Programação

IFMG CODAAUT

Prof. Marco Antonio M. Carvalho

UFOP
Universidade Federal de Ouro Preto

PROEX
Pró-reitoria de extensão da UFOP

decom
departamento de computação

INSTITUTO FEDERAL
MINAS GERAIS

Lembretes

- **Lista de discussão**
  - Endereço:
    - programaacao@googlegroups.com
  - Solicitem acesso:
    - http://groups.google.com/group/programaacao
- **Página com material dos treinamentos**
  - http://www.decom.ufop.br/marco/extensao/obi/
- **Repositório online de problemas das edições passadas da OBI**
  - http://br.spoj.com/problems/obi/sort=-7
- **Moodle**
  - http://programaacao.net.br/login/index.php

Avisos

Na aula de hoje

- *Strings*
  - Estrutura
  - Leitura
  - Impressão
  - Processamento
- Problemas Selecionados
- Um Problema de Lógica

Strings

Estrutura

- Utilizamos objetos da classe **string**.

```
#include <string>

string str;
```

Leitura (uma palavra)

- Utilizamos o **cin** e o **operador >>**.

```
#include <iostream>
#include <string>
using namespace std;
.
.
.
cin >> str;
```

Leitura (uma linha)

- Utilizamos o método **getline** para lermos strings que possuam espaços em branco.

```
//lê o conteúdo de uma linha até encontrar o enter
getline(cin, str);

//lê o conteúdo de uma linha até encontrar o caractere A
getline(str, 'A');
```

Impressão

- Utilizamos o **cout** e o **operador <<**.

```
#include <iostream>
#include <string>
using namespace std;
.
.
.
cout << "s = “<< str <<endl;
```

Determinar o tamanho

- Utilizamos o método **length**, que retorna um número inteiro.

```
#include <iostream>
#include <string>
using namespace std;

…

 string str("hello");

 cout<<str.length();
```

Comparação Entre *Strings*

- Podemos utilizar os operadores ==, !=, < e > para compararmos strings diretamente.

```
string str("hello");
string str2("world");

if(str == str2)
   cout<<"iguais!"<<endl;
```

Concatenação de *Strings*

- Utilizamos o método **append**, ou o **operador +=**.

```
//versão que usa o método append
str2 = "hello";
str2.append(" world");
cout << str2 << endl;

//versão que usa o operador +=
str3+="hello ";
str3+="world";
cout << str3 << endl;
```

Encontrar uma *Substring*

- Uma *substring* é uma *string* que pertence a outra. Ex.: "Federal" e "Instituto Federal"
- Utilizamos o método **find**, que retorna o índice do primeiro caractere da primeira ocorrência da substring.

```
int pos = str2.find(substr2);
//npos indica o final de qualquer string
if (pos != string::npos)
      cout << pos - 1 << endl;
```

Primeira Ocorrência de Um Caractere (Dentre Um Conjunto)

- O método **find_first_of** retorna o índice da primeira ocorrência de um caractere de um conjunto específico.

```
string str2 = "quinze de maio de 1984";
cout<<str2.find_first_of("0123456789")<<endl;
```

Última Ocorrência de Um Caractere (Dentre Um Conjunto)

- O método **find_last_of** retorna o índice da última ocorrência de um caractere de um conjunto específico.

```
//procura por uma string, ao inves de um caractere
cout<<str2.find_last_of("d")<<endl;
```

Editar/Examinar Caracteres

- Um objeto *string* pode ser tratado como um vetor do tipo **char**
  - Para isto, usamos o operador de índice **[]**.
- Desta forma, podemos aplicar operações da biblioteca **cctype** sobre os caracteres isoladamente.

Editar/Examinar Caracteres

```
#include <cctype>
...
for (int i = 0; i<str.length(); i++)
   str[i] = toupper(str[i]);
// ou tolower(ch),
// isalpha(ch), isdigit(ch)...

cout<<str;
```

Ordenação

- Quando ordenamos strings, podemos estar interessados em duas coisas:
  - Ordenar os caracteres de uma mesma *string*
    - "ouro" se torna "ooru".
  - Ordenar diferentes *strings* em um vetor, como em um dicionário
    - vet[0] = "Proetiosum", vet[1] = "Tamen", vet[2] = "Nigrum" se torna
    - vet[0] = "Nigrum", vet[1] = "Proetiosum", vet[2] = "Tamen".

Ordenação de Caracteres de uma *String*

```
#include <algorithm>
#include <string>
.
.
.
//ordem crescente
sort(str2.begin(), str2.end());
```

- Utilize o método **resize** para redimensionar uma string de tamanho pré-definido.

Ordenação de *Strings*

```
#include <algorithm>
#include <string>
#include <vector>

vector<string> S;
// assumindo que S não é vazio
// ordem crescente
sort(S.begin(), S.end());
```

- Utilize o método **resize** para redimensionar um *vector* de tamanho pré-definido.

Outras Possibilidades…

- Encontrar a primeira ocorrência de um caractere que não pertença a uma *string*
  - **find_first_not_of**().
- Encontrar a última ocorrência de um caractere que não pertença a uma *string*
  - **find_last_not_of**().
- Encontrar a quantidade de caracteres de uma string que ocorram contiguamente no início de outra *string*
  - **strspn**().

Casamento de Padrões

Casamento de Padrões

- Dada uma *string* padrão P, ela pode ser encontrada na *string* maior T?
  - Não codifiquem a versão gulosa (ingênua).
  - Solução mais fácil: método **find**
  - Ou ainda, algoritmos avançados
    - KMP (Knuth, Morris, Pratt);
    - Vetor de sufixos.

Problemas Selecionados

Problemas Selecionados

- http://br.spoj.com/problems/PAR/
- http://br.spoj.com/problems/CHAMADA1/
- http://br.spoj.com/problems/AUTO08/

Um Problema de Lógica

Um Problema de Lógica

- Três amigos vão a um restaurante, e a conta dá 25 reais
  - Cada um paga com uma nota de 10 reais;
  - Como o troco não pode ser dividido por 3, o garçom surrupia dois reais e devolve um real para cada;
  - No final, cada um pagou 9 reais (nota de R$10 e R$1 de troco)
    - 3x9,00 = 27,00;
  - O garçom surrupiou R$2,00
    - R$27,00+R$2,00= R$29,00
- Onde está o R$1,00 restante?

Perguntas?