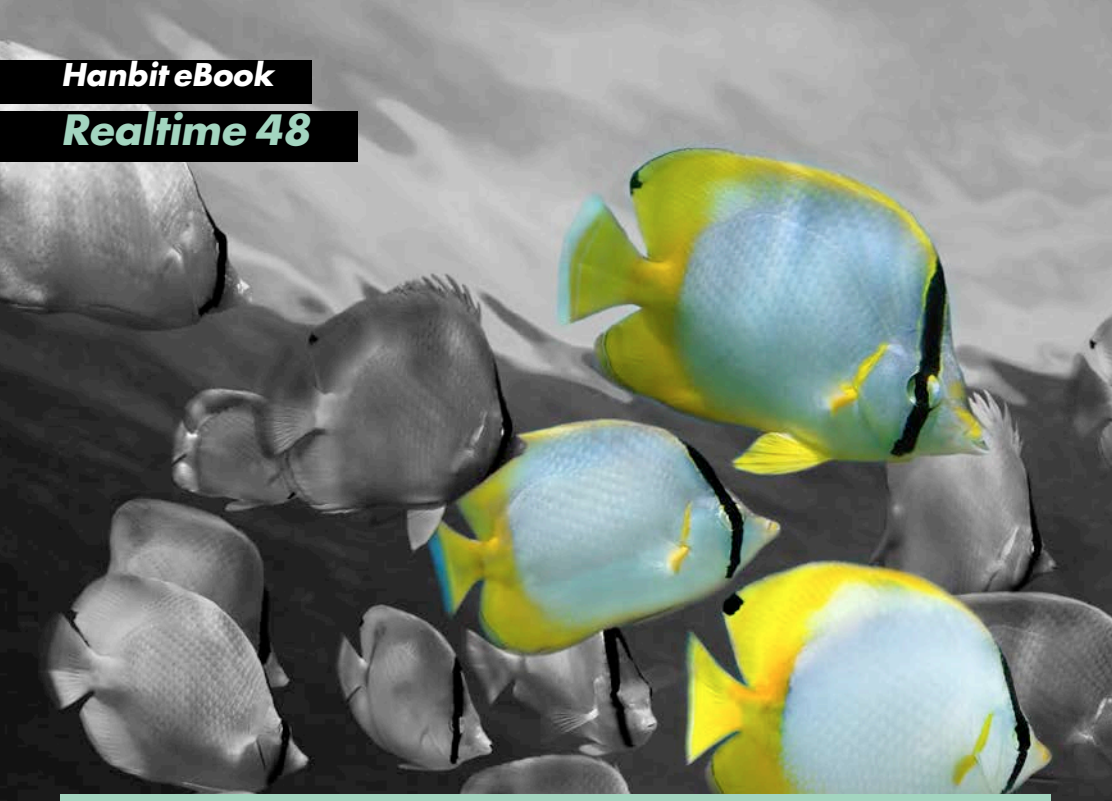
Hanbit eBook
Realtime 48



BACK TO THE BASIC

C++11 핵심 노트

핵심 주제 12가지로 배우는 C++11

이주한 지음

한빛미디어 Hanbit Media, Inc.

BACK TO THE BASIC

C++11 핵심 노트

핵심 주제 12가지로 배우는 C++11

BACK TO THE BASIC C++11 핵심 노트 핵심 주제 12가지로 배우는 C++11

초판발행 2013년 11월 29일

지은이 이주한 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-653-1 15000 / **정가** 9,900원

책임편집 배용석 / **기획** 이중민 / **편집** 이순옥
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 박진희
마케팅 박상용, 김옥현

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

지은이_ **이주한**

삼성전자에서 디지털 영상처리 및 네트워크 전송 기술 분야에서 8년간 근무했으며, 삼성 스마트 TV 개발에 참여했다. 멀티미디어 네트워크 전송 기술과 관련해 40여 개의 국내 국제 특허의 제1저자 및 주요 저자로 등록되어 있기도 하다. 2009년에 가족과 함께 호주로 이민한 후 Australian Associated Press 사 등에서 소프트웨어 엔지니어로 근무했고, 현재는 멀티미디어 코덱 기술과 관련한 프리랜서 개발자로 일하고 있다. 애자일 프로세스와 모바일 웹 플랫폼 기술에 관심이 있으며 다양한 실험을 즐긴다. 주말에 두 아이(윤서, 윤재)와 공원에서 함께 노는 것이 요즈음 가장 큰 즐거움이다.

