Aula 18 – Arranjos (parte 1)

Norton T. Roman & Luciano A. Digiampietri

Arranjos

- Considere o código para calcular o valor da piscina:
- Qual o problema?

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA:
                    return(area*1500);
    case VINIL: return(area*1100);
    case FIBRA: return(area*750);
    case PLASTICO: return(area*500);
    default: return(-1);
  }
}
```

Arranjos

- Considere o código para calcular o valor da piscina:
- Qual o problema?
  - Todos os preços estão declarados dentro do método
  - Se o código crescer, fica mais difícil achar, em caso de mudança

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA:
                    return(area*1500);
    case VINIL: return(area*1100);
    case FIBRA: return(area*750);
    case PLASTICO: return(area*500);
    default: return(-1);
  }
}
```

Arranjos

- Considere o código para calcular o valor da piscina:
- Qual o problema?
  - Todos os preços estão declarados dentro do método
  - Se o código crescer, fica mais difícil achar, em caso de mudança
- Que fazer?

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA:
                    return(area*1500);
    case VINIL: return(area*1100);
    case FIBRA: return(area*750);
    case PLASTICO: return(area*500);
    default: return(-1);
  }
}
```

Arranjos

- Poderíamos agrupar essa informação, sob a forma de constantes

```
/* materiais da piscina */
#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

/* preços dos materiais */
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

Arranjos

- Poderíamos agrupar essa informação, sob a forma de constantes
- Tornaria mais fácil a manutenção do código

```
/* materiais da piscina */
#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

/* preços dos materiais */
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

Arranjos

- Poderíamos agrupar essa informação, sob a forma de constantes
- Tornaria mais fácil a manutenção do código
- Basta?

```
/* materiais da piscina */
#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

/* preços dos materiais */
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

Arranjos

- Ainda temos que relacioná-las

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA: return(area*
                          P_ALVENARIA);
    case VINIL: return(area*P_VINIL);
    case FIBRA: return(area*P_FIBRA);
    case PLASTICO: return(area*
                           P_PLASTICO);
    default: return(-1);
  }
}
```

Arranjos

- Ainda temos que relacioná-las
- E como faríamos se quiséssemos calcular o preço médio dos materiais?

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA: return(area*
                           P_ALVENARIA);
    case VINIL: return(area*P_VINIL);
    case FIBRA: return(area*P_FIBRA);
    case PLASTICO: return(area*
                            P_PLASTICO);
    default: return(-1);
  }
}
```

Arranjos

```
(P_ALVENARIA + P_VINIL +
 P_FIBRA + P_PLASTICO)/4
```

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA: return(area*
                        P_ALVENARIA);
    case VINIL: return(area*P_VINIL);
    case FIBRA: return(area*P_FIBRA);
    case PLASTICO: return(area*
                         P_PLASTICO);
    default: return(-1);
  }
}
```

Arranjos

```
(P_ALVENARIA + P_VINIL +
 P_FIBRA + P_PLASTICO)/4
```

- Deve haver um meio melhor, que mantenha o agrupamento, simplifique o código e facilite esse tipo de cálculo

```
double valorPiscina(
        double area, int material) {

  switch (material) {
    case ALVENARIA: return(area*
                        P_ALVENARIA);
    case VINIL: return(area*P_VINIL);
    case FIBRA: return(area*P_FIBRA);
    case PLASTICO: return(area*
                        P_PLASTICO);
    default: return(-1);
  }
}
```

Arranjos

- Usamos Arranjos (Array):
  - Estruturas de dados, de tamanho fixo, que permitem armazenar um conjunto de valores de um mesmo tipo

Arranjos

- Usamos Arranjos (Array):
  - Estruturas de dados, de tamanho fixo, que permitem armazenar um conjunto de valores de um mesmo tipo

Em vez de termos

```
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

Podemos fazer

```
double precos[] = {1500, 1100, 750, 500};
```

Arranjos

Deixou de ser constante

```
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

```
double precos[] = {1500, 1100, 750, 500};
```

Arranjos

Deixou de ser constante

```
const double P_ALVENARIA = 1500;
const double P_VINIL = 1100;
const double P_FIBRA = 750;
const double P_PLASTICO = 500;
```

Mas deixou o código mais enxuto

