Aula: Ponteiros e Funções

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2020/1
Departamento de Computação – UFOP

Aula: Ponteiros e Funções

Aula: Ponteiros e Funções

Ponteiros

Um ponteiro (apontador ou *pointer*) é um tipo especial de variável que armazena um **endereço de memória**

- Ponteiros são declarados utilizando o caractere especial `*`:

```
1  int *pi;     // pi é um ponteiro do tipo int
2  char *pc;    // pc é um ponteiro do tipo char
3  float *pf;   // pf é um ponteiro do tipo float
4  double *pd; // pd é um ponteiro do tipo double
```

- Vários podem ser declarados em uma única linha:

```
1  int *p1, *p2, *p3;
```

Ponteiros

O **conteúdo** da memória apontada por um ponteiro se refere ao valor armazenado no endereço de memória para o qual o ponteiro aponta.

- Este conteúdo (valor) pode ser alterado usando o operador *
- Exemplo:

```
1  int main()
2  {
3      int x = 10, y = 0;
4      int *px = &x;
5   → y = *px; // y recebe o conteúdo do endereço apontado por px
6      printf("y = %d\n", y);
7      return 0;
8  }
```

- O que será impresso?

```
1  y = 10
```

Ponteiros

Exemplo:

```c
int main()
{
    int x = 0;
    int *px;
    px = &x;
    *px = 99;
    printf("x = %d\n", x);
    return 0;
}
```

O que será impresso?

```
x = 99
```

Ponteiros

Exemplo:

```c
int main()
{
    int x = 100;
    int *px = &x;
    printf("valor de x    = %d\n", x);
    printf("endereço de x = %p\n", &x); // %p: formato para ponteiro
    printf("endereço de x = %p\n", px); // %p: formato para ponteiro
    printf("valor de x    = %d\n", *px);
    return 0;
}
```

Exemplo de saída (computador com 64 bits):

```
valor de x    = 100
endereço de x = 0x7ffedfc1e378
endereço de x = 0x7ffedfc1e378
valor de x    = 100
```

Tipos de ponteiros

Há vários tipos de ponteiros:

- Ponteiros para caracteres
- Ponteiros para inteiros
- Ponteiros para vetores
- Ponteiros para ponteiros para inteiros
- etc...

Você especifica o tipo de ponteiro!

Aula: Ponteiros e Funções

Endereços

- Quais são as características da variável `x`, declarada a seguir?

```
1  int x = 9;
```

  - Tipo: `int`
  - Nome: `x`
  - Endereço de memória ou referência: 0xbfd267c4
  - Valor: 9

- Para acessar o endereço de uma variável, utilizamos o operador `&`

```
1  int x = 9;
2  printf("O endereço de memória de x é %p\n", &x);
```

Memória

Endereço	Valor
00010000	??
00010001	??
00010002	??
00010003	??
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

- A memória é formada por várias células.
- Cada célula contém um endereço e um valor (veja exemplo ao lado).
- O tamanho do endereço e do valor dependem da arquitetura (32/64 bits).

Memória

Endereço	Valor	
00010000	??	i
00010001	??	
00010002	??	
00010003	??	
00010004	??	
00010005	??	
00010006	??	
00010007	??	
00010008	??	
00010009	??	
0001000A	??	
0001000B	??	
0001000C	??	
0001000D	??	

Exemplo:

- O caractere `i` ocupa 1 byte na memória

```
1  int main()
2  {
3      char i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- Inteiro `i` ocupa 4 bytes na memória

```
1  int main()
2  {
3      int i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- Ponto flutuante `i` ocupa 4 bytes na memória

```
1  int main()
2  {
3      float i;
4      return 0;
5  }
```

Memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	
00010005	
00010006	
00010007	
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

i

Exemplo:

- Double `i` ocupa 8 bytes na memória

```
1  int main()
2  {
3      double i;
4      return 0;
5  }
```

Memória

Endereço	Valor	
00010000	??	c
00010001		
00010002		
00010003		
00010004	??	i
00010005		
00010006		
00010007		
00010008	??	f
00010009		
0001000A		
0001000B		
0001000C	??	d
0001000D		
0001000E		
0001000F		

Exemplo:

- Note os quatro ponteiros...

```
1  int main()
2  {
3      char *c;
4      int *i;
5      float *f;
6      double *d;
7      return 0;
8  }
```

- Todos requerem o mesmo tamanho (32/64 bits).
- Lembre-se: um ponteiro armazena um **endereço de memória**, independente do tipo.

