Aula 20 – Arranjos (parte 3)

Norton T. Roman & Luciano A. Digiampietri

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?
- Primeira tentativa:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;

  for (x=0;x<4;x++) printf("%i, ",a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ",a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- Como fazemos para copiar um arranjo em outro?
- Primeira tentativa:
- Aparentemente funciona:

```
0, 1, 2, 3,
0, 1, 2, 3,
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;

  for (x=0;x<4;x++) printf("%i, ",a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ",a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- E se fizermos:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ",a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ",a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- E se fizermos:
- Teremos

```
0, 1, 2, 9,
0, 1, 2, 9,
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ",a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ",a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- E se fizermos:
- Teremos

  0, 1, 2, 9,
  0, 1, 2, 9,

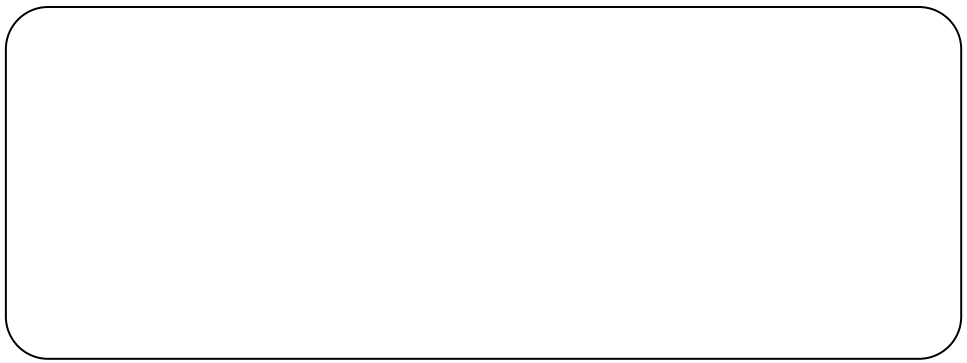
- O que houve?
  Mudamos também a2

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ",a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ",a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- Voltemos à memória

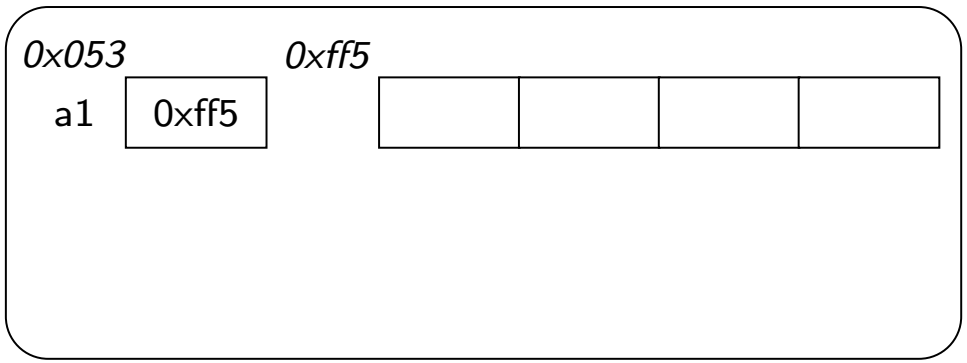
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Voltemos à memória
- Quando a1 é declarada, há memória alocada para a1 e também reservamos memória (para o arranjo), com o *malloc*.

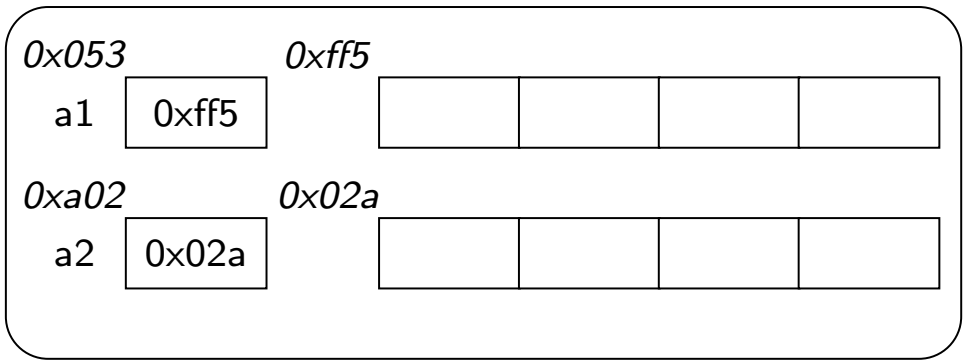
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Voltemos à memória
- Quando a1 é declarada, há memória alocada para a1 e também reservamos memória (para o arranjo), com o *malloc*.
- Quando a2 é declarada, há memória alocada para a2 e também reservamos memória com o *malloc*

