

Linguagem de Programação

Cadeias de Caracteres

ECT2303

helton.maia@ufrn.br

Introdução

Em C++, uma cadeia de caracteres é uma sequência de caracteres (letras, números, símbolos, etc.) que são armazenados em um array de caracteres do tipo `char`. Cada caractere da cadeia é armazenado em uma posição do array, começando na posição 0.

Uma cadeia de caracteres é denotada por uma sequência de caracteres entre aspas duplas, por exemplo: `"Olá mundo!"`. O último caractere da cadeia é sempre um caractere nulo `'\0'`, que indica o fim da cadeia.

Introdução

Para declarar e inicializar uma cadeia de caracteres em C++, pode-se usar a seguinte sintaxe:

```
char minhaCadeia[ ] = "Exemplo de cadeia de caracteres";
```

Nesse exemplo, a cadeia de caracteres é armazenada em um array chamado `minhaCadeia`. O tamanho do array é automaticamente definido pelo compilador com base no tamanho da cadeia de caracteres.

Definindo e Inicializando

Exemplo:

```
char str[] = "C++";
```

Obs: o caractere nulo \0 foi adicionado ao final da string, automaticamente.

Definindo e Inicializando

Outras possibilidades:

```
char str[] = "C++";
```

```
char str[4] = "C++";
```

```
char str[] = {'C', '+', '+', '\\0'};
```

```
char str[4] = {'C', '+', '+', '\\0'};
```

Definindo e Inicializando

- Assim como estudado anteriormente em arrays, é possível alocar mais memória do que realmente se deseja utilizar.

Exemplo: `char str[50] = "ECT";`

Cadeias de Caracteres - Exemplo 1

Escreva um programa no qual seja possível armazenar 50 caracteres, incluindo o delimitador de fim. Faça a leitura da entrada de dados, pedindo para o usuário digitar uma string. Por fim, imprima tudo o que foi digitado.

Cadeias de Caracteres - Exemplo 1

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      char str[50];
6
7      cout << "Entre com uma string: ";
8      cin >> str;
9      cout << "Voce digitou: " << str << endl;
10
11     return 0;
12 }
13
```


Notou algum problema no exemplo 1?

Cadeias de Caracteres - Exemplo 1 (*updated*)

Escreva um programa no qual seja possível armazenar 50 caracteres, incluindo o delimitador de fim. Faça a leitura da entrada de dados, pedindo para o usuário digitar uma *string* “frase”. Por fim, **imprima tudo o que foi digitado.**

- O operador >> considera o espaço “ ” como fim.

Cadeias de Caracteres - Exemplo 1 (*updated*)

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      char str[50];
6
7      cout << "Entre com uma string: ";
8      cin.getline(str, 50);
9
10     cout << "Voce digitou: " << str << endl;
11     return 0;
12 }
13
```

Cadeias de Caracteres - Leitura dos dados

```
1  #include <iostream>
2  | #define TAM 8
3  using namespace std;
4
5  int main(){
6      char str1[TAM];
7      char str2[TAM];
8
9      cout << "Digite a string 1: ";
10     cin.getline(str1, TAM);
11     cout << "Digita a string 2: ";
12     cin >> str2;
13
14     cout << str1 << " " << str2;
15     return 0;
16 }
17
```

String 1: casa de

'c'	'a'	's'	'a'	' '	'd'	'e'	'\0'
-----	-----	-----	-----	-----	-----	-----	------

String 2: casa de

'c'	'a'	's'	'a'	'\0'			
-----	-----	-----	-----	------	--	--	--

→ **Obs:** Para impressão (cout), não é necessário imprimir cada caractere separadamente.

Cadeias de Caracteres

Buffer de entrada em C / C ++

O que é um *buffer* ?

Cadeias de Caracteres

Buffer de entrada em C / C ++

- Em C++, um *buffer* é uma área de memória reservada para armazenar temporariamente dados antes de serem processados ou transferidos para outro local.
- Em particular, um buffer de entrada é usado para armazenar dados que foram lidos de uma entrada, como um arquivo ou o teclado, enquanto um buffer de saída é usado para armazenar dados que serão gravados em uma saída, como um arquivo ou a tela.

Cadeias de Caracteres

Buffer de entrada em C / C ++

O que é um buffer?

