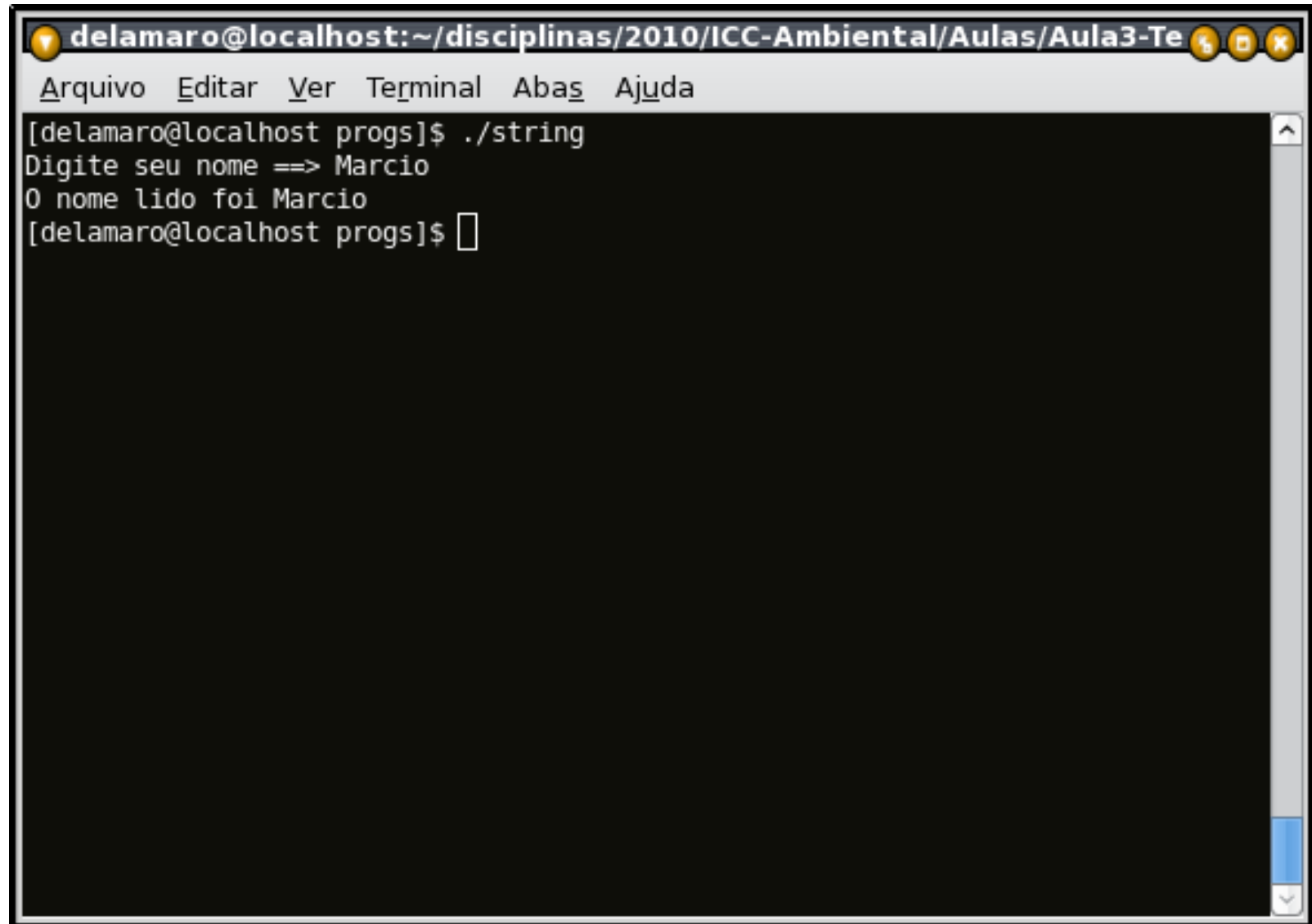
# String – exemplo

```
#include <stdio.h>

int main()
{
char nome[50];

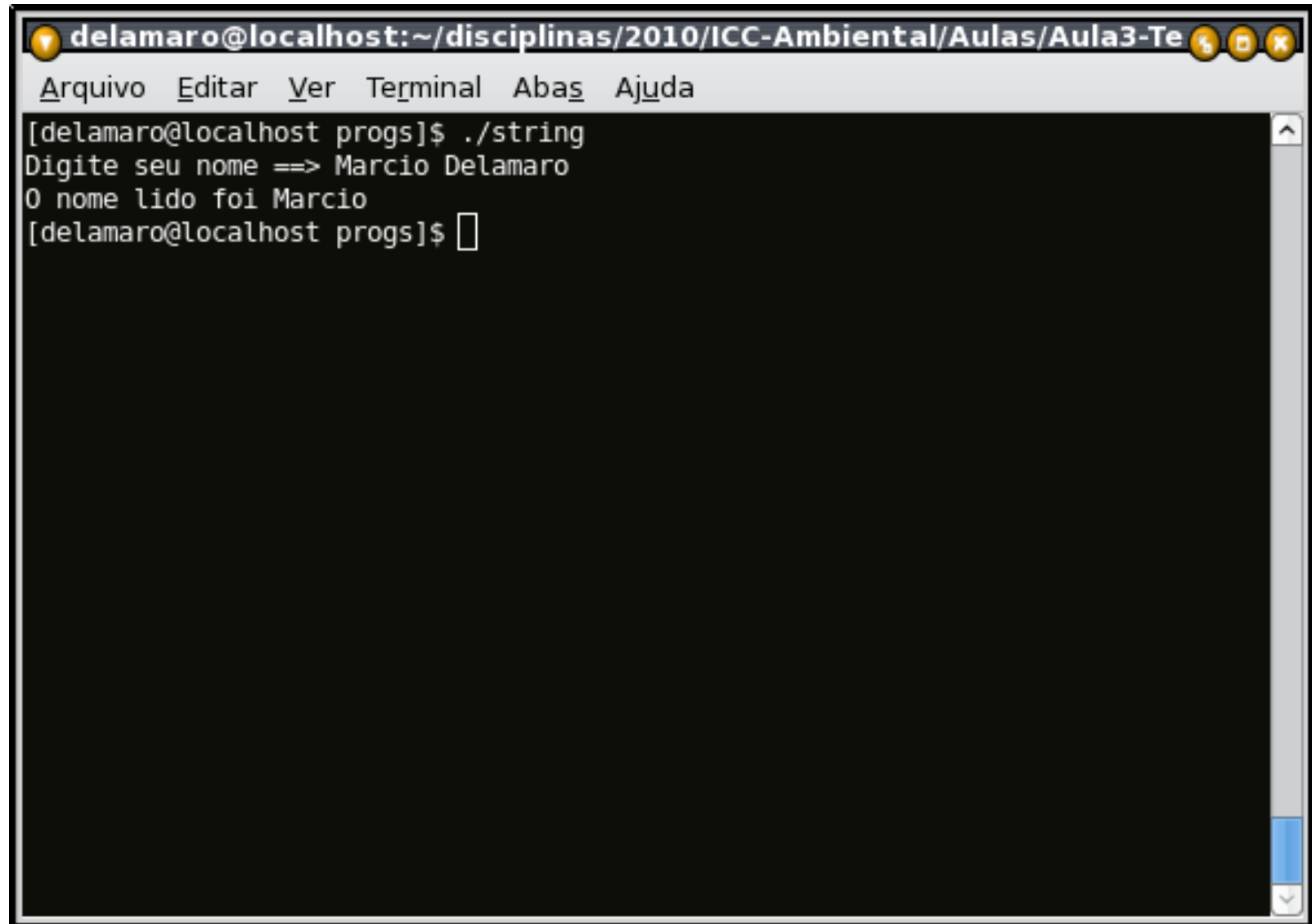
    printf("Digite seu nome ==> ");
    scanf("%s", nome);
    printf("O nome lido foi %s\n", nome);
}
```

