

Arrays [Vetores] na Stack

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Vetores

Vetores

- Coleção de itens armazenado em espaços sequenciais de memória
 - ◆ Podem ser acessado *aleatoriamente* com o índice do vetor
- Usados para armazenar tipos de elementos similares
 - ◆ Todos os elementos precisam ser do mesmo tipo de dado
- Seu tamanho é *fixo*!

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

<- Array Indices

Array Length = 9

First Index = 0

Last Index = 8

Fonte: <https://www.geeksforgeeks.org/arrays-in-c-cpp/>

Array Declaration in C

`int a[3];`

2192	451	13918
------	-----	-------

`int a[3]={1, 2, 3};`

1	2	3
---	---	---

`int a[3]={1, 1, 1};`

1	1	1
---	---	---

`int a[3]={ };`

0	0	0
---	---	---

`int a[3]={ 0 };`

0	0	0
---	---	---

`int a[3]={ 1 };`

1	0	0
---	---	---

`int a[3]={ [0...1]=3 };`

3	3	0
---	---	---

`int a[]={ [0...1]=3 };`

3	3
---	---

`int *a;
int* a;
int * a;
int*a;`



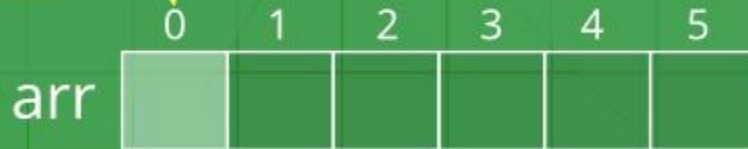
Fonte: <https://www.geeksforgeeks.org/arrays-in-c-cpp/>

Array in C

array variable

`arr [0];`

index of the element
to be accessed



Fonte: <https://www.geeksforgeeks.org/arrays-in-c-cpp/>

Vetores

```
int main ()
{
    int i, array[10];
    i = 0;
    for(i = 0; i < 10; i++)
    {
        array[i] = i*2;
        printf("Array: %d\n", array[i]);
    }
    return 0;
}
```

Vetores

```
int main ()
{
    int i, array[3];
    for(i = 0; i < 3; i++)
    {
        printf("Digite um valor:\n");
        scanf("%d", &array[i]);
        printf("Digitou: %d\n", array[i]);
    }
    return 0;
}
```


Vetores - Endereços [1]

```
int main()
{
    int arr[5], i;
    printf("Size of integer in this compiler is %lu\n", sizeof(int));
    for (i = 0; i < 5; i++)
        printf("Address arr[%d] is %p\n", i, &arr[i]);
    return 0;
}
```

Vetores

- C não acusa erro de *out of bounds*
 - ◆ Ou seja, você pode acessar regiões de memória da *stack* não alocadas pro seu vetor...
- Deixa você inicializar array com mais valores do que seu tamanho
 - ◆ Dá apenas um *warning*

Out of Bounds - Funciona “Normalmente” [1]

```
int main()
{
    int arr[2];
    printf("%d ", arr[3]);
    printf("%d ", arr[-2]);
    return 0;
}
```

Code Inicializar Mais Elementos no Vetor [1]

```
int main()
{
    // Array declaration by initializing it with more
    // elements than specified size.
    int arr[2] = { 10, 20, 30, 40, 50 };
    printf("%d", arr[2]); //Imprime 0
    return 0;
}
```

Vetores

→ É possível saber o tamanho de um vetor usando a função *sizeof()*

```
int main()
{
    int array[] = {10, 20, 30, 40};
    int n = sizeof(array)/sizeof(array[0]);
    printf("%d", n);
}
```

Strings

Strings

- É um vetor de *char*.
- Porém, termina com o caractere especial '\0' (Null)
 - ◆ Se não termina, não é *string*, é um vetor de *char*
- É possível inicializar uma string passando para ela uma sequência de caracteres entre aspas duplas
 - ◆ Caso não seja definido um tamanho, '\0' é adicionado automaticamente ao final

