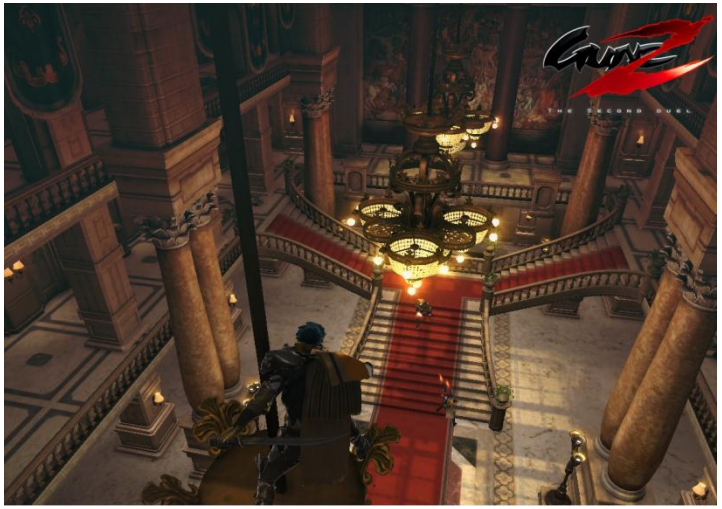
Boost 라이브러리와 C++11

마이에트 엔터테인먼트 건즈2팀
서버 프로그래머
최흥배
MS Visual C++ MVP
Twitter : @jacking75

MAIET entertainment



Visual C++ MVP 2008 ~

VS 스터디팀 회원으로 활동 중. http://vsts2010.tistory.com/
현재 **'미리 보는 C++11'** 연재 중


SPEAKER
손오공
Choi Heung Bae
Korea
iCon


Define the Future
KGC2008
The 5th Generation of Gaming
최흥배
Dyson
Pass
SPEAKER


Define the Future
KGC2009
Korea Games Conference
CORE OF GAME
2009.10.7(Wed.) ~ 10.9(Fri.)
900046
Heung Bae, Choi
Maiet Entertainment
SPEAKER


Define the Future
KGC2010
Korea Games Conference
No. 900041
최흥배
마이에트 엔터테인먼트
SPEAKER

