

Estruturas de dados

Ponteiros e passagem de parâmetros por referência

Passagem de parâmetros por referência

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

```
void troca(int x,int y);
```

```
int main()
```

```
{
```

```
int a,b;
```

```
a=3;
```

```
b=5;
```

```
troca(a,b);
```

```
printf("%d %d\n",a,b);
```

```
return 0;
```

```
}
```

```
void troca(int x,int y)
```

```
{
```

```
int aux;
```

```
aux=x;
```

```
x=y;
```

```
y=aux;
```

```
}
```



Passagem de parâmetros por referência

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

```
void troca(int x,int y);
```

```
int main()
```

```
{
```

```
int a,b;
```

```
a=3;
```

```
b=5;
```

```
troca(a,b);
```

```
printf("%d %d\n",a,b);
```

```
return 0;
```

```
}
```

```
void troca(int a,int b)
```

```
{
```

```
int aux;
```

```
aux=a;
```

```
a=b;
```

```
b=aux;
```

```
}
```



Passagem de parâmetros por referência

Como permitir que a função troca **altere** uma variável que foi passada como parâmetro?

PROBLEMA:

Escreva uma função chamada **calcEqSegundoGrau** que receba como entrada os coeficientes (A,B e C) de uma equação do 2º grau e retorne o valor das raízes. Retornar também um inteiro que indique se existe ou não raízes no conjunto dos reais.

Entrada: Coeficientes A,B e C

Retorno: As duas raízes da equação (se pertencerem a **R**).

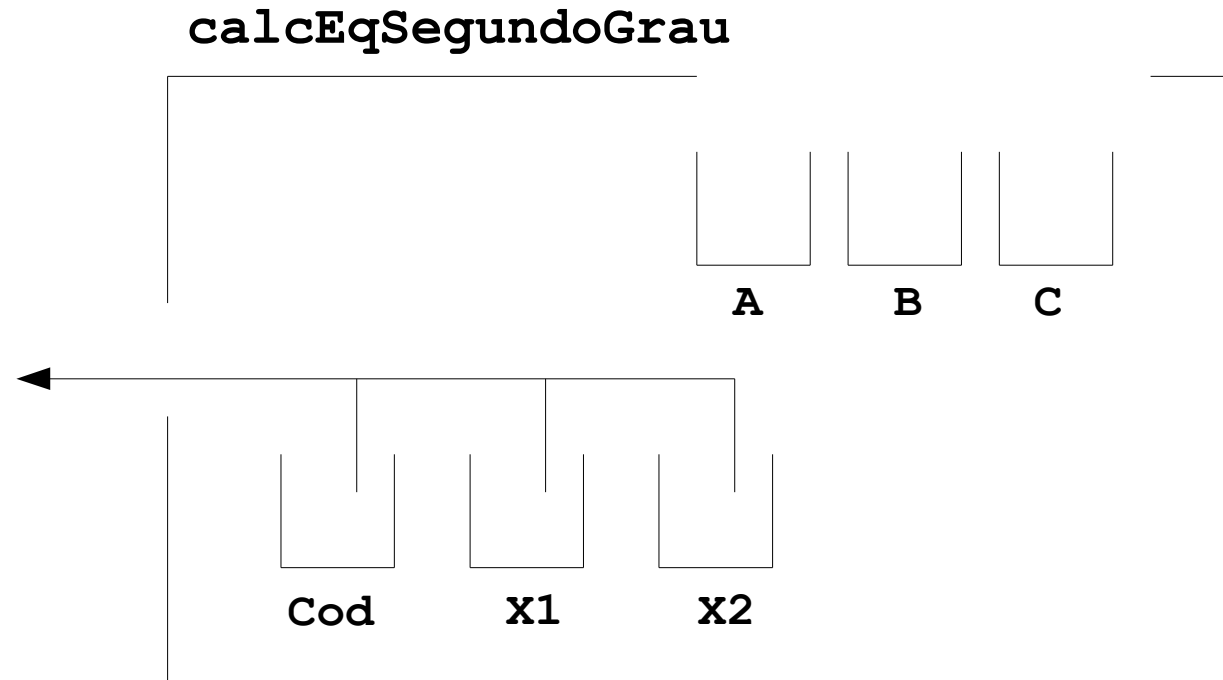
O código **0** se foi possível calcular as raízes (em **R**) e **1** caso contrário.

$$Ax^2 + Bx + C = 0$$

Passagem de parâmetros por referência

Como retornar 3 valores ???