# String – exemplo

A terminal window with a title bar that reads "delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te". The window has a menu bar with "Arquivo", "Editar", "Ver", "Terminal", "Abas", and "Ajuda". The terminal content shows a shell prompt "[delamaro@localhost progs]\$ ./string", followed by the program's output: "Digite seu nome ==> Marcio", "0 nome lido foi Marcio", and a final shell prompt "[delamaro@localhost progs]\$".

```
delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te
Arquivo  Editar  Ver  Terminal  Abas  Ajuda
[delamaro@localhost progs]$ ./string
Digite seu nome ==> Marcio
0 nome lido foi Marcio
[delamaro@localhost progs]$
```

# String – exemplo



A terminal window titled "delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te" with standard window controls. The menu bar includes "Arquivo", "Editar", "Ver", "Terminal", "Abas", and "Ajuda". The terminal output shows the execution of a program named "string" which prompts for a name and outputs the name read.

```
delamaro@localhost:~/disciplinas/2010/ICC-Ambiental/Aulas/Aula3-Te  
Arquivo  Editar  Ver  Terminal  Abas  Ajuda  
[delamaro@localhost progs]$ ./string  
Digite seu nome ==> Marcio Delamaro  
O nome lido foi Marcio  
[delamaro@localhost progs]$
```

