String em C++

Uma das formas de se manipular cadeias de caracteres, também chamadas de **strings** em C++ é armazená-las como vetores de char. Esta é a forma tradicional utilizada pela linguagem C. Então, definimos inicialmente o tamanho do vetor e depois podemos usar comandos da linguagem para manipular estes dados. Não podemos copiar uma string para outra com o comando de atribuição =, temos comandos para executar esta operação. Para manipular este tipo de string é preciso ter certo cuidado, pois o vetor sempre tem um tamanho definido e caso façamos um acesso a um endereço fora do limite do vetor, invadiremos outras áreas de memória de maneira indevida, e portanto, poderemos fazer o programa parar de funcionar

Podemos também em C++, manipular strings através da classe String. Inicialmente veremos a manipulação de strings representadas por vetores de char.

Os caracteres são também representados através de números que geralmente têm oito bits, esses números são traduzidos na tabela ASCII de 128 caracteres, como existem inúmeras regiões no mundo com características linguísticas próprias, a tabela ASCII é estendida por um bloco de caracteres acima dos 128 mais baixos que varia de acordo com as necessidades de cada língua. A parte superior da tabela ASCII é conhecida como parte estendida e é referenciada por páginas de códigos para cada propósito linguístico, isso quer dizer que podemos ter os mesmos números significando caracteres diferentes para cada região do mundo.

Cada caractere em C++ ocupa um byte na memória. Dessa forma, 'C' é o caractere C (ocupa apenas 1 byte na memória) enquanto que, "C" é um vetor de caracteres (ocupa 2 bytes na memória onde 1 byte é reservado para o finalizador de strings). O tamanho da string deve sempre incluir o finalizador de strings ('\0'). A função do finalizador de strings é única e estritamente para definir quais são as posições preenchidas dentro de vetor de caracteres das posições que não foram preenchidas

Verificando o tamanho da string

Para uma string armazenada em um vetor de char, podemos verificar quantos elementos deste vetor estão sendo efetivamente utilizados, ou seja, o tamanho da string digitada através do comando **strlen** presente no arquivo de cabeçalho **cstring**. Usa-se a função com o vetor entre parênteses: **strlen(vetor)**.

```
//Verificando tamanho de String
#include <iostream>
#include <cstring>

using namespace std;

int main (){
 char nome[50];
 cout << "Digite um nome: " << endl;
 cin.getline(nome,50);
 cout << "O nome armazenado é :" << nome << " que tem " \
<< strlen(nome) <<  " caracteres." << endl;
 return 1;
}
```

Um exemplo da saída deste programa seria:

O nome armazenado é: José Carlos que tem 11 caracteres.

Copiando um String para Outra

O comando **strcpy** pode ser usado para copiar os valores entre duas strings. A sintaxe seria: **strcpy**(destino, origem). Podemos especificar quantos dos primeiros caracteres queremos copiar, utilizando o comando **strncpy**. A sintaxe é **strncpy**(destino, origem, x), onde o terceiro parâmetro x deve ser substituído por um número inteiro que definirá a quantidade de caracteres que será copiada, partindo do inicio da string de origem. A seguir, um exemplo.

```
// Copiando strings
int main (){
 char nome1[50], nome2[50], nome3[50];
 cout << "Digite um nome: " << endl;
 cin.getline(nome1,50);
 //strcpy(destino, origem)
 strcpy(nome2,nome1);
 strncpy(nome3,nome1,7);
 cout << "Usando comandos de copias de strings" << endl;
 cout << nome1 << endl << nome2 << endl << nome3 << endl;
 return 1;
}
```

Uma saída para este programa seria:

Usando comandos de copias de strings

Maria Tereza

Maria Tereza

Maria T

Comparando duas strings

O comando **strcmp** pode ser utilizado para comparar duas strings. A sintaxe é: **strcmp**(string1,string2)

Se as **strings forem iguais a função retorna zero,** se string1 for maior a função retorna um valor menor que zero e se string2 for maior a função retorna um valor maior que zero.

Unindo duas Strings

Podemos unir ou concatenar duas strings com o comando **strcat**, cuja sintaxe é **strcat**(destino, origem). Este comando adiciona ao final da string destino o conteúdo presente na string origem.