저자 서문

어떤 분은 이미 거대할 대로 거대해진 C++가 이제는 마치 머리카락 대신 수백 마리의 뱀을 휘감은 메두사와 같은 모습으로 변해버렸다고 말합니다. 이미 차고 넘치는 C++ 문법을 다 사용하지도 못할뿐더러 배워야 할 모범 사례Best Practice라고 불리는 다양한 테크닉으로도 숨이 막힐 지경인데, 심지어 새로운 문법이라니요. 메두사의 눈을 보면 그 누구라도 딱딱한 돌덩이로 변해버리듯, 새로운 C++11 표준을 마주하는 순간 프로그래머의 손가락이 돌처럼 굳어버릴 것이라고 지레 겁을 먹습니다.

그리스 고대 신화를 읽어본 사람이라면 메두사는 괴기스러운 겉모습과는 달리 원래 과거뿐만 아니라 현재와 미래까지도 꿰뚫어 볼 수 있는 지혜의 여신이었다는 사실을 알 것입니다. 이런 맥락에서 C++는 메두사의 원모습과 비슷하다고 할 수 있습니다. 잘만 사용하면 메모리를 마음대로 주무를 수 있는 포인터부터, 코드가 아름다울 수도 있다고 느껴지는 객체 지향 프로그래밍, 그리고 코드가 또 다른 코드를 만들어내는 경이로운 신세계 템플릿 메타프로그래밍까지, C++는 그야말로 '원하는 것을 말씀하세요'라고 속삭이는 요정 지니 같은 프로그래밍 언어이기 때문입니다.

다양한 일을 할 수 있다는 말은 숙달하는 데 오랜 시간을 투자해야 한다는 말과도 일맥상통합니다. 솔직히 말해서 C++는 숙달하기 정말 쉽지 않은 언어입니다. 2~3년 정도만 익히면 어느 정도 쓸 줄 안다고 자부할 수 있는 언어도 많지만, C++ 만큼은 예외입니다. 방대한 문법은 둘째치고 C를 기본으로 한 언어인 만큼 C 언어를 이해하는 것은 필수입니다. 실행 환경은 또 어떻습니까? C++는 자바나 C#처럼 컴파일러나 실행 환경이 독점적으로 제공되지 않으므로 운영체제 플랫폼에 따라, 그리고 컴파일러 벤더에 따라 똑같은 코드가 동작하기도 하고, 아예 처음부터 컴파일

에러를 발생시키기도 합니다. 멀티플랫폼을 위한 C++ 코드를 한 번이라도 작성해 보았다면 #ifdef와 #elif, #endif로 점철되는 코드를 앞에 두고 현기증을 느꼈던 기억이 적어도 한 번쯤은 있으리라 생각됩니다.

이번에 새롭게 도입된 C++11의 기능들은 새로운 문법과 라이브러리를 추가해 프로그래머 여러분을 다시 한번 곤경에 빠뜨리려는 것이 아닙니다. 오히려 앞에서 말한 컴파일러 벤더별로 천차만별인 지원 상황을 일소하고, 강력한 표준화 프로세스를 통해 작성하기도 쉽고 읽기도 편한 C++를 만들려는 노력의 결과물입니다.

이러한 C++의 변화는 컴퓨팅 환경의 변화와도 무관하지 않습니다. 싱글코어에서 멀티코어로, 64KB 메모리 안에서 마른 수건 짜듯 해야 했던 개발 환경에서 이제는 몇 기가의 메모리 정도는 넉넉히 사용할 수 있는 개발 환경으로 변했습니다. 인터넷에 연결되지 않은 독립 PC 환경에서 모든 종류의 기기가 IP로 연결된 커넥티드 기기 환경으로 변했습니다. C++는 이러한 컴퓨팅 전반에 이르는 변화를 끌어안고 앞으로 나아가려는 노력의 산물입니다.