Aula: Ponteiros e Funções

Ponteiros

Breve revisão:

Um ponteiro (apontador ou *pointer*) é um tipo especial de variável que armazena um **endereço de memória**

- Ponteiros são declarados utilizando o caractere especial `*`:

```
1  int *pi;     // pi é um ponteiro do tipo int
2  char *pc;    // pc é um ponteiro do tipo char
3  float *pf;   // pf é um ponteiro do tipo float
4  double *pd; // pd é um ponteiro do tipo double
```

- Vários podem ser declarados em uma única linha:

```
1  int *p1, *p2, *p3;
```

Ponteiros

O **conteúdo** da memória apontada por um ponteiro se refere ao valor armazenado no endereço de memória para o qual o ponteiro aponta.

- Este conteúdo (valor) pode ser alterado usando o operador *:

```
1  int main()
2  {
3      int x = 10, y = 100;
4      int *px = &x;
5   →  *px = *px + 1; // conteúdo de px recebe o conteúdo de px mais 1
6      printf("x = %d", x);
7      printf("y = %d", y);
8      return 0;
9  }
```

- O que será impresso?

```
1  x = 11
2  y = 100
```

Exemplo: ponteiros e memória

Endereço	Valor
00010000	??
00010001	??
00010002	??
00010003	??
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??

Exemplo de uso:

```
1  int main()
2  {
3   → int i;
4      i = 15;
5      char c = 's';
6      int *p = &i;
7      *p = 25;
8      return 0;
9  }
```

- A memória para o inteiro `i` será alocada.

Exemplo: ponteiros e memória

Endereço	Valor
00010000	??
00010001	
00010002	
00010003	
00010004	??
00010005	??
00010006	??
00010007	??
00010008	??
00010009	??
0001000A	??
0001000B	??
0001000C	??
0001000D	??
0001000C	??
0001000D	??

i

Exemplo de uso:

```
1  int main()
2  {
3      int i;
4   →  i = 15;
5      char c = 's';
6      int *p = &i;
7      *p = 25;
8      return 0;
9  }
```

- A memória para o inteiro `i` foi alocada.
- O conteúdo de `i` será alterado para 15 (representação será em decimal, mas na prática é tudo em binário).

Exemplo: ponteiros e memória

Endereço	Valor	
00010000	15	i
00010001		
00010002		
00010003		
00010004	??	
00010005	??	
00010006	??	
00010007	??	
00010008	??	
00010009	??	
0001000A	??	
0001000B	??	
0001000C	??	
0001000D	??	
0001000C	??	
0001000D	??	

Exemplo de uso:

```
1  int main()
2  {
3      int i;
4      i = 15;
5   →  char c = 's';
6      int *p = &i;
7      *p = 25;
8      return 0;
9  }
```

- O conteúdo de `i` foi alterado para 15 (representação é em decimal, mas na prática é tudo em binário).
- A memória para o `char c` será alocada e inicializada com `'s'`.

Exemplo: ponteiros e memória

Endereço	Valor	
00010000	15	i
00010001		
00010002		
00010003		
00010004	s	c
00010005	??	
00010006	??	
00010007	??	
00010008	??	
00010009	??	
0001000A	??	
0001000B	??	
0001000C	??	
0001000D	??	
0001000C	??	
0001000D	??	

Exemplo de uso:

```
1  int main()
2  {
3      int i;
4      i = 15;
5      char c = 's';
6   →  int *p = &i;
7      *p = 25;
8      return 0;
9  }
```

- A memória para o `char c` foi alocada e inicializada com `'s'`.
- O ponteiro de inteiro `p` será declarado e inicializado com o endereço de memória de `i`.