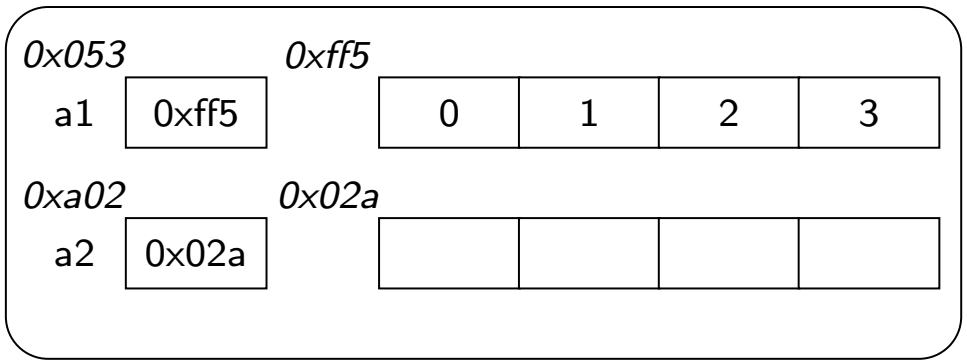
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Voltemos à memória
- Quando a1 é declarada, há memória alocada para a1 e também reservamos memória (para o arranjo), com o *malloc*.
- Quando a2 é declarada, há memória alocada para a2 e também reservamos memória com o *malloc*
- São atribuídos valores aos elementos de a1

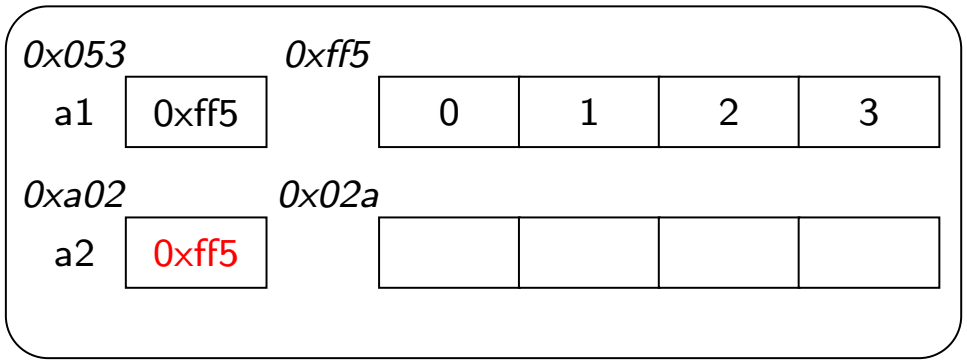
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2

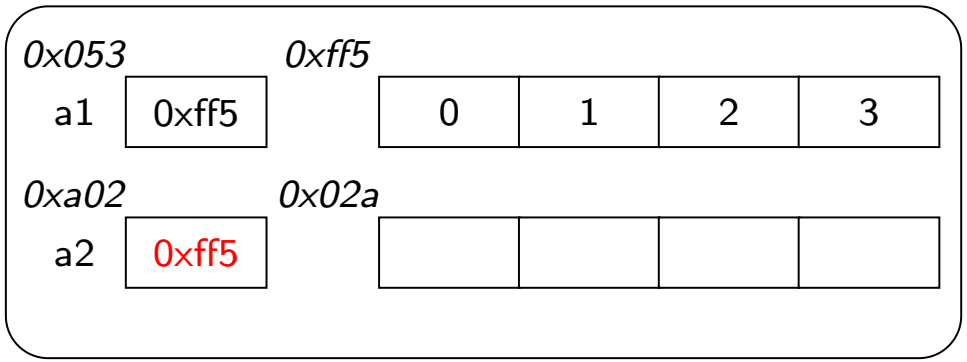
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2
- Copiamos o endereço (referência) do arranjo

