Declaração de Constante

#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
int main()
{
 float nota [LIMITE];
 float media, somanotas;
 int ind;
  //ind varia de 0 a limite - 1
 for (ind = 0; ind < LIMITE; ind++)
       {
       printf("Informe nota do %d º aluno",ind + 1);
       scanf("%f", &nota[ind]);
...
...
printf("Alunos: %d", LIMITE);
media = somanotas / LIMITE;
indice = LIMITE;

LIMITE = 7;
scanf("%d",LIMITE);
```

Se, em vez de 30 alunos, forem 60, 100, 200, 500, etc., basta trocar o valor de LIMITE: flexibilidade e segurança.

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.

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**

1 caractere

Variável caractere: **char caract;**

Leitura:

1) **scanf** - função genérica de leitura:
scanf(" %c", &caract);
Atenção: colocar um espaço em branco antes do %c, para que caracteres de fim de linha, de espaço ou de tabulação, ainda existente no *buffer* do dispositivo, não interfira na leitura efetuada.

2) **getchar** - função específica para leitura de caracteres:
caract = getchar( );

Função getchar

```
// Leitura de 1 caractere:
// Forma geral: getchar( )
//testa funcao getchar
#include <stdio.h>
#include <stdlib.h>

int main( )
{
    char ch;
    int i;
    system("color 71");
    printf("Forneca um caractere: ");
    ch = getchar( );
    printf("O caractere digitado: %c\n", ch);
    system("pause");
    return 0;
}
```

```
C:\Cora\Disciplinas\INFO11
Forneca um caractere: *
O caractere digitado: *
Pressione qualquer tecla para continuar
```

+ de 1 caractere = *string*

Strings são cadeias ou seqüências de caracteres.

Em C, as ***strings*** são vetores de caracteres que têm como característica apresentar o caractere delimitador de fim '\0' (caractere da posição 0 da tabela ASCII – não é o dígito 0).

Importante: toda *string* é um vetor de caracteres, mas nem todo vetor de caracteres é uma *string*.

Exemplo: vetor de caracteres **vet** de 9 posições, apenas com as posições de 0 a 6 ocupadas (outras posições com "lixo"):

	0	1	2	3	4	5	6	7	8
vet	b	r	a	s	i	l	\0	?	?

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 maior 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];

Função strlen

- **Retorna o tamanho de uma string, sem contar o '\0'.**
- **Formato:** strlen(string);
- **Exemplo:**

```
#include <string.h>
int main( )
{
 char string_primeiro[40], string_segundo[40];
 system("color f1");
 printf("Forneca um texto: ");
 gets(string_primeiro);
 printf("Forneca um texto: ");
 gets(string_segundo);
 printf("\n%s tem comprimento %d\n",
       string_primeiro, strlen(string_primeiro) );
 printf("\n%s tem comprimento %d\n",
       string_segundo, strlen(string_segundo) );
 system("pause");
 return 0;
}
```

```
G:\ARRAYSUN\stringlen.exe
Forneca um texto: Brasil
Forneca um texto: Rio Grande do Sul

Brasil tem comprimento 6

Rio Grande do Sul tem comprimento 17
Pressione qualquer tecla para continu
```

Função sizeof

- Retorna o tamanho de um tipo ou uma variável.
- **Formato:** sizeof (variável);
- **Exemplo:**

```
....
int main( )
{ int i;
  char primeiro[40], segundo[12];
  printf("Forneca um texto: ");
  gets(primeiro);
  printf("Forneca um texto: ");
  gets(segundo);
  printf("\ntamanho reservado para primeiro:%d\n",sizeof(primeiro));
  printf("\ntamanho reservado para segundo:%d\n",sizeof segundo );
  printf("\ntamanho reservado para i:%d\n",sizeof i);
  printf("\ntamanho reservado para inteiros:%d\n", sizeof (int)); // aqui
precisa estar entre parênteses
  system("pause");
  return 0;
}
```

```
C:\Cora\Disciplinas\INFO
Forneca um texto: Rio Grande do Sul
Forneca um texto: Brasil

tamanho reservado para primeiro:40

tamanho reservado para segundo:12

tamanho reservado para i:4

tamanho reservado para inteiros:4
Pressione qualquer tecla para contin
```

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'.

Exemplo 3:
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.

Inicialização de *strings*

```
// Exemplos de strings:
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXIMO 20
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;
}
```

```
C:\Cora\Disciplinas\INF0120...
Nome 1 = Ana, tamanho 3 - 15
Nome 2 = Ana, tamanho 3 - 15
Nome 3 = Ana, tamanho 3 - 4

Nome 2 = Ana, tamanho 3 - 15
Nome 2 = Ana-b, tamanho 5 - 15
```

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)
```


Leitura e escrita de *strings* com *gets* e *puts*

Leitura

O *gets* permite colocar na variável todos os caracteres introduzidos, sem encerrar com o branco.

Ex.:
```
char nome_cliente[20];
gets(nome_cliente);
// se usuário digitou Maria do Socorro Silva,
//todo o nome estará armazenado em nome_cliente
```

Escrita

Permite a escrita de 1 único *string* (constante literal ou variável), mudando automaticamente de linha após a impressão.

