INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

ENSINO MÉDIO INTEGRADO - INFORMÁTICA
Disciplina de Linguagem de Programação
Aula 09: Vetor - *String*

Gil Eduardo de Andrade

Conceitos Preliminares
(https://docs.microsoft.com/pt-br/cpp/c-language/?view=msvc-170)

Introdução

Um vetor pode ser definido como é um conjunto de elementos (variáveis) que apresentam um mesmo tipo e são referenciados por um nome comum. Essas variáveis que compõem um vetor são acessadas através de um índice inteiro, onde o índice de menor valor (índice zero) corresponde ao primeiro elemento do vetor, enquanto o índice de maior valor corresponde ao último elemento. A declaração de vetores é efetuada, de forma geral, da seguinte maneira:

```
tipo nome_vetor[tamanho];
```

- **tipo:** indica qual é o tipo das variáveis que compõem o vetor. Exemplo: ***int***, ***float***, ***double***;
- **nome_vetor:** indica o nome pelo qual o vetor é referenciado;
- **tamanho:** indica o tamanho do vetor, o número de elementos total que ele contém;

Sintaxe e Utilização

Considerando a descrição apresentada anteriormente, podemos então declarar um vetor do ***tipo char*** de ***tamanho 10***. Teríamos:

```
char nome[10];
```

Ao declararmos o vetor "***nome***", garantimos que o espaço de memória necessário para armazenar todos os seus elementos seja reservado. Contudo, é importante

observar que apenas declaramos o vetor, ou seja, ainda não especificamos os elementos (valores) que ele deve armazenar.
Antes desta especificação, dizemos que o vetor possui "lixo de memória" armazenado em suas posições, que são valores quaisquer gerados por outros programas que utilizaram esse mesmo espaço de memória num momento anterior.

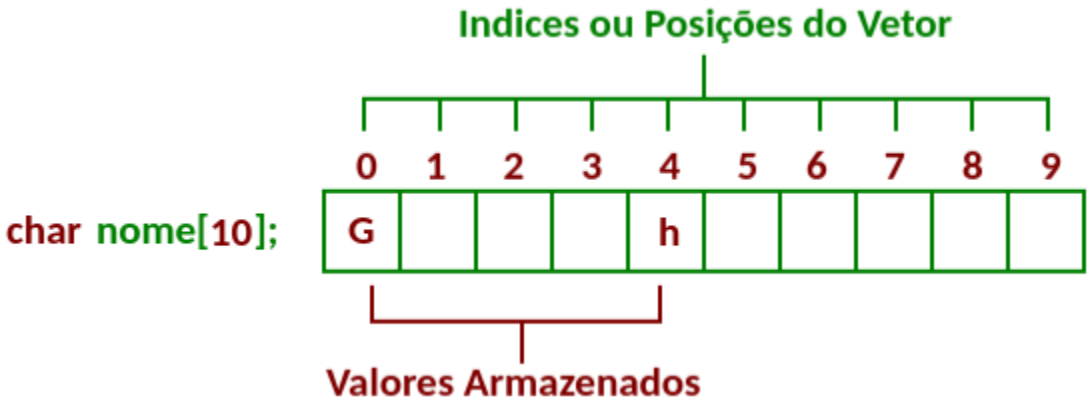
Armazenando Valores no Vetor

Como mencionado, o tamanho do vetor indica quantos elementos podemos armazenar nele. Para que seja possível especificar qual elemento do vetor receberá um dado que queremos armazenar, utilizamos um índice (valor inteiro) que indica a posição do vetor que receberá o dado.
Os índices do vetor variam de ***0*** até seu ***tamanho-1***, ou seja, o vetor **char nome[10]** possui os ***índices***: **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** e **9**. Para armazenar dados neste vetor utilizamos a seguinte sintaxe:

- **nome[0] = 'G';**
- **nome[4] = 'h';**

Representação Gráfica

Um vetor, dentro da programação, pode ser representado por um retângulo na horizontal, dividido em um número de partes exatamente igual ao seu tamanho. Onde cada uma destas partes é referenciada por um índice.



Definição de String

Quando trabalhamos com vetores de caracteres (char), dizemos que estamos trabalhando com Strings. Dentro desse contexto, podemos definir Strings como um conjunto de caracteres, ou ainda, um conjunto de elementos do tipo char.
A declaração de Strings (vetores **char**) acontece da mesma forma como visto para vetores inteiros (**int**).