Exemplo: ponteiros e memória

Endereço	Valor	
00010000	15	i
00010001		
00010002		
00010003		
00010004	s	c
00010005	00	
00010006	01	p
00010007	00	
00010008	00	
00010009	??	
0001000A	??	
0001000B	??	
0001000C	??	
0001000D	??	
0001000C	??	
0001000D	??	

Exemplo de uso:

```
1  int main()
2  {
3      int i;
4      i = 15;
5      char c = 's';
6      int *p = &i;
7   →  *p = 25;
8      return 0;
9  }
```

- O ponteiro de inteiro `p` foi declarado e inicializado com o endereço de memória de `i`.
- O conteúdo da memória apontada por `p` será atualizado para 25.

Exemplo: ponteiros e memória

Endereço	Valor	
00010000	25	i
00010001		
00010002		
00010003		
00010004	s	c
00010005	00	
00010006	01	p
00010007	00	
00010008	00	
00010009	??	
0001000A	??	
0001000B	??	
0001000C	??	
0001000D	??	
0001000C	??	
0001000D	??	

Exemplo de uso:

```
1  int main()
2  {
3      int i;
4      i = 15;
5      char c = 's';
6      int *p = &i;
7      *p = 25;
8   → return 0;
9  }
```

- O conteúdo da memória apontada por `p` foi atualizado para 25.
- Agora o método `main` será finalizado...

Exemplos: uso de ponteiros

Exemplo:

```c
int main()
{
    int x = 0;
    int *px;
    px = &x;
    *px = x - 5;
    printf("x = %d\n", x);
    return 0;
}
```

O que será impresso?

```
x = -5
```

Exemplos: uso de ponteiros

Exemplo:

```c
int main()
{
    int x = 0;
    int *px = &x;
    int *py;
    py = &(*px);
    *py = 10;
    printf("x = %d\n", x);
    return 0;
}
```

O que será impresso?

```
x = 10
```

Exemplos: uso de ponteiros

Exemplo:

```c
int main()
{
    int x = 1000;
    int *px = &x;
    int y = *&*px; // ou *(&(*px))
    printf("y = %d\n", y);
    return 0;
}
```

O que será impresso?

```
y = 1000
```

Exemplos: uso de ponteiros

Exemplo:

```
1  int main()
2  {
3      int x = 1;
4      int *px = &x;
5      *px = *px * 10;
6      printf("  x = %d\n", x)
7      printf(" &x = %p\n", &x)
8      printf(" px = %p\n", px);
9      printf("*px = %d\n", *px);
10     return 0;
11 }
```

Exemplo de saída (computador com 64 bits):

```
1    x = 10
2   &x = 0x7ffedfc1e378
3   px = 0x7ffedfc1e378
4  *px = 10
```

Aula: Ponteiros e Funções

Passagem por valor

Qual o problema da função a seguir?

```
1  void naoTroca(int a, int b)
2  {
3      int aux = a;
4      a = b;
5      b = aux;
6  }
```

- Os parâmetros são passados por valor!
- Assim, valores são passados para `a` e `b`.
- Logo: a função não efetua a troca de fato (vide aula anterior)!

Passagem de ponteiros

Ao contrário de C++, C não implementa passagem por referência...

- A solução é utilizar **ponteiros** para simular a passagem por referência.

```
1  void troca(int *a, int *b)
2  {
3      int aux = *a;
4      *a = *b;
5      *b = aux;
6  }
```

- A função recebe ponteiros para duas variáveis.
- Em seguida, troca o conteúdo das memórias apontadas.

Como usar essas funções?