Dessa forma **não** é possível...



O comando **return** só permite retornar **1** valor.

Ponteiros

Um ponteiro é a representação de um **endereço**.

Ponteiro constante: é um endereço.

Ponteiro variável: é uma variável que armazena endereços.

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

```
int main()
```

```
{
```

```
int n;
```

```
n=3;
```

```
...
```

```
}
```

Endereço
da variável n

1000

1001

1002

1003

1004

1005

1006

Memória

3

n

Conteúdo da
variável n

Ponteiros

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

```
int main()
```

```
{  
int n;
```

Formato para inteiro
sem sinal

```
n=3;
```

```
printf("%u\n", &n);
```

```
return 0;
```

```
}
```

Endereço
da variável n

1000

1001

1002

1003

1004

1005

1006

1000

Memória

Conteúdo da
variável n

3

n

operador **&** : obtém o **endereço** de memória
do seu operando.

Ponteiros

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

```
int main()  
{  
  int n, *p;
```

Declaração de um
ponteiro para
inteiro

```
  n = 3;  
  p = &n;  
  printf("%u\n", p);  
  return 0;  
}
```

variável que armazena
um endereço.

Memória

1000

1001

1002

1003

3

n

...

1100

1101

1102

1103

1000

p

Ponteiros

O operador **&** não pode ser aplicado a constantes e expressões

Ex: ~~&3~~

~~&(x+1)~~

Declaração de um ponteiro

int *p;



Indica que p é
um ponteiro

Indica o tipo da variável cujo
endereço será armazenado no
ponteiro

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

```
int main()
{
    int *px;
    float x;
```

```
    x = 2.5;
    px = &x;
    . . . . .
```

```
}
```

**Incorreto!!
Por que??**

Ponteiros

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

```
int main()
```

```
{  
  float x, *px;
```

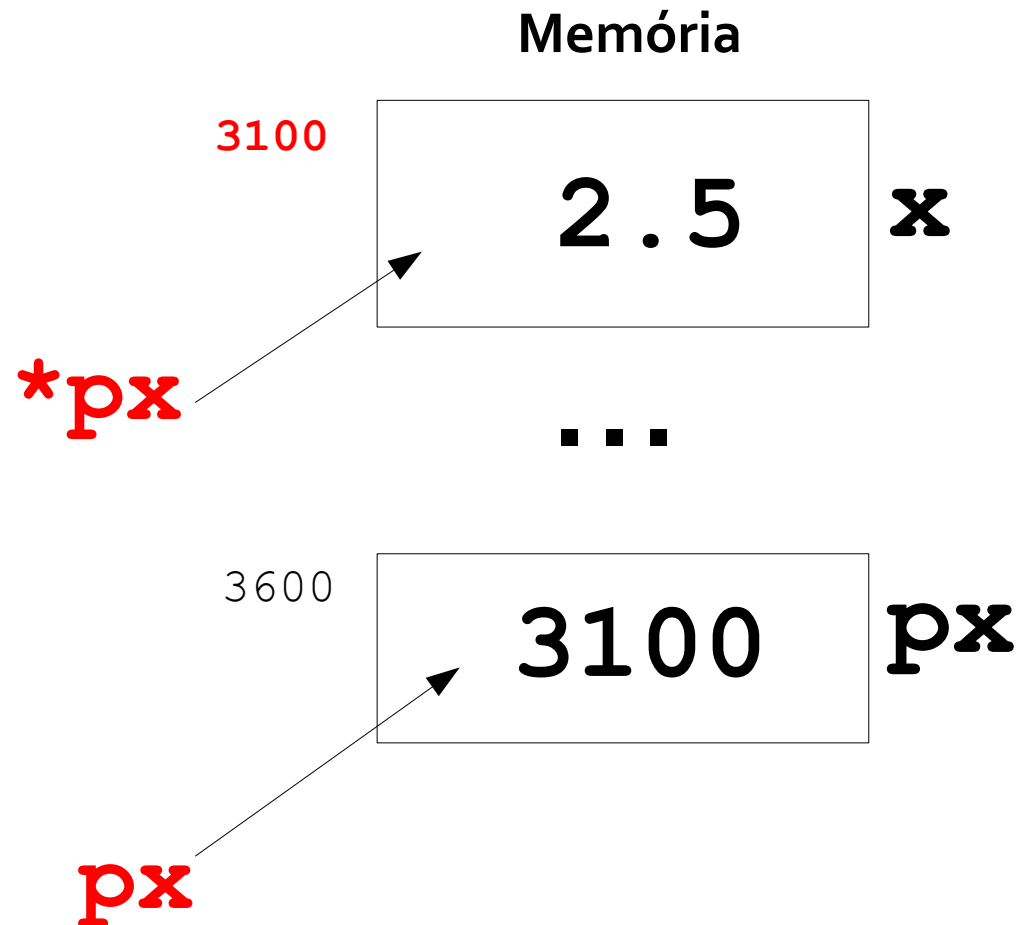
```
  x = 2.5;
```

```
  px = &x;
```

```
  printf("%f\n", *px);
```

```
  return 0;
```

```
}
```



