Aula 19 – Arranjos (parte 2)

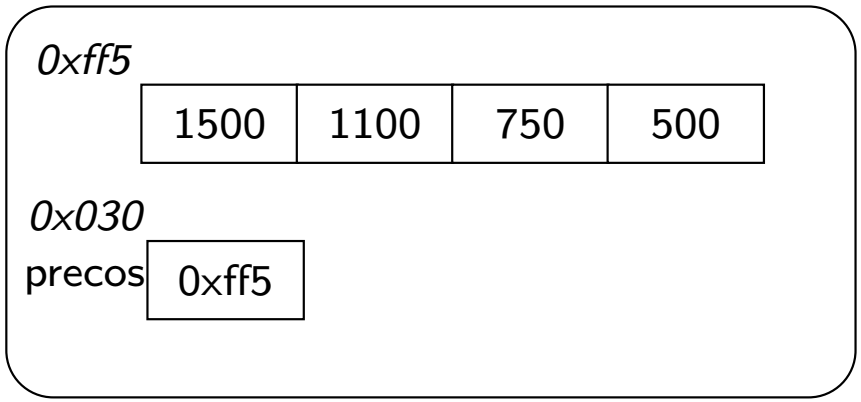
Norton T. Roman & Luciano A. Digiampietri

Arranjos

```
#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;
}
```

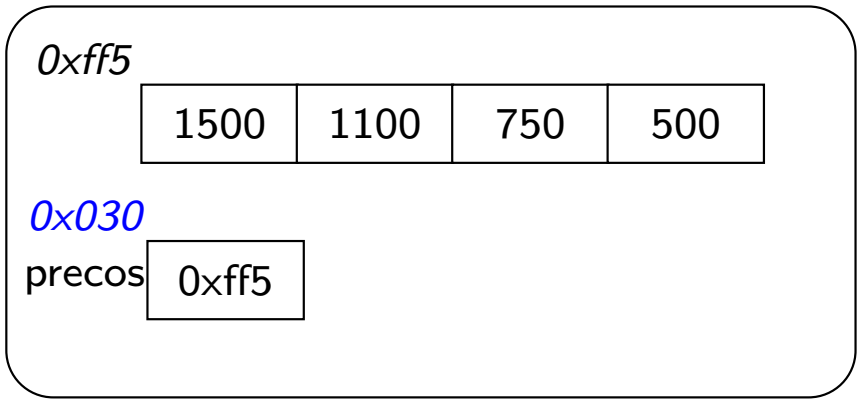
Índices na Memória

- Como o computador faz para achar o elemento de índice `i` do arranjo `precos`?



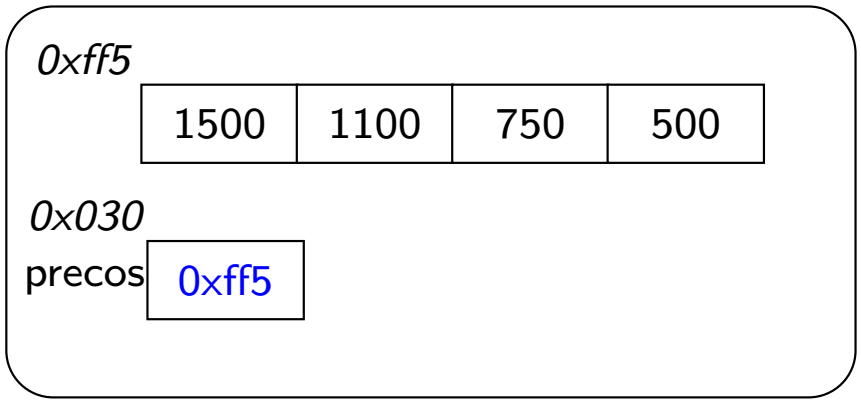
Índices na Memória

- Como o computador faz para achar o elemento de índice `i` do arranjo `precos`?
- Primeiro, vai à região da memória correspondente a `precos`



