#define IDENTIFICADOR valor Valor constante: identificador será substituído, em tempo de execução, pelo valor associado. IDENTIFICADOR escrito em maiúsculas. Entre identificador e valor, nada além de espaços deve ser colocado. Pode ser utilizado, mas não pode ser alterado durante o programa.

```
Uso de Constante
       #define IDENTIFICADOR valor
// Exemplo de uso de constante:
#define LIMITE 30
                                      printf("Alunos: %d", LIMITE);
int main()
                                      media = somanotas / LIMITE;
                                      indice = LIMITE;
 float nota [LIMITE];
 float media, somanotas;
                                        LIMITE = Z
 int ind:
  //ind varia de 0 a limite - 1
                                       scanf("%d",LIMITE);
  for (ind = 0; ind < LIMITE; ind++)
      printf("Informe nota do %d ° aluno", ind + 1);
scanf("%f", &nota[ind]);
                                    5e, em vez de 30 alunos, forem 60,
                                    100, 200, 500, etc., basta trocar o
                                   valor de LIMITE: flexibilidade e
                                   segurança
                  Profa. Cora H.F. Pinto
```

Lembretes importantes!!!

O índice da primeira posição de um vetor é zero.
Ex.: num[0] = 10;

O sistema não controla a correção dos índices usados. Quem deve garantir que os índices estejam dentro do intervalo correto é o programador!!!!!!

Não existe vinculação permanente entre um valor ou variável e um vetor.

Qualquer índice (variável ou constante) usado para acessar um vetor deve corresponder a um valor dentro do intervalo de índices válidos para o vetor: se o acesso ocorrer fora do esperado, valores e resultados imprevisíveis serão obtidos.

Side 3

Pola Cora H.F. Pinto Ribeiro

Ex:Ler o nome de 30 alunos e a nota correspondente.
Calcular e informar a média da turma, seguido dos nomes
dos alunos com nota inferior à média da turma.

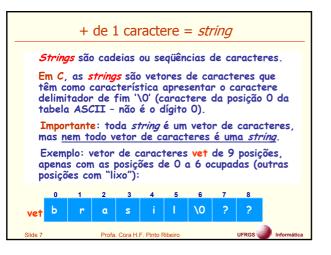
Como obter e
armazenar nomes ?

Nome : cadeia de caracteres ou string
No C:

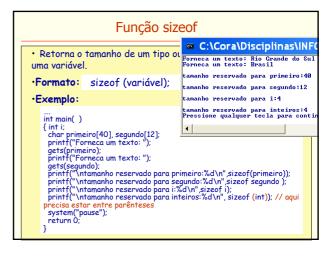
• caractere : char – 1 caractere

• strings: vetor de caracteres – + de 1

```
Função getchar
// Leitura de 1 caractere:
// Forma geral: getchar()
#include <stdio.h>
#include <stdlib.h>
                          C:\Cora\Disciplinas\INF01:
int main( )
                        Forneca um caractere: *
O caractere digitado: *
Pressione qualquer tecla para continuar
   char ch;
   int i:
   system("color 71");
printf("Forneca um caractere: ");
   ch = getchar();
   printf("O caractere digitado: %c\n", ch);
   system("pause");
   return 0;
                      Profa. Cora H.F. Pinto Ribeiro
                                                           UFRGS
```



STRINGS Declaração: char nome_da-string [elem]; O tamanho de uma string deve sempre prever a inclusão do caractere delimitador '\0'. Antes da declaração de uma variável string, contar o número máximo de caracteres a serem nela armazenados e somar 1 a este número. Exemplo: sejam 2 variáveis char dia_da_semana[?] e mes[?]: o dia da semana com major número de caracteres é segunda_feira (13) e o mês com maior número de caracteres é fevereiro (9). logo, essas variáveis devem ser declaradas no mínimo como: char dia_da_semana[14], mes[10]; Outros exemplos: char primeiro_nome[15], ultimo_sobrenome[25]; UFRGS Informática Profa. Cora H.F. Pinto Ribeiro



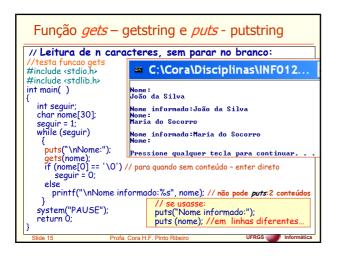
```
Inicialização de strings
Pode ser feita de várias maneiras :
Exemplo 1:
    char primeiro nome[15] = "Ana";
    O sistema insere os caracteres indicados entre as aspas duplas no
    vetor primeiro_nome, a partir da posição 0, e insere na posição 3 do
   arranjo o caractere '\0'.
Exemplo 2:
    char primeiro_nome[15] = \{'A', 'n', 'a'\};
    O sistema insere os caracteres entre chaves, a partir da posição 0. Se
    o tamanho do vetor for superior ao número de caracteres nele
    armazenados, as posições não ocupadas serão preenchidas com '\0'.
    char primeiro _nome[ ] = "Ana";
    O sistema determina o número de caracteres entre as aspas duplas,
    soma um para o caractere terminador, e cria uma string com o
    tamanho igual a tamanho da string + 1.
                     Profa, Cora H.F. Pinto Ribeiro
```

```
Inicialização de strings
 // Exemplos de strings:
                                                    C:\Cora\Disciplinas\INF0120...
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXIMO 20
                                                  Nome 2 = Ana, tamanho 3 - 15
Nome 2 = Ana-b, tamanho 5 - 15
int main( )
 char nome1[15] = "Ana";
char nome2[15] = {'A', 'n', 'a'};
char nome3[] = "Ana";
 system("color f1"):

printf("Nome 1 = %s, tamanho %d - %d\n", nome1,strlen(nome1), sizeof(nome1));

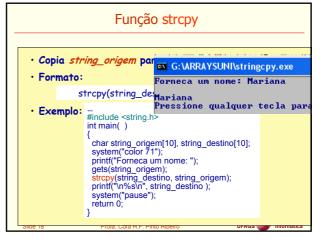
printf("Nome 2 = %s, tamanho %d - %d\n", nome2,strlen(nome2), sizeof(nome2));
 printf("Nome 3 = %s, tamanho %d - %d\n\n", nome3, strlen(nome3), sizeof(nome3));
 nome2[4]='b'; // posição 3 permanece com'\0'
printf("Nome 2 = %s, tamanho %d - %d\n", nome2,strlen(nome2), sizeof(nome2));
 nome2[3]='-'; // substitui '\0' da posição
printf("Nome 2 = %s, tamanho %d - %d\n", nome2,strlen(nome2), sizeof(nome2));
 system("pause>>null");
 return 0:
                                    Profa. Cora H.F. Pinto Ribeiro
```