O operador ***** obtém o **valor** da variável localizada no endereço armazenado no ponteiro.

Ponteiros

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

```
int main()
```

```
{  
float x, *px, a;
```

```
x = 2.5;
```

```
px = &x;
```

```
a = *px;
```

```
printf("%f\n", a);
```

```
return 0;
```

```
}
```

Memória

3100

2.5

x

***px**

...

3600

3100

px

px

...

3700

2.5

a

Copia **x** para **a** de forma **indireta**.

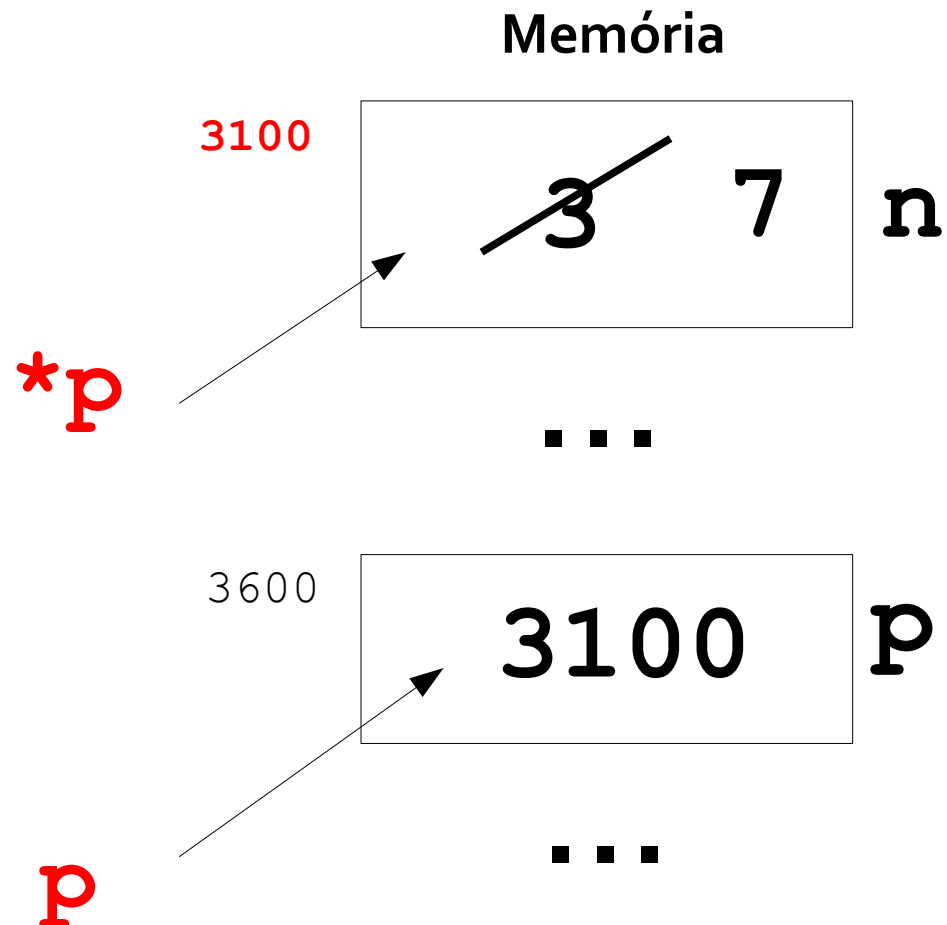
Ponteiros

É possível **alterar** o valor de uma variável de forma indireta.
Basta que seu **endereço** esteja armazenado em um **ponteiro**.