Capturando Strings

Para que seja possível efetuar a leitura de uma String, sabendo que a mesma é composta por vários elementos do tipo char, utilizamos a função ***gets()***, em contrapartida a função ***scanf()***. A função ***gets()*** permite capturar uma String contendo espaços, como ***"Gil Eduardo"***. A função ***scanf()*** até permite a captura de Strings, pela utilização do parâmetro ***"%s"***, contudo, o ***scanf()*** possui uma limitação, ele captura caracteres até encontrar o primeiro espaço. Exemplos:

- `scanf("%s", &nome);`
- `gets(nome);`

Exibindo Strings

Quando precisamos exibir o conteúdo de uma String, considerando que o mesmo é composto, normalmente, por vários caracteres, utilizamos o comando printf() com o modificador %s. Exemplo:

- `printf("%s", nome);`

Final de String - Caractere '\0'

Quando precisamos percorrer uma String, utilizando, normalmente, um laço de repetição, é preciso identificar o seu término, como condição de parada para o laço de repetição. Para que isso seja possível, na linguagem C, as Strings possuem o caractere especial **'\0'** na sua última posição, indicando o seu término.

Sendo assim, ao efetuarmos a leitura de uma String digitada pelo usuário, da qual não sabemos o seu tamanho, podemos utilizar um laço de repetição e o caractere '\0' para percorrer toda essa String. Exemplo:

- ***for(a=0; rua[a] != '\0'; a++) { }***

Constantes - *#define*

O comando ***#define*** permite especificar valores que serão constantes ao longo do código-fonte C que está sendo implementado. O ***#define*** deve ser utilizado entre o trecho de código onde efetuamos a inclusão das bibliotecas e o código relativo a função ***main()***.

Sintaxe e Utilização

#define NOME_CONSTANTE valor

- **#define:** comando utilizado para definir uma constante em C;
- **NOME_CONSTANTE:** nome especificado para constante;
- **valor:** valor atribuído a constante;

Exemplo:

- **#define PI 3.1415**

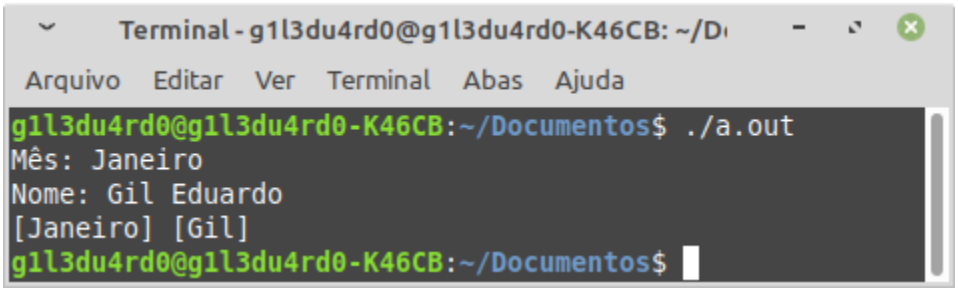
Codificação – Linguagem de Programação C

Capturando e Exibindo String - *scanf()*

```c
#include <stdio.h>

int main() {

   // Declara duas Strings

   char mes[20];

   char nome[20];

   // Captura a String utilizando scanf("%s")

   printf("Mês: ");

   scanf("%s", mes);

   // Captura a String utilizando scanf("%s")

   printf("Nome: ");

   scanf("%s", nome);

   // Exibe a String utilizando printf("%s")

   printf("[%s] [%s]", mes, nome);

   printf("\n");

   return 0;

}
```

Capturando uma String - comando *scanf()*.