이 책은 C++를 처음 접하는 독자를 대상으로 하는 책이 아닙니다. 그보다는 C++를 다루어 본 경험이 있고, 여타 언어도 한두 개 정도 다루어 본 적이 있는 분들이 변화하는 C++ 언어 표준을 접할 때 읽으면 좋은 책입니다. 이미 C++를 어느 정도 안다고 가정했기 때문에 객체 지향 개념이나 자료구조 같은, 프로그래밍 기초는 다루지 않습니다. 또한 상대적으로 중요한 개념 설명이 필요한 부분에는 전체 코드를 삽입했지만, 대부분은 간단한 코드이므로 실제로 작동하는 예제를 만들려면 main 함수 정도는 작성할 수 있어야 합니다.

C++11에서 도입된 변화가 개발자로서 여러분의 삶에 얼마나 많은 변화를 불러일으킬지는 분명하지 않습니다. 하지만 C++ 개발자로서 계속 경력을 쌓아나가고 싶다면, C++에 부는 변화의 바람에 주목해야 합니다. 마이크로소프트와 같은 메이저 IT 업체가 왜 그렇게 많은 인력과 시간을 C++에 투자하는지 다시 한번 생각해 보기 바랍니다. 훌륭한 목수는 연장을 다듬는 데 소홀함이 없습니다. C++11을 배우는 데 투자한 시간은 코드의 성능과 생산성 향상이라는 보상으로 반드시 돌아올 것이라 믿습니다.

집필을 마치며

이주한

대상 독자 및 참고 사항

초급 **초중급** 중급 중고급 고급

이 책은 C++11의 핵심 개념을 소개하는 책입니다. C++의 기본을 알고 있으며 C++11에 관심이 있는 분이라면 누구나 읽을 수 있습니다. 또한 이 책의 샘플 코드를 실행하려면 다음에 소개하는 환경이 갖춰져 있어야 합니다.

- 마이크로소프트 Visual C++ 2012
- GCC 4.7 이상 버전과 컴파일러를 사용할 수 있는 개발 환경

이 책은 국내에서 가장 많이 사용하는 C++ 통합 개발 환경인 Visual C++ 2012에서의 실행 여부를 테스트했습니다. 하지만 '4장 유니폼 초기화'의 샘플 코드는 Visual C++ 2012에서 완전하게 지원하지 않음을 미리 알립니다. 4장에서 설명하는 샘플 코드는 GNC GCC 4.7 버전 기준으로 작성해 테스트했으며, 여러분도 샘플 코드를 테스트하려면 GCC 컴파일러를 사용해야 합니다(참고로 GCC 컴파일러에서 C++11 코드를 컴파일하려면 컴파일 옵션에 -std=c++11 항목을 반드시 추가해야 합니다).

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

1 | Hello! C++11 World

1998년 C++ 표준이 발표된 이래, C++는 아주 오랫동안 다양한 소프트웨어를 만드는 데 사용됐습니다. 즉, 여러분을 가르쳤던 나이 지긋한 교수님께서 당시로써는 생소한 프로그래밍이라는 것을 처음 배우던 때부터, 10대 천재 프로그래머가 멋진 모바일 게임을 만들어 내는 요즘까지도, C++는 소프트웨어 업계에서 묵묵히 자기 할 일을 해내는 사실상 업계 표준처럼 사용됐던 것입니다.

C++가 사실상의 산업 표준이라는 말을 믿기 어려운가요? 여러분이 지금 사용하는 운영체제는 어떤 언어로 작성됐을까요? 마이크로소프트 윈도우나 애플의 맥Mac OS 같은 PC 운영체제나 iOS나 블랙베리 같은 모바일 기기 운영체제의 전부 혹은 핵심 부분은 C++로 만든 것입니다. 컴퓨터의 바탕화면을 아름답게 꾸미고, 마우스를 이용한 컴퓨팅 혁명을 불러왔던 GUIGraphical User Interface도 한 뭉치의 C++ 코드입니다. 리포트를 작성하거나 프레젠테이션 자료를 만드는 데 사용하는 오피스 애플리케이션도 C++로 만들어졌으며, 거의 모든 웹 브라우저도 C++ 애플리케이션입니다. 이른바 기업 정보화에 한몫한 데이터베이스, 이제는 없어서는 안 될 구글 검색 엔진, 컴퓨터 게임, 자동차 내비게이션, 이베이나 아마존 같은 온라인 상거래 사이트, 심지어 페이스북 같은 소셜 웹 사이트도 C++ 없이는 한순간도 동작할 수 없는 애플리케이션들입니다. 개발자 여러분이 C++ 프로그램을 작성하는 데 사용하는 마이크로소프트의 Visual Studio 같은 통합 개발 환경IDE, Integrated Development Environment과 C++ 컴파일러도 C++로 만든 것이기도 합니다.