C++11

Boost 라이브러리와 C++11

유용한 Boost 라이브러리 소개

C++11

C++98

C++03


NOKIA
NOKIA
6160

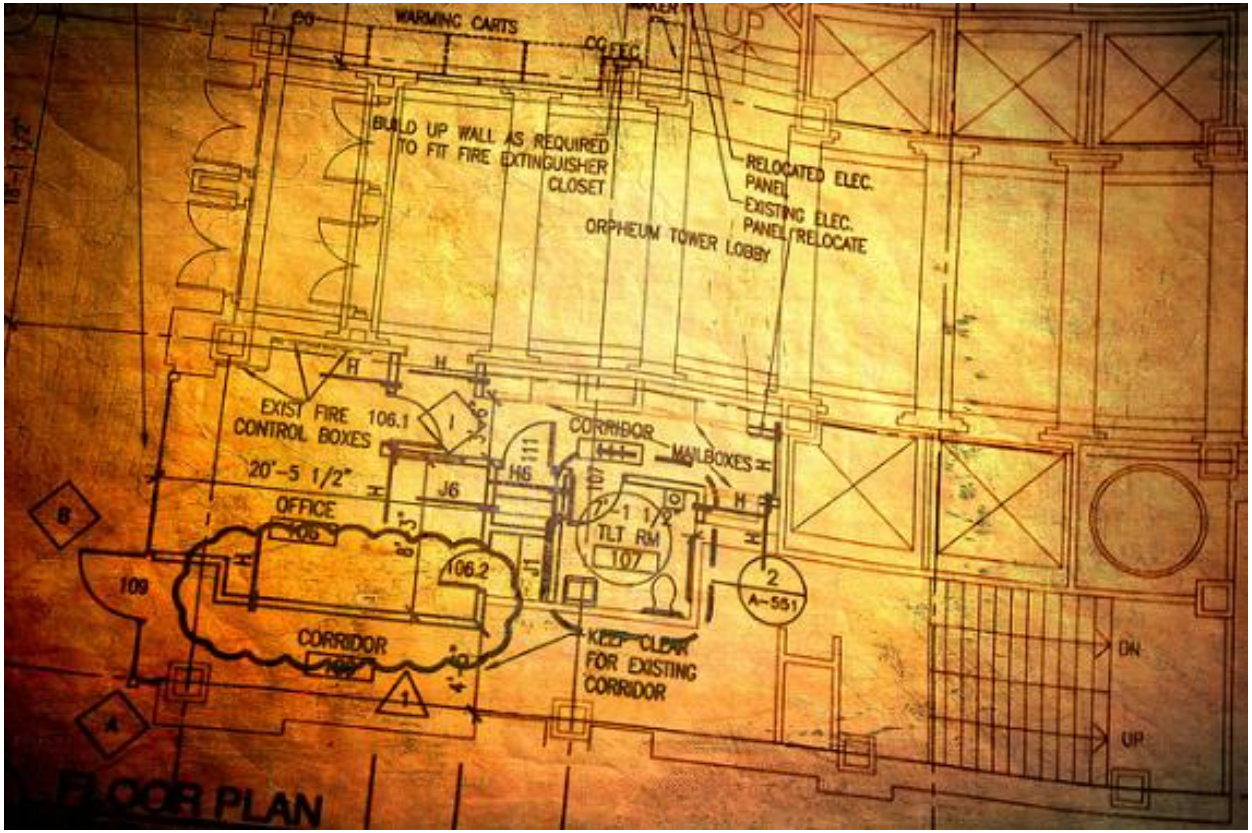
2006년까지 새로운 표준에
들어갈 기능을 제안 받음

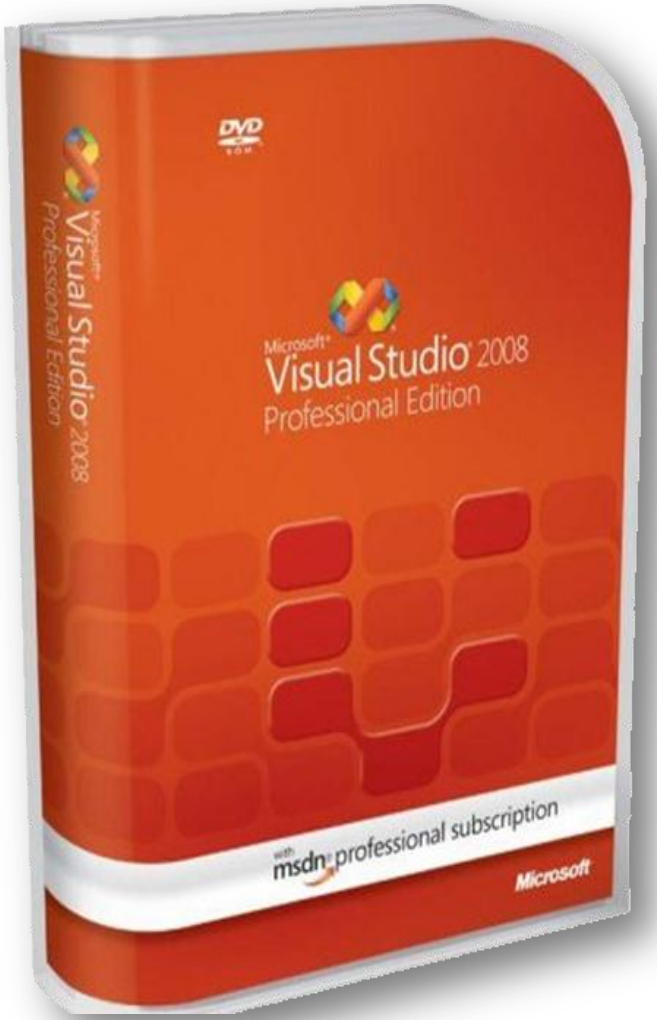
당초 계획은 2009년까지
표준을 확정하는 것.