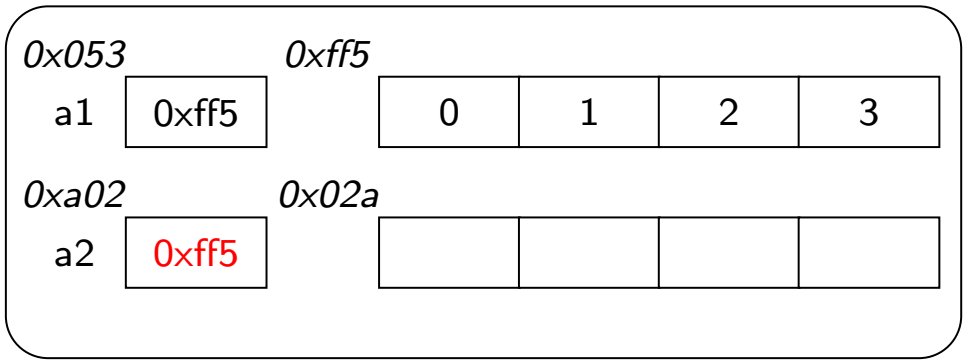
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos a2 = a1, copiamos o conteúdo de a1 para dentro de a2
- Copiamos o endereço (referência) do arranjo
- Perdemos a referência ao arranjo 0x02a

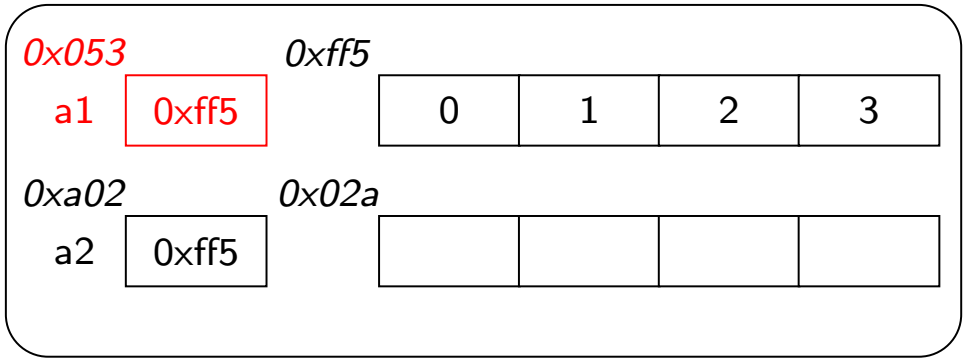
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos `a1[3] = 9`, vamos ao endereço de memória de a1 e lemos seu conteúdo – endereço do arranjo

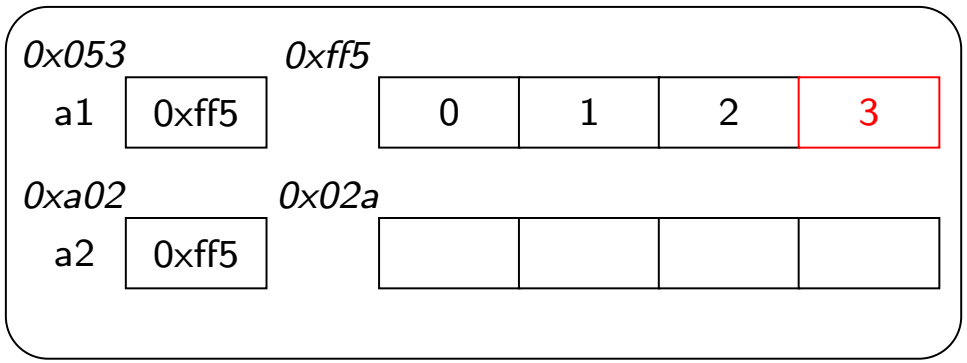
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Ao fazermos `a1[3] = 9`, vamos ao endereço de memória de `a1` e lemos seu conteúdo – endereço do arranjo
- Vamos ao endereço correspondente a $\texttt{0xff5} + 3 \times 4$
  - Quarto elemento do arranjo (com sizeof(*int*) igual a 4 bytes)