Strings

```
int main()
{
    //{'t', 'e', 's', 't'};
    char str[] = "test";
    printf("Size of: %lu\n", sizeof(str));
    printf("Content: %s", str);
    return 0;
}
```


Strings

- *strlen()* é uma função que nos fornece o tamanho da *string*
- Ela não funciona em vetores de *char*, pois ela procura o caractere `'\0'`
- Ela é o identificador adequado do tamanho da *string* e não o *sizeof()*, uma vez que este conta também o `'\0'`

Strings - strlen() vs sizeof()

```
int main()
{
    char str[] = "string";
    printf("String length: %lu\n", strlen(str)); //6
    printf("String size: %lu\n", sizeof(str)); //7
}
```

Strings

- Leitura de strings pode ser feita com um *scanf()* e a máscara “%s”
 - ◆ Adiciona automaticamente o ‘\0’ ao final.
 - ◆ Lembre de alocar espaço suficiente para a entrada +1, para o ‘\0’
- Para imprimir, a máscara “%s” pode ser usada. Assim como imprimir caractere por caractere, procurando pelo ‘\0’ ao final.

Strings

```
int main ()
{
    int i;
    char palavra[10];
    printf("Digite uma palavra:\n");
    scanf("%s", palavra);

    i = 0;
    while(palavra[i] != '\0')
    {
        printf("%c ", palavra[i]);
        i++;
    }
    printf("\n%s", palavra);
    return 0;
}
```

Strings

- Com *strlen()* e *sizeof()*, podemos checar que o `'\0'` não é adicionado automaticamente a não ser na inicialização com aspas duplas ou lendo a entrada com um `"%s"`.
- ◆ Porém, em alguns sistemas a stack é zerada na inicialização do programa e, como `'\0'` corresponde ao valor inteiro 0 (zero), temos a impressão de que está tudo certo com o programa ao executarmos.

Strings - '\0' não é automático!

```
int main()
{
    char str[6] = {'s', 't', 'r', 'i', 'n', 'g'};
    char str2[10];
    str2[0] = 't';
    printf("String length: %lu\n", strlen(str));
    printf("String size: %lu\n", sizeof(str));
    printf("String length: %lu\n", strlen(str2));
    printf("String size: %lu\n", sizeof(str2));
}
```

Strings

- Para ler uma *string* com espaços pode-se usar a uma expressão regular no *scanf()* (é preferível ao *gets()*, que foi removido desde o C11)
 - ◆ `scanf("%[^\n]%*c", str);`
- Uma expressão regular é uma expressão escrita em linguagem formal que identifica cadeias de caracteres de interesses e é interpretado por um programa específico.

Strings

→ `%[^\\n]*c`

◆ `[^\\n]`

- Lê qualquer coisa diferente de `\\n` até encontrá-lo

◆ `%*c`

- Lê qualquer caractere uma única vez (no caso, o `\\n`)
- Descarta o caractere lido (*)

→ `%[^\\n]` seguido por um espaço também funciona, pois também descarta o caractere `'\\n'` lido ao final

Strings com espaço

```
int main()
{
    char str[20], str2[20];
    scanf("%[^\\n]*c", str);
    printf("%s\\n", str);
    scanf("%[^\\n]*c", str2);
    printf("%s\\n", str2);
    return 0;
}
```

Referências

- https://www.ic.unicamp.br/~norton/disciplinas/mc1022s2005/06_10.html
- <http://www.cplusplus.com/doc/tutorial/arrays/>
- <https://www.geeksforgeeks.org/arrays-in-c-cpp/>
- <https://www.geeksforgeeks.org/g-fact/>
- <https://www.geeksforgeeks.org/strings-in-c-2/>
- <https://www.geeksforgeeks.org/difference-strlen-sizeof-string-c-reviewed/>
- <http://www.cplusplus.com/reference/cstring/>