그래서 **C++0x**라고 부르기로 함

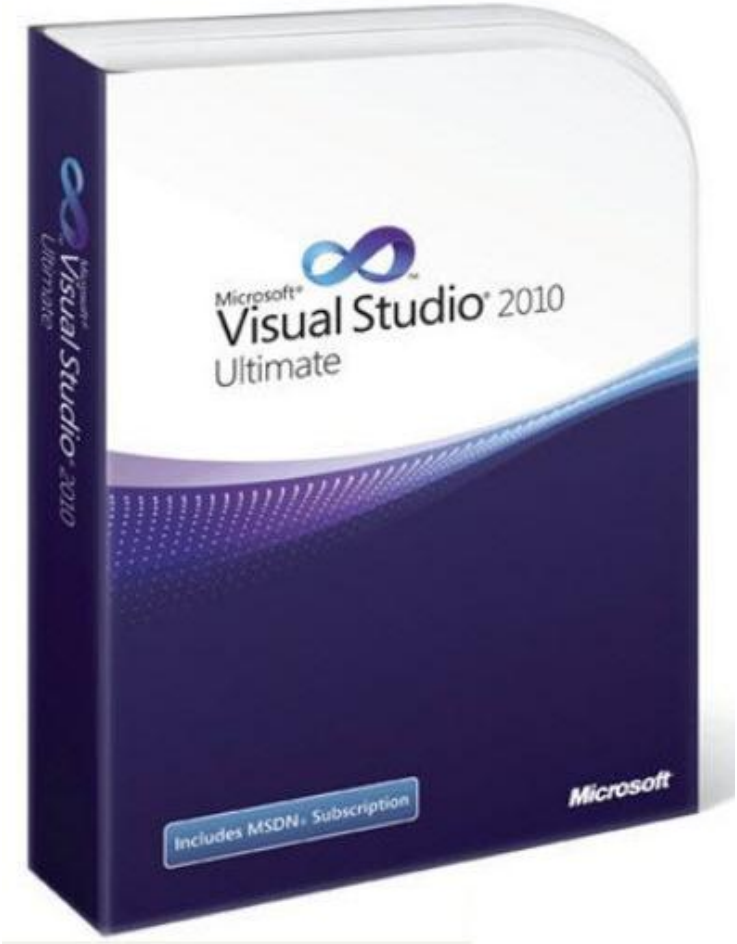
2011년 8월 12일에 ISO에서 승인을 받음

C++11


WARMING CARTS
COFFEE
BUILD UP WALL AS REQUIRED TO FIT FIRE EXTINGUISHER CLOSET
RELOCATED ELEC. PANEL
EXISTING ELEC. PANEL TO BE RELOCATED
ORPHEUM TOWER LOBBY
EXIST FIRE CONTROL BOXES
106.1
20'-5 1/2"
OFFICE
105
106.2
CORRIDOR
109
CORRIDOR
MAILBOXES
H6
J6
TLT RM
107
KEEP CLEAR FOR EXISTING CORRIDOR
2
A-551
DN
UP
FLOOR PLAN



- C++0x TR1 라이브러리 추가
- VC++ 버전 9



- auto
- static_assert
- rvalue Reference
- lambda
- decltype
- nullptr
- unique_ptr
- make_shared
- STL 추가

C++11 Core Language Features	VC10	VC11
Rvalue references v0.1, v1.0, v2.0, v2.1, v3.0	v2.0	v2.1*
ref-qualifiers	No	No
Non-static data member initializers	No	No
Variadic templates v0.9, v1.0	No	No
Initializer lists	No	No
`static_assert`	Yes	Yes
auto v0.9, v1.0	v1.0	v1.0
Trailing return types	Yes	Yes
Lambdas v0.9, v1.0, v1.1	v1.0	v1.1
`decltype` v1.0, v1.1	v1.0	v1.1**
Right angle brackets	Yes	Yes
Default template arguments for function templates	No	No
Expression SFINAE	No	No
Alias templates	No	No
Extern templates	Yes	Yes
`nullptr`	Yes	Yes
Strongly typed enums	Partial	Yes
Forward declared enums	No	Yes
Attributes	No	No
`constexpr`	No	No
Alignment	TR1	Partial
Delegating constructors	No	No
Inheriting constructors	No	No
Explicit conversion operators	No	No
`char16_t` and `char32_t`	No	No
Unicode string literals	No	No
Raw string literals	No	No
Universal character names in literals	No	No
User-defined literals	No	No
Standard-layout and trivial types	No	Yes
Defaulted and deleted functions	No	No
Extended friend declarations	Yes	Yes
Extended `sizeof`	No	No
Inline namespaces	No	No
Unrestricted unions	No	No
Local and unnamed types as template arguments	Yes	Yes
Range-based for-loop	No	No
`override` and `final` v0.8, v0.9, v1.0	Partial	Partial
Minimal GC support	Yes	Yes
`noexcept`	No	No

