Exercício:

Construa um programa que leia duas strings fornecidas pelo usuário, através da entrada padrão, verifique se estas possuem o mesmo tamanho, caso possuam, as compare. Se forem iguais, retorne uma mensagem na saída padrão indicando este fato. Caso não possuam o mesmo tamanho, concatene-as e retorne o resultado desta operação na saída padrão.

UNIVASF

```c
#include<stdio.h>
#include<string.h>
int main()
{
    char string1[100],string2[100];
    printf("\nEntre com a  primeira string: ");
    scanf ("%99[^\n]", string1);
    printf("\nEntre com a  segunda string:  ");
    scanf ("%99[^\n]", string2);
    if (strlen(string1)==strlen(string2))
    {
       if (!strcmp(string1,string2))
          printf("\nAs strings sao iguais!\n");
    } else {
       strcat(string1,string2);
       puts (string1);
    }
}
```

```c
#include<stdio.h>
#include<string.h>
int main(){
    char string1[100],string2[100];
    printf("\nEntre com a  primeira string: ");
    scanf("%99[^\n]",string1);
   printf("\nEntre com a  segunda string:  ");
   setbuf(stdin,NULL);
   scanf("%99[^\n]",string2);
    if (strlen(string1)==strlen(string2)){
        if (!strcmp(string1,string2))
           printf("\nAs strings sao iguais!\n");
    } else {
        if (strlen(string1)+strlen(string2)<=99){
            strcat(string1,string2);
            puts (string1);
        }else
            puts ("\nAs strings concatenadas nao podem ser armazenadas na variavel declarada!\n");
    }
}
```

3. Strings (continuação)

Exercício:

Com base no que vimos, construa um programa em C que leia duas *strings*, fornecidas pelo usuário, através da entrada padrão, e verifique se a segunda *string* lida está contida no final da primeira, retornando o resultado da verificação na saída padrão.

UNIVASF

```c
#include <stdio.h>
#include <string.h>
int main () {
    char str1[100],str2[100],logico=1;
    int tam_str1, tam_str2;
    printf ("\nEntre com a primeira string: ");
    scanf("%99[^\n]",str1);
    printf ("\nEntre com a segunda string: ");
    setbuf(stdin,NULL);
    scanf("%99[^\n]",str2);
    tam_str1=strlen(str1);
    /*tam_str2=strlen(str2);*/
    for (tam_str2=0; str2[tam_str2]; tam_str2++);
    if (tam_str2>tam_str1)
        logico=0;
    else
        for (;tam_str2;tam_str1--,tam_str2--)
            if (str1[tam_str1-1]!=str2[tam_str2-1]) {
                logico=0;
                break;
            }
    if (logico) {
        printf ("\nA segunda string esta contida no final");
        printf (" da primeira!\n");
    } else {
        printf ("\nA segunda string nao esta contida no ");
        printf ("final da primeira!\n");
    }
}
```

```c
#include <stdio.h>
#include <string.h>
#define tamanho 100
int main () {
    char str1[tamanho],str2[tamanho],logico=1;
    int tam_str1, tam_str2;
    printf ("\nEntre com a primeira string: ");
    fgets(str1,tamanho,stdin);
    printf ("\nEntre com a segunda string: ");
    fgets(str2,tamanho,stdin);
    for (tam_str1=0; str1[tam_str1]; tam_str1++);
    for (tam_str2=0; str2[tam_str2]; tam_str2++);
    if (tam_str2>tam_str1)
        logico=0;
    else
        for (;tam_str2;tam_str1--,tam_str2--)
            if (str1[tam_str1-1]!=str2[tam_str2-1]) {
                logico=0;
                break;
            }
    if (logico) {
        printf ("\nA segunda string esta contida no final");
        printf (" da primeira!\n");
    } else {
        printf ("\nA segunda string nao esta contida no ");
        printf ("final da primeira!\n");
    }
}
```

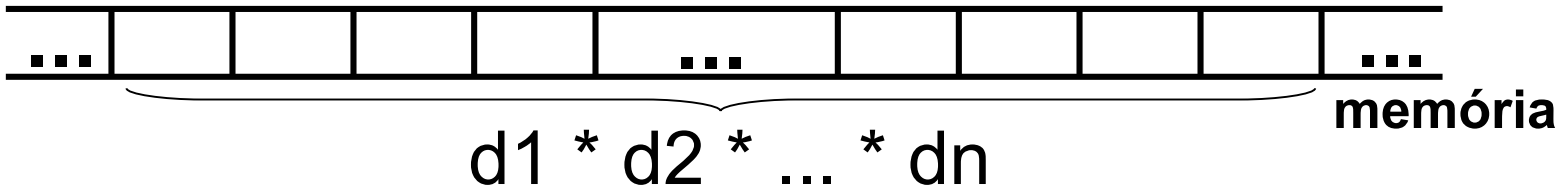
```c
#include <stdio.h>
#include <string.h>
#define tamanho 100
int main ()
{
    char str1[tamanho],str2[tamanho],logico=1;
    int tam_str1, tam_str2;
    printf ("\nEntre com a primeira string: ");
    fgets(str1,tamanho,stdin);
    printf ("\nEntre com a segunda string: ");
    fgets(str2,tamanho,stdin);
    if (strlen(str1)>=strlen(str2) &&
    !strcmp(&str1[strlen(str1)-strlen(str2)],str2)) {
        printf ("\nA segunda string esta contida no final da");
        printf (" primeira!\n");
    }else{
        printf ("\nA segunda string nao esta contida no ");
        printf ("final da primeira!\n");
    }
}
```