혹자에게는 C++라는 프로그래밍 언어가 추억의 70, 80 트로트 메들리와도 같은 구닥다리일지도 모르겠습니다. 하지만 21세기에 정보화라는 이름 아래 누리는 여러 가지 편리함과 즐거움을 가능케 했던 핵심 정보 기술 가운데 하나가 바로 C++

라는 점은 그 누구도 부정할 수 없는 사실입니다. C++는 웹 플랫폼 기술과 클라우드 기술이 찬란히 빛나는 요즘에도 변화의 흐름을 놓쳐서는 안 될 중요한 프로그래밍 언어 가운데 하나입니다. 특히 C++0x를 거쳐 정식 표준으로 확정된 C++11이야말로 C++ 프로그래머들에게 흥분과 기대를 불러일으키는 커다란 선물 꾸러미가 아닐까 생각합니다.

이러한 C++11이 2011년 8월 12일에 ISO C++ 표준 위원회Standard Committee의 만장일치로 통과되었습니다. 그동안 C++0x 등으로 불렸으나 표준으로 인정받은 해에 맞춰 기술하는 원칙에 따라 C++11이라는 정식 명칭이 탄생한 것입니다(표준으로 인정받은 다음에도 지속해서 C++를 발전시키기 위해 C++14를 논의하고 있습니다. 단, 실제 2014년도에 표준으로 승인받을 수 있을지는 알 수 없습니다).

최근 스마트폰과 태블릿 PC가 등장하고 클라우드와 빅데이터 구축 등이 IT의 새로운 화두로 떠오르면서 C++에 대한 관심이 조금 낮아진 것 같습니다. 하지만 앞서 설명한 것처럼 C++는 지금도 다양한 분야에서 가장 많이 사용되는 프로그래밍 언어입니다. 1장에서는 이러한 C++의 최신 표준인 C++11의 특징을 간략하게 정리해볼까 합니다.

1.1 C++11의 특징

C++11은 그야말로 천지가 개벽할 만한 변화를 한가득 품었습니다. 심지어 C++의 아버지로 불리는 비야네 스트라스트럽Bjarne Stroustrup조차 "새로운 C++ 표준은 내 자신이 애초에 디자인했던 언어가 아닌 전혀 새로운 언어처럼 느껴진다"라고 놀라움을 표현할 정도입니다. 이런 소회는 자신이 디자인했던 C++가 가질 수 없었던 매끄러움과 극복할 수 없었던 한계를 이겨내고 한층 더 향상된 효율성을 이뤄낸 C++11 표준에 보내는 찬사(http://www.stroustrup.com/C++11FAQ.html)이기도 합니다.

먼저 위키피디아의 설명을 잘 살펴볼 필요가 있습니다. 위키피디아에서는 C++11의 특징을 다음처럼 설명합니다.

- 안정성 및 C++98(가능하면 C 언어와도)과의 호환성 유지
- 핵심 언어의 확장보다는 표준 라이브러리를 이용한 새로운 기능 추가
- 프로그래밍 기술을 발전시킬 변화 선호
- 특정 애플리케이션에서만 유용한 새로운 기술보다는 시스템이나 라이브러리 디자인에 유용하게 C++를 개선
- 이전의 안전하지 않은 기술에 대해 좀 더 안전한 대안을 제공하여 타입 안전성 증가
- 성능 향상과 하드웨어 직접 조작 능력 강화
- 실 세계의 문제를 해결할 수 있는 해법 제시
- '부담 최소화'의 원칙(어떤 유틸리티가 필요로 하는 추가적인 지원은 그 유틸리티를 사용할 때만 필요해야 함)
- 전문 프로그래머가 필요로 하는 어떤 유틸리티도 제거하지 않고도, 쉽게 가르치거나 배울 수 있어야 함

또한 문법에서도 편의성이 크게 향상됐습니다. 이를 위키피디아에서는 다음처럼 소개합니다.