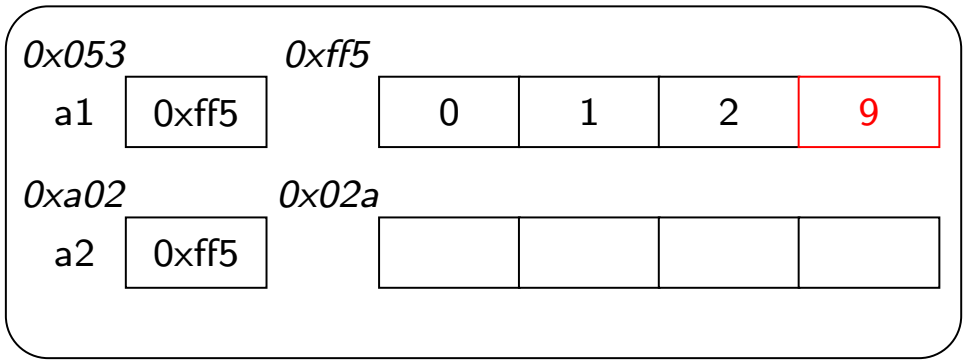
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Modificamos o valor que lá estava

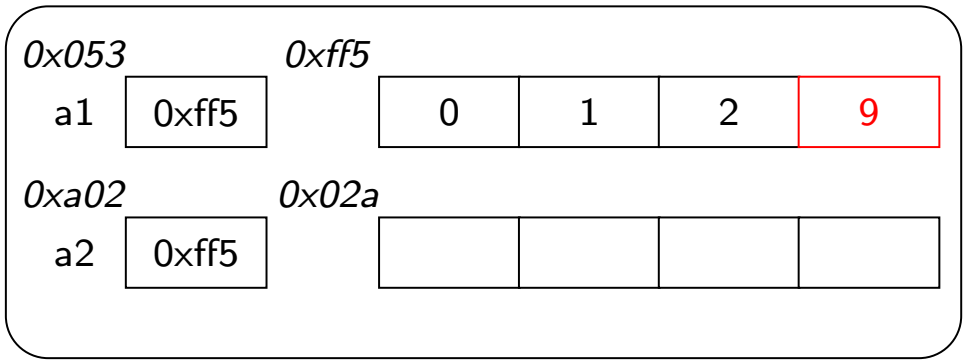
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Modificamos o valor que lá estava
- Como a2 também referencia esse mesmo arranjo, parece que o mudamos também

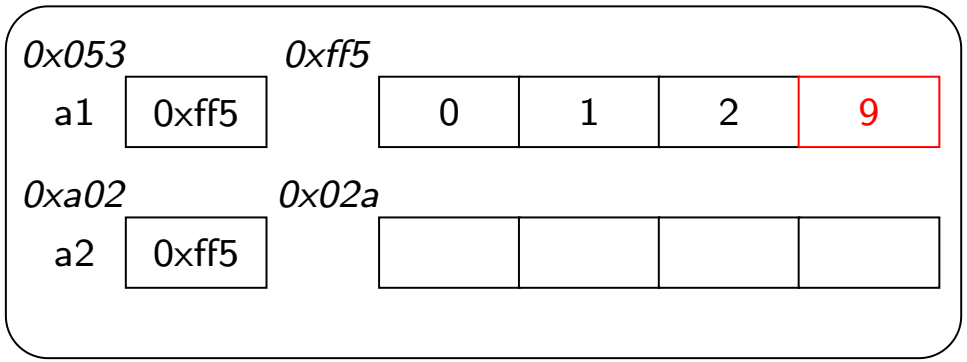
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Na verdade, fizemos tanto a1 quanto a2 referenciarem o mesmo arranjo na memória

