INF333 - Programacao Competitiva
Strings

Profs. Andre Gustavo, Salles Magalhaes

Strings C++ (#include <string>) vs Strings C

- Em geral, prefiro as strings do C++
- São basicamente vector<char>
  - Menos chance de erro
  - Uso bem mais fácil
- Infelizmente algumas funções das “strings C” não estão disponiveis para strings C++ (é possível misturar as duas!)
- Exemplos de construcao de strings, concatenacao, c_str() (...const char *), etc.

Stringstream

- Stringstream : permite criar um “stream” bidirecional a partir de uma string (ou simplesmente “jogando dados” no stream).
  - Exemplos:
    - Converter string de número em um número
    - Pegar uma string correspondente ao conteudo de um stringstream
    - Criar strings no formato: arquivo_1_final.txt, arquivo_2_final.txt, ….
    - Criar uma stringstream a partir de uma string (ex: string com 3 numeros)
    - Ler linhas e criar um vector com cada palavra de cada linha.

Exemplos de funcoes uteis para strings (estilo C++)

- #include <string>
- Parte do material retirado de http://www.cplusplus.com/reference/string/string/

Iteradores em strings do C++

- Strings possuem iteradores, begin(), end(), rbegin(), rend(), podem ser atravessadas com o range-based for do C++11.

```
string str ("Test string");
string::iterator it;
for ( it=str.begin() ; it < str.end(); it++ )
     cout << *it;
cout << “\n”;
for(char &c:str) c = toupper(c); //converte tudo para maiuscula
for(const char c:str) cout << c;
```

```
#inclue <cctype>

toupper(char)
tolower(char)
islower(char)
….
```

```
Test string
TEST STRING
```

string::compare

- str.compare(beg,len,str2): retorna 0 se str for igual a str2, <0 se for “menor” (lexicograficamente) e >0 se for “maior”. Os parâmetros opcionais beg e len comparam str2 considerando apenas os len (ou menos, se ultrapassar o tamanho) primeiro caracteres após a posição beg.

```
string str1 ("green apple");
string str2 ("red apple");
if (str1.compare(str2) != 0) cout << str1 << " is not " << str2 << "\n";
if (str1.compare(6,5,"apple") == 0) cout << "still, " << str1 << " is an apple\n";
if (str2.compare(str2.size()-5,5,"apple") == 0) cout << "and " << str2 << " is also an apple\n";
```

```
green apple is not red apple still,
green apple is an apple
and red apple is also an apple
```

Operadores

- Os operadores == , !=, > e < tambem podem ser utilizados (mas não permitem limitar o espaco de comparacao)

“teste”==”teste” (true)
“Teste”!=”teste” (true)
“Capivara”<”Jacare” (true)
“AAAAAAAAAAAAAAAAAAAAAA” < “Z” (true)
“222”<”9” (true)

string::find_first_of

- A posição (ou string::npos, se não achar) para a primeira ocorrencia para um dos caracteres passados como parametro.

```
...
string str("Replace the vowels in this sentence by asterisks.");
size_t found;
found = str.find_first_of("aeiou");
while (found != string::npos) {
        str[found] = '*';
        found = str.find_first_of("aeiou",found+1); //recomeca a busca a partir do proximo caractere
}
cout << str << endl;
```

```
R*pl*c* th* v*w*ls *n th*s s*nt*nc* by *st*r*sks.
```

string::find_first_not_of

- A posição (ou string::npos, se não achar) para a primeira ocorrencia para um caractere que não casa com um dos passados como parametro.

```
...
string str ("look for non-alphabetic characters...");
size_t found;
found = str.find_first_not_of("abcdefghijklmnopqrstuvwxyz ");
if (found != string::npos) {
    cout << "First non-alphabetic character is " << str[found]; cout << " at position " << int(found) <<
endl;
}
```

```
First non-alphabetic character is - at position 12
```

string::replace

- Varias versoes da função replace (para substituir parte de uma string por outra)

```
...
string base="this is a test string.";
string str2="n example";
string str3="sample phrase";
// Using positions:                    0123456789*123456789*12345
string str=base;                   // "this is a test string."
str.replace(9,5,str2);             // "this is an example string."
str.replace(19,6,str3,7,6);        // "this is an example phrase."
str.replace(8,10,"just all",6);    // "this is just a phrase."
str.replace(8,6,"a short");        // "this is a short phrase."
str.replace(22,1,3,'!');           // "this is a short phrase!!!"
```

string::replace

- Varias versoes da função replace (para substituir parte de uma string por outra)

```
string base="this is a short phrase!!!";
string str3="sample phrase";
string str4="useful.";
String str = "this is a short phrase!!!";

// Using iterators:                        0123456789*123456789*
string::iterator it = str.begin();   // ^
str.replace(it,str.end()-3,str3);    // "sample phrase!!!"
str.replace(it,it+6,"replace it",7); // "replace phrase!!!"
it+=8;                               //          ^
str.replace(it,it+6,"is cool");      // "replace is cool!!!"
str.replace(it+4,str.end()-4,4,'o'); // "replace is cooool!!!" it+=3; // ^
str.replace(it,str.end(),str4.begin(),str4.end()); // "replace is useful."
```