Índices na Memória

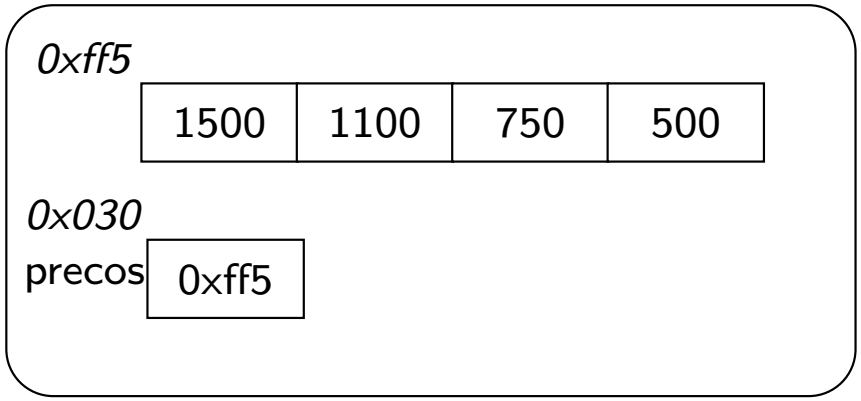
- Como o computador faz para achar o elemento de índice `i` do arranjo `precos`?
- Primeiro, vai à região da memória correspondente a `precos`



- Lê seu conteúdo – endereço do primeiro byte do arranjo

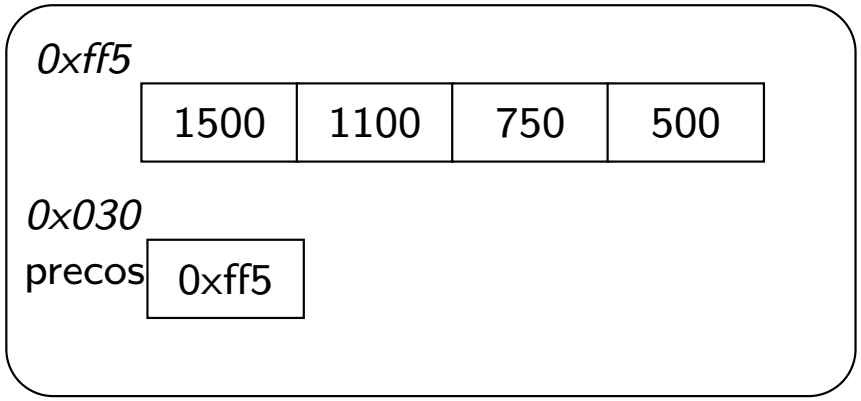
Índices na Memória

- Calcula então a posição do elemento de índice `i`:



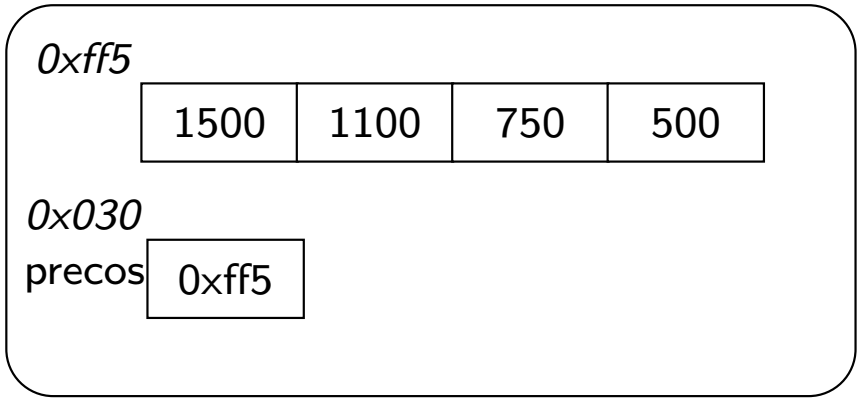
Índices na Memória

- Calcula então a posição do elemento de índice `i`:
- Sabendo que cada elemento tem 8B (double), e que $0 \leq \texttt{i} \leq \texttt{n} - 1$