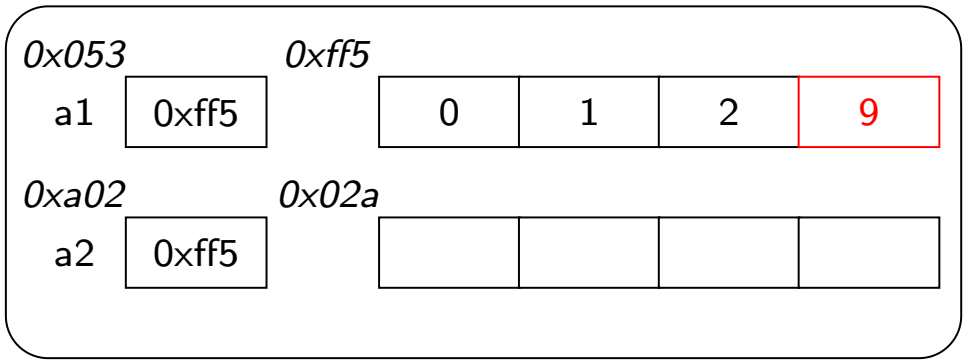
```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Na verdade, fizemos tanto a1 quanto a2 referenciarem o mesmo arranjo na memória
- Perdendo o originalmente referenciado por a2

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  a2 = a1;
  a1[3] = 9;
  ...
}
```



Copiando Arranjos

- Fazer `a2 = a1` não dá muito certo
- Que fazer?

Copiando Arranjos

- Fazer a2 = a1 não dá muito certo
- Que fazer? Copiar termo a termo os valores do arranjo correspondente a a1 para o referenciado por a2

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

int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  for (x=0;x<4;x++) a2[x] = a1[x];
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ", a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ", a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- Note que corremos o arranjo via seu índice

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

int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  for (x=0;x<4;x++) a2[x] = a1[x];
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ", a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ", a2[x]);
  printf("\n");

  return 0;
}
```

Copiando Arranjos

- Note que corremos o arranjo via seu índice
- E a saída será:

```
0, 1, 2, 9,
0, 1, 2, 3,
```

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

int main() {
  int* a1 = (int*) malloc(sizeof(int)*4);
  int* a2 = (int*) malloc(sizeof(int)*4);
  int x;
  for (x=0;x<4;x++) a1[x] = x;

  for (x=0;x<4;x++) a2[x] = a1[x];
  a1[3] = 9;
  for (x=0;x<4;x++) printf("%i, ", a1[x]);
  printf("\n");
  for (x=0;x<4;x++) printf("%i, ", a2[x]);
  printf("\n");

  return 0;
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina

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

int main() {
  double* precos = (double*)
            malloc(sizeof(double)*4);
  precos[0] = 1500;
  precos[1] = 1100;
  precos[2] = 750;
  precos[3] = 500;

  double media = 0;
  int i;
  for (i=0;i<4;i++) media += precos[i];
  media = media/4;

  printf("%8.2f\n", media);
  return 0;
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina
- Como podemos generalizá-lo?

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

int main() {
  double* precos = (double*)
            malloc(sizeof(double)*4);
  precos[0] = 1500;
  precos[1] = 1100;
  precos[2] = 750;
  precos[3] = 500;

  double media = 0;
  int i;
  for (i=0;i<4;i++) media += precos[i];
  media = media/4;

  printf("%8.2f\n", media);
  return 0;
}
```

Arranjos

- Considere agora nosso código para calcular o preço médio dos materiais da piscina
- Como podemos generalizá-lo?
- Criando um método que calcule a média dos elementos de um arranjo genérico

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

int main() {
  double* precos = (double*)
             malloc(sizeof(double)*4);
  precos[0] = 1500;
  precos[1] = 1100;
  precos[2] = 750;
  precos[3] = 500;

  double media = 0;
  int i;
  for (i=0;i<4;i++) media += precos[i];
  media = media/4;

  printf("%8.2f\n", media);
  return 0;
}
```

Arranjos

- Como?

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}
```

Arranjos

- Como?

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}
```

Arranjos

- Como?

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}
```

- Arranjos podem ser passados como parâmetro também (ou, neste caso, o endereço que referencia um arranjo)

Arranjos

- Como?

