ESTRUTURA DE DADOS - 03A - 2022.2

Entrada e Saída de dados

Entrada e Saída

A biblioteca padrão de entrada e saída do C++ é a iostream. Ela deverá ser incluída no início de todo código em C++. Para podermos usá-la deveremos colocar a linha de código: `#include <iostream>`

O comando `using namespace std` é utilizado para que não se repita o std em todos os comandos padrões da linguagem.

A estrutura de comunicação com o meio externo em modo texto é composta por um conjunto de objetos. Estas, em conjunto com operadores e funções de formatação possibilitam uma forma de comunicação mais intuitiva. Devido à abstração de elementos do mundo real por recursos da orientação a objetos, a forma de entender o código torna-se mais natural.

Na biblioteca **iostream**, temos os seguintes objetos:

- **cin**: Este objeto fornece entrada de dados "bufferizada" através do dispositivo de entrada padrão, que geralmente é o teclado;
- **cout:** Este objeto fornece saída de dados "bufferizada" através do dispositivo de saída padrão, que geralmente é o teclado;
- **cerr:** Este objeto fornece saída de dados ***não*** "bufferizada" para o dispositivo de erro padrão, que é inicialmente definido para a tela.

Buffer

Para entendermos um pouco mais sobre buffer, se faz necessário recordar um pouco sobre o funcionamento da memória e suas operações relacionadas ao buffer.

Bufferização é um meio de sincronização entre dispositivos de velocidades diferentes, tais como memória e dispositivos de armazenamento mecânicos, como discos magnéticos. Para evitar que as operações do dispositivo mais lento interfiram no desempenho do programa pode-se fazer com que os dados sejam colocados em uma memória mais rápida e depois sejam enviadas ao dispositivo mais lento a medida que ele tenha disponibilidade para recebê-los, desta forma temos os seguintes modos de escrita em dispositivos de saída:

- **unbuffered:** significa que qualquer mensagem ou dados serão escritos imediatamente. É o caso da escrita no dispositivo cerr;
- **buffered:** significa que os dados serão mantidos num buffer de memória até que o dispositivo de destino solicite, ou que um comando de descarregamento seja executado, ou quando o buffer estiver cheio. O problema é que se o programa é interrompido antes do buffer ser escrito esses dados são perdidos.

Objeto cout

O objeto **cout** representa o **stream de saída** no C++. Este stream é uma espécie de sequência (fluxo) de dados a serem impressos na tela.
Para realizar a impressão, usa-se o "operador de inserção" que "insere" dados dentro do stream.

<< Operador de Inserção

O operador << sobrecarregado executa a saída (imprime na tela) com streams em C++. O objeto cout é usado em conjunto com ele para a impressão de dados.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Imprimindo o famoso HELLO WORLD!!!\n";

  // imprimindo uma linha usando múltiplos comandos
  cout << "Teste com ";
  cout << "dois couts\n";

  // usando o manipulador endl
  // endl gera um caractere nova linha, e também descarrega o buffer de saída
  cout << "Escrevendo uma linha..." << endl;

  cout << "Mais uma vez...\n";
  cout << flush;    // agora apenas esvaziando o buffer de saída, sem gerar nova linha

  return 0;
}
```

O operador << sempre retorna uma referência ao objeto **cout**. Desta forma, é possível encadear vários cout, permitindo uma combinação de constantes e variáveis, como mostra o exemplo abaixo:

```
int a = 10;
int b = 12;
// imprime uma string e o resultado da soma entre as variáveis a e b
cout << "Encadeando saídas: Somandos dois números " << a + b << endl;
```

Então podemos enviar constantes literais e variáveis separadas pelo operador de inserção.

Uma característica muito importante do C++, presente nas instruções logo acima, é o polimorfismo notável na operação de apresentação dos dados na saída; Note que os tipos de dados que são passados para o cout são diversos, ou seja, não importa qual o tipo de dado que será entregue ao cout, de alguma maneira ele sempre formatará de uma maneira legível na tela do monitor

---

Objeto cin

O objeto **cin** representa o **stream de entrada** no C++. Ele realiza a leitura de um seqüência de dados, sem espaços e sem tabulações, vindas do teclado. Para coletar estes dados armazenados, usa-se o "operador de extração" que "extrai" dados do stream.

Operador de extração >>

O operador >> sobrecarregado executa a entrada com streams em C++, usando o comando cin para aquisição de dados. Variáveis podem ser usadas para o armazenamento das informações.

```
#include <iostream>
using namespace std;
int main()
{
  int Num1;
  int Num2;
  cout << "Lendo o primeiro número...: ";
  cin >> Num1;
  cout << endl;
  cout << "Lendo o segundo número....: ";
  cin >> Num2;
  cout << endl;
```

```
  return 0;
}
```

Da mesma forma que o operador de inserção, >> sempre retorna uma referência ao objeto cin, permitindo o encadeamento de vários cin. Exemplo:

```
float c;
float d;
cin >> c >> d;    // entra com um float, pressiona <ENTER>, entra com outro float,
pressiona <ENTER>
```

Entrada de Strings

Em determinadas ocasiões, deseja-se coletar dados que contenham strings com tabulações, espaços em branco e/ou novas linhas. Frases são exemplos onde este tipo de situação ocorre. No entanto, o operador >> sobrecarregado ignora tais caracteres.

Para englobar essas situações, C++ oferece o uso da função-membro **getline**. Esta função remove o delimitador do stream (isto é, lê o caractere e o descarta) e armazena o mesmo em uma variável definida pelo usuário.

Abaixo, um exemplo de sua utilização.

```
#include <iostream>
using namespace std;
int main()
{
  const TAM = 80;
  char guarda[TAM];
  cout << "Digite uma frase com espaços: " << endl;
  cin.getline(guarda, TAM);
  cout << "A frase digitada é: \n" << guarda << endl;
  return 0;
}
```

Agora um outro exemplo, usando o tipo de dado std::string:

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

int main()
{
    string s1, s2;

    cout << "Ola'!" << endl;
    getline(cin, s1);
    cout << "String lida 1:" << s1 << "*" << endl;
    getline(cin, s2);
    cout << "String lida 2:" << s2 << "*" << endl;

    getline(cin, s1, '&'); // para a leitura no '&'
    // note que o caracter '&' NÃO vai para 's1'
    cout << "String lida 1:" << s1 << "*" << endl;

    return 0;
}
```

Última atualização: terça, 4 mai 2021, 09:52

◄ Variáveis e operadores aritméticos

Seguir para...

Manipulando a saída com a biblioteca iomanip ►

Av. José de Freitas Queiroz, 5003
Cedro – Quixadá – Ceará CEP: 63902-580
Secretaria do Campus: (88) 3411-9422

Obter o aplicativo para dispositivos móveis