UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Fundamentos de Programação

CP41F

Vetores (arrays). Vetores de caracteres (strings). Busca em vetores. Matrizes (arrays bidimensionais e multidimensionais).

Aula 11

Prof. Daniel Cavalcanti Jeronymo

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação – 1º Período

Plano de Aula

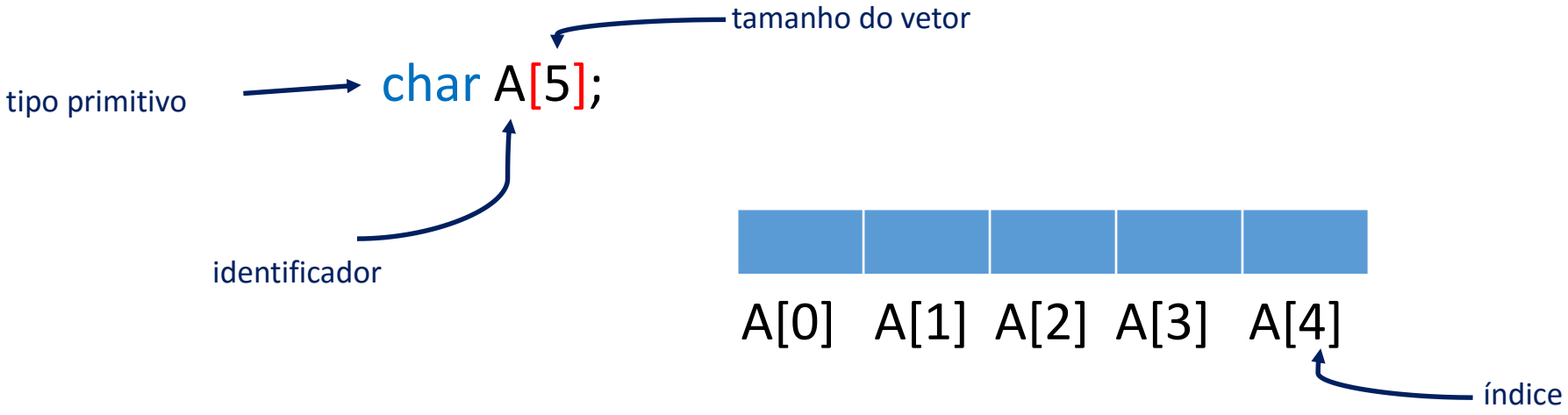
- Vetores (*arrays*)
- Busca em vetores
- Vetores de caracteres (*strings*)
- Matrizes (*arrays* bidimensionais e multidimensionais)

Vetores

- Blocos contíguos de memória

- Não são objetos, portanto não tem propriedades nem métodos associados (diferente de Java, C#, etc.)

- Blocos formados de tipos primitivos



Vetores

- Tamanho do bloco de memória depende do tipo

1 byte – sizeof(char)

```
char A[5];
```

- sizeof(A) == 5*sizeof(char)

4 bytes – sizeof(int)

```
int B[10];
```

- sizeof(B) == 10*sizeof(int)

Vetores

- Elementos são identificados por **índices**

- O índice começa em **zero**

- O *i*-ésimo elemento encontra-se no índice *i-1*

A[0] = 85; A[1] = 84; A[2] = 70; A[3] = 80; A[4] = 82;

85	84	70	80	82
A[0]	A[1]	A[2]	A[3]	A[4]

Vetores

- Elementos são acessados como qualquer variável

```
A[0] = -1;

int n = A[3];

printf("valor: %d\n", A[1]);
```

- Índices podem conter expressões:

```
A[3-2] = A[1] + 1;
```

Vetores

- Acesso ilegal – índice inválido

```
A[-1] = 0;

A[10] = 5;
```

- Atribuição ilegal – significado: “endereço de A recebe lista”

```
A = {1,2,3,4,5};
```

Vetores

- Declaração

```
tipo nome[tamanho];
```

- O tamanho pode ser uma constante ou uma variável constante

```
const int arraySize = 10;

int B[arraySize];
```

- Declaração de múltiplos vetores com o mesmo tipo e diferentes tamanhos:

```
char A[10], B[5], C[2];
```

Vetores

- Inicialização

```
int n[5] = {1, 2, 3, 4, 5};

char v[100] = {0};
```

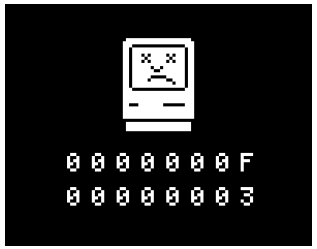
- No caso de muitos inicializadores, erro de sintaxe
- Tamanho do vetor pode ser omitido, lista de inicializadores determina tamanho

```
int n[] = {1, 2, 3, 4, 5};
```