Vetores Multidimensionais

UNIVASF

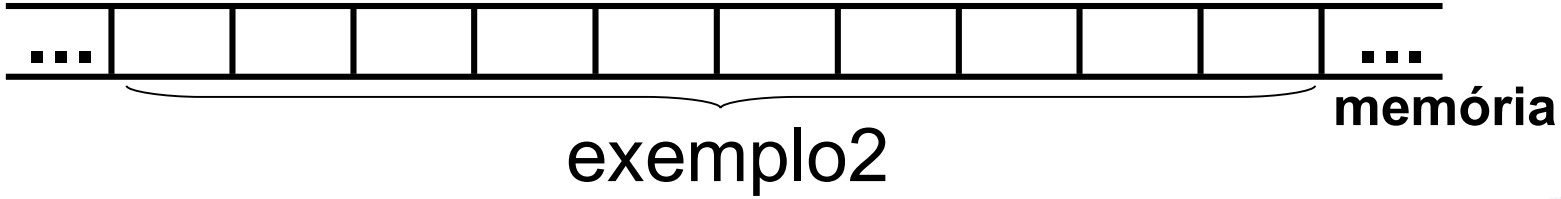
Vetores Multidimensionais

tipo_da_variavel nome_da_variavel [n];



Exemplo:

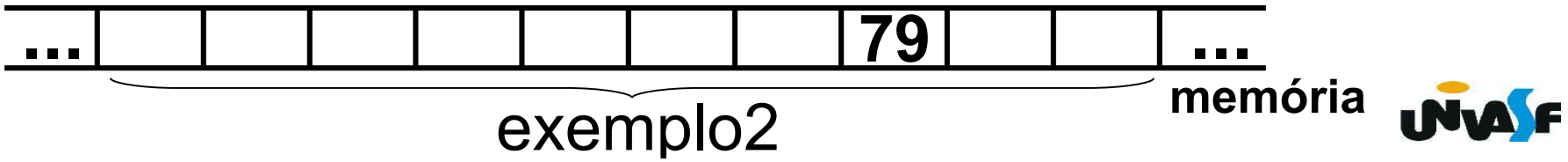
int exemplo2 [2][5];



Vetores Multidimensionais

int exemplo2[2][5];

O armazenamento de vetores multidimensionais se dá da seguinte forma: na primeira posição armazena-se o elemento com todos os índices zero, no caso do exemplo acima o elemento referenciado por exemplo2[0][0], seu sucessor é o elemento com o índice mais à direita incrementado em uma unidade (exemplo2[0][1]); quando o referido índice chegar ao seu valor máximo (exemplo2[0][4]) é incrementado o índice que o antecede e o seu valor é zerado (exemplo2[1][0]) e assim sucessivamente. Sendo assim, temos



Vetores Multidimensionais

Exemplo:

O programa abaixo declara uma matriz 3x4 de inteiros e a inicializa com valores fornecidos pelo usuário.

```c
#include <stdio.h>
#define n_l 3
#define n_c 4
int main() {
   int matriz[n_l][n_c], i, j;
   for (i=0;i<n_l;i++)
      for (j=0;j<n_c;j++) {
            printf ("\nEntre com matriz[%d][%d]=",i+1,j+1);
            scanf ("%d",&matriz[i][j]);
      }
}
```

Vetores Multidimensionais

Assim como os vetores unidimensionais os vetores multidimensionais também podem ser inicializados na declaração.

Exemplo:

float matriz [3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

float matriz [ ][2]={1,2,3,4,5,6,7,8,9,10,11,12};

/* Uma dimensão pode ser omitida quando a inicialização se dá na declaração */

~~*float matriz [ ][ ]={1,2,3,4,5,6,7,8,9,10,11,12};*~~

/* Mais de uma dimensão não podem ser omitidas quando a inicialização se dá na declaração */

UNIVASF

Vetores Multidimensionais

Exercício: Construa um programa, na linguagem C, que declare uma matriz 7x4 de números em ponto flutuante, a inicialize com valores fornecidos pelo usuário, através da entrada padrão, e a retorne na saída padrão com o layout a seguir:

```
|    x.xx      x.xx    ...      x.xx |
|    x.xx      x.xx    ...      x.xx |
|       .         .      .         . |
|       .         .      .         . |
|       .         .      .         . |
```

10 10 ...

```c
#include <stdio.h>
#define nl 7
#define nc 4
int main() {
   float matriz[nl][nc];
   int i, j;
   for (i=0;i<7;i++)
      for (j=0;j<nc;j++) {
         printf ("\nEntre com matriz[%d][%d]=",i+1,j+1);
         scanf ("%f",&matriz[i][j]);
       }
   for (i=0;i<nl;i++) {
      printf("\n|");
      for (j=0;j<nc;j++)
         printf ("%10.2f",matriz[i][j]);
      printf(" |");
    }
}
```

Vetores Multidimensionais

Exercício: Construa um programa, na linguagem C, que declare uma matriz 5x5 de números naturais, a inicialize com valores fornecidos pelo usuário, através da entrada padrão, e, após a inicialização, efetue uma pesquisa nos elementos retornando na saída padrão a soma dos elementos localizados abaixo da diagonal principal.

UNIVASF

```c
#include <stdio.h>
#define nl 5
#define nc 5
int main() {
   int matriz[nl][nc], i, j, aux;
   for (i=0;i<nl;i++)
      for (j=0;j<nc;j++)
      do {
            printf ("\nEntre com um valor natural para matriz[%d][%d]=",i+1, j+1);
            scanf ("%f",&matriz[i][j]);
      } while (matriz[i][j]<0);
   for (aux=0, i=0;i<nl;i++)
      for (j=0;j<nc;j++)
         if (i>j)
            aux+= matriz[i][j];
   printf ("A soma dos elementos posicionados abaixo da diagonal principal é %d\n",aux);
}
```