Aula 18
Strings

Roteiro

Strings

- A linguagem C não possui o tipo *string* explicitamente, mas podemos considerar um vetor de caracteres como uma *string*.
- Em C uma string é sempre terminada pelo caracter especial: '\0'
- **Portanto sempre declare uma string com um caracter a mais do que precisa!**
  - Se por exemplo estivermos trabalhando com strings de 10 caracteres:

```
char st[11];
```

Strings

- Para ler ou imprimir uma string do teclado usamos o operador especial $\%s$.

```
int main(){
   char st[80];
   int a;

   printf("\nEntre com nome:");
   scanf("%s",st);
   printf("\nEntre com idade:");
   scanf("%d",&a);
   printf("\n Digitado: %s e %d\n",st,a);
}
```

- **Note que para strings não é utilizado o** & **no comando** *scanf*.

Strings

- Para ler strings **incluindo espaços** use a opção: %[^ \n].

```
int main(){
  char st[80];
  int a;

  printf("\nEntre com nome:");
  scanf("%[^\n]",st);
  printf("\nEntre com idade:");
  scanf("%d",&a);
  printf("\n Digitado: %s e %d\n",st,a);
}
```

Inicialização de Strings

- Em algumas situações, ao criarmos uma string, pode ser útil atribuir valores já na sua criação.
- No caso de strings, podemos atribuir diretamente uma constante string para a variável.

Exemplo

char st[100] = "sim isto é possível";

Strings: Exemplos

- Ler uma string de até 80 caracteres e salvar a inversa desta em um vetor.
- Imprimir a inversa da string lida.

Strings: Exemplos

```
int main(){
  char st[81], stInv[81];
  int tam, i, j;

  printf("Entre com o string: ");
  scanf("%s",st);

  //Primeiro determinamos o tamanho da string
  tam = 0;
  while(st[tam] != '\0'  && tam < 81){
    tam++;
  }

  //Depois escrevemos os caracteres na inversa
  stInv[tam] = '\0';
  j = tam-1;
  i = 0;
  while(i<tam){
    stInv[j] = st[i];
    i++;
    j--;
  }

  printf("A inversa e: %s\n",stInv);
}
```

Strings: Exemplos

A mesma coisa mas com laço **for**:

```
int main(){
  char st1[81], stInversa[81];
  int i, j , tam;

  printf("Digite um texto (max. 80):");
  scanf("%s",st1);

  for(tam=0; (st1[tam] != '\0') && (tam < 81) ; tam++)
    ;

  stInversa[tam] = '\0';
  for(j = tam-1, i = 0 ; j >= 0 ;  j--, i++){
    stInversa[j] = st1[i];
  }

  printf("A inversa e: %s\n", stInversa);
}
```

string.h

- A biblioteca **string.h** possui várias funções úteis para se trabalhar com strings.
- Vamos apresentar algumas funções comuns:
  - **char *strcat(char *s1, const char *s2)** : Para fazer a concatenação de strings.
  - **int strcmp(const char *s1, const char *s2)** : Para fazer a comparação lexicográfica (utilizada em ordenação) de duas strings.
  - **char *strcpy(char *s1, const char *s2)** : Para fazer a cópia de strings.
  - **int strlen(const char *s1)** : Para se determinar o tamanho de uma string.

string.h

Exemplo de uso da função **strcat** para fazer concatenação de strings.

- A função recebe duas strings como parâmetro e concatena a string segundo parâmetro no final da string primeiro parâmetro.
- Deve haver espaço suficiente na primeira string, caso contrário ocorrerá um erro.

```
#include <stdio.h>
#include <string.h>

int main(){
  char s1[80]="ola ", s2[80]="turma de 102!";

  //concatena s2 no final de s1
  strcat(s1, s2);

  printf("%s\n", s1);
}
```

Saída será

```
ola turma de 102!
```

string.h

Exemplo de uso da função **strcmp** para fazer comparação de strings.

- A função recebe duas strings **s1** e **s2** como parâmetro e devolve:
  - 0 caso as duas strings sejam iguais.
  - um valor menor que 0 caso **s1** seja lexicograficamente menor que **s2**.
  - um valor maior que 0 caso **s1** seja lexicograficamente maior que **s2**.

```
#include <stdio.h>
#include <string.h>

int main(){
  char s1[80]="aab", s2[80]="aac";
  int r;
  r = strcmp(s1, s2);
  if(r < 0)
    printf("%s vem antes que %s\n", s1, s2);
  else if(r>0)
    printf("%s vem antes que %s\n", s2, s1);
  else
    printf("sao iguais\n");
}
```

Saída será

```
aab vem antes que aac
```

string.h

Exemplo de uso da função **strcpy** para fazer cópia de strings.

- A função recebe duas strings como parâmetro e copia a string segundo parâmetro na string primeiro parâmetro.

```
#include <stdio.h>
#include <string.h>

int main(){
  char s1[80], s2[80]="ola pessoal";

  strcpy(s1, s2);
  printf("%s\n", s1);
}
```

Saída será

```
ola pessoal
```

string.h

Exemplo de uso da função **strlen** para calcular o tamanho de uma string.