Índices na Memória

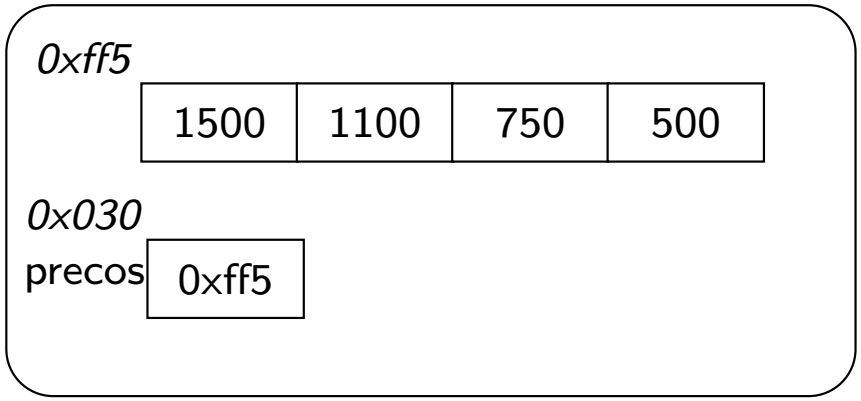
- Calcula então a posição do elemento de índice `i`:
- Sabendo que cada elemento tem 8B (double), e que $0 \leq \texttt{i} \leq \texttt{n} - 1$



- O elemento estará a $8 \times \texttt{i}$ bytes do início do arranjo

Índices na Memória

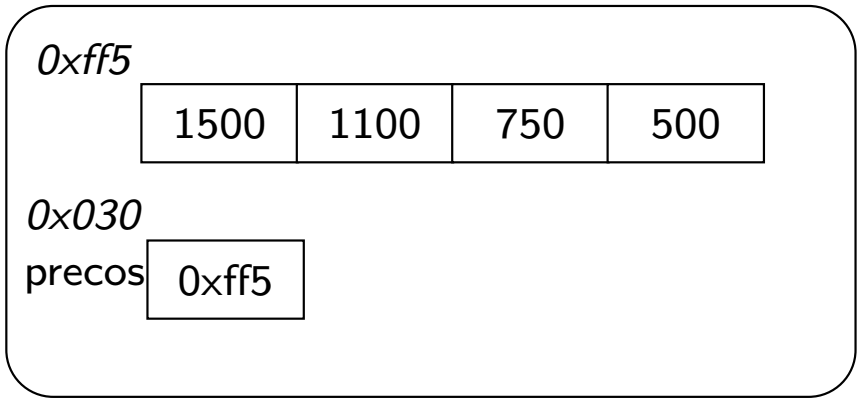
- Calcula então a posição do elemento de índice `i`:
- Sabendo que cada elemento tem 8B (double), e que $0 \leq \texttt{i} \leq \texttt{n} - 1$



- O elemento estará a $8 \times \texttt{i}$ bytes do início do arranjo
- O endereço será $\texttt{0xff5} + 8 \times \texttt{i}$

Índices na Memória

- Vai à região da memória correspondente a esse endereço e lê seu conteúdo



- Por isso o índice começa no 0. Se `i` $= 0$, o endereço visitado será o do início do arranjo

Arranjos

- Voltemos agora a `valorPiscina`...

```
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

- Voltemos agora a `valorPiscina`...
- Como ficaria usando o arranjo `precos`?

```
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

```
double valorPiscina(double
                    area, int material) {
  if (material<ALVENARIA ||
                     material>PLASTICO ||
                      area<0) return(-1);

  return(area*precos[material]);
}
```

Arranjos

- Incluímos o teste para a área

```
double valorPiscina(double
                    area, int material) {
  if (material<ALVENARIA ||
                     material>PLASTICO ||
                      area<0) return(-1);

  return(area*precos[material]);
}
```

Arranjos

- Incluímos o teste para a área
- Usamos o código do tipo do material como índice em `precos`

```
double valorPiscina(double
                    area, int material) {
  if (material<ALVENARIA ||
                     material>PLASTICO ||
                      area<0) return(-1);

  return(area*precos[material]);
}
```