- 템플릿에서의 가변 인자
- 새로운 문자열 리터럴

- 사용자 정의 리터럴
- 멀티태스킹 메모리 모델
- TLS(Thread-local storage)
- 특수 멤버 함수의 기본값 사용 및 삭제에 대한 명시적 표시
- long long int 타입
- 정적 assertion
- 멤버에 대한 sizeof 허용

위키피디아의 설명을 종합하면 C++11의 특징은 기존 ISO C++ 표준과의 호환성을 유지하면서 최근 객체 지향/스크립트 언어에서 제공하는 편리함을 추가하기 위함임을 알 수 있습니다. 즉, 다음과 같습니다.

- 스크립트 언어에서 볼 수 있는 타입의 유연성 강화
- 표준 라이브러리(STL)를 기반에 두고 언어를 향상함
- 시스템 종속적이 아닌 다양한 시스템에서 사용할 수 있도록 C++를 개선
- 애플리케이션마다 독립적일 수 있게, 개발 도구에 영향을 받지 않도록 언어 구조를 설계

또한 문법에서의 편의성 향상 부분을 살펴보면 주로 타입을 쉽게 다루고 이를 초기화하는 방법에 많은 할애를 했음을 알 수 있습니다.

1.2 C++11 컴파일러

C++11은 승인된 표준 중에서는 최신 표준입니다. 따라서 개발 도구의 핵심 중 하나인 컴파일러의 C++11 지원 여부가 중요하지 않을 수 없습니다. 이번에는 C++11의 컴파일러를 간략하게 살펴보겠습니다.

1.2.1 컴파일러 지원

C++11을 지원하는 컴파일러에는 무엇이 있을까요? 아마도 Visual C++를 사용한다면 MSVC 컴파일러를 떠올릴 것이고, 리눅스를 사용한다면 GCC를 떠올릴 겁니다(물론 다른 컴파일러도 많습니다).

먼저 C++11을 지원하는 컴파일러를 살펴볼 필요가 있습니다. C++11의 컴파일러 지원 상황은 아파치 커뮤니티에서 공개한 'C++0x 컴파일러 지원'을 참고하면 좋습니다. 표 1-1에서는 'C++0x 컴파일러 지원' 표 전체 내용 중 국내에서 많이 사용하는 주요 컴파일러 관련 부분만 소개합니다.

표 1-1 C++11을 지원하는 컴파일러별 특징(2013년 5월 13일 기준)

C++11 주요 기능	GCC	Intel C++	MSVC	Sun/ Oracle C++	Clang
Alignas	4.8				3.0
Alignof	4.5				2.9
Atomic operations	4.4	13.0	11.0		3.1
Auto	4.4(v1.0)	11.0(v0.9)	10.0(v0.9)		Yes
C99 preprocessor	4.3	11.1		5.9	Yes
Concepts [removed]	ConceptGcc				
Constexpr	4.6	13.0			3.1
decltype	4.3(v1.0) 4.8.1(v1.1)	11.0(v1.0)	10.0(v1.0) 11.0(v1.1)		2.9

C++11 주요 기능	GCC	Intel C++	MSVC	Sun/ Oracle C++	Clang
Defaulted And Deleted Functions	4.4	12.0			3.0
Delegating Constructors	4.7		11.0 nov'12		3.0
Explicit conversion operators	4.5	13.0	11.0 nov'12		3.0
Extended friendDeclarations	4.7	11.0	10.0***		2.9
extern template	3.3	9	6.0		Yes
Forward declarations for enums	4.6		11.0		3.1
Inheriting Constructors	4.8				3.3
Initializer Lists	4.4	13.0	11.0 nov'12		3.1
Lambda expressions and closures	4.5(v1.1)	11.0(v0.9) 12.0(v1.0)	10.0(v1.0) 11.0(v1.1)		3.1
Local and Unnamed Types as Template Arguments	4.5	12.0	10.0		2.9
long long	Yes	Yes	Yes	Yes	Yes
Namespace Association	4.4				2.9
New character types	4.4				2.9
New function declaration syntax for deduced return types	4.4	12.1	10.0		2.9
nullptr	4.6	12.1*	10.0		2.9
Unicode String Literals	4.4	11.0		5.7	3.0
Raw String Literals	4.5		11.0 nov'12		Yes
User-defined Literals	4.7				3.1