O comando **strncat**(destino, origem, nr_caracteres) pode ser utilizado para adicionar apenas alguns dos primeiros caracteres da string de origem.

Verificando a ocorrência de um caractere

O exemplo a seguir, percorre o vetor que armazena a string e informa em qual posição está a primeira ocorrência de um caractere.

```
int main (){
 char nome[50];
 int tam;
 char ch;
 cout << "Digite um nome: " << endl;
 cin.getline(nome,50);
 cout << "Digite uma letra: " << endl;
 cin >> ch;
 tam = strlen(nome);
 for (int i =0; i < tam; i++){
   if (nome[i]==ch){
      cout << "A primeira ocorrência do caractere a ocorre em " << i << endl;
      break;
   }
   if (i == (tam -1)){
      cout << "Não existe o caractere " << ch << " na string digitada." << endl;
   }
 }
 return 1;
}
```

Verificando a ocorrência de uma substring

```
int main(){
   char palavra[15], subs[15];
   cout << "Digite uma palavra: " << endl;
   cin.getline(palavra,25);
   cout << "Digite uma substring: " << endl;
   cin.getline(subs,25);
   if (strstr(palavra,subs))
      cout << "Existe a substring : " << subs << " em " <<palavra << endl;
   else
      cout << "Não existe a substring : " << subs << " em " <<palavra << endl;
   cout << palavra << " " << subs << endl;
   return 1;
}
```

Além dos comandos citados anteriormente, a tabela a seguir apresenta um resumo de alguns comandos que são utilizados na manipulação de strings. Estes comondos estão presentes no cabeçalho ***cstring***.

Tabela com resumo dos comandos:

Comando	Função
strlen(nome_vetor)	Informa o tamanho da string armazenada
strcpy(destino, origem)	Copia conteúdo da string origem para a destino
strncpy(destino, origem,x)	Copia os x primeiros caracteres da string origem para a destino
strcat(destino, origem)	Adiciona ao final do vetor destino o conteúdo do vetor origem
strcmp(string1,string2)	Verifica se as duas strings são iguais.
strncat(destino, origem,x)	Adiciona ao final do vetor destino os x primeiros caracteres do vetor origem
strupr(string)	Converte os caracteres da string para caixa alta ou deixa todos em maiúsculo
strlwr(string)	Converte os caracteres da string para caixa baixa ou deixa todos em minúsculas.
strset(string,caractere)	Substitui todos os caracteres de uma string pelo caractere passado como parâmetro.
tolower	Converter um caractere em minúsculo
toupper	Converte um caractere minúsculo em maiúsculo.
isalnum	Verifica se o caractere é alfanumérico
isalpha	Verificar se o caractere é uma letra do alfabeto
iscntrl	Verificar se o caractere é um caractere de controle
isdigit	Verificar se o caractere é um digito decimal
isgraph	Verifica se o caractere tem representação gráfica
islower	Verifica se o caractere é minúsculo
isprint	Verifica se o caractere é imprimível.
ispunct	Verifica se o caractere é um ponto
isspace	Verificar se o caractere é um espaço em branco
isupper	Verifica se o caractere é uma letra maiúscula
isxdigit	Verifica se o caractere é um dígito hexadecimal

Exemplo:

```
int main(){
   char ch;
   cout << "Digite uma letra: " << endl;
   cin >> ch;
   cout << (char) tolower(ch) << " " <<(char) _toupper(ch) << endl;
   if ( isalnum(ch))
      cout << "\nVoce digitou um caractere alfanumérico”;
   if ( isalpha(ch))
      cout << "\nVoce digitou um letra do alfabeto";
   if ( iscntrl(ch))
      cout << "\nVoce digitou um caractere de controle";
   if ( islower(ch))
   cout << "\nVoce digitou um caractere minúsculo";
   if ( isprint(ch))
   cout << "\nVoce digitou caractere imprimível";
   if ( ispunct(ch))
      cout << "\nVoce digitou um ponto";
   if ( isspace(ch))
      cout << "\nVoce digitou um espaço";
   if ( isupper(ch))
      cout << "\nVoce digitou um caractere maiúsculo";
   if ( isxdigit(ch))
      cout << "\nVoce digitou um caractere hexadecimal";
   return 1;
}
```