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}
```

- Arranjos podem ser passados como parâmetro também (ou, neste caso, o endereço que referencia um arranjo)
- Esta função assume um arranjo com tamanho quatro (melhoraremos isso em outra aula)

Arranjos

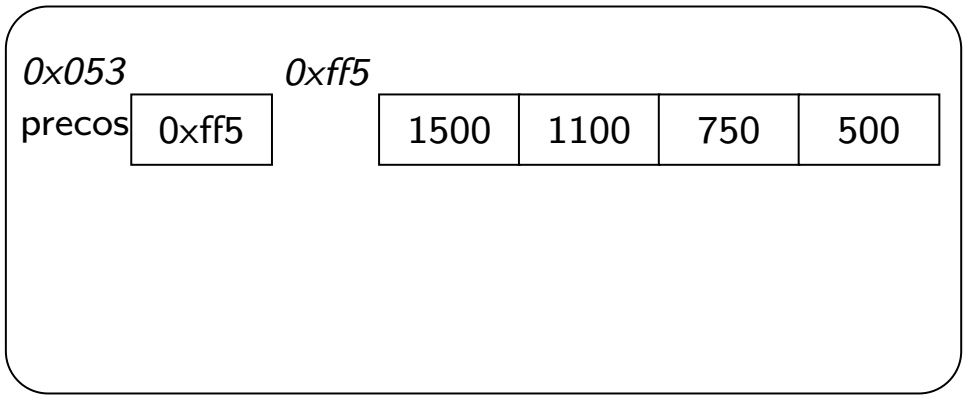
- O que acontece quando passamos um arranjo como parâmetro?

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```

Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória

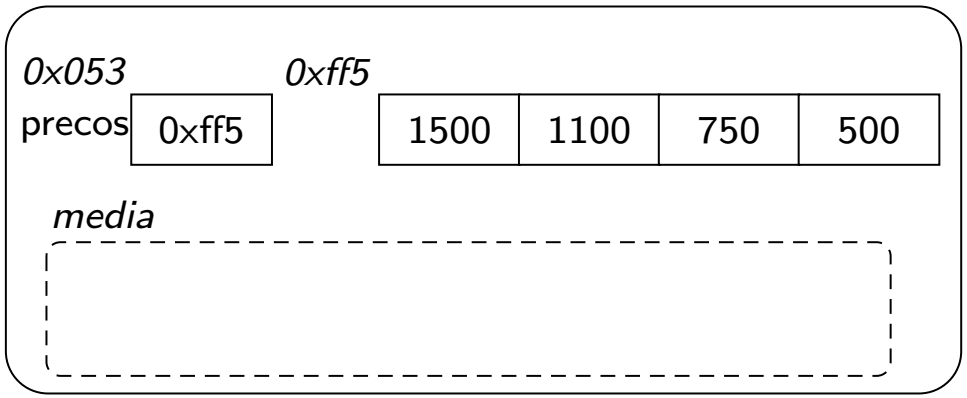
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória
- O computador separa espaço para o método invocado

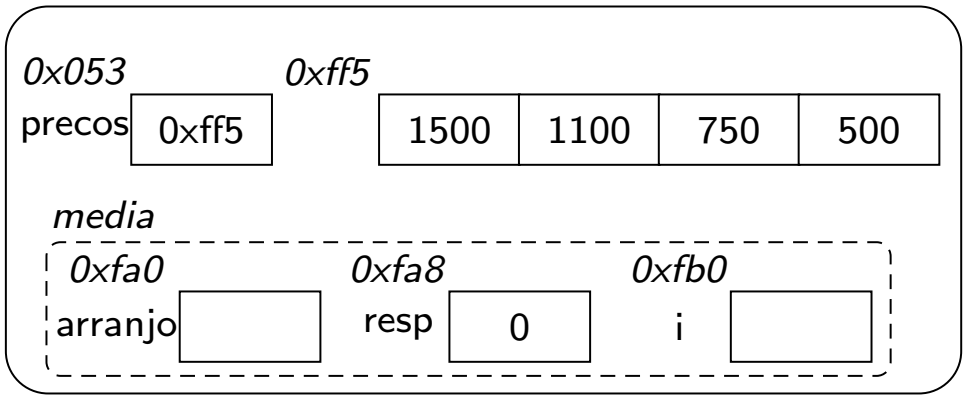
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- O que acontece quando passamos um arranjo como parâmetro?
- Lembre que o arranjo passado já está na memória
- O computador separa espaço para o método invocado
  - Para seu parâmetro e variável local

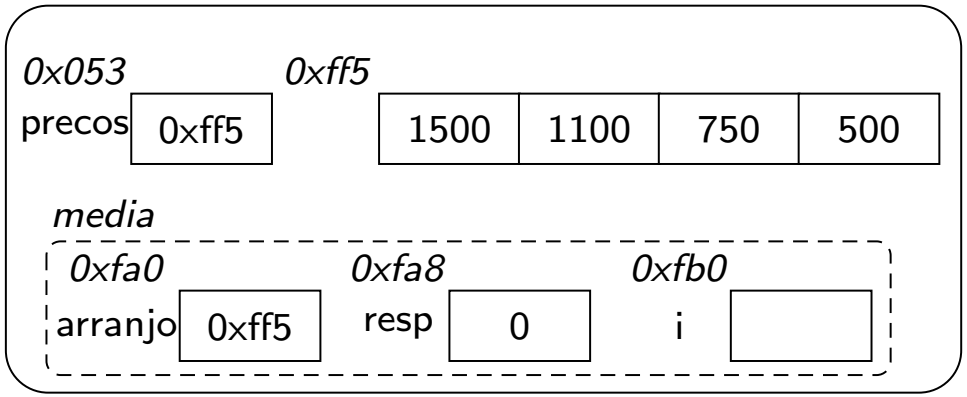