C++11 주요 기능	GCC	Intel C++	MSVC	Sun/ Oracle C++	Clang
Right Angle Brackets	4.3	11.0	8.0		Yes
R-Value References, std::move	4.3(v1.0) 4.5(v2.1) 4.6(v3.0)	11.1(v2.0) 12.0(v2.0)	10.0(v2.0) 11.0(v2.1)		Yes
static_assert	4.3	11.0	10.0		2.9
Strongly-typedenums	4.4	12.0	11.0		2.9
Template aliases	4.7	12.1			3.0
Thread-Local Storage	4.8(4.4****)	11.1***	10.0***	5.9***	3.3(2.9****)
Unrestricted Unions	4.6				3.0
Built-in Type Traits	4.3	10.0	8.0		3.0
Variadic Templates	4.3(v0.9) 4.4(v1.0)		11.0 nov'12		2.9(1.0)
Range-based for-loop	4.6	12.1(v0.9)	11.0		3.0
override and final	4.7	13.0	8.0(v0.8)*** 11.0(v1.0)		2.9
Attributes	4.8	12.0(v0.8)***			3.3
ref-qualifiers	4.8.1	12.1			2.9
Non-static data member initializers	4.7				3.0
Dynamic initialization and destruction with concurrency (Magic statics)	4.3				2.9

2012년 당시의 상황을 알고 싶다면 C++Roscks에서 소개하는 문서를 참고해도 좋습니다.

1.2.2 GCC

C나 C++를 다뤄본 개발자라면 대다수가 GCC를 알 것입니다. GCC는 현재 C++11을 가장 완벽하게 지원하는 컴파일러입니다. 앞에서 소개한 표처럼 GCC 버전에 따라 C++11 전체 기능을 얼마나 지원하는지 차이가 있지만 GCC 컴파일러의 최신 버전을 사용한다면 C++11 기능 대부분을 사용할 수 있습니다.

1.2.3 MSVC

현재 국내에서 C++ 개발 도구로 널리 사용되는 것은 역시나 Visual C++입니다. 그런데 C++11의 모든 기능을 지원할 것으로 예상했던 Visual C++ 2012(VC++11)에서는 현재 C++11을 지원하는 컴파일러 중 C++11의 기능 지원이 가장 소극적으로 이루어졌습니다.

Visual C++의 C++11 지원 현황과 사용법을 살펴보려면 Visual C++ 2013을 기준으로 C++11 기능을 소개하는 http://msdn.microsoft.com/ko-kr/library/hh567368.aspx 문서를 참고하기 바랍니다. 한글 문서이므로 참고하기 편할 것입니다.

또한 영문 문서며, 2011년 C++11 표준이 승인된 지 얼마 되지 않은 후에 발표한 문서(최근 수정일은 2012년 3월 2일)지만 MSDN의 Visual C++ 공식 팀 블로그의 'C++11 Features in Visual C++11'에서 Visual C++ 2012(VC++11)와 Visual C++ 2011(VC++10)의 C++11 지원 차이점을 설명합니다. 실무 개발 환경에서 아직 Visual C++ 2013(VC++12)을 사용하기 부담스럽다면 이 팀 블로그를 참고해서 C++11을 사용할 것을 추천합니다. 그리고 기존 MSVC 컴파일러에서 부족한 부분을 보완하기 위해 오래된 버전의 Visual C++에 VC++11 컴파일러를 추가로 설치할 수 있도록 지원합니다. 자세한 내용은 'Announcing November CTP of the C++ compiler, now with more C++11'에서 확인할 수 있습니다.

1.2.4 LLVM과 Clang

지금까지 널리 쓰이던 GCC와 MSVC 컴파일러에 대해서 설명했습니다. 그런데 뭔가 하나 부족하다는 생각이 들지는 않으신가요? 과거에는 크게 신경 쓸 필요가 없었지만 스마트폰 시대가 열리면서 주목하기 시작한 애플 개발 환경을 빼놓을 수 없습니다. 당연히 애플 개발 환경에서도 C++11을 중요하게 생각하며 이를 지원하는 개발 환경으로 LLVM과 Clang이 있습니다.