C++11 Core Language Features: Concurrency	VC10	VC11
Reworded sequence points	N/A	N/A
Atomics	No	Yes
Strong compare and exchange	No	Yes
Bidirectional fences	No	Yes
Memory model	N/A	N/A
Data-dependency ordering	No	Yes
Data-dependency ordering: function annotation	No	No
`exception_ptr`	Yes	Yes
`quick_exit` and `at_quick_exit`	No	No
Atomics in signal handlers	No	No
Thread-local storage	Partial	Partial
Magic statics	No	No

C++11 Core Language Features: C99	VC10	VC11
`__func__`	Partial	Partial
C99 preprocessor	Partial	Partial
`long long`	Yes	Yes
Extended integer types	N/A	N/A

Boost 라이브러리와 C++11

C++ ?

C#, Java ?

http://www.flickr.com/photos/adrianamorett/5470167595/

boost C++ LIBRARIES

"...one of the most highly regarded and expertly designed C++ library projects in the world."
— Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

WELCOME TO BOOST.ORG!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages both commercial and non-commercial use.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Ten Boost libraries are already included in the C++ Standards Committee's Library Technical Report (TR1) and will be in the new C++0x Standard now being finalized. C++0x will also include several more Boost libraries in addition to those from TR1. More Boost libraries are proposed for TR2.

GETTING STARTED

Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the Getting Started Guide to download and install Boost. Popular Linux and Unix distributions such as Fedora, Debian, and NetBSD include pre-built Boost packages. Boost may also already be available on your organization's internal web server.

BACKGROUND

Read on with the introductory material to help you understand what Boost is about and to help in educating your organization about Boost.

COMMUNITY

Boost welcomes and thrives on participation from a variety of individuals and organizations. Many avenues for participation are available in the Boost Community.