Ex.:
```
char nome_cliente[20];
(...)
puts("Informe nome da cliente:");
gets(nome_cliente);
```


Função *gets* – getstring e *puts* - putstring

```
// Leitura de n caracteres, sem parar no branco:
//testa funcao gets
#include <stdio.h>
#include <stdlib.h>
int main( )
{
   int seguir;
   char nome[30];
   seguir = 1;
   while (seguir)
    {
     puts("\nNome:");
     gets(nome);
     if (nome[0] == '\0') // para quando sem conteúdo – enter direto
        seguir = 0;
     else
        printf("\nNome informado:%s", nome); // não pode puts:2 conteúdos
    }
   system("PAUSE");
   return 0;
}
```

```
// se usasse:
puts("Nome informado:");
puts (nome); //em  linhas diferentes...
```

C:\Cora\Disciplinas\INF012...

```
Nome:
João da Silva

Nome informado:João da Silva
Nome:
Maria do Socorro

Nome informado:Maria do Socorro
Nome:

Pressione qualquer tecla para continuar. . .
```


STRINGS

Como *string* não é um tipo em C, não se pode atribuir uma *string* a outra:

~~string1 = string2;~~

Para isso: funções para manipulação de *strings*.

Funções para manipular *strings*

- Utilização da biblioteca #include <string.h>
- Funções:
  - strcpy
  - strcat
  - strcmp

Função strcpy

- **Copia *string_origem* par**
- **Formato:**

  strcpy(string_des
- **Exemplo:**

```
...
#include <string.h>
int main( )
{
  char string_origem[10], string_destino[10];
  system("color 71");
  printf("Forneca um nome: ");
  gets(string_origem);
  strcpy(string_destino, string_origem);
  printf("\n%s\n", string_destino );
  system("pause");
  return 0;
}
```

G:\ARRAYSUNI\stringcpy.exe

```
Forneca um nome: Mariana

Mariana
Pressione qualquer tecla para
```

Função strcat

- A string_origem, sem alteração, é anexada ao final da string_destino.
- Formato: strcat(string_destino, string_origem);
- Exemplo:

```
#include <string.h>
int main( )
{
  char string_origem[20], string_destino[40];
  system("color f1");
  printf("Forneca um texto: ");
  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;
}
```

ATENÇÃO:
A string_destino deve ter tamanho suficiente para armazenar o resultado de strcat!

```
G:\ARRAYSUNI\stringcat.exe
Forneca um texto: Lindo dia!
O texto digitado foi: Lindo dia!
Pressione qualquer tecla para continuar. . .
```

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);
- Exemplo:

Lembrar que as maiúsculas vêm antes das minúsculas na tabela ASCII.

```
#include <string.h>
int main( )
{
  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 = -1
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.**

1 - Escreva um programa que lê um texto contendo até MAXIMO caracteres (consistindo), lê um caractere e informa a 1ª posição do texto onde este caractere ocorre ou que não existe tal caractere no texto, se for o caso.

```
// Exemplos de uso de funções:
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // para usar funções de manipulação de strings
#define MAXIMO 20
int main( )
{
  char caract, texto[MAXIMO + 1];
  int tamanho, i;
  system("color f1");

  // leitura, consistindo se não ultrapassa limite de caracteres:
  do
    {
      printf("Digite um texto (tamanho maximo %d): ", MAXIMO);
      gets(texto);
      tamanho = strlen(texto);
      if (tamanho>MAXIMO)
        printf("Tamanho maximo deve ser %d!\n", MAXIMO);
    }
  while ( tamanho >MAXIMO);

  // obtém caractere e procura primeira ocorrência dele no texto:
```



```
.....
 // obtém caractere e procura primeira ocorrência dele no texto:
  printf("Informe caractere a localizar: ");
  caract = getchar();
  i = 0; // posiciona no primeiro caractere
  while (texto[i] != caract && texto[i] != '\0') // analisa conteúdo da posição
  {
        i++;        // se não é o buscado nem terminou, incrementa i
  }
  if (texto[i] == caract)  // analisa conteúdo da posição i
     printf("Primeira ocorrência de %c em %d\n", caract, i);
  else
     printf("%c não encontrado! \n", caract);  // porque é '\0'

  system("pause");
  return 0;
}
```

STRINGS

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' NÃO são 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* !

2 - Escreva um programa que lê um texto contendo até MAXTXT caracteres (consistindo), lê uma palavra contendo até MAXPAL caracteres (consistindo) e informa se esta palavra existe no texto.

```
// Exemplos de uso de strings:
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // para usar funções de manipulação de strings
#define MAXTXT 40
#define MAXPAL 10
int main( )
{
 char texto[MAXTXT + 1], palavra[MAXPAL + 1],
       auxtxt[MAXTXT + 1], auxpal [MAXPAL + 1];
 int tamanho, i;
 system("color f1");

 // leitura do texto, consistindo se não ultrapassa limite de caracteres:
 do
  {
   printf("Digite um texto (tamanho maximo %d): ", MAXTXT);
   gets(texto);
   tamanho = strlen(texto);
   if (tamanho>MAXTXT)
     printf("Tamanho maximo deve ser %d!\n", MAXTXT);
  }
 while ( tamanho >MAXTXT);

// leitura da palavra:  - próximo slide -
```



```
.....
 // leitura da palavra, consistindo se não ultrapassa limite de caracteres:
 do
  {
   printf("Digite uma palavra (tamanho maximo %d): ", MAXPAL);
   gets(palavra);
   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(auxpal , strlwr(palavra));

// verifica se a palavra existe no texto:
```

Fazer para a próxima aula!