Lab. 12

Cadeias de Caracteres (strings)

ECT2303 - T02 - 19.1

12.1 Declarando e inicializando uma string

Quando você declara uma string, geralmente especifica um valor inicial, como mostrado a seguir:

```
char s[100];
char chapter[] = "Aula sobre strings em C++";
char section[64] = "Strings";
char myArray[8] = {'S', 't', 'r', 'i', 'n', 'g', 's', '\0'};
```

Em C++, uma cadeia de caracteres (strings) é um vetor de caracteres terminada por zero (`'\0'`) ou `NULL`.

Não alocar espaço suficiente para armazenar o caractere nulo que encerra uma string gera um erro.

12.2 Leitura de Strings

Para ler uma string s do usuário, pode-se usar o comando `cin`, diretamente na variável do tipo string:

```
1 int main () {
2     char str [10];
3
4     cin >> str ;
5     cout << str << endl ;
6
7     return 0;
8 }
```

Na leitura com o comando `cin`, o caractere especial `'\0'` é inserido automaticamente.

Para ler strings com espaços em branco, a função **`cin.getline (myStr, size )`** deve ser utilizada. Os parãmetros da função **`cin.getline`** são:

- **`myStr`**: variável do tipo string onde texto deve ser armazenado;
- **`size`**: número máximo de caracteres a serem lidos incluindo o **`'\0'`**.

```
1  int main () {
2      char s [51];
3
4      cin.getline (s , 51) ; // le 50 caracteres
5      cout << s << endl;
6
7      return 0;
8  }
```

OBS: Usar a função **`cin.ignore()`** logo após o **`cin`** evita que o **`'\n'`** seja considerado como a entrada da string.

```
1  const int STRMAX = 31;
2  int main () {
3      char s [ STRMAX ];
4      int x ;
5
6      cout << " Informe um inteiro :\n " ;
7      cin >> x ;
8      cin.ignore () ;
9      cout << " Inteiro informado : " << x << endl ;
10     cout << " Informe uma string :\n " ;
11     cin.getline (s , STRMAX ) ;
12     cout << " String informada : " << s << endl ;
13
14     return 0;
15 }
```

12.3 Funções da biblioteca **`cstring`**

A biblioteca **`cstring`** possui algumas funções úteis para manipular dados de string:

- **`strlen(s)`**: retorna o tamanho da string `s`.

```
1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  int main()
7  {
8      char str[] = "TESTANDO A FUNCAO STRLEN";
9
10     cout <<"A string " << str <<" possui " << strlen(str) << "
       caracteres." << endl;
11
```

```
12      return 0;
13  }
```

- **strcpy(dest, orig)**: copia o conteúdo da string `orig` para a string `dest`.

```
1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  int main()
7  {
8      char str[] = "TESTANDO A FUNCAO STRCPY";
9      char copia[30];
10
11     strcpy(copia, str);
12
13     cout << copia << endl;
14
15     return 0;
16
```

- **strcat(s1, s2)**: concatena o conteúdo da string `s2` na string `s1`.

```
1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  int main()
7  {
8      char str[100] = "Origem";
9
10     strcat(str, " DESTINO");
11
12     cout << str << endl;
13
14     return 0;
15 }
```

- **strcmp(s1, s2)**: compara a string `s1` com `s2`. Retorna 0 se elas forem iguais; Retorna um número negativo se `s1` for menor do que `s2`; Retorna um número positivo se `s1` for maior do que `s2`

Exemplo 12.3.1. Implementar um programa utilizando funções da biblioteca **cstring** para:

1. Ler uma string do usuário;
2. Copiar a string lida para uma segunda string;
3. Checar se a cópia da string é igual à palavra "ECT2303", imprimindo uma mensagem conforme o caso.

```
1  int main () {
2      char str [10] , copia [10] , palavra [10] = " ECT2303 ";
3      cin >> str;
4      strcpy ( copia , str );
5      if ( strcmp ( copia , palavra ) == 0)
6          cout << " Iguais \ n ";
7      else
8          cout << " Diferentes \ n ";
9      return 0;
10 }
```

12.4 Manipulação de Caracteres

A biblioteca de manipulação de caracteres inclui várias funções que executam testes e manipulações úteis de dados to tipo caractere. Algumas dessas funções são:

- **isdigit (c)**: retorna **true** se **c** é um dígito e **false** caso contrário.
- **isalpha (c)**: retorna **true** se **c** é uma letra e **false** caso contrário.
- **isalnum (c)**: retorna **true** se **c** é uma letra ou um dígito e **false** caso contrário.
- **islower (c)**: retorna **true** se **c** é uma letra minúscula e **false** caso contrário.
- **isupper (c)**: retorna **true** se **c** é uma letra maiúscula e **false** caso contrário.
- **tolower (c)**: se **c** é uma letra maiúscula, retorna **c** como uma letra minúscula. Caso contrário, retorna o argumento inalterado.
- **toupper (c)**: se **c** é uma letra minúscula, retorna **c** como uma letra maiúscula. Caso contrário, retorna o argumento inalterado.