- Área de armazenamento temporário;
- Dispositivos de entrada/saída padrão contém um buffer de entrada/saída;
- No C/C ++, os fluxos são armazenados em buffer, por exemplo, no caso de entrada padrão, quando pressionamos uma tecla, essa informação não é enviada para o programa, e sim armazenada em buffer pelo sistema operacional até que realmente seja alocada para o programa.

Cadeias de Caracteres

Limpando o *buffer* de entrada em C / C ++

```
1  #include<iostream>
2  #define MAX 100
3
4  using namespace std;
5  int main(){
6      int num;
7      char palavra[MAX];
8
9      cout << "Entre com um inteiro: ";
10     cin >> num;
11     cin.ignore();
12     cout << "Informe uma frase: ";
13     cin.getline(palavra,MAX);
14
15     return 0;
16 }
17
```

Obs: Caso, antes da leitura de uma cadeia de caracteres um outro tipo de dado tenha sido lido (números, caracteres, ...), é necessário a utilização do comando `cin.ignore()`. Desta forma o buffer de leitura é limpo.

Manipulação de Cadeias de Caracteres (biblioteca `cstring`)

`strlen()` prototype

```
size_t strlen( const char* str );
```

Retorna a quantidade de caracteres contidos em `str`, sem contar com o caractere nulo.

Manipulação de Cadeias de Caracteres (biblioteca cstring)

Exemplo: strlen()

```
1  #include <cstring>
2  #include <iostream>
3
4  using namespace std;
5
6  int main()
7  {
8      char str1[] = "ECT";
9      char str2[] = "UFRN";
10
11     int len1 = strlen(str1);
12     int len2 = strlen(str2);
13
14     cout << "Tamanho da string 1 = " << len1 << endl;
15     cout << "Tamanho da string 2 = " << len2 << endl;
16
17     return 0;
18 }
19
```

Manipulação de Cadeias de Caracteres (biblioteca `cstring`)

`strcpy()` prototype

```
char* strcpy( char* dest, const char* src );
```

Recebe dois argumentos: ***dest*** e ***src***. Esta função copia a cadeia de caracteres de *src* para o local de memória apontado para o *dest*. O caractere nulo também é copiado.

Manipulação de Cadeias de Caracteres (biblioteca cstring)

Exemplo: strcpy()

```
1  #include <cstring>
2  #include <iostream>
3
4  using namespace std;
5
6  int main(){
7
8      char src[] = "Ola Turma!";
9      char dest[20];
10
11     strcpy(dest,src);
12     cout << dest;
13
14     return 0;
15 }
16
```

Manipulação de Cadeias de Caracteres (biblioteca cstring)

strcat() prototype

```
char* strcat( char* dest, const char* src );
```

Recebe dois argumentos: **dest** e **src**. Esta função anexa(concatena) uma cópia da cadeia de caracteres de src ao final da string dest. O caractere de terminação nulo no final de dest é substituído pelo primeiro caractere de src. A string final resultante, contém também um caractere de terminação nulo.

Manipulação de Cadeias de Caracteres (biblioteca cstring)

Exemplo: strcat()

```
1  #include <cstring>
2  #include <iostream>
3
4  using namespace std;
5
6  int main(){
7      char dest[50] = "Primeira parte,";
8      char src[50] = " segunda parte.";
9
10     strcat(dest, src);
11     cout << dest ;
12
13     return 0;
14 }
15
```

Manipulação de Cadeias de Caracteres (biblioteca `cstring`)

`strcmp()` prototype

```
int strcmp( const char* str1, const char* str2 );
```

Compara o primeiro caractere, entre `str1` e `str2`. Sendo iguais, o próximo par de caracteres é comparado e assim, o procedimento é repetido até que seja encontrada uma diferença ou o caractere `'\0'`. A função retorna zero (0), quando iguais.

Manipulação de Cadeias de Caracteres (biblioteca cstring)

Exemplo: strcmp()

```
1  #include <cstring>
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      char lhs[] = "UFRN";
8      char rhs[] = "UFRN1";
9      int result;
10
11     result = strcmp(lhs, rhs);
12
13     if(result == 0)
14         cout << "cadeias iguais" << endl;
15     else cout << "cadeias diferentes" << endl;
16
17     return 0;
18 }
```


?