Eis um exemplo de uso das funções apresentadas:

```c
int main()
{
    int a = 1;
    int b = 2;
    naoTroca(a, b); // valores a e b são passados (e não há troca)
    printf("a = %d, b = %d", a, b); // a = 1, b = 2
}
```

```c
int main()
{
    int a = 1;
    int b = 2;
    troca(&a, &b); // endereço de memória de a e b são passados
    printf("a = %d, b = %d", a, b); // a = 2, b = 1
}
```

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome
0x1000		
0x1004		
0x1008		
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome
0x1000		
0x1004		
0x1008		
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome
0x1000		
0x1004	100	x
0x1008	200	y
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		

main

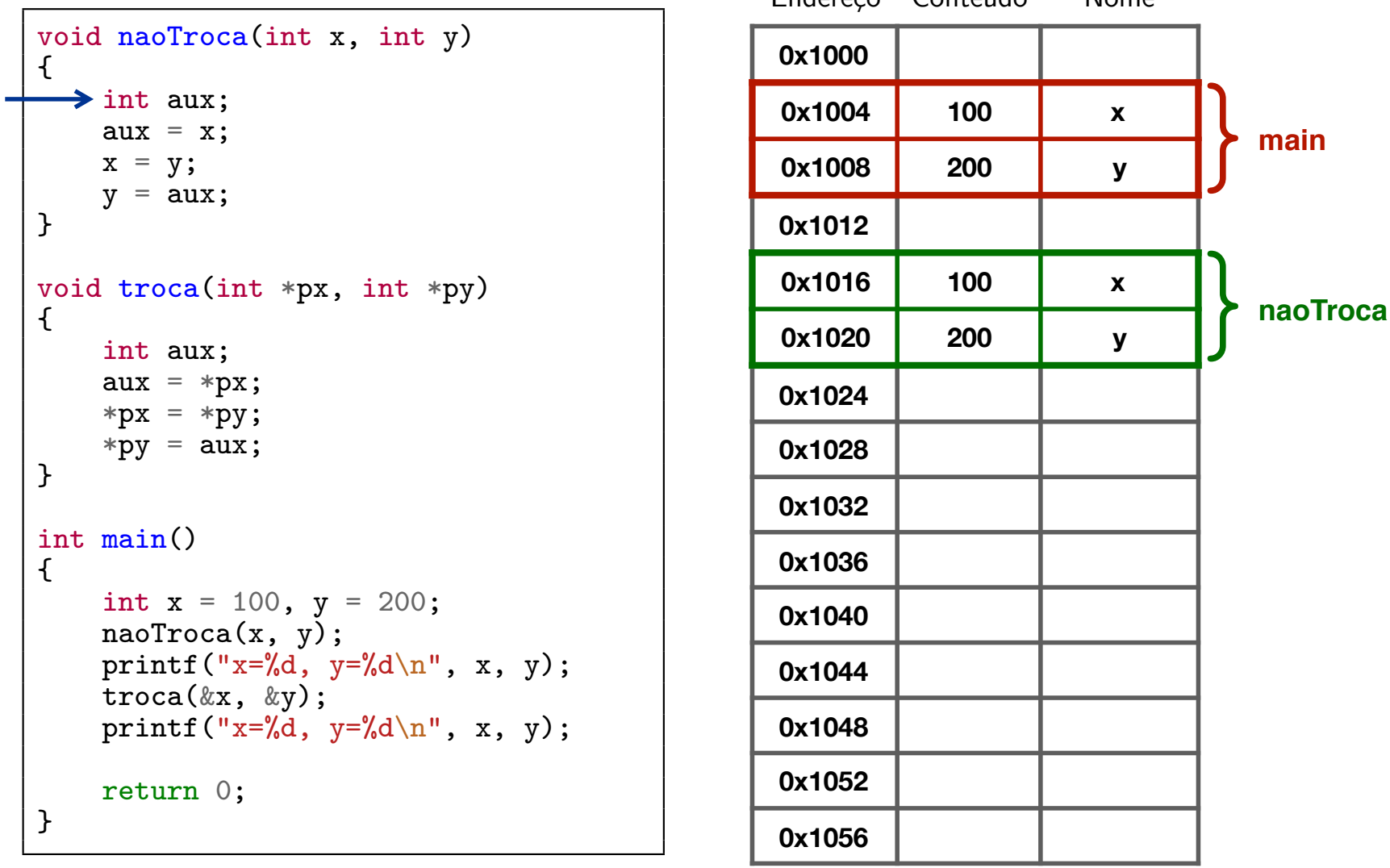
Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016		x	naoTroca
0x1020		y	naoTroca
0x1024			
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