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de `precos`

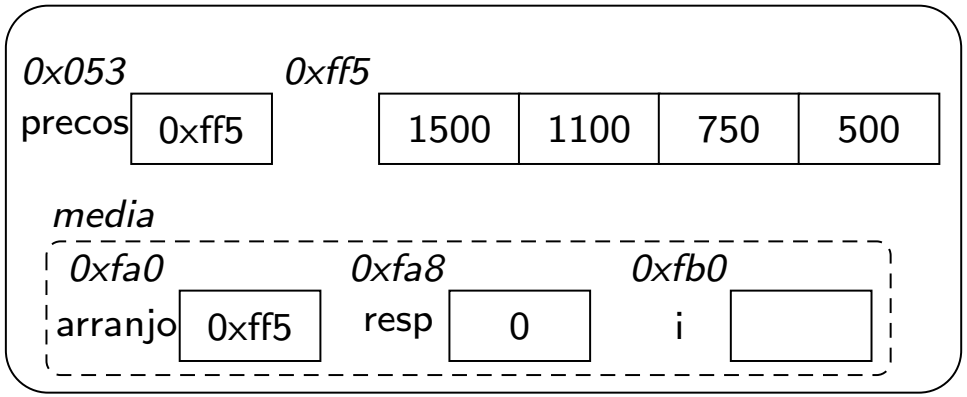
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de `precos`
  - Ou seja, o endereço do arranjo referenciado por `precos`

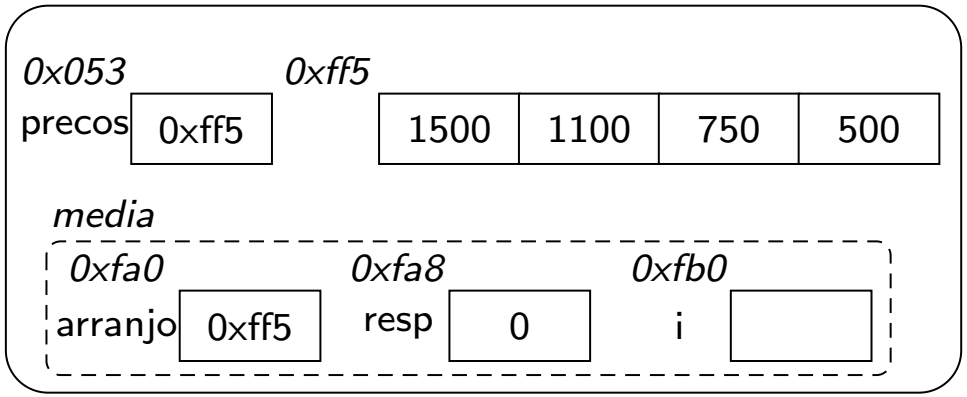
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- Copiando para seu parâmetro o conteúdo de `precos`
  - Ou seja, o endereço do arranjo referenciado por `precos`
- Com isso, ao modificarmos qualquer valor em `arranjo`, dentro de `media`, mudaremos `precos` também

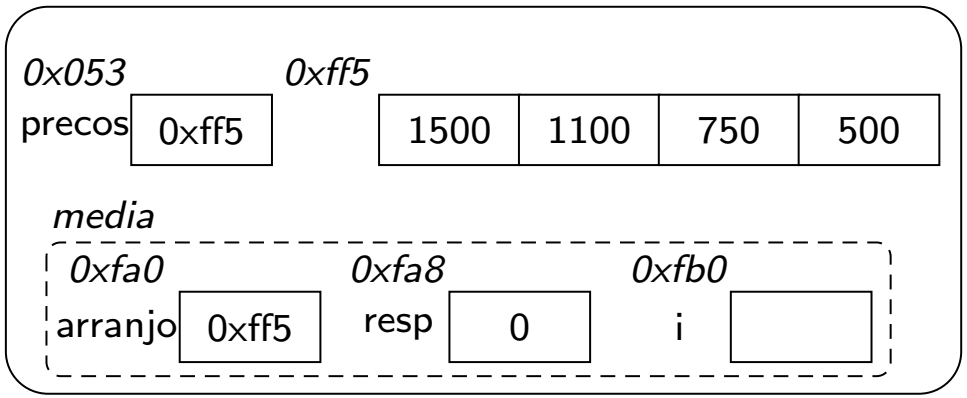
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- Pois tanto `arranjo` quanto `precos` referenciam a mesma região de memória

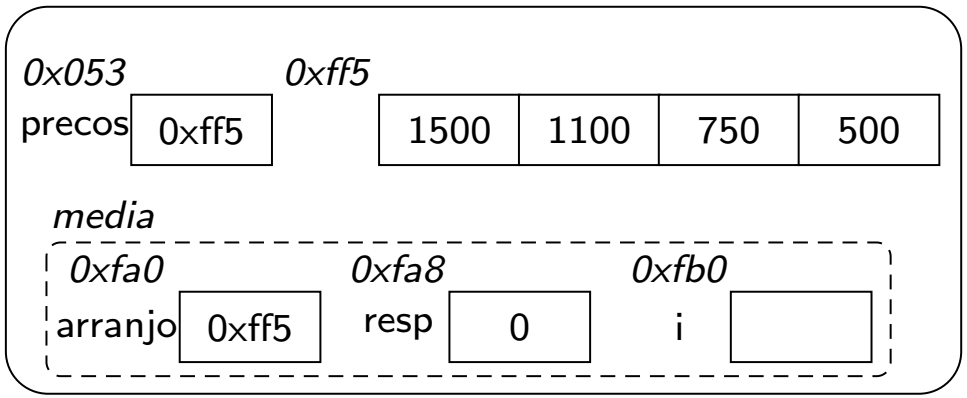
```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Arranjos

- Pois tanto `arranjo` quanto `precos` referenciam a mesma região de memória
- Passagem de parâmetro por referência

```
double media(double* arranjo) {
  int i;
  double resp = 0;
  for (i=0;i<4;i++) resp += arranjo[i];
  return resp/4;
}

int main() {
  ...
  printf("%8.2f\n", media(precos));
  return 0;
}
```



Passagem de Parâmetros

Por valor:

- O conteúdo de uma determinada região da memória é copiado para outra

Por referência:

- O conteúdo de uma determinada região da memória é copiado para outra

Passagem de Parâmetros

Por valor:

- O conteúdo de uma determinada região da memória é copiado para outra
- Esse conteúdo representa o valor para alguma variável

Por referência:

- O conteúdo de uma determinada região da memória é copiado para outra
- Esse conteúdo representa um endereço de memória $\rightarrow$ é uma referência a outra região da memória

Passagem de Parâmetros

Por valor:

- Modificações em uma das regiões não afetam a outra

Por referência:

- Modificações na região referenciada são sentidas por todas as referências àquela região

Aula 20 – Arranjos (parte 3)

Norton T. Roman & Luciano A. Digiampietri