```
double precos[] = {1500, 1100, 750, 500};
```

Arranjos

- `double precos[] = {1500, 1100, 750, 500};` diz ao compilador para reservar espaço na memória para 4 doubles

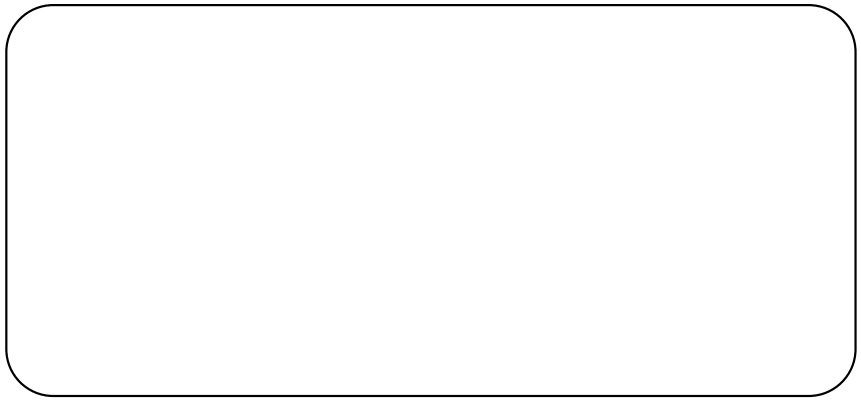
Arranjos

- `double precos[] = {1500, 1100, 750, 500};` diz ao compilador para reservar espaço na memória para 4 doubles
- Armazenando os valores 1500, 1100, 750, 500 neles

Arranjos na Memória

- O que acontece ao fazermos

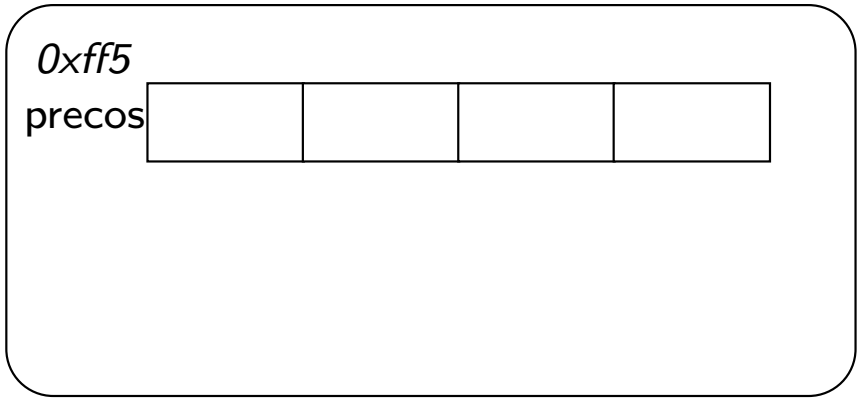
```
double precos[] =
{1500, 1100, 750, 500};?
```



Arranjos na Memória

- O que acontece ao fazermos