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

```
int main()  
{  
  int n, *p;
```

```
  n = 3;  
  p = &n;  
  *p = 7; Altera n  
  printf("%d\n", n);  
  return 0;  
}
```



Passagem de parâmetros por referência

PROBLEMA:

Escreva uma função chamada **somaDez** que receba como entrada um inteiro e acrescente 10 na variável passada como argumento.

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

```
void somaDez(int x) ;
```

```
int main()
```

```
{  
int n;
```

```
n = 2;
```

```
somaDez(n) ;
```

```
printf("%d\n",n) ;
```

```
return 0;
```

```
}
```

Não funciona!

Continuação...

Por quê???

```
void somaDez(int x)
```

```
{  
x = x + 10;  
}
```

Passagem de parâmetros por referência

Como permitir que a função `somaDez` **altere** uma variável `n` que foi passada como parâmetro? Passando para a função o **endereço** de `n`.

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

```
void somaDez(int *x) ;
```

```
int main()
```

```
{  
  int n;
```

```
  n = 2;
```

```
  somaDez(&n) ;
```

```
  printf("%d\n", n) ;
```

```
  return 0;
```

```
}
```

Continuação...

```
void somaDez(int *x)
```

```
{
```

```
  *x = *x + 10;
```

```
}
```

Passagem de parâmetros por referência

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

```
void troca(_____);
```

```
int main()
```

```
{
```

```
int a,b;
```

```
a=3;
```

```
b=5;
```

```
troca(_____);
```

```
printf("%d %d\n",a,b);
```

```
return 0;
```

```
}
```

```
void troca(_____)
```

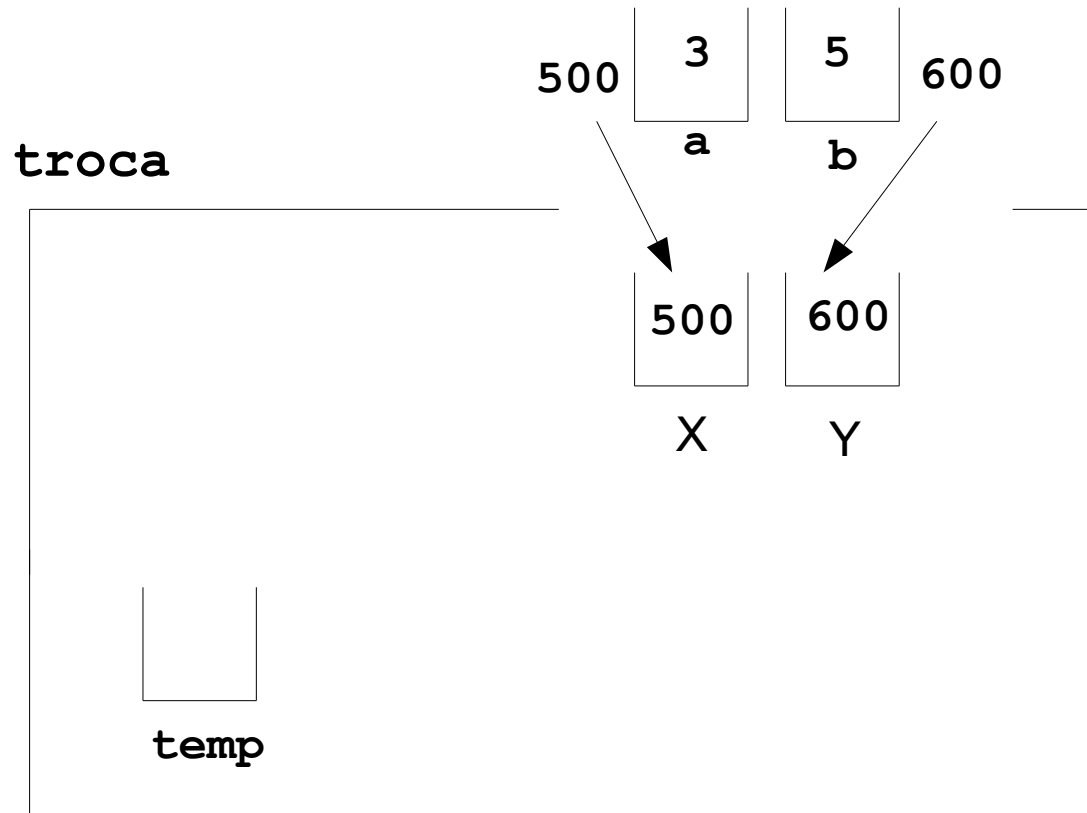
```
{
```

```
}
```

Como implementar
a função troca?