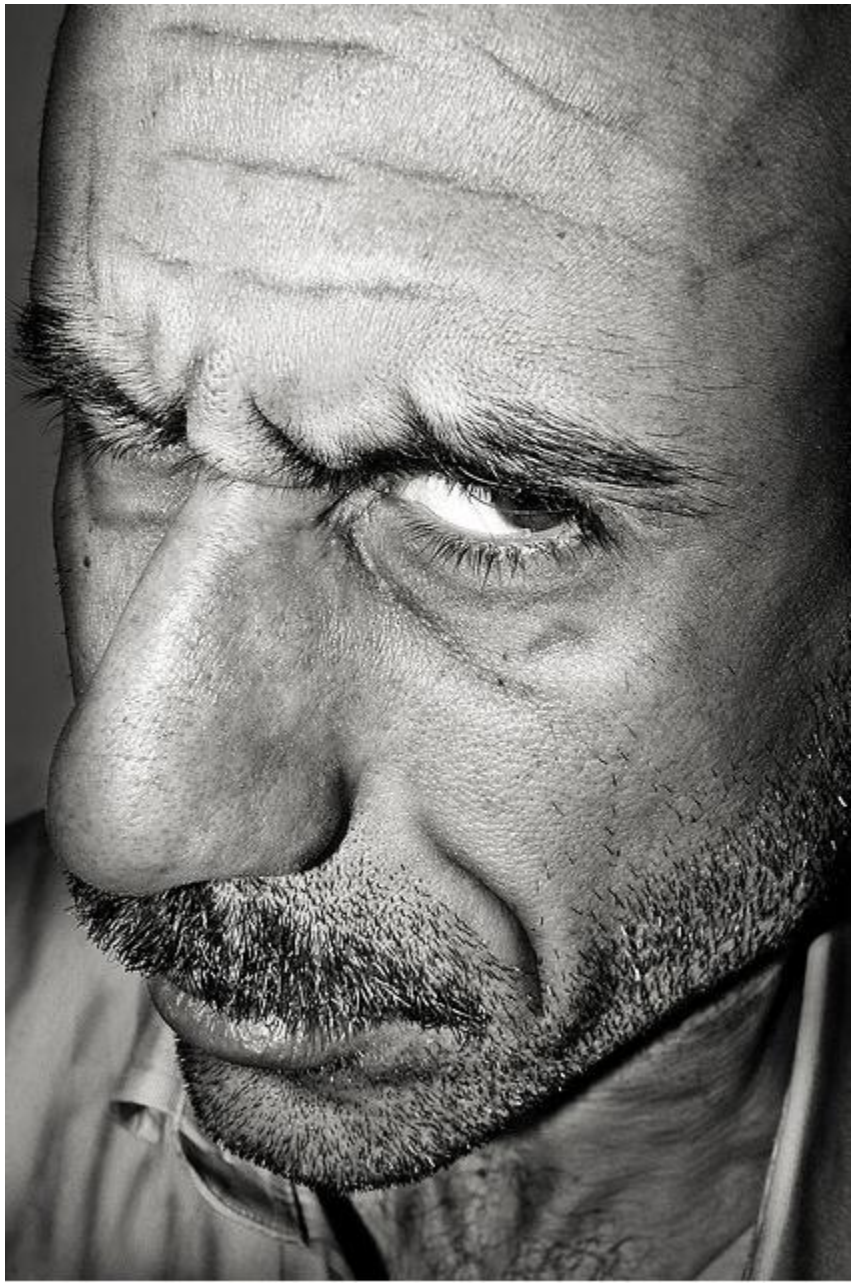
DOWNLOADS

- Version 1.47.0 (release notes)
  July 11th, 2011 18:19 GMT

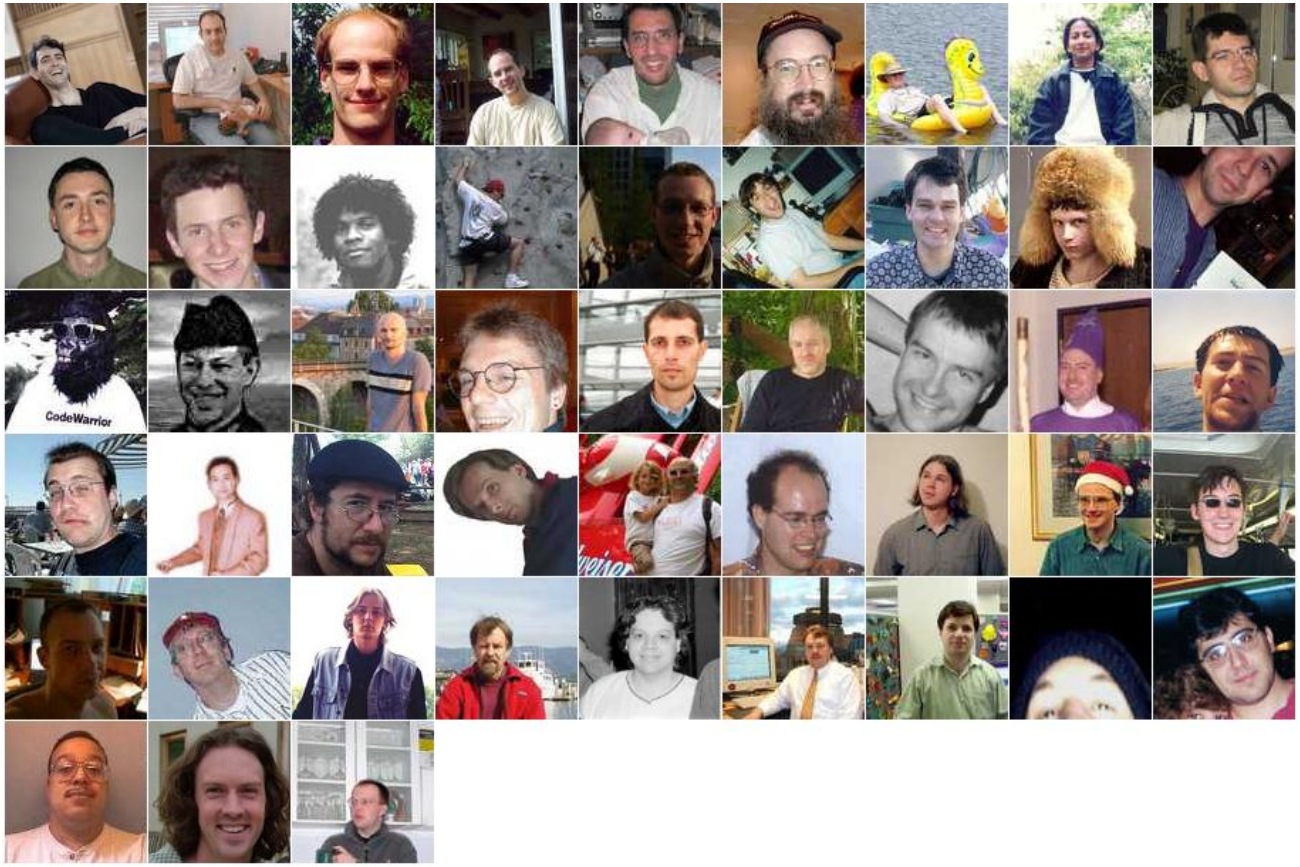
More Downloads... (RSS)