LLVM은 원래 다양한 프로그래밍 언어의 정적/동적 컴파일러 집합과 도구를 만들어내는 오픈 소스 프로젝트로 시작한 것입니다. 자신만의 중간 코드IR, Intermediate Representation 언어를 정의하고, 이 중간 코드로 코드를 생성/수행하는 일을 하며, 프로그래밍 언어와 CPU에 독립적인 최적화기, x86, x86-64, ARM, SPARC 등 다양한 CPU를 지원합니다. 또한 프로그램을 분석하는 루틴을 만들 때 아주 편리합니다. LLVM의 자세한 설명은 LLVM Compiler Infrastructure을 자세히 살펴보시기 바랍니다.

> LLVM은 창시자라고 할 수 있는 크리스 래트너Chris Lattner의 대학원 연구 과제로 시작했던 프로젝트인데, 크리스 래트너가 애플에 입사하면서 이 프로젝트를 전면 도입했고 애플이 주도적으로 관리하고 있습니다. 물론 오픈 소스 프로젝트임에는 변함없습니다.

Clang은 LLVM 컴파일러 컴포넌트를 기반으로 해 C, C++, Objective-C, Objective-C++ 프로그래밍 언어를 지원하기 위한 컴파일러 프론트엔드입니다. GCC 프론트엔드보다 빠르고, 메모리 소모가 적으며 에러 메시지가 명확하다는 특징이 있습니다.

지원하는 언어를 보면 알겠지만 애플이 주도하며, 구글에서도 C++를 사용할 때 LLVM에 기반을 둔 Clang을 사용하길 권장합니다. 당연히 Xcode에는 포함되어

있으며 리눅스 패키지에도 포함되어 있습니다. LLVM 2.6 버전 이후로는 정식 릴리스의 일부로 자리잡았습니다. 일리노이 대학교/NCSA 오픈 소스 라이선스로 이용할 수 있습니다.

> Objective-C++는 아직도 개발 중인 언어로 Objective-C와 C++를 혼합해서 사용하려는 목적이 있습니다. 일반적으로 Objective-C 문법으로는 C++ 객체나 표준 라이브러리를 호출할 수가 없는데 이를 위해서 생겨난 개념입니다. GCC 4.1 버전부터 지원합니다.

2 | auto 키워드

원래 C++에서 auto 키워드는 변수의 저장 공간을 지정하려는 용도로 만들어졌습니다. 이런 키워드들을 일컬어 스토리지 클래스 지정자storage class specifier라고 부릅니다. 스토리지 클래스 지정자에는 auto 이외에도 register, static, extern, mutable과 같은 키워드들이 있으며, 여전히 많은 C++ 프로그래머의 사랑을 받으며 다양한 용도로 사용되어 왔습니다.

특히 static이나 extern 같은 키워드는 C 라이브러리와 함께 작성해야 하는 용도로 프로그램에 등장하는 단골 메뉴이며, register나 mutable 같은 키워드는 임베디드 멀티스레드 프로그래밍에서 없어서는 안 될 존재들입니다.

하지만 앞에서 소개한 키워드와 비교했을 때 auto의 경우는 뭔가 편리해 보일 것 같은 이름임에도 사용 빈도가 내리막을 걷다 못해 아무도 찾지 않는 죽은 키워드가 되었습니다. 이유는 auto 키워드가 로컬 변수의 스토리지 클래스를 지정하는 기본 설정이었기 때문입니다.

기본 설정이란 의미는 무엇일까요? 여기에서는 선언해도 그만, 선언하지 않아도 그만이라는 의미입니다. 즉, 다음 구문은 완전히 똑같은 문장입니다.

```
int i = 0;
auto int i = 0;
```

얼른 일을 끝내고 퇴근해야 할 시간도 부족한 마당인데, auto라는 네 글자를 입력하는 것도 사실은 시간 낭비입니다. 결국 auto 키워드는 야심찬 이름에도 아무도 찾지 않아 쓸쓸히 용도 폐기의 길을 향하고 있었습니다.