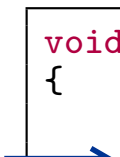
Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```



Exemplo de execução

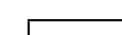
```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```



Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	
0x1012			
0x1016	100	x	naoTroca
0x1020	200	y	
0x1024		aux	
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```



Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016	100	x	naoTroca
0x1020	200	y	naoTroca
0x1024	100	aux	naoTroca
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016	200	x	naoTroca
0x1020	200	y	naoTroca
0x1024	100	aux	naoTroca
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1   void naoTroca(int x, int y)
2   {
3       int aux;
4       aux = x;
5       x = y;
6       y = aux;
7   }
8
9   void troca(int *px, int *py)
10  {
11      int aux;
12      aux = *px;
13      *px = *py;
14      *py = aux;
15  }
16
17  int main()
18  {
19      int x = 100, y = 200;
20      naoTroca(x, y);
21      printf("x=%d, y=%d\n", x, y);
22      troca(&x, &y);
23      printf("x=%d, y=%d\n", x, y);
24
25      return 0;
26  }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	
0x1012			
0x1016	200	x	naoTroca
0x1020	100	y	
0x1024	100	aux	
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032		px	troca
0x1036		py	troca
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032	0x1004	px	troca
0x1036	0x1008	py	troca
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032	0x1004	px	troca
0x1036	0x1008	py	troca
0x1040		aux	troca
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	100	x	main
0x1008	200	y	main
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032	0x1004	px	troca
0x1036	0x1008	py	troca
0x1040	100	aux	troca
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	200	x	main
0x1008	200	y	
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032	0x1004	px	troca
0x1036	0x1008	py	
0x1040	100	aux	
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	200	x	main
0x1008	100	y	main
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032	0x1004	px	troca
0x1036	0x1008	py	troca
0x1040	100	aux	troca
0x1044			
0x1048			
0x1052			
0x1056			

Exemplo de execução

```
1  void naoTroca(int x, int y)
2  {
3      int aux;
4      aux = x;
5      x = y;
6      y = aux;
7  }
8
9  void troca(int *px, int *py)
10 {
11     int aux;
12     aux = *px;
13     *px = *py;
14     *py = aux;
15 }
16
17 int main()
18 {
19     int x = 100, y = 200;
20     naoTroca(x, y);
21     printf("x=%d, y=%d\n", x, y);
22     troca(&x, &y);
23     printf("x=%d, y=%d\n", x, y);
24
25     return 0;
26 }
```

Endereço	Conteúdo	Nome	
0x1000			
0x1004	200	x	main
0x1008	100	y	
0x1012			
0x1016			
0x1020			
0x1024			
0x1028			
0x1032			
0x1036			
0x1040			
0x1044			
0x1048			
0x1052			
0x1056			

Aula: Ponteiros e Funções

Exercícios

Exercício 1

Indique e corrija o erro do código a seguir:

```c
int main()
{
    int valor;
    scanf("%d", &valor);

    int *p = &valor;
    p = p * p;
    printf("Valor ao quadrado = %d\n", valor);

    return 0;
}
```

Exercício 2

Crie uma função que duplica o conteúdo da memória apontada por um ponteiro `p`. Utilize o protótipo a seguir:

```
1  void duplica(int *p);
```

Exercício 3

Faça uma única função que converte um valor em metros para: (i) jardas; (ii) pés; e (iii) polegadas. Use a função no método `main()`.

Dica: utilize o protótipo abaixo:

```
1  void dist(float metros, float *jardas, float *pes, float *polegadas);
```

- Lembre-se que 1 metro é igual a aproximadamente 1,094 jardas, 3,281 pés, e 39,3701 polegadas.

Perguntas?