NEWS

- Version 1.47.0
  New Libraries: Chrono, Geometry, Phoenix and Ratio. Updated Libraries: Accumulators, Asio, Config, DateTime, Dynamic Bitset, Foreach, Function, Function Types, Graph, Iostreams, Iterator, Lexical Cast, Logic, Math, Meta State Machine, MultiIndex, Proto, Random, Range, Spirit, Tokenizer, Utility, Uuid, Wave
  July 11th, 2011 18:19 GMT
- Version 1.46.1
  Bug fixes: Asio, Fusion, Graph, Icl, Math, Polygon, Proto, Property Tree, Signals2, TR1, Unordered.
  March 12th, 2011 15:45 GMT
- Version 1.46.0
  New Libraries: Icl. Updated Libraries: Array, Asio, Bind, Concept Check, Filesystem, Fusion, Hash, Iostreams, Iterator, Math, Meta State Machine, Optional, Pool, Program Options, Proto, Signals, Spirit, Tokenizer, Unordered, Wave. Updated Tools: Boostbook, Inspect, Quickbook.
  February 21st, 2011 20:36 GMT



믿을 수 있어 ?


CodeWarrior

TR 1

array functional
shared_ptr weak_ptr
regex random
unordered_map
unordered_set

boostcon
BOOST LIBRARIES CONFERENCE 2011

home about location community program registration

news

BoostCon 2011 Trip Report

Boris Kolpackov of Code Synthesis has published his BoostCon 2011 trip report. If you've written publicly about your BoostCon experience please let us know so we can share your comments with others.

more...

Thank you!

I big thank you to attendees, presenters and volunteer organizers for making BoostCon 2011 a huge success! Check back as materials, photos and links to videos of the presentations will be put online. Please mark your

BoostCon 2011

May 15-20, 2011 – Aspen, Colorado

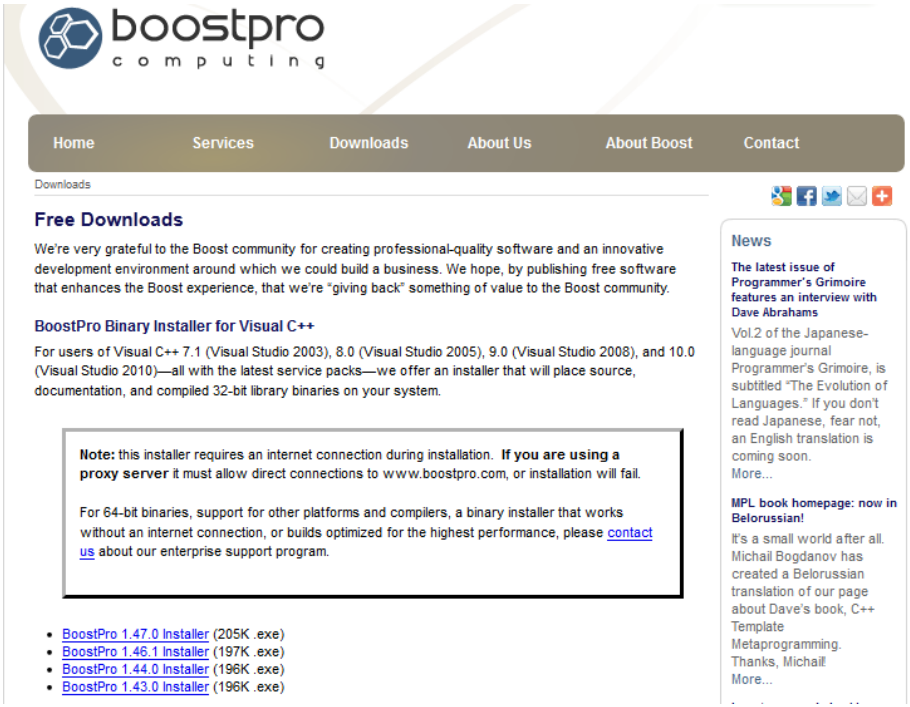
Promising to be **the** main face-to-face event for all things Boost, *BoostCon 2011* opens the door to your C++ future. From using the Boost libraries to writing and maintaining them, from evangelizing to deploying Boost within your organization, from infrastructure and process to vision and mission, and from TR1 to TR2, BoostCon brings together the sessions, the colleagues, and the inspiration to support your work with Boost for the next year.