Arranjos

- Incluímos o teste para a área
- Usamos o código do tipo do material como índice em `precos`

```
double valorPiscina(double
                    area, int material) {
  if (material<ALVENARIA ||
                     material>PLASTICO ||
                      area<0) return(-1);

  return(area*precos[material]);
}
```

- É importante certificar-se que `precos[0]` tem o preço de `ALVENARIA`, que `precos[1]` tem o preço de `VINIL` etc

Arranjos

- Elementos em arranjos podem ser atribuídos a outras variáveis, como uma variável comum:
  - `double x = precos[0];`
- Desde que o tipo da variável seja o mesmo do tipo armazenado no arranjo
  - Do contrário, um type cast será necessário

Arranjos

- Elementos em arranjos podem ser atribuídos a outras variáveis, como uma variável comum:
  - `double x = precos[0];`
- Desde que o tipo da variável seja o mesmo do tipo armazenado no arranjo
  - Do contrário, um type cast será necessário
- E o que acontece se tentamos acessar um elemento fora dos limites do arranjo?

Arranjos

Código

Saída

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[-1]);
   return 0;
}
```

Arranjos

Código

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[-1]);
   return 0;
}
```

Saída

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

Arranjos

Código

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[-1]);
   return 0;
}
```

Saída

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[4]);
   return 0;
}
```

Arranjos

Código

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[-1]);
   return 0;
}
```

Saída

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[4]);
   return 0;
}
```

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

Arranjos

Código

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[-1]);
   return 0;
}
```

Saída

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   double* precos = (double*) malloc(sizeof(double)*4);
   printf("%8.2f\n", precos[4]);
   return 0;
}
```

```
$ gcc Teste.c -o Teste
$ ./Teste
    0.00
```

Compilará... e NÃO irá gerar erro de execução.

Arranjos

Código

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

int main() {
   double* precos;
   printf("%8.2f\n", precos[1]);
   return 0;
}
```

Saída

Arranjos

Código

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

int main() {
   double* precos;
   printf("%8.2f\n", precos[1]);
   return 0;
}
```

Saída

```
$ gcc Teste2.c -o Teste2
$ ./Teste2
Segmentation fault (core dumped)
```

Arranjos

Código

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

int main() {
   double* precos;
   printf("%8.2f\n", precos[1]);
   return 0;
}
```

Saída

```
$ gcc Teste2.c -o Teste2
$ ./Teste2
Segmentation fault (core dumped)
```

Compilará... mas irá gerar erro de execução.
A variável *precos* está "apontando" para um endereço nulo, não é possível encontrar um endereço válido a partir de um endereço nulo.

Arranjos

- E como podemos dar um novo valor a um elemento do arranjo?

Arranjos

- E como podemos dar um novo valor a um elemento do arranjo?
  - `arranjo[índice] = novo_valor`
  - `precos[2] = 500;`

Arranjos

- E como podemos dar um novo valor a um elemento do arranjo?
  - `arranjo[índice] = novo_valor`
  - `precos[2] = 500;`
- Vimos que, para criar um arranjo, basta fazer
  - `double precos[] = {1500, 1100, 750, 500};`

Arranjos

- E como podemos dar um novo valor a um elemento do arranjo?
  - `arranjo[índice] = novo_valor`
  - `precos[2] = 500;`
- Vimos que, para criar um arranjo, basta fazer
  - `double precos[] = {1500, 1100, 750, 500};`
- Seria a única maneira?

Arranjos

```
int main() {
  double precos2[4];

  precos2[ALVENARIA] = 1500;
  precos2[VINIL] = 1100;
  precos2[FIBRA] = 750;
  precos2[PLASTICO] = 500;
}
```

Arranjos

- Criamos um arranjo de 4 elementos