(Arquivo-fonte: 10 - Vetor String/captura_scanf.c)



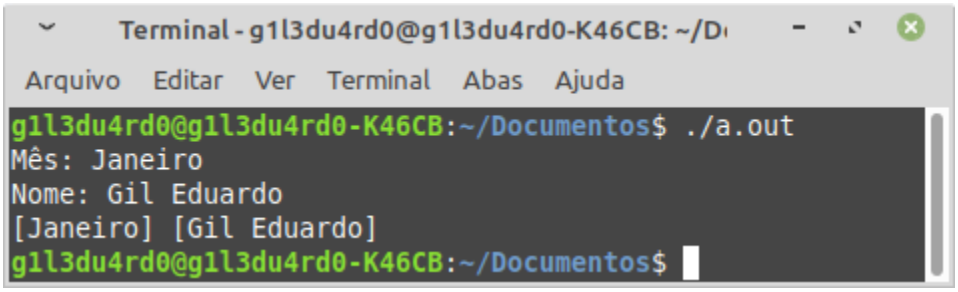
Resultado da execução do código anterior

Capturando e Exibindo String - *gets()*

```c
#include <stdio.h>

int main() {

   // Declara duas Strings

   char mes[20];

   char nome[20];

   // Captura a String utilizando gets()

   printf("Mês: ");

   get(mes);

   // Captura a String utilizando gets()

   printf("Nome: ");

   gets(nome);

   // Exibe a String utilizando printf("%s")

   printf("[%s] [%s]", mes, nome);

   printf("\n");

   return 0;

}
```

Capturando uma String - comando *gets()*.

(Arquivo-fonte: 10 - Vetor String/captura_gets.c)



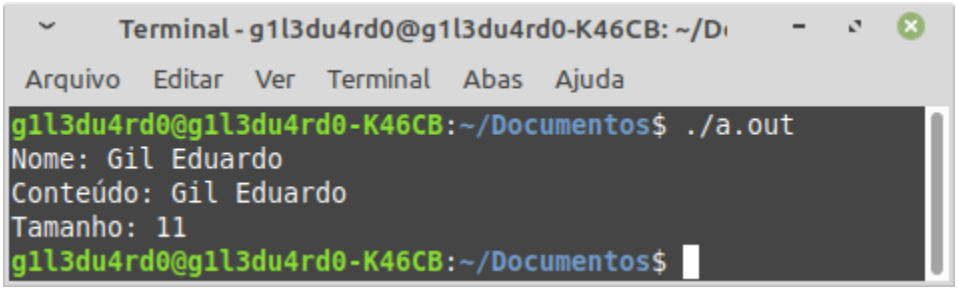
Resultado da execução do código anterior

Percorrendo uma String - '\0'

```c
#include <stdio.h>

int main() {

   // Declara um Strings

   char nome[20];

   int a;

   // Captura a String utilizando gets()

   printf("Nome: ");

   gets(nome);

   // Percorre a String até o '\0' - Exibe o Conteúdo e Calcula o Tamanho

   printf("Conteúdo: ");

   for(a=0; nome[a] != '\0'; a++) {

       printf("%c", nome[a]);

   }

   printf("\nTamanho: %i", a);

   printf("\n");

   return 0;

}
```

Percorrendo uma String - Caractere Especial '\0'.

(Arquivo-fonte: 10 - Vetor String/percorre_string.c)



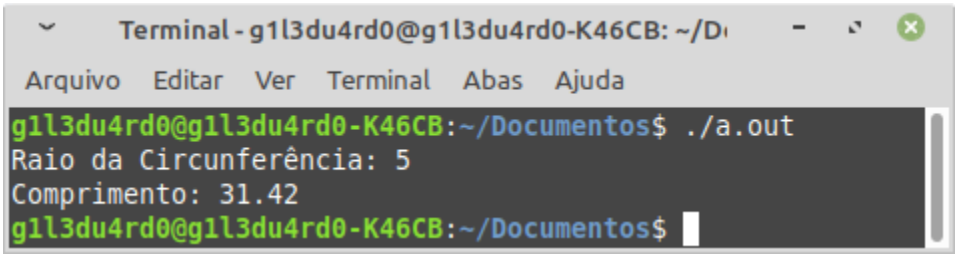
Resultado da execução do código anterior

Definindo Constantes - *#define*

```c
#include <stdio.h>

// Define a constante de nome PI e valor 3.1415
#define PI 3.1415

int main() {

   double raio, comp;

   printf("Raio da Circunferência: ");

   scanf("%lf", &raio);

   // Calcula o comprimento da circunferência

   comp = 2 * PI * raio;

   printf("Comprimento: %.2lf", comp);

   printf("\n");

   return 0;

}
```

Definindo Constantes - Comando *#define*

(Arquivo-fonte: 10 - Vetor String/constante_define.c)



Resultado da execução do código anterior