# Alternativa

- A função **gets** lê um string até o fim da linha (até um <enter>)
- O parâmetro é o nome da variável aonde o string deve ser colocado
- `#include <stdio.h>`

```
int main()  
{  
    char nome[50];  
  
    printf("Digite seu nome ==> ");  
    gets(nome);  
    printf("O nome lido foi %s\n", nome);  
}
```

# Exercício

- Escreva um programa que leia 3 ângulos em radianos, no 1o quadrante, e mostre o valor do seno, cosseno e tangente de cada um deles
- Use a sua própria versão de cada uma dessas funções
- O seno deve ser calculado usando a série de Taylor, com 5 termos
- O cosseno deve ser computado usando a função do seno
- A tangente deve ser computada usando o seno e o cosseno

# Programa principal

```
int main()
{
double x;

    printf("Entre com o 1o angulo: ", &x);
    scanf("%lf", &x);
    printf("Tg: %lf -- seno: %lf -- cosseno: %lf\n",
           tangente(x), seno(x), cosseno(x));

    printf("Entre com o 2o angulo: ", &x);
    scanf("%lf", &x);
    printf("Tg: %lf -- seno: %lf -- cosseno: %lf\n",
           tangente(x), seno(x), cosseno(x));

    printf("Entre com o 3o angulo: ", &x);
    scanf("%lf", &x);
    printf("Tg: %lf -- seno: %lf -- cosseno: %lf\n",
           tangente(x), seno(x), cosseno(x));
}
```



# Função seno

```
double seno(double); // protótipo
double seno(double x)
{
double sen, denominador, numerador;
    // primeiro termo
    sen = x;
    // segundo termo
    numerador = x * x * x;
    denominador = 6;
    sen = sen - (numerador / denominador);
    // terceiro termo
    numerador = numerador * x * x;
    denominador = denominador * 4 * 5;
    sen = sen + (numerador / denominador);
    // quarto termo
    numerador = numerador * x * x;
    denominador = denominador * 6 * 7;
    sen = sen - (numerador / denominador);
    // quinto termo
    numerador = numerador * x * x;
    denominador = denominador * 8 * 9;
    sen = sen + (numerador / denominador);
    return sen;
}
```

# Função cosseno, tangente

```
double cosseno(double);  
double tangente(double);  
  
double cosseno(double x)  
{  
    double sen2;  
  
    sen2 = seno(x) * seno(x);  
    return sqrt(1.0 - sen2);  
}  
  
double tangente(double x)  
{  
    return seno(x) / cosseno(x);  
}
```

# Variáveis locais

- As variáveis que utilizamos são variáveis **locais**
- Elas só existem dentro de uma função
- Não podem ser acessadas em outra função
- Podem existir duas variáveis com o mesmo nome em funções diferentes

# Variáveis globais

- Às vezes é necessário definir uma variável que possa ser compartilhada entre todas as funções
- Elas existem dentro de todas as funções
- Não é bom ter variáveis locais com o mesmo nome de uma variável global
- Ao modificar uma variável global, estamos modificando sempre a mesma posição da memória

# Variável global – exemplo

```
#include <stdio.h>

int global = 3;

int f();

int main(void)
{
    printf("%d\n", global); // Imprime 3
    f(); // chama função f
    printf("%d\n", global); // imprime 5
    return 0;
}

int f()
{
    global = 5;
    return 0;
}
```

# Inicialização de variável

- Uma variável global é sempre inicializada (0 para números)
- Uma variável local nunca é inicializada (valor desconhecido)
- É sempre possível inicializar a variável explicitamente na declaração
  - `int a = 10, b = 0;`

# Variável global – Exercício

- Modifique o programa trigonométrico para que no final da sua execução seja apresentada uma mensagem dizendo quantas vezes cada uma das funções seno, cosseno e tangente foram chamadas.

# Variável global – solução

```
// contadores
int cont_seno = 0, cont_cos = 0, cont_tan = 0;

int main()
{
    ...
    printf("Seno foi chamada %d vezes\n", cont_seno);
    printf("Cosseno foi chamada %d vezes\n", cont_cos);
    printf("Tangente foi chamada %d vezes\n", cont_tan);
}

double cosseno(double x)
{
    double sen2;
    cont_cos = cont_cos + 1;
    sen2 = seno(x) * seno(x);
    return sqrt(1.0 - sen2);
}

double tangente(double x)
{
    cont_tan = cont_tan + 1;
    return seno(x) / cosseno(x);
}

double seno(double x)
{
    double sen, denominador, numerador;
    cont_seno = cont_seno + 1;
    ...
}
```