Ponteiros e Vetores

- Vetores como ponteiros

Há uma **diferença** entre o nome de um vetor e um ponteiro que deve ser **destacada**: um ponteiro é uma variável. Mas, o nome de um vetor não é uma variável. Isto significa, que não se consegue alterar o endereço que é apontado pelo "nome do vetor". Logo:

```
int vetor[10], *ponteiro, i;
ponteiro = &i;
/* as operações a seguir são inválidas */
vetor = vetor + 2;
vetor++;
vetor = ponteiro;
```

UNIVASF

Ponteiros e Vetores

- Vetores como ponteiros

Se testarmos as operações anteriores em compiladores. Eles darão, por exemplo, uma mensagem de erro do tipo:

- Lvalue;
- incompatible types in assignment.

UNIVASF

Ponteiros e Vetores

- Vetores como ponteiros

Exercício:

Construa um programa que declare uma matriz [3,4] de inteiros e a inicializa da seguinte forma:

$$\begin{bmatrix} 01 & 02 & 03 & 04 \\ 05 & 06 & 07 & 08 \\ 09 & 10 & 11 & 12 \end{bmatrix}$$

Depois, a imprima na saída padrão com o layout apresentado. As manipulações da matriz deve ser feitas utilizando um ponteiro.

UNIVASF

```c
#include <stdio.h>
#define nl 3
#define nc 4
int main () {
     int matriz[nl][nc],*p,i;
     for (i=0, p=&matriz[0][0];i<nl*nc;i++)
          *(p++)=i+1;
     for (i=0, p=matriz[0];i<nl*nc;i++)
          if (!(i%nc))
               printf ("| %02d ",*(p+i));
          else
               if (i%4==nc-1)
                    printf("%02d |\n",*(p+i));
               else
                    printf("%02d ",*(p+i));
}
```

01	02	03	04
05	06	07	08
09 (8)	10 (9)	11 (10)	12 (11)

Ponteiros e Vetores

- Ponteiros como vetores

Sabemos agora que:

- o nome de um vetor é um ponteiro constante;
- podemos indexar o nome de um vetor.

Logo, podemos também indexar um ponteiro qualquer.

O programa mostrado a seguir funciona perfeitamente:

UNIVASF

Ponteiros e Vetores

- Ponteiros como vetores

```c
#include <stdio.h>
int main ()
{
    int matrx [] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    int *p;
    p=matrx;
    printf ("O terceiro elemento do vetor e: %d", p[2]);
}
```

OBS.: Podemos constatar que p[2] é equivale a *(p+2).

Ponteiros e Vetores

- Ponteiros como vetores

Contudo, uma observação é necessária:

```c
#include <stdio.h>
int main () {
    int matrx [2][] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    int *p;
    p=matrx[0];
    printf ("O terceiro elemento da segunda linha da matriz eh: %d",p[7]);
    printf ("O terceiro elemento da segunda linha da matriz eh: %d",p[1][2]);
}
```

p=&matrx[0][0];

UNIVASF

Ponteiros e Vetores

- Strings

Seguindo o raciocínio anterior, nomes de strings, são do tipo **char***. Isto nos permite explorar os conceitos apresentados para resolver problemas como, por exemplo, o apresentado no exercício a seguir.

UNIVASF

Ponteiros e Vetores

Exercício:

Construa um programa que declare um vetor de strings com 10 elementos e o inicialize com nomes fornecidos pelo usuário através da entrada padrão e em seguida o retorne na saída padrão. A manipulação do vetor deve ser feita por meio de um ponteiro.

UNIVASF

```c
#include <stdio.h>
#define tamanho 10
#define comprimento_max 100
int main () {
    char vetor_strings[tamanho][comprimento_max],*p;
    int i;
    for (i=0, p=vetor_strings[0]; i<tamanho; i++)
    {
        printf ("\nEntre com a string[%d]: ",i+1);
        scanf("%99[^\n]", p); /*caso seja necessário lembre-se de limpar o buffer de entrada*/
        p+=comprimento_max;
    }
    printf ("\nString digitadas:");
    for (i=0, p=&vetor_strings[0][0]; i<tamanho; p+=comprimento_max, i++)
        printf ("\nString[%d]: %s", i+1, p);
}
```

Ponteiros

Strings Constantes – Modificador de Acesso *const*

UNIVASF

Ponteiros

- Inicializando Ponteiros

Podemos inicializar, ponteiros de um jeito, no mínimo interessante.

