Exemplo de documento em docbook

COLLABORATORS			
	TITLE : Exemplo de documento em docbook		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Agostinho Brito	4 de dezembro de 2013	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Sumário

Lista de Figuras

Prefácio

Esta mera introdução à programação em linguagem C++ foi desenvolvida com foco no ensino de orientação a objetos em cursos de engenharia. Tais cursos geralmente demandam apenas conhecimentos básicos da linguagem, dadas as limitações de tempo impostas nos currículos e a carga de outras disciplinas presentes nos currículos. Empestar de conteúdos mais avançados em programação a rotina de futuros engenheiros pode sobrecarregar a audiência e colocar o público para correr.

Serão apresentados ao leitor conceitos rudimentares de orientação a objetos em C++, construtores e destrutores, alocação dinâmica de memória, herança, polimorfismo, classes abstratas, tratamento de exceções, sobrecarga de operadores e de funções. Apresentamos também alguns recursos da biblioteca padrão de gabaritos STL (www.sgi.com/tech/stl), uma espécie de canivete suíco em algoritmos e estruturas de dados. Acreditamos que tais conceitos são razoáveis para alunos de engenharia que desejam se aventurar nos meandros da programação em C++.

Os exemplos apresentados no texto foram testados com sucesso em um sistema Linux.

Capítulo 1

Introdução

Desde que começaram a se espalhar, as linguagens de programação vêm acompanhando de perto a evolução do *hardware* em que operam. As primeiras linguagens que surgiram por aí foram projetadas para dispositivos bem mais modestos que o seu telefone celular. Em verdade, possuíam limitações de memória e velocidade nos sistemas da época. A arquitetura dos antigos processadores forçava os pioneiros a realizar verdadeiros malabarismos para implementar programas que hoje conseguimos desenvolver com grande facilidade em equipamentos baratos comprados na loja da esquina. Para leitores curiosos e com tempo livre, vale a pena gastar alguns minutos lendo sobre a História da programação e ver as dificuldades enfrentadas pelos nossos antepassados.

Se olharmos para os programas de computador criados no passado e compararmos com os criados no presente, não é muito difícil concluir que é preciso cada vez mais esforço para conseguir prosperar na área de programação de computadores. Antigamente, a oferta de desenvolvedores de *software* e de ferramentas de desenvolvimento era bem mais escassa. Hoje, com a disseminação dos meios de educação em massa, de ferramentas de desenvolvimento gratuitas e de mecanismos eficientes para comercialização de programas de computador, está bem mais difícil encontrar compradores para programas do tipo *"hello, world!"*.

Provavelmente, o paradigma mais conhecido é o procedural (presente na linguagem de programação C), mas há outras opções também disponíveis por aí, capazes de permitir a solução de algumas classes de problemas de forma mais eficiente. Em todo caso, os paradigmas de programação mais comuns são os seguintes:

- Procedural
- Funcional
- Lógico
- Orientado a objeto

Cada um desses paradigmas será brevemente apresentado nas seções seguintes, ilustrando classes de problema cada um se aplica.

1.1 Paradigma de programação procedural

O paradigma de programação procedural geralmente é o preferido dos iniciantes. Foi um dos primeiros paradigmas de programação adotados, pois permitia (e ainda permite) que diversos algoritmos, principalmente aqueles que envolviam computação numérica, fossem criados com relativa facilidade.

Para o ensino da programação, o paradigma procedural facilita a introdução aos alunos dos conceitos básicos de concepção de algoritmos, de entrada e saída de dados e de tratamento de memória. Tais conceitos são normalmente tratados em cursos introdutórios de programação para alunos de engenharia e normalmente atendem às exigências curriculares da maioria dos cursos.

Programas de computador escritos em linguagens que seguem este paradigma geralmente são compostos por um conjunto de módulos (ou funções), cada um realizando uma determinada parte do processo. Cada programa é composto de uma rotina principal e diversas subrotinas auxiliares, sendo dada ênfase principal à ordem dos eventos no sistema desenvolvido, conforme ilustrado na Figura 1.1.

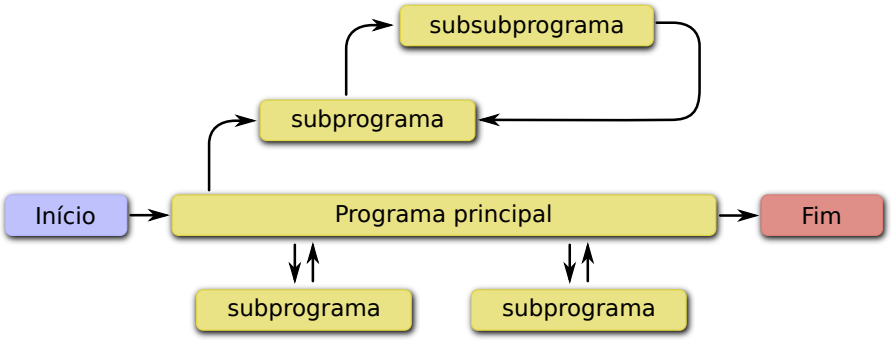

Figura 1.1: Paradigma de programação procedural

Normalmente, linguagens puramente procedurais oferecem um melhor controle do fluxo do código e são mais fáceis de entender. Como o foco é dado na sequência de eventos, elas geralmente são bastante comuns nas comunidades que lidam com a concepção e implementação de algoritmos numéricos. As bibliotecas mais difundidas para uso em computação numérica são escritas em linguagens procedurais como C ou FORTRAN.

Example 1.1 Programa `testa-ambiente.cpp` para testar sua ferramenta de desenvolvimento.

```
#include <iostream>

using namespace std;

int main(void){
  char s[40];
  cout << "Digite seu nome: ";
  cin >> s;
  cout << "Seu nome eh" << s << "\n";
  return 0;
}
```

O programa do Exemplo 1.1 pode ser compilado e executado em um terminal Linux a partir das seguintes linhas de comando:

```
$ g++ testa-ambiente.cpp -o testa-ambiente
$ ./testa-ambiente

Digite seu nome: Jose
Seu nome é Jose
```

1.2 Descriçao do programa `testa-ambiente.cpp`

O programa inicia com uma diretiva de inclusão que insere para processamento o arquivo `iostream`. A diretiva #include instrui o compilador a usar um pré-processador que lê e processa o arquivo referenciado antes de processar o restante do programa.

```
#include <iostream>
```

Capítulo 2

Capitu sobre qualquer coisa

Este é o meu primeiro paragrafo