C++ Programming

Step-by-Step

Asadullah Shah

IIUM PRESS
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

C++ PROGRAMMING: STEP BY STEP

Editors

Asadullah Shah

IIUM Press

Published by:
IIUM Press
International Islamic University Malaysia

First Edition, 2011
©IIUM Press, IIUM

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without any prior written permission of the publisher.

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Bibliography p.
Includes Index
ISBN

ISBN: 978-967-418-090-4

Member of Majlis Penerbitan Ilmiah Malaysia - MAPIM
(Malaysian Scholarly Publishing Council)

Printed by :
IIUM PRINTING SDN. BHD.
No. 1, Jalan Industri Batu Caves 1/3
Taman Perindustrian Batu Caves
Batu Caves Centre Point
68100 Batu Caves
Selangor Darul Ehsan

CONTENTS

32. MORE ON CONST KEYWORDS

Asadullah Shah and Osama Mahfooz
Department of Computer Science, Faculty of Information and Communication Technology, International Islamic University Malaysia, Malaysia

Abstract

It is simple in concept: variables declared with 'const' added become constants and cannot be altered by the program. However it is also used to budge in a substitute for one of the missing features of C++ and there it gets horridly complicated and sometimes frustratingly restrictive.

32.1 Const iterators

As we've already seen, in order to enforce const, C++ requires that const functions return only const pointers and references. Since iterators can also be used to modify the underlying collection, when an STL collection is declared const, then any iterators used over the collection must be const iterators. They're just like normal iterators, except that they cannot be used to modify the underlying data. (Since iterators are a generalization of the idea of pointers, this makes sense.)

Const iterators in the STL are simple enough: just append "const_" to the type of iterator you desire. For instance, we could iterator over a vector as follows:

```
std::vector<int> vec;
vec.push_back( 3 );
vec.push_back( 4 );
vec.push_back( 8 );
```