Passagem de parâmetros por referência

Passar os **endereços** de **a** e **b** para a função



Os ponteiros **X** e **Y** permitem a alteração de **a** e **b**.

Passagem de parâmetros por referência

Como permitir que a função troca **altere** uma variável que foi passada como parâmetro?

PROBLEMA:

Escreva uma função chamada **calcEqSegundoGrau** que receba como entrada os coeficientes (A,B e C) de uma equação do 2º grau e retorne o valor das raízes. Retornar também um inteiro que indique se existe ou não raízes no conjunto dos reais.

Entrada: Coeficientes A,B e C

Saída: As duas raízes da equação (se pertencerem a **R**).

Retorno: O código 0 se foi possível calcular as raízes (em **R**) e 1 caso contrário.

$$Ax^2 + Bx + C = 0$$

Passagem de parâmetros por referência

```
#include <stdio.h>

int calcEqSegundoGrau(float a,float b,float c, float *x1,float *x2);

int main()
{
float r1,r2,a,b,c;

printf("Informe os coeficientes da equação:");
scanf("%f %f %f",&a,&b,&c);
if (calcEqSegundoGrau(a,b,c,&r1,&r2)==1)
    printf("Não possui raízes reais\n");
else {
    printf("R1: %f \n",r1);
    printf("R2: %f \n",r2);
}
return 0;
}

int calcEqSegundoGrau(float a,float b,float c, float *x1,float *x2)
{
....
}
```

Passagem de parâmetros

Passagem de parâmetros **por valor**:

Uma cópia da variável utilizada como argumento é passada para a função.

Não permite que a função altere o valor do parâmetro real.

Passagem de parâmetros **por referência**:

O endereço da variável utilizada como argumento é passada para a função.

Permite que a função altere o valor do parâmetro real.

Passagem de parâmetros

Podemos classificar os parâmetros em 3 categorias.

De entrada: Parâmetros que têm seus valores estabelecidos fora da função e não são modificados dentro dela. Normalmente utilizamos passagem de parâmetros **por valor**.

De saída: Parâmetros que têm seus valores estabelecidos dentro da função. Utilizamos passagem de parâmetros **por referência**.

De entrada/saída: Parâmetros que têm seus valores estabelecidos fora da função mas que são alterados dentro dela. Utilizamos passagem de parâmetros **por referência**.

Três versões para a função soma

Versão 1

Dois parâmetros passados **por valor**.
Resultado devolvido com **return**.

Nome: Soma

Entrada: 2 inteiros

Saída: Nenhuma

Retorno: valor da soma

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

```
int soma(int a,int b);
```

```
int main()  
{  
    int x,y,s;
```

```
    x=10;
```

```
    y=20;
```

```
    s=soma(x,y);
```

```
    printf("%d\n",s);
```

```
    return 0;
```

```
}
```

E



E



```
int soma(int a, int b)
```

```
{
```

```
    return a+b;
```

```
}
```

Três versões para a função soma

Versão 2

Dois parâmetros passados **por valor**.
Resultado obtido com parâmetro
passado por referência

Nome: Soma

Entrada: 2 inteiros

Saída: valor da soma

Retorno: nenhum

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

```
void soma(int a,int b,int *r);
```

```
int main()
```

```
{
```

```
int x,y,s;
```

```
x=10;
```

```
y=20;
```

```
soma(x,y,&s);
```

```
printf("%d\n",s);
```

```
return 0;
```

```
}
```

E



E



S



```
void soma(int a, int b, int *r)
```

```
{
```

```
*r = a + b;
```

```
}
```

Três versões para a função soma

Versão 3

Um parâmetro passado **por valor**.
Resultado obtido com parâmetro
passado por referência (modifica o
segundo operando).

Nome: Soma

Entrada: 1 inteiro

Entrada/Saída: valor da soma

Retorno: nenhum

```
#include <stdio.h>

void soma(int a,int *b);

int main()
{
    int x,y;

    x=10;
    y=20;
    soma(x,&y);
    printf("%d\n",y);
    return 0;
}
```

E E/S
↓ ↓ ↑
void soma(int a, int *b)
{
 *b = a + *b;
}