UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO
CIC107 – PROGRAMAÇÃO DE COMPUTADORES I

14ª Aula prática

Cadeias de Caracteres (*Strings*) e Estruturas Heterogêneas

Em C e C++ é possível o tratamento de *strings* ou cadeias de caracteres. Um valor constante de string nessas linguagens é definido entre aspas duplas. O seguinte comando imprime uma *string* constante:

```
cout << "Olá Mundo !"  << endl;
```

Em se tratando de C++ temos o tipo de dados **`string`** que oferece várias facilidades para o tratamento de cadeias de carcteres. Como no exemplo abaixo:

```
#include <string>
#include <iostream>

using namespace std;

main()
{
     string curta = "Duas palavras.";
     string longa;
     longa = "Uma frase inteira está escrita aqui.";

     cout << "String Curta: " << curta << endl;
     cout << "Caracteres em Curta: " << curta.length() << endl;

     cout << "String Longa: " << longa << endl;
     cout << "Caracteres em Longa: " << longa.length() << endl;
}
```

Um tipo de dados string em C++ pode ser visto como um "vetor inteligente" de caracteres. Ele suporta as operações de vetores, como o acesso indexado e além disso possui recursos adicionais, como a mudança automática de tamanho, quando necessário. No exemplo a seguir pode-se ver como uma string suporta o acesso indexado de forma natural.

```
     ...
     string str;
     cin >> str;
     if ( str == "naturesa" )
          str[6] = 'z';
     ...
```

Note que em C++ **'z'** indica o caractere **z** e **`"z"`** indica uma string de tamanho 1 que contém o caractere z na primeira posição (posição 0).

Eventualmente precisamos lidar com tipos de dados mais complexos do que somente números ou cadeias de caracteres. A identificação de um livro, por exemplo, inclui os seguintes campos:

Título
Autor
Nr. Edição
Ano

Se quisermos armazenar esses dados em somente uma variável, em C/C++, podemos usar o recurso de `structs`. Como segue:

```
#include <string>
#include <iostream>

using namespace std;

typedef struct Livro            // Definindo um novo tipo de
dados: Livro
{
     string titulo;
     string autor;
     int edicao;
     int ano;
};

main()
{
     Livro l;     // variável l é uma estrutura do tipo Livro

     cout << "Digite os dados do livro: " << endl;
     cout << "Título: ";
     cin >> l.titulo;
     cout << "Autor: ";
     cin >> l.autor;
     cout << "Edição: ";
     cin >> l.edicao;
     cout << "Ano: ";
     cin >> l.ano;

     cout << "Referência ao livro:" << endl;
     cout << l.autor << ". " << l.titulo << ". " << l.edicao << ".
" << l.ano << endl;
}
```

Vetores de Estruturas Heterogêneas:

Do mesmo modo do que os tipos normais de C++, podemos trabalhar com vetores de estruturas heterogêneas. O exemplo abaixo mostra um programa que lê um conjunto de coordenadas no espaço bidimensional e imprime quais os pontos mais distantes.

```
#include <string>
#include <iostream>
#include <cmath>

using namespace std;
```

```
#define N_PONTOS 5

typedef struct Ponto2D    // Definicao de um noto tipo de dados:
Ponto2D
{
     double x;
     double y;
};

double distancia (double x1, double y1, double x2, double y2)
{
     return sqrt(
                    pow( x1 - x2, 2.0 )  +
                    pow( y1 - y2, 2.0 )
                 );
}

int main()
{
   Ponto2D pontos[N_PONTOS];

   for ( int i=0 ; ( i<N_PONTOS ) ; i++ )
   {
      cout << "Digite as coordenada X e Y do ponto " << i << "
(separadas por espaço): ";
      cin >> pontos[i].x;
      cin >> pontos[i].y;
   }

   double maiorDistancia = -1.0;
   int p1, p2;

   for ( int i=0 ; ( i<N_PONTOS-1 ) ; i++ )
   {
      for ( int j=i+1 ; ( j<N_PONTOS ) ; j++ )
      {
         double dist = distancia(pontos[i].x, pontos[i].y,
pontos[j].x, pontos[j].y);
         if ( dist > maiorDistancia )
         {
            maiorDistancia = dist;
            p1 = i;
            p2 = j;
         }
      }
   }
   cout << "Os pontos mais distantes são " << p1 << " e " << p2 <<
endl;
   cout << "Com distância de: " << maiorDistancia << endl;
}
```

Exercícios:

1. Faça um programa que leia uma string e um caractere isolado. Ao final, diga se a string contém o caractere dado.
2. Faça um programa que leia 2 strings. Ao final, diga se a segunda string está contida na primeira.

3. Faça uma função que dado uma string retorne a mesma escrita com os caracteres na ordem inversa. A função deve ter o seguinte cabeçalho:

```
string inversa( string str );
```

4. Escreva um programa que leia registros de dados pessoais: nome, sobrenome e idade de 5 pessoas. Ao final você deve checar para cada par de elementos desse vetor se eles correspondem a parentes. No caso positivo, imprima os dados dos parentes encontrados.