Vetores

- *array* != *vector*

- O mesmo nome para os dois tipos de dados em português: vetor



- *Array*: Região de memória contínua destinada a um tipo de dado

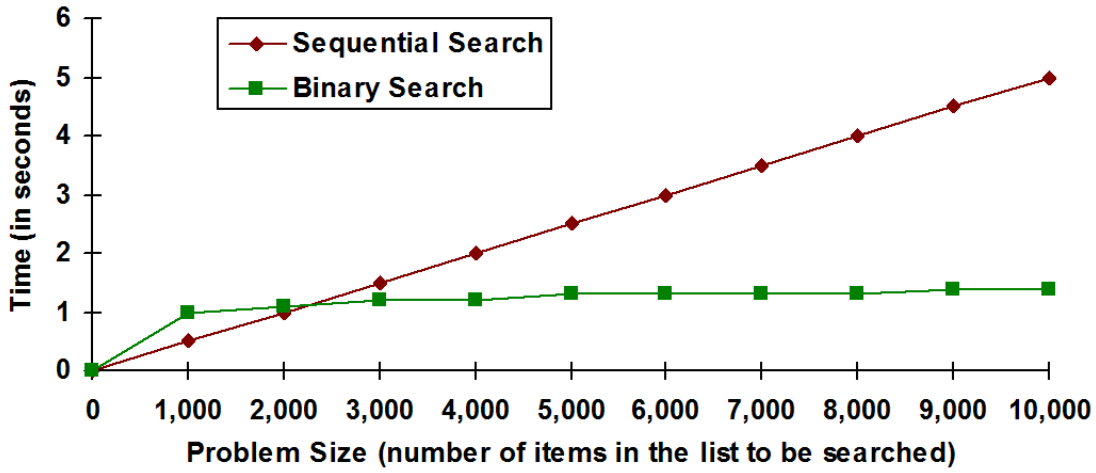
- *Vector*: Estrutura de dado similar ao *array*, com maior funcionalidade (tamanho dinâmico, acesso seguro, etc)

Busca em Vetores

- Para encontrar um valor, vetores devem ser percorridos

- Diferentes métodos:
  - Busca sequencial
  - Busca sequencial com sentinela
  - Busca binária
  - Busca interpolada
  - Busca exponencial
  - Etc...

Futuro: Estruturas de dados!



Busca em Vetores

- Busca sequencial

```c
#include <stdio.h>
int main()
{
   int i = 0, n = 0;
   int vec[] = {1, 3, -9, 5, -9, 4, 439, 123, 0, -1, 404};
   int target = 15;

   /* quantidade de elementos no vetor */
   n = sizeof(vec)/sizeof(int);

   /* caso o valor seja encontrado, i é a posição. caso não seja encontrado, i == n */
   while(i < n)
   {
      if(vec[i] == target)
         break;

      i++;
   }

   printf("Pos: %d\n", i);
   return 0;
}
```

Busca em Vetores

- Busca sequencial com sentinela

```
#include <stdio.h>
int main()
{
   int i = 0, n = 0;
   int vec[] = {1, 3, -9, 5, -9, 4, 439, 123, 0, -1, 404, 0xDEADC0DE};
   int target = 4;

   /* tamanho do vetor desconsiderando o sentinela no final */
   n = sizeof(vec)/sizeof(int) - 1;

   /* definindo o valor do sentinela */
   vec[n] = target;

   /* caso o valor seja encontrado, i é a posição. caso não seja encontrado, i == n */
   for(i = 0; vec[i] != target; i++);

   printf("Pos: %d\n", i);
   return 0;
}
```

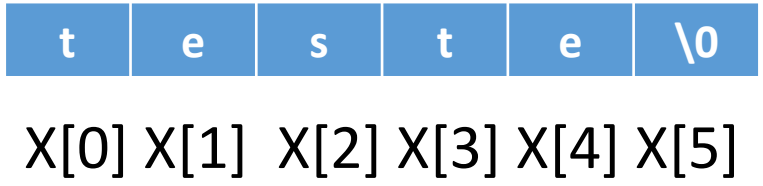
Strings

- Vetores de caracteres

```
char X[6] = “teste”;

char X[] = “teste”;

char X[6] = {‘t’, ‘e’, ‘s’, ‘t’, ‘e’, ‘\0’};
```



- O final da string é marcado pela terminação nula (*null terminated*)

Strings

- Cuidados!

- Biblioteca de strings: string.h

- Atribuição ilegal

```
char A[20];
A = “testando”;
```

- Alternativa correta

```
strcpy(A, “testando”);
```

Strings

- Cuidados!

- Biblioteca de strings: string.h

- Comparação incorreta (não é ilegal, erro de semântica)

```
char string1[] = "teste1";

char string2[] = "teste1";

printf("%d\n", string1 == string2);
```

- Alternativa correta

```
strcmp(string1, string2);
```

Strings

- Cuidados!

- Biblioteca de strings: string.h

- Concatenação ilegal

```
string3 = string1 + string2;
```

- Alternativa correta

```
strcat(string1, string2); /* resultado em string1 */
```

Strings

- Cuidados!

- *String literals* são constantes!
- Armazenadas na memória como apenas leitura.

- Atribuição ilegal (violação de acesso)

```
char *str = “teste”;

str[0] = ‘p’;
```

- Qual a diferença para vetor? Pra casa: compare o demonstrado acima com um array char str[], qual a diferença no assembly?

Strings

- Exercício

- Como contar a quantidade de letras numa string?

Vetores multidimensionais

- Um vetor é dado por:

```
tipo nome[tamanho];
```

- Um vetor multidimensional é um vetor de vetores:

```
tipo nome[tamanho1][tamanho2]...;
```

- Inicialização:

```
int A[2][3] = {{1,2,3}, {4,5,6}};
```