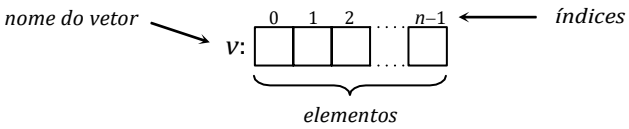
1. Estrutura de Dados: Conjunto Homogêneo Unidimensional (Vetor)

- **Definição 1:** Um *vetor* é uma coleção de espaços (variáveis) de um mesmo tipo. Esses espaços compartilham o mesmo nome e são organizados como posições consecutivas (adjacentes) de memória. Cada uma dessas variáveis denomina-se *elemento* e é identificada por um *índice*. Se v é um vetor com n posições, seus elementos são $v[0]$, $v[1]$, $v[2]$, ..., $v[n-1]$.

  Exemplo 1:



Um vetor e seus elementos

- **Definição 2:** A forma geral para declarar um vetor em C é:

  tipo nome_var [*tamanho*];

  Exemplo 2:

  ...

  int notas[10];

  ....

- **Definição 3:** Em linguagem C, a primeira posição do vetor é definida pelo índice 0. Portanto, no exemplo anterior foi declarada uma matriz que tem dez elementos, notas[0] até notas[9].

2. Instanciar um Vetor

- **Definição 4:** Atribuição de valores em um vetor segue as regras de uma variável simples. A diferença está na necessidade de indicar o índice (posição) que receberá a atribuição de um valor.

  Exemplo 3:

```
notas[1]=0;
indice=2;
notas[indice]=0;
float moeda[10]={1.0,10.1,5.2,7}; /* as demais posições são definidas
        automaticamente pelo compilador como 0.*/
```

Exemplo 4:

```
#include <stdio.h>

#define max 4

int main()
{
   int A[max] = {9, 3, 2, 7};
   int i;

   for(i=0; i<max; i++)
      printf("%d", A[i]);

   return 0;
}
```

3. Vetor de tamanho implícito:

- **Definição 5:** Quando um vetor é instanciado, o seu tamanho pode ser omitido. Nesse caso, o compilador determina o tamanho do vetor contando os elementos fornecidos na lista de valores iniciais.

Exemplo 5:

```
#include <stdio.h>

#define max 7

int main() {
   char ds[] = {'D', 'S', 'T', 'Q', 'Q', 'S', 'S'};

   for(int i=0; i<max; i++)
      printf("%c", ds[i]);

   return 0;
}
```

```
//Propriedade válida para vetor numérico.
#include <stdio.h>
#define max 7

int main()
{
       int v[] = {1,2,3,4,5,6,7};

       for(int i=0; i<max; i++)
          printf("%d ", v[i]);

       return 0;
}
```

Como o tamanho é omitido, o compilador cria o vetor *ds* com 7 posições.

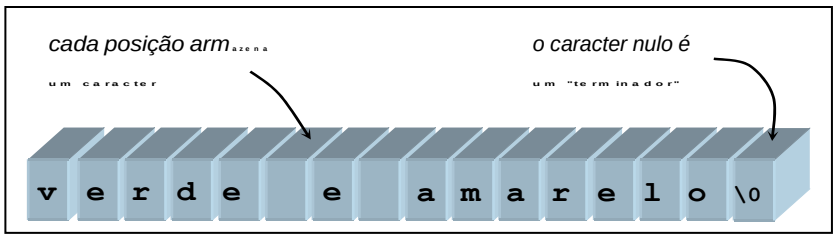
4. Strings

- **Definição 6:** Em linguagem C, string não é um tipo de dados básico como é, por exemplo, em Pascal. Uma *string* em linguagem C é uma série de caracteres terminada com um caracter nulo[1], representado por '\0'.

Vetor é uma estrutura de dados capaz de armazenar uma série de elementos do mesmo tipo e a *string* é uma série de caracteres. Assim, uma string é definida como um vetor de caracteres.

[1] *O caracter nulo '\0' é o primeiro da tabela ASCII e tem código igual a zero. Cuidado para não confundir com o caracter '0', que tem código ASCII 48.*

Uma *string* pode ser instanciada delimitando o texto por aspas, como por exemplo, "*verde e amarelo*". Internamente, essa *string* é armazenada conforme ilustrado na figura a seguir.



- **Definição 7:** Devido à necessidade do '\0', os vetores que armazenam strings devem ter uma posição a mais do que o número de caracteres a serem armazenados. **O programador é responsável por considerar o espaço adicional no momento da alocação.**

Exemplo 6:

```
#include <stdio.h>
int main()
{
   char n[21];
   printf("Qual o seu nome? ");
   scanf("%s",n);
   printf("Olá, %s!",n);
   return 0;
}
```

A chamada `scanf` permite a leitura de uma *string* e realiza o armazenamento no vetor *n*. O <*enter*> digitado para finalizar a entrada é automaticamente substituído por '\0'.

Exemplo 7 (Instanciar STRINGS):

```
#include <stdio.h>
int main()
{
   char n1[] = "um";                 /* inclui automaticamente o '\0'*/
   char n2[] = {'d','o','i','s','\0'};   /* deve ser incluído '\0' */
   char n3[] = "Verde e Amarelo";
   printf("\n %s \n %s \n %s ", n1, n2, n3);

   return 0;
}
```

a) A saída da *string n1* certamente termina após a letra *m* ter sido exibida, pois o '\0' é encontrado.

b) Considerando a *string n2*, entretanto, não há como saber quando a saída terminará se não for incluído o '\0'. No exemplo, a *string n2* será exibida corretamente. Entretanto, caso '\0' não seja incluído, o

compilador irá exibir todos os caracteres armazenados após o *s*, inclusive "lixo" de memória ou valores contidos em posições vizinhas. A saída do programa pode ser algo do tipo: `doisΩë5ßpẅ☺` ou qualquer outra coisa após *dois*;

c) Com isso, os vetores *n1, n2* e *n3* têm tamanhos 3, 5 e 16, respectivamente.

Referência

- SALES, André Barros de; AMVAME-NZE, Georges Daniel. Linguagem C: roteiro de experimentos para aulas práticas [recurso eletrônico]. Florianópolis: UFSC, 2016. Disponível em: <http://repositorio.unb.br/handle/10482/21540>.
  - ❑ Páginas 107 a 115.
  - ❑ Realizar os Experimentos e Atividades de Fixação
  - ❑ Complementar com leitura das páginas 135 a 141.