```
int main() {
  double precos2[4];

  precos2[ALVENARIA] = 1500;
  precos2[VINIL] = 1100;
  precos2[FIBRA] = 750;
  precos2[PLASTICO] = 500;
}
```

Arranjos

- Criamos um arranjo de 4 elementos
- Inicializamos então esse arranjo

```
int main() {
  double precos2[4];

  precos2[ALVENARIA] = 1500;
  precos2[VINIL] = 1100;
  precos2[FIBRA] = 750;
  precos2[PLASTICO] = 500;
}
```

Arranjos

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

```
double precos2[4];

precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

Arranjos

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

```
double precos2[4];

precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

- Útil se conhecemos os valores de antemão

Arranjos

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

```
double precos2[4];

precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

- Útil se conhecemos os valores de antemão
- E se esses valores são poucos

Arranjos

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

- Útil se conhecemos os valores de antemão
- E se esses valores são poucos

```
double precos2[4];

precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

- Útil se não conhecemos os valores de antemão

Arranjos

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

- Útil se conhecemos os valores de antemão
- E se esses valores são poucos

```
double precos2[4];

precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

- Útil se não conhecemos os valores de antemão
- Ou se esses valores são muitos

Arranjos

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

```
double* precos2 = (double*)
     malloc(sizeof(double)*4);
precos2[ALVENARIA] = 1500;
precos2[VINIL] = 1100;
precos2[FIBRA] = 750;
precos2[PLASTICO] = 500;
```

- Útil se conhecemos os valores de antemão
- E se esses valores são poucos

- Útil se não conhecemos os valores de antemão
- Ou se esses valores são muitos
- Necessário se estivermos usando alocação dinâmica

Arranjos

- Arranjos podem ser criados de qualquer tipo em C:
  - `tipo nomeDaVariavel[tamanho_do_arranjo]`
  - OU
  - `tipo* nomeDaVariavel = (tipo*) malloc(sizeof(tipo)*tamanho_do_arranjo)`
- Em que `tamanho_do_arranjo` é o número de elementos que ele conterá
- Seus índices variando de 0 a `tamanho` − 1

Arranjos

Ex:

- `float precos2[4];`
- `double precos2[4];`
- `int tamanhos[10];`
- etc (veremos mais adiante)

Arranjos

- Vamos calcular o preço médio dos materiais de nossa piscina

Arranjos

- Vamos calcular o preço médio dos materiais de nossa piscina
- Como fazer?

Arranjos

- Vamos calcular o preço médio dos materiais de nossa piscina
- Como fazer?
  - Somando todos os preços, e dividindo pelo número deles

Arranjos

- Vamos calcular o preço médio dos materiais de nossa piscina
- Como fazer?
  - Somando todos os preços, e dividindo pelo número deles

```
#include <stdio.h>

double precos[] = {1500, 1100,
                           750, 500};
int main() {
    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

- Funciona... mas e se tivermos que aumentar o tamanho do arranjo?

Arranjos

- Funciona... mas e se tivermos que aumentar o tamanho do arranjo?
- Teremos que mudar o limite do `for` também

```
#include <stdio.h>

double precos[] = {1500, 1100,
                           750, 500};
int main() {
    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

- Funciona... mas e se tivermos que aumentar o tamanho do arranjo?
- Teremos que mudar o limite do `for` também
- Deve haver um meio melhor...

```
#include <stdio.h>

double precos[] = {1500, 1100,
                           750, 500};
int main() {
    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 - comprimento

- Em C, diferentemente de outras linguagens, não há um atributo predefinido com seu comprimento

- Então é comum a definição de constantes para armazenar esse valor

- ou a criação de estruturas com mais de um campo (sendo um deles o arranjo propriamente dito e outro o comprimento)

Arranjos - comprimento

- Em C, diferentemente de outras linguagens, não há um atributo predefinido com seu comprimento

- Então é comum a definição de constantes para armazenar esse valor
- ou a criação de estruturas com mais de um campo (sendo um deles o arranjo propriamente dito e outro o comprimento)

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

Aula 19 – Arranjos (parte 2)

Norton T. Roman & Luciano A. Digiampietri