한편 새로운 키워드를 찾아 삼매경이던 C++11 표준 위원회에서는 이 먼지 쌓인 키워드를 발견해냈고, 이렇게 쓸쓸히 버려두기보다는 새로운 의미를 부여한 후 멋지게 재활용하기로 결심합니다. 그 결과 C++11에서 auto 키워드는 컴파일러에게 타입을 알아내라고 지시하라는 의미가 되었습니다. 즉, 프로그래머가 직접 타입을 지정하지 않아도 변수 이름만 컴파일러에게 전달하면 변수에 지정된 값에 맞춰서 컴파일러가 자동으로 타입을 지정합니다. 다음 소개하는 예를 살펴보면 쉽게 이해할 수 있습니다.

```
auto i = 100;           // i 변수의 타입은 int
auto l = 100L;          // l 변수의 타입은 long
auto p = new Person(); // p 변수의 타입은 Person 포인터
```

그럼 실제로 auto 키워드를 사용한 예를 살펴보겠습니다.

```
#include "stdafx.h"
#include <iostream>

using namespace std;
struct Person {
  int age;
  char name[5];
};
class CPerson {
  public:
    CPerson():age(1), name("anonymous"){}
    CPerson(int age, std::string name):age(age), name(name){}
    ~CPerson(){};
  public:
    int GetAge(){return age;}
  private:
```

```
    int age;
    std::string name;
    // auto nationality; // error
};

int main(int argc,  char** argv) {
  // char* type
  auto name = "JohnL";
  cout<<name<<endl;

  // integer
  auto iNum = 1;
  cout<<iNum<<endl;

  // pointer
  auto* piNum = &iNum;
  cout<<*piNum<<endl;

  // reference
  auto& riNum = iNum;
  cout<<riNum<<endl;
  riNum = 10;
  cout<<riNum<<endl;
  cout<<*piNum<<endl;

  // object type
  auto* p1 = new Person;
  auto* p2 = new CPerson;

  cout<<"Age : "<<p2->GetAge()<<endl;
  getchar();
  return 0;
}
```

실행 결과는 다음과 같습니다.

```
JohnL
1
1
1
10
Age : 1
```

타입의 이름이 길거나 참조나 포인터를 지정할 경우는 입력해야 할 코드 길이가 좀 깁니다. 하지만 auto 키워드는 이런 코드 입력을 줄여줍니다. int 타입의 변수를 선언하는 경우를 제외하면 적어도 같거나 짧습니다. 특히 다음 예처럼 STL 컨테이터에서 반복자iterator 등을 지정할 때는 auto 키워드가 입력해야 할 코드양을 상당히 줄여줍니다.

```
#include <iostream>
#include <vector>

using namespace std;

int main(int argc, char** argv) {
  vector<int> vInt;
  for(auto i=0; i < 10; ++i) {
    vInt.push_back(i);
  }

  // C++03 표준
  cout<<"C++03 standard"<<endl;
  vector<int>::iterator it = vInt.begin();
  while(it != vInt.end()) {
    cout<<*it<<endl;
```

```
    it++;
  }

  // C++11 표준
  cout<<"C++11 standard"<<endl;
  auto it2 = vInt.begin();
  while(it2 != vInt.end()) {
    cout<<*it2<<endl;
    it2++;
  }
  getchar();
  return 0;
}
```

실행 결과는 다음과 같습니다.

```
C++01 standard
0
1
2
3
4
5
6
7
8
9
C++11 standard
0
1
2
3
```

```
4
5
6
7
8
9
```

기존 STL의 반복자를 사용해야 할 경우에는 typedef를 이용해 길고 긴 반복자 타입을 짧게 줄이는 선언부터 하고 시작해야 했습니다. 또한 다른 프로그래머가 이미 작성했던 typedef를 읽는 일은 또 어떻습니까. 한두 개라면 부담없이 읽을 수 있겠지만, 십여 개의 typedef가 작성된 코드를 읽는 것만큼 고역인 일도 없습니다. 하지만 이제 새롭게 태어난 auto 키워드를 사용하면, 이런 불편에서 해방될 수 있을 뿐 아니라 코드 작성 시간을 줄여 여러분의 퇴근 시간을 조금이나마 앞당길 수 있을지도 모릅니다.

사실 지금까지 설명한 동적 타입 지정은 자바스크립트 같은 웹을 위한 언어나 파이썬 같은 스크립트 언어에서는 오랫동안 편리하게 사용되던 개념입니다. 그런데 지금까지 C++는 기본적으로 C의 개념을 상속받은 언어이므로 명시적으로 타입을 지정해야 했습니다.

C++11에서는 기존의 명시적인 타입 지정도 지향하면서 동적인 타입 지정도 허용하도록 바뀌었습니다. 지금부터는 변수들의 타입 지정 같은 사소한 일에 얽매이지 말고 프로그램 로직이나 구성 설계에 좀 더 신경 쓰기 바랍니다.