- A função recebe uma string como parâmetro e devolve o número de caracteres na string até o '\0'.

```
#include <stdio.h>
#include <string.h>

int main(){
  char s1[80]="ola pessoal";
  int t;

  t = strlen(s1);
  printf("%d\n", t);
}
```

Saída será

```
11
```

Processamento de Texto

- Como exemplo de funções com strings vamos implementar duas funcionalidades básicas de processadores de texto:
  1. Contar o número de palavras em um texto.
  2. Fazer a busca de uma palavra em um texto.

Processamento de Texto

Função para contar o número de palavras em uma string.

```
int numPalavras(char s[]){
  int i=0, n=0;

  while(s[i]!='\0'){
    while(s[i]==' ')
      i++;
    //no fim do laço achou o começo de uma palavra ou o fim do texto

    if(s[i]!='\0'){ //se achou uma palavra
      n++; //incrementa número de palavras
      while(s[i]!=' '  && s[i]!='\0')//passa pela palavra
           i++;
    }
  }
  return n;
}
```

Processamento de Texto

```
#include <stdio.h>
#include <string.h>
int numPalavras(char s[]);

int main(){
  char s1[200];
  printf("Digite um texto: \n");
  scanf("%[^\n]", s1);
  printf("O número de palavras é: %d\n", numPalavras(s1));
}

int numPalavras(char s[]){
  int i=0, n=0;

  while(s[i]!='\0'){
    while(s[i]==' ')
      i++;
    if(s[i]!='\0'){ //se achou uma palavra
      n++; //incrementa numero de palavras
      while(s[i]!=' '  && s[i]!='\0')//passa pela palavra
         i++;
    }
  }
  return n;
}
```

Processamento de Texto

Achar palavras em um texto.

- Fazer função que acha a posição de ocorrência de uma palavra em um texto a partir de uma posição inicial dada.
- **int achaPal(int ini, char palavra[], char texto[])**

Exemplo:

```
Texto=a tete tetete
Palavra=tete

A partir de 0, a resposta é 2.
A partir de 3, a resposta é 7.
A partir de 8, a resposta é 9.
```

Processamento de Texto

Ideia do algoritmo:

- Para cada possível posição no texto onde a palavra pode iniciar checamos se a palavra ocorre naquela posição ou não.
- Seja **ini** a posição a partir de onde se faz a busca, e **tamT** (respectivamente **tamP**) o tamanho do texto (tamanho da palavra respectivamente).
- Posições válidas no texto são de **ini** até **tamT - tamPa**.

```
int achaPal(int ini, char palavra[], char texto[]){
  int tamP = strlen(palavra), tamT = strlen(texto);
  int i;
  for(i=ini; i <= tamT - tamP; i++){
      //Checar se palavra ocorre exatamente na posição i do texto
  }
}
```

Processamento de Texto

Como testar se palavra ocorre exatamente em uma posição **i**?

- Checar se todos os caracteres da palavra são iguais do texto a partir de **i**.

```
//Checa se palavra ocorre na posição i do texto
int j=0;
while( j<tamP && palavra[j] == texto[i+j])
  j++;
if(j==tamP) //se atingiu o fim do laço porque j==tamP,
   return i;      // então palavra ocorre em i
```

Processamento de Texto

A função retorna -1 caso não haja ocorrência da palavra no texto a partir de uma posição inicial **ini**.

```
int achaPal(int ini, char palavra[], char texto[]){
  int tamP = strlen(palavra), tamT = strlen(texto);
  int i;
  for(i=ini; i <= tamT - tamP; i++){
    int j=0;
    while(j<tamP && palavra[j] == texto[i+j] )
      j++;
    if(j==tamP)
      return i;
  }
  return -1;
}
```

Processamento de Texto

```
int main(){
  char s1[200];
  printf("Digite um texto: \n");
  scanf("%[^\n]", s1);
  int ini=0;
  while(1){
    ini = achaPal(ini, "tete", s1);
    if(ini == -1)
      break;
    printf("Ocorrencia: %d\n", ini);
    ini = ini+1;
  }
}
int achaPal(int ini, char palavra[], char texto[]){
  int tamP = strlen(palavra), tamT = strlen(texto);
  int i;
  for(i=ini; i <= tamT - tamP; i++){
    int j=0;
    while(j<tamP && palavra[j] == texto[i+j])
      j++;
    if(j==tamP)
      return i;
  }
  return -1;
}
```

Exercício

- Escreva um programa que lê uma string de até 50 caracteres, e imprime “Palindromo” caso a string seja um palindromo e “Nao Palindromo” caso contrário.
- OBS: Um palindromo é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (acentos e espaços em brancos são descartados).
- Exemplo de palindromo: Saudável leva duas.

Strings: Exemplos

- Escreva uma função **void inversa(char s[])** que recebe uma string como parâmetro e deixa ela invertida, ou seja, ao final da execução da função, **s** contém a string original invertida.
- Refaça a função tal que não seja utilizado nenhum vetor adicional! Ou seja devemos computar a inversa no próprio vetor original.