Leitura e escrita de strings com scanf e printf Leitura Formato: %s, mas SEM & antes do nome da variável. O scanf encerra a leitura do string quando encontra: um caractere branco, ou um caractere de fim de linha ou de tabulação. Ex: char nome_cliente[20]; scanf("%s", nome_cliente); // se usuário digitou Maria do Socorro //em nome do cliente, apenas Maria foi armazenada Escrita O formato usado também é o %s. Ex:: char nome_cliente[20]; (...) printf("O nome do cliente eh %s\n", nome_cliente)









```
Função strcat

    A string_origem, sem alteração, é anexada ao final da

string_destino.

    Formato: strcat(string_destino, string_origem);

Exemplo:
                                                                                        ATENÇÃO:
      #include <string.h>
int main( )
                                                                                  A string_destino
                                                                                  deve ter tamanha
                                                                                  suficiente para
        char string_origem[20], string_destino[40];
system("color f1");
printf("Forneca um texto: ");
                                                                                  armazenar o
                                                                                  resultado de
       gets(string_origem);
strcpy(string_destino, "O texto digitado foi: ");
strcat(string_destino, string_origem);
printf("\n%s\n", string_destino);
system("pause");
return 0;
Forneca um texto: Lindo
Forneca um texto: Lindo
                                                                                  strcat
                                                O texto digitado foi: Lindo dia!
```

```
Função strcmp
· Compara duas strings, s1 e s2, caractere a caractere, com base na
posição dos caracteres na tabela ASCII.
         Se s1 e s2 forem iguais, retorna zero

    ✓ Se s1 for menor que s2, retorna um valor menor que zero
    ✓ Se s1 for maior que s2, retorna um valor maior que zero.

       √Formato:
                             strcmp(s1, s2);
                                                                        Lembrar que as maiúsculas
vêm antes das minúsculas
· Exemplo:
#include <string.h>
int main( )
                                                                        na tabela ASCII.
 printf("\n Ana X ana = %d", strcmp("Ana","ana")); //Ana < ana
printf("\n ana X Ana = %d", strcmp("ana","Ana")); // ana > Ana
printf("\n Ana X Ana = %d", strcmp("Ana","Ana")); // Ana = = Ana
 system("pause>>null");
 system("pause");
return 0;
                                      C:\Cora\Dis...
                                     Ana X ana = -
ana X Ana = 1
Ana X Ana = 0
```

```
Funções de manipulação de strings

Função strlen

Formato: strlen(string);

Retorna tamanho do string, sem contar o '\0'.

Função strcpy

Formato: strcpy(string_destino, string_origem);

Copia string_origem para string_destino.

Função strcat

Formato: strcat(string_destino, string_origem);

Anexa string_origem ao final de string_destino.

Função strcmp

Formato: strcmp(s1, s2);

Compara duas strings, s1 e s2, caractere a caractere, com base na posição dos caracteres na tabela ASCII, e informa se iguais - 0, se s1 < s2 - -1 ou se s1 > s2 - +1.
```

```
Funções de manipulação de strings

Função strupr

Formato: strupr(string);

Converte todas as letras minúsculas da string para maiúsculas.

Função strlwr

Formato: strlwr(string);

Converte todas as letras maiúsculas da string para minúsculas.
```

ATENÇÃO → As strings são representadas entre aspas duplas e os caracteres entre apóstrofos. → Por definição, toda string tem o caractere terminador '\0' ao final. → Assim, "A" e 'A' <u>NÃO são</u> a mesma coisa!! "A": vetor de 2 caracteres- 'A' e '\0'. 'A': um único caractere. → Uma string é sempre um vetor de caracteres (com '\0' ao final), mas um vetor de caracteres nem sempre é uma string!

```
// leitura da palavra, consistindo se não ultrapassa limite de caracteres:
do
{
    printf("Digite uma palavra (tamanho maximo %d): ", MAXPAL);
    gets(palavra);
    if (tamanho - strlen(palavra);
    if (tamanho - MAXPAL)
    printf("Tamanho maximo deve ser %d!\n", MAXPAL);
}
while (tamanho > MAXPAL);
// converte texto e palavra para minúsculas:
    strcpy(auxtxt , strlwr(texto));
    strcpy(auxtxt , strlwr(texto));
    strcpy(auxpal , strlwr(palavra));
// verifica se a palavra existe no texto:

Fazer para a próxima aula!

Slide 27 Profa Cora H.F. Pinto Ribeiro

UFRGS
```