Curso de C++ para Resolução de Problemas

Projeto de Ensino e Aprendizagem de Programação para Olimpíadas

Strings

Strings

Strings são objetos que representam sequências de caracteres. Enquanto que na linguagem C era necessário declarar um vetor de caracteres, por exemplo `char texto[50]`, e utilizar a biblioteca `<string.h>` para sua manipulação, em C++ utilizamos a biblioteca `<string>` e o objeto `string` que esta implementa, fazendo uso de todos os seus recursos de maneira muito mais fácil e prática, como veremos a seguir.

- Funções Membro

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

using namespace std;

int main (){
    string s0 ("String Inicial");
    cout << s0 << endl;
    string s1;
    s1 = "Nova String";
    cout << s1 << " de tamanho " << s1.length() << endl;
    for(int i=0; i<(int)s1.size(); ++i)
        cout << s1[i];
    cout << endl;
    s0.clear();
    if(s0.empty()) cout << "string s0 está vazia" << endl;
    if(!s1.empty()) cout << "string s1 não está vazia" << endl;
    return 0;
}
```

Para declarar uma string vazia fazemos `string s1`. Já se estamos interessados em também inicializar o conteúdo da string em sua declaração, usamos o **construtor** do objeto string da forma `string s0 ("String Inicial")`. Para saber o tamanho de uma determinada string utilizamos a função **`length`** ou a função **`size`**, ambas produzem o mesmo resultado, isto é, retornam o tamanho da string como um número inteiro sem sinal, por isso a conversão `(int)s1.size()` foi utilizada dentro da estrutura de repetição `for`. Através do operador **`[]`** é possível caractere por caractere da string, do mesmo jeito que fazíamos para um vetor de caracteres na linguagem C. Já as funções **`clear`** e **`empty`** servem para limpar o conteúdo de um string e verificar se uma string é vazia, respectivamente. Naturalmente, a função **`empty`** retorna um valor **booleano** (`true` ou `false`), se comportando como uma expressão lógica por si só.

- **Sobrecarga de Funções Não-Membro**

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

using namespace std;

int main (){
    string nome("Charlie");
    string espaco(1,' ');
    string sobrenome("Brown");
    string nomeCompleto = nome + espaco + sobrenome;
    cout << nomeCompleto << endl;
    if(nome>sobrenome)
           cout << nome << " vem depois de " << sobrenome << " na ordem
alfabética" << endl;
    if(nome=="Charlie")
        cout << "Os nomes são iguais" << endl;
    string outroSobrenome("Brownie");
    if(sobrenome!=outroSobrenome){
           cout << sobrenome << " é diferente de " << outroSobrenome <<
endl;
        swap(sobrenome,outroSobrenome);
        nomeCompleto = nome + espaco + sobrenome;
    }
    cout << "Novo nome: " << nomeCompleto << endl;
    string frase;
    getline(cin,frase);
    cout << frase << endl;
    return 0;
}
```

Para **concatenar** strings simplesmente fazemos uso do operador **+**, assim acrescentando o conteúdo de uma ao final de outra. Também podemos **comparar** lexicograficamente, isto é, em ordem alfabética, strings por meio dos operadores relacionais **>**, **>=**, **<**, **<=**, **==**, **!=**. A função **swap**

troca o conteúdo de duas strings, assim como mostrado no exemplo `swap(sobrenome,outroSobrenome)`. Já em `getline(cin,frase)` usamos a função **`getline`** para atribuir o conteúdo de uma linha do *buffer* de entrada padrão `cin` para a string `frase`.

- **Funções Úteis**

A biblioteca `<cctype>` da linguagem C nos oferece algumas funções úteis quando estamos lidando com strings e manipulação de caracteres, como podemos ver a seguir.

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

using namespace std;

int main (){
      string exemplo ("10 pessoas já me disseram que o curso de C++ é
bastante interessante.");
    int digito, alfabetico, maiusculo, minusculo;
    digito = alfabetico = maiusculo = minusculo = 0;
    for(int i=0; exemplo[i]!='\0'; i++){
        if(isdigit(exemplo[i]))
            digito++;
        else if(isalpha(exemplo[i])){
            alfabetico++;
            if(isupper(exemplo[i]))
                maiusculo++;
            else if(islower(exemplo[i]))
                minusculo++;
        }
    }
    cout << "Caracteres numéricos: " << digito << endl;
    cout << "Caracteres alfabéticos: " << alfabetico << endl;
    cout << "Caracteres alfabéticos maiúsculos: " << maiusculo << endl;
    cout << "Caracteres alfabéticos minúsculos: " << minusculo << endl;
    string fim ("EnFiMEstEExEmPlOAcAbOu");
    cout << fim << endl;
    for(int i=0; i<(int)fim.length(); i++)
        fim[i] = toupper(fim[i]);
    cout << fim << endl;
    for(int i=0; i<(int)fim.size(); i++)
        fim[i] = tolower(fim[i]);
    cout << fim << endl;
    return 0;
}
```

Para saber se um caractere é numérico usamos **`isdigit`**. Para verificar se um caractere é alfabético utilizamos **`isalpha`**, assim como também podemos usar **`isupper`** e **`islower`** para verificar se um caractere é maiúsculo ou minúsculo, respectivamente. Já as funções **`toupper`** e **`tolower`** convertem determinado caractere para maiúsculo e minúsculo, respectivamente.

PROGRAMATHON

Projeto de Ensino e Aprendizagem de Programação para Olimpíadas

EQUIPE

Reginaldo M. Kuroshu (Coordenador Geral do Projeto)
Alexandre Hild Aono
Bruno Bernardo de Moura
Danilo Gustavo Hansen Laboissiere
Diogo Augusto Hansen Laboissiere
Lucas de Alencar Barbosa
Marcos Castro de Souza
Nathan de Melo Cruz
Rodrigo de Farias Ramires
Thauany Moedano
Victor de Sá Nunes
Willian da Silva Zocolau

APOIO

Pró-Reitoria de Extensão (PROEX-UNIFESP)

REALIZAÇÃO

Instituto de Ciência e Tecnologia (ICT-UNIFESP)

CONTATO

programathon.unifesp@gmail.com

PARA MAIS INFORMAÇÕES ACESSE

programathon-unifesp.github.io