Curso de C++ para Resolução de Problemas

Projeto de Ensino e Aprendizagem de Programação para Olimpíadas

Entrada e Saída

- Biblioteca *iostream*
- Espaço de nomes padrão *namespace std*
- *cout*
  - Imprimir frases e constantes
  - Imprimir variáveis
  - Imprimir com precisão
- *cin*
  - Ler entrada padrão
  - Ler múltiplas entradas

Biblioteca *iostream*

```
#include <iostream>
```

Bibliotecas são coleções de classes, funções e variáveis escritas na própria linguagem para facilitar o desenvolvimento de aplicações. Assim, para utilizar as funções de entrada e saída padrão de C++ é necessário incluir a biblioteca `iostream`.

Espaço de nomes padrão *namespace std*

```
using namespace std;
```

Entidades nomeadas, tais como variáveis, funções e tipos compostos de dados precisam ser declarados antes de serem usados em C++. É justamente através dos espaços de nomes que realizamos essas declarações. Portanto, para fazer uso dos principais recursos da linguagem definimos a utilização do **espaço de nomes padrão**.

● *COUT*

- **IMPRIMIR FRASES E CONSTANTES**

```
cout << "Imprimindo o classico Hello World!\n";
cout << "Imprimindo uma linha ";
cout << "utilizando dois couts" << endl;
```

Assim como na linguagem C, definimos o texto a ser impresso entre aspas duplas. Em seguida, redirecionamos esse conteúdo para a saída padrão **cout** utilizando o operador **<<** e finalizamos o comando com o **;**. O operador **endl** tem o mesmo efeito de **\n**, isto é, pular uma linha.

```
cout << "Imprimindo valor inteiro do numero pi: " << 3 << endl;
cout << 3.14 << " corresponde ao valor real com duas casas decimais do numero pi" << endl;
```

Para imprimir constantes o procedimento é semelhante. Lembre-se que o conteúdo a ser exibido na saída padrão mantém a mesma ordem de redirecionamento adotada no código, isto é, da esquerda para a direita.

- **IMPRIMIR VARIÁVEIS**

```
int a = 10, b = 12;
cout << "Somando " << a << " mais " << b;
cout << " temos " << a+b << " como resultado" << endl;
```

Também é possível imprimir o conteúdo de variáveis e/ou realizar operações, sejam lógicas ou aritméticas, no próprio comando de saída.

- **IMPRIMIR COM PRECISÃO**

```
#include <iomanip>
```

Primeiramente, devemos incluir esta biblioteca para conseguir manipular os especificadores de formato da saída padrão de C++.

```
double pi = 3.14159265;
cout << fixed << setprecision(2);
cout << "Valor de pi " << pi << " com duas casas decimais" << endl;
cout << fixed << setprecision(6);
cout << "Valor de pi " << pi << " com seis casas decimais" << endl;
```

Para definir a precisão em termos da quantidade de casas decimais a ser utilizada na impressão usamos o especificador de formato **fixed** junto ao **setprecision**, passando como parâmetro a quantidade desejada de dígitos decimais de precisão.

● *CIN*

- **LER ENTRADA PADRÃO**

```
float x, y;
cout << "Informe valores reais de x e y: ";
cin >> x >> y;
```

Para ler da entrada padrão e atribuir os conteúdos respectivos às variáveis, redirecionamos o fluxo de dados de **`cin`** para x e y usando o operador **`>>`**. A ordem de precedência de atribuição de conteúdo às variáveis mantém-se igual a ordem estabelecida no código (da esquerda para a direita), ou seja, primeiro atribuindo valor a variável x e depois a variável y.

```
const int tamanho = 50;
char texto[tamanho];
cout << "Digite uma palavra: ";
cin >> texto;
```

Para ler strings (cadeias de caracteres) podemos atribuir o conteúdo da entrada padrão **`cin`** para a variável `texto` usando **`>>`** diretamente, assim ignorando todo o conteúdo informado após encontrar o primeiro espaço em branco.

```
cout << "Digite uma frase: ";
cin.getline(texto,tamanho);
cout << "Digite uma frase com no maximo 9 caracteres: ";
cin.getline(texto,10);
```

Outra opção é usar **`cin.getline`**, especificando como parâmetros a cadeia de caracteres e o tamanho da string a ser lida da entrada padrão, que pode ser tanto exatamente o tamanho da variável `texto` definida como pode ser qualquer quantidade de caracteres inferior a esse tamanho. Lembre-se de que o `'\0'` é utilizado internamente para identificar o final do conteúdo em uma cadeia de caracteres, logo a quantidade de posições que podem ser ocupadas neste caso é sempre igual ao `tamanho-1`.

- **LER MÚLTIPLAS ENTRADAS**

```
int n;
while(cin >> n){}
```

Para ocasiões em que é necessário realizar a leitura de dados da entrada padrão até que se chegue ao fim do arquivo (EOF) ou em que é preciso processar vários casos de teste e não se sabe a quantidade exata deles, fazemos a leitura dentro de um laço de repetição **`while`**. Desse modo, o procedimento `cin >> n` retorna `true` (verdadeiro) enquanto houver informação a ser lida da entrada padrão, garantindo que chegaremos até o final do arquivo (EOF) ou que todos os casos de teste serão executados.

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