Quando usar funções da biblioteca de manipulação de caracteres não deixe de incluir o arquivo de cabeçalho **<cctype>**.

12.5 Exercícios de Fixação

1. Implemente as funções contidas na biblioteca `cstring`:
   (a) `int strlen(char s[]);`
   (b) `void strcpy(char dest[], char orig[]);`
   (c) `void strcat(char s1[], char s2[]);`
   (d) `int strcmp(char s1[], char s2[]).`

   A função `main` deve ler duas strings e imprimir na tela o resultado das funções.

2. Implemente uma função que receba como parâmetro de entrada uma string e como parâmetro de saída uma outra string. A função a ser implementada deve armazenar na string de saída a string de entrada na ordem inversa. A função `main` deve ler uma string e exibir na tela a string computada pela função.

   Exemplo de execução:

   ```
   Informe uma frase:
   Esta e uma frase
   Frase invertida:
   esarf amu e atsE
   ```

3. Implemente uma função que receba como parâmetro de entrada uma string e dois números inteiros `p` e `q`. A função a ser implementada deve armazenar em um parâmetro de saída a string delimitada pelos índices `p` e `q`, observando se estes números encontram-se no intervalo $[0, \texttt{n}-1]$, onde `n` é o tamanho da string de entrada. O valor `p` informa o índice inicial e `q` o final. A função `main` deve ler uma string, dois números inteiros denotando os índices e exibir na tela a string delimitada pelos índices informados utilizando uma chamada à função implementada.

   Exemplo de execução:

   ```
   -- Exemplo 1:
   Informe uma string: Tangamandapio
   Informe os indices da substring: 4 9
   Substring resultante: amanda
   -- Exemplo 2:
   Informe uma string: Tangamandapio
   Informe os indices da substring: 5 20
   Substring resultante: mandapio
   ```

4. Implemente uma função que receba como parâmetro de entrada uma string `s1`, uma string `s2` e um número inteiro `p`. A função a ser implementada deve armazenar em uma string de saída `s3` a string `s2` inserida na posição `p` de `s1`, sem remover qualquer caractere de `s1`. A função `main` deve ler duas strings, um número inteiro e exibir na tela a string resultante utilizando uma chamada à função implementada.

   Exemplo de execução:

   ```
   Informe uma string: programacao
   Informe uma string a ser inserida: codigo
   Informe a posicao: 3
   String resultante: procodigogramacao
   ```

5. Implemente uma função que receba como parâmetro de entrada uma string e como parâmetro de saída um vetor de inteiros de 26 posições. A função a ser implementada deve armazenar no vetor a contagem de cada caractere minúsculo que aparece

na string: na posição 0 deve ser armazenada a quantidade de `'a'`, na posição 1 a quantidade de `'b'` e assim por diante até a posição 25, que deve armazenar a quantidade de `'z'`. A função `main` deve ler uma string e exibir na tela quantas vezes aparece cada caractere na frase utilizando a função implementada.

Exemplo:

```
Informe uma frase:
estudos de linguagem de programacao
Contagem de caracteres:
a: 4    c: 1    d: 3    e: 4    g: 3    i: 1    l: 1    m: 2
n: 1    o: 3    p: 1    r: 2    s: 2    t: 1    u: 2
```

6. Um palíndromo é uma palavra/frase que pode ser lida tanto da esquerda para a direita quanto da direita para a esquerda. Exemplos: osso; ame o poema; subi no onibus. Implemente uma função que receba como parâmetro uma string e retorne verdadeiro caso ela seja um palíndromo ou falso caso contrário. Existem duas versões para o problema:

   (a) Considerando espaços em branco como parte da string:
   `subi no onibus não é palíndromo` ;

   (b) Desconsiderando espaços em branco como parte da string:
   `subi no onibus é palíndromo` ;

12.6 Referências Bibliográficas

1. MANZANO, J.A.; OLIVEIRA, J.F.; **Algoritmos - Lógica para Desenvolvimento de Programação**. Editora Erica.

2. ASCENCIO, A.F.G.; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores - Algoritmos, Pascal e C/C++**. 3ed. Editora Pearson.

3. DEITEL, H. M.; DEITEL, P. J. **C++ Como Programar**. 3ed. Editora Bookman.