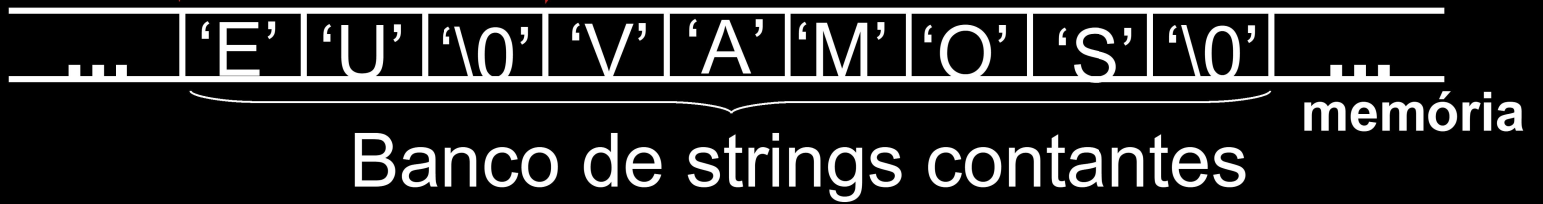
Para isto, precisamos entender como a linguagem de programação C trata as strings constantes.

Toda string constante que o programador insere no programa é colocada num banco de strings que o compilador cria. No local onde está uma string constante no programa, o compilador coloca o endereço do início desta string no banco de strings constantes.

UNIVASF

String Constante 1

String Constante 2



Banco de strings contantes

strcmp (char *, char *)

strcmp (“EU”, “VAMOS”)

Ponteiros

- Inicializando Ponteiros

O que isto tem a ver com a inicialização de ponteiros?

É que, para uma string constante que vamos usar várias vezes, podemos fazer:

```
char *str1="String constante.";
```

Aí poderíamos, em todo lugar que precisarmos da string, usar a variável **str1**. Devemos apenas tomar cuidado ao usar este ponteiro. Pois, se o alterarmos vamos perder a referência para a string e se o usarmos para alterar a string podemos facilmente corromper o banco de strings constantes que o compilador criou.

UNIVASF

Ponteiros

- Inicializando Ponteiros

OBS.: Em C existem modificadores de acesso, um exemplo é o modificador *const* que permite criar uma constante.

Exemplo:

```
const int numero = 32;
```

Logo, podemos fazer:

```
const char *str1="String constante.";
```

Desta forma, o endereço apontado pelo ponteiro não pode ser alterado. Mas, ainda é possível corromper o banco de strings constantes, ou seja, ainda devemos ter cuidado com este tipo de manipulação.

UNIVASF

Ponteiros

Funções strchr() e strstr() - string.h

Ponteiros

Agora podemos concluir nosso estudo das Funções Básicas para manipulação de Strings.

- strchr

Sua forma geral é:

*char *strchr (const char *str, int ch);*

A função *strchr()* devolve um ponteiro para a primeira ocorrência do byte menos significativo de *ch* na string apontada por *str*. Se não for encontrada nenhuma coincidência, será devolvido um ponteiro nulo. (string.h)

UNIVASF

Ponteiros

/* Exemplo strchr */

```c
#include <string.h>
int main()
{
    char *p;
    p = strchr("Isto eh um teste.",' ');
    puts(p);
}
```

Ponteiros

- strstr

Sua forma geral é:

*char *strstr (const char *str1, const char *str2);*

A função *strstr*() devolve um ponteiro para a primeira ocorrência da string apontada por *str2* na string apontada por *str1*. Ela devolve um ponteiro nulo se não for encontrada nenhuma coincidência.
Obs.: Função presente em string.h.

UNIVASF

Ponteiros

/* Exemplo strstr */

```c
#include <string.h>
int main()
{
    char *p;
    p = strstr("Isto é um teste", "te");
    puts(p);
}
```

Ponteiros

Exercício:

Construa um programa que receba duas strings fornecidas pelo usuário, verifique se a segunda string está presente na primeira e, caso esteja retorne a posição do caractere na primeira string onde a primeira ocorrência da segunda string inicia, caso contrário retorne zero.

UNIVASF