```
double precos[] =
{1500, 1100, 750, 500};?
```

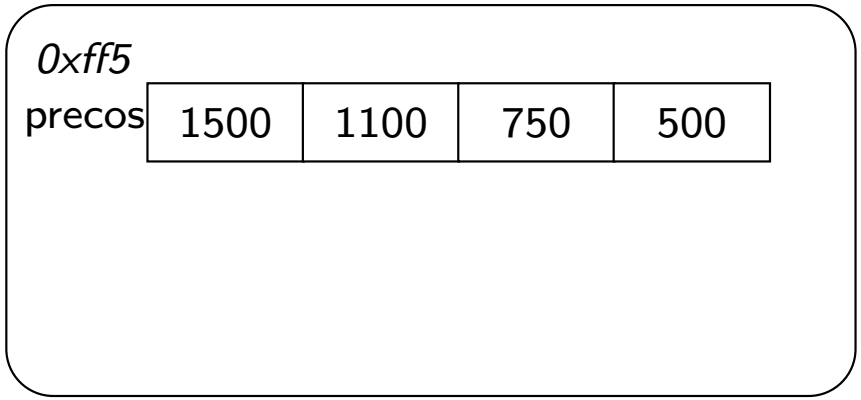


- O compilador aloca espaço suficiente para quatro *double* consecutivos (32B)

Arranjos na Memória

- O que acontece ao fazermos

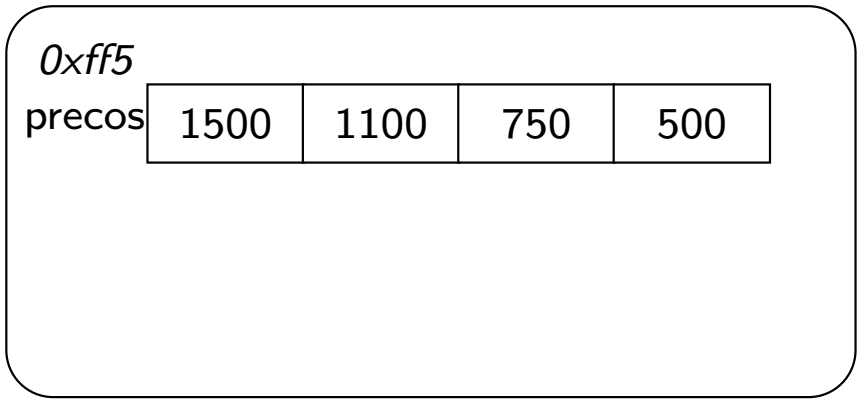
```
double precos[] =
{1500, 1100, 750, 500};?
```



- O compilador aloca espaço suficiente para quatro *double* consecutivos (32B)
- Guarda os valores da inicialização lá

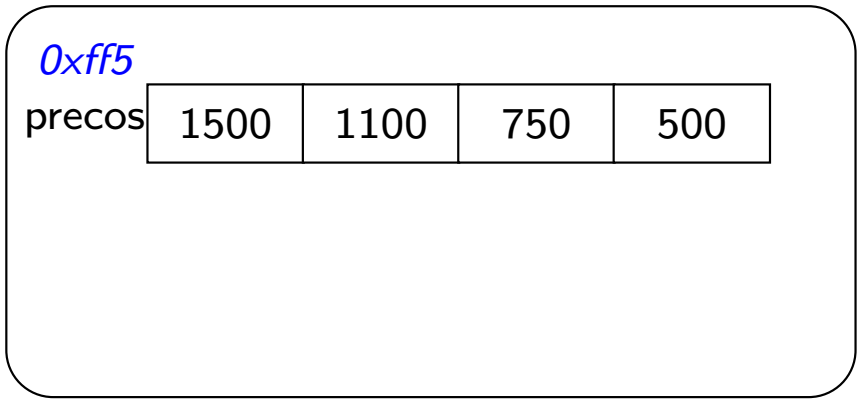
Arranjos na Memória

- Para chegar ao primeiro elemento do arranjo, o computador:



Arranjos na Memória

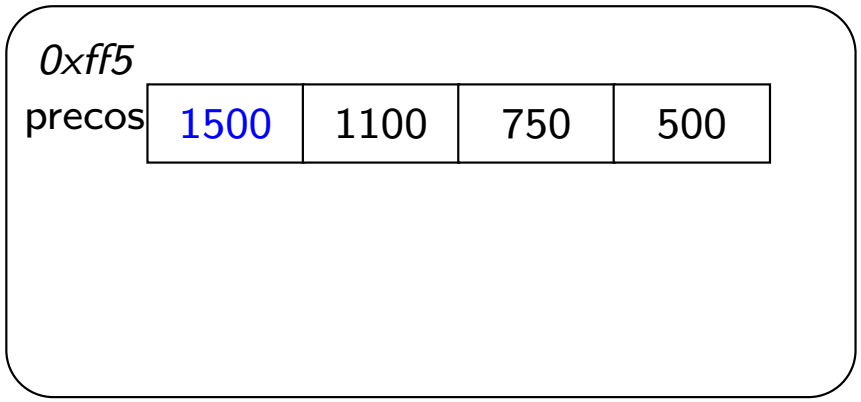
- Para chegar ao primeiro elemento do arranjo, o computador:



- Vai à região da memória correspondente a `preços`

Arranjos na Memória

- Para chegar ao primeiro elemento do arranjo, o computador:



- Vai à região da memória correspondente a `preços`
- Lê seu conteúdo

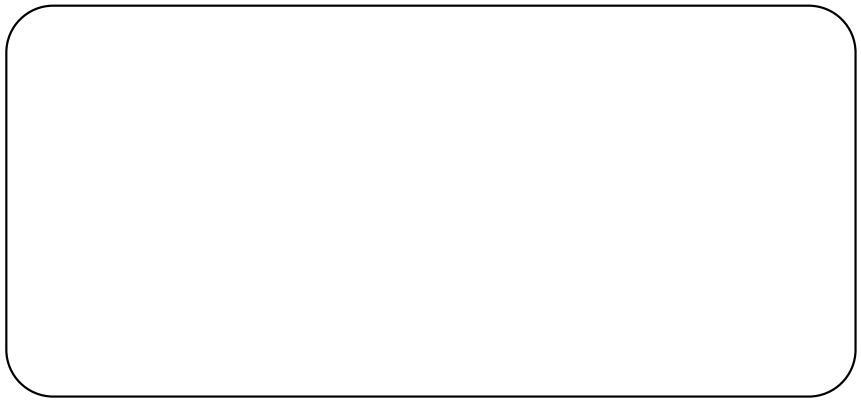
Alocação Dinâmica

- Como utilizar a função *malloc* para alocar a memória para nosso arranjo?

Alocação Dinâmica

O que acontece ao fazermos o seguinte:

```
double* precos = (double*) malloc(sizeof(double)*4);
```

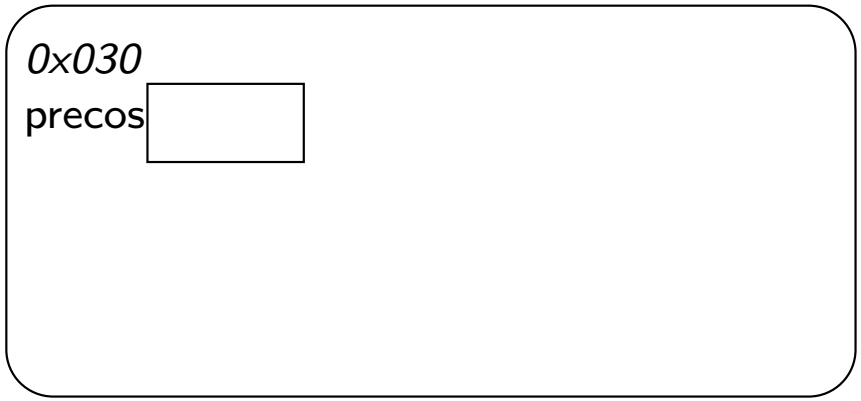


Alocação Dinâmica

O que acontece ao fazermos o seguinte:

```
double* precos = (double*) malloc(sizeof(double)*4);
```

- O compilador aloca a variável `precos`

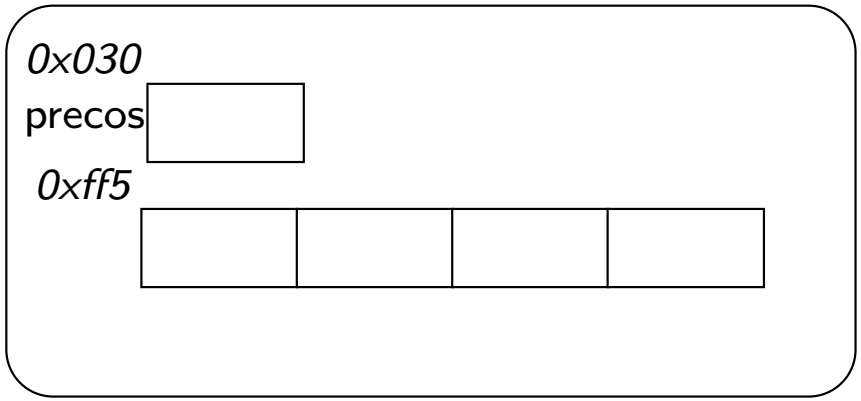


Alocação Dinâmica

O que acontece ao fazermos o seguinte:

```
double* precos = (double*) malloc(sizeof(double)*4);
```

- O compilador aloca a variável `precos`
- Em seguida a função *malloc* aloca espaço suficiente para quatro *double* consecutivos (32B)

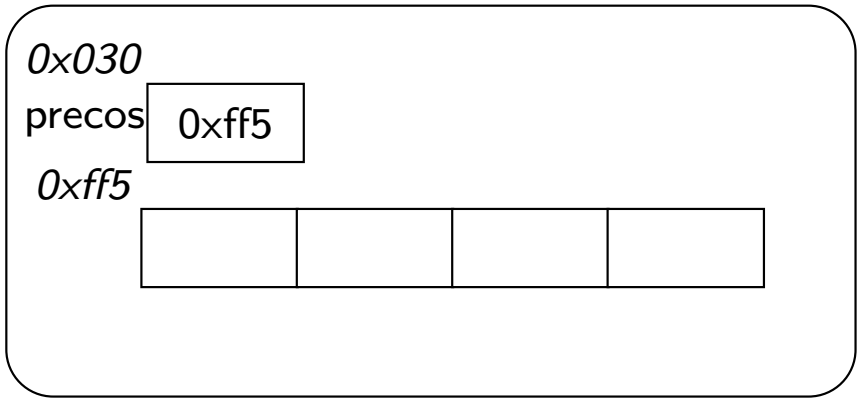


Alocação Dinâmica

O que acontece ao fazermos o seguinte:

```
double* precos = (double*) malloc(sizeof(double)*4);
```

- O compilador aloca a variável `precos`
- Em seguida a função *malloc* aloca espaço suficiente para quatro *double* consecutivos (32B)
- E retorna o endereço dessa memória que é armazenado em `precos`



Arranjos na Memória

- Em C, como podemos atribuir um valor ou ler o valor de um elemento do arranjo?

Arranjos na Memória

- Em C, como podemos atribuir um valor ou ler o valor de um elemento do arranjo?
- Fazendo `arranjo[índice]`
  - Onde índice é um inteiro de 0 a $n - 1$, com n sendo o número de elementos do arranjo
  - 0 corresponde ao primeiro elemento, 1 ao segundo, etc

Arranjos na Memória

- Em C, como podemos atribuir um valor ou ler o valor de um elemento do arranjo?
- Fazendo `arranjo[índice]`
  - Onde índice é um inteiro de 0 a $n - 1$, com n sendo o número de elementos do arranjo
  - 0 corresponde ao primeiro elemento, 1 ao segundo, etc
  - Isto funcionará para as duas formas de criação de arranjos que vimos na aula de hoje

Arranjos na Memória – Exemplo

```
#include <stdio.h>
int main() {
    double precos[] = {1500, 1100, 750, 500};
    printf("%8.2f\n", precos[0]);
    printf("%8.2f\n", precos[1]);
    printf("%8.2f\n", precos[2]);
    printf("%8.2f\n", precos[3]);
    return 0;
}
```

Arranjos na Memória – Exemplo

```
#include <stdio.h>
int main() {
   double precos[] = {1500, 1100, 750, 500};
   printf("%8.2f\n", precos[0]);
   printf("%8.2f\n", precos[1]);
   printf("%8.2f\n", precos[2]);
   printf("%8.2f\n", precos[3]);
   return 0;
}
```

Ou:

```
#include <stdio.h>
int main() {
   double precos[] = {1500, 1100, 750, 500};
   int i;
   for (i=0; i<4; i++) printf("%8.2f\n", precos[i]);
   return 0;
}
```

Arranjos na Memória

- Uma vez que o índice pode ser qualquer inteiro, podemos melhorar a legibilidade do código

```
#include <stdio.h>
#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

int main() {
   double precos[] = {1500, 1100, 750, 500};
   int i;
   for (i=ALVENARIA; i<=PLASTICO; i++) {
      printf("%8.2f\n", precos[i]);
   }
   return 0;
}
```

Alocação Dinâmica

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

#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

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

   int i;
   for (i=ALVENARIA; i<PLASTICO; i++) {
      printf("%8.2f\n", precos[i]);
   }
   return 0;
}
```

Alocação Dinâmica

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

#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

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

   int i;
   for (i=ALVENARIA; i<PLASTICO; i++) {
      printf("%8.2f\n", precos[i]);
   }
   return 0;
}
```

Alocação Dinâmica

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

#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

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

   int i;
   for (i=ALVENARIA; i<PLASTICO; i++) {
      printf("%8.2f\n", precos[i]);
   }
   return 0;
}
```

Alocação Dinâmica

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

#define ALVENARIA 0
#define VINIL 1
#define FIBRA 2
#define PLASTICO 3

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

   int i;
   for (i=ALVENARIA; i<PLASTICO; i++) {
      printf("%8.2f\n", precos[i]);
   }
   return 0;
}
```

Aula 18 – Arranjos (parte 1)

Norton T. Roman & Luciano A. Digiampietri