string::substr

- Varias versoes da função substr (retorna a substring de uma string)

```
string str="We think in generalities, but we live in details.";
// quoting Alfred N. Whitehead
string str2, str3;
size_t pos;

str2 = str.substr (12,12); // "generalities"
pos = str.find("live"); // posicao de "live" em str
str3 = str.substr (pos); // pega de "live" até o fim
cout << str2 << ' ' << str3 << endl;
```

```
generalities live in details.
```

string::insert

- Varias versoes da função insert

```
string str="to be question";
string str2="the ";
string str3="or not to be";
string::iterator it;
str.insert(6,str2); // to be (the )question
str.insert(6,str3,3,4); // to be (not )the question
str.insert(10,"that is cool",8); // to be not (that is )the question
str.insert(10,"to be "); // to be not (to be )that is the question
str.insert(15,1,':'); // to be not to be(:) that is the question it =
str.insert(str.begin()+5,','); // to be(,) not to be: that is the question
str.insert (str.end(),3,'.'); // to be, not to be: that is the question(...)
str.insert (it+2,str3.begin(),str3.begin()+3); // (or )
cout << str << endl;
```

to be, or not to be: that is the question...

string::erase

- Varias versoes da função erase (para remover parte de uma string)

```
string str ("This is an example phrase.");
string::iterator it;
str.erase (10,8);
cout << str << endl; // "This is an phrase."
it=str.begin()+9;
str.erase (it); //remove um caractere...
cout << str << endl; // "This is a phrase."
str.erase (str.begin()+5, str.end()-7);
cout << str << endl; // "This phrase."
```

Exemplo de problema

Empresas tem mudado de nome com mais frequência ultimamente: umas se juntam, outras são compradas, e há ainda as que mudam de nome por questão de marketing. Um problema é saber o nome atual de uma empresa ao ler documentos antigos.

Sua empresa, Digiscam (antes Algorist Technologies), te passou a tarefa de manter um banco de dados com as mudanças de nome e fazer as substituições apropriadas em alguns documentos. O programa terá como entrada uma lista de mudanças de nome e várias linhas do texto a ser corrigido. Devem ser substituídas apenas ocorrências exatas dos nomes. Haverá no máximo 100 mudanças de nome, e no máximo 1000 caracteres por linha.

Entrada:
4
"Anderson Consulting" to "Accenture"
"Enron" to "Dynegy"
"DEC" to "Compaq"
"TWA" to "American"
5
Anderson Accounting begat Anderson Consulting, which
offered advice to Enron before it DECLARED bankruptcy,
which made Anderson
Consulting quite happy it changed its name
in the first place!

Saida:
Anderson Accounting begat Accenture, which
offered advice to Dynegy before it CompaqLARED bankruptcy,
which made Anderson
Consulting quite happy it changed its name
in the first place!!

Algum possivel caso dificil nesta questao?

- Primeiro pegamos as trocas de strings (removendo “)
- Para facilitar, trocamos as aspas por “\n” e usamos o getline para pegar as partes importantes.

```
int nt;

cin >> nt;

vector<pair<string,string> > trocas(nt);
cin.ignore();
for(int i=0;i<nt;i++) {
    string temp;
    getline(cin, temp);
    std::replace( temp.begin(), temp.end(), '\"', '\n'); // <algorithm>
    stringstream ss(temp);

    getline(ss,temp); //primeira linha vazia...
    getline(ss,trocas[i].first);
    getline(ss,temp); //linha com "to"
    getline(ss,trocas[i].second);
}
```

- Para cada linha a ser alterada, procure cada uma das palavras a serem trocadas e a troque.

```
    int nl;
    cin >> nl;
    cin.ignore(); //remove o \n
    for(int i=0;i<nl;i++) {
        string linha;
        getline(cin, linha);
        for(const auto &troca:trocas) {
            auto it = linha.find(troca.first);//encontre a chave p/ser
trocada
            while(it!=string::npos) {
                linha.replace(it, troca.first.size(),troca.second);
                it = it + troca.second.size(); //para continuarmos  após….
                it = linha.find(troca.first,it);
            }
        }
        cout << linha << "\n";
    }
```

Mais assuntos sobre strings

- Infelizmente algumas funções das “strings C” não estão disponiveis para strings C++. Exemplo: strtok (cstring)

```
char str[] ="- This, a sample string.";
char * pch;
pch = strtok (str," ,.-");
while (pch != NULL) {
    cout << pch << endl;
    pch = strtok (NULL, " ,.-");
}
```

```
This
a
sample
string
```

Mais assuntos sobre strings

- Um tema que as vezes cai na maratona e’ algoritmos para pesquisa eficiente em strings (ex: KMP, RK, BM, etc).

- Uma busca em forca bruta de uma string de tamanho K em uma de tamanho N teria complexidade: O(NK)

- KMP, por exemplo, tem complexidade O(K)