To reflect the breadth of the Boost community, the conference includes sessions aimed at two constituencies: Boost end-users and hard-core Boost library and tool developers. The program fosters interaction and engagement within *and across* those two groups, with an emphasis on hands-on, participatory sessions.

C++ Now!

#include <boost/array>

```
D:\boost_1_42_0>bjam stage variant=debug  --stagedir="d:\boost_1_42_0" --with-fi
lesystem --with-thread --with-date_time --with-program_options --layout=versione
d threading=multi toolset=msvc-10.0
WARNING: No python installation configured and autoconfiguration
         failed.  See http://www.boost.org/libs/python/doc/building.html
         for configuration instructions or pass --without-python to
         suppress this message and silently skip all Boost.Python targets
...patience...
...found 825 targets...
...updating 72 targets...
common.mkdir d:\boost_1_42_0\lib
common.mkdir bin.v2
common.mkdir bin.v2\libs
common.mkdir bin.v2\libs\date_time
common.mkdir bin.v2\libs\date_time\build
common.mkdir bin.v2\libs\date_time\build\msvc-10.0
common.mkdir bin.v2\libs\date_time\build\msvc-10.0\debug
common.mkdir bin.v2\libs\date_time\build\msvc-10.0\debug\link-static
common.mkdir bin.v2\libs\date_time\build\msvc-10.0\debug\link-static\threading-m
ulti
common.mkdir bin.v2\libs\date_time\build\msvc-10.0\debug\link-static\threading-m
ulti\gregorian
compile-c-c++ bin.v2\libs\date_time\build\msvc-10.0\debug\link-static\threading-
multi\gregorian\greg_month.obj
greg_month.cpp
compile-c-c++ bin.v2\libs\date_time\build\msvc-10.0\debug\link-static\threading-
multi\gregorian\greg_weekday.obj
greg_weekday.cpp
compile-c-c++ bin.v2\libs\date_time\build\msvc-10.0\debug\link-static\threading-
multi\gregorian\date_generators.obj
```



C++ 강력하지만 생산성이 없다.....

C++ 강력함과 생산성이 뛰어난...


boost
C++ LIBRARIES

유용한 Boost 라이브러리 소개

01.Accumulators
02.Any
03.Array
04.Asio
05.Assign
06.Bimap
07.Bind
08.Circular Buffer
09.Compressed Pair
10.Concept Check
11.Conversion
12.CRC
13.Date Time
14.Dynamic Bitset
15.Enable If
16.Exception
17.Filesystem
18.Flyweight
19.Foreach
20.Format

21.Function
22.Function Types
23.Fusion
24.GIL
25.Graph
26.Interprocess
27.Interval
28.Intrusive
29.IO State Server
30.Iostreams
31.Iterators
32.Lambda
33.Math
34.Member Function
35.MPL
36.Multi Array
37.Multi Index
38.Numeric Conversion
39.Operators
40.Optional

41.Parameter
42.Pointer Container
43.Pool
44.Preprocessor
45.Property Map
46.Proto
47.Python
48.Random
49.Range
50.Ref
51.Scope Exit
52.Serialization
53.Signals2
54.Smart Pointers
55.Spirit
56.Statechart
57.Static Assert
58.String Algo
59.Swap
60.System

61.Test
62.Thread
63.Timer
64.Tokenizer
65.Tribool
66.Tuple
67.Typeof
68.uBLAS
69.Units
70.Unordered
71.Utility
72.Variant
73.Wave
74.Xpressive

출처 : http://d.hatena.ne.jp/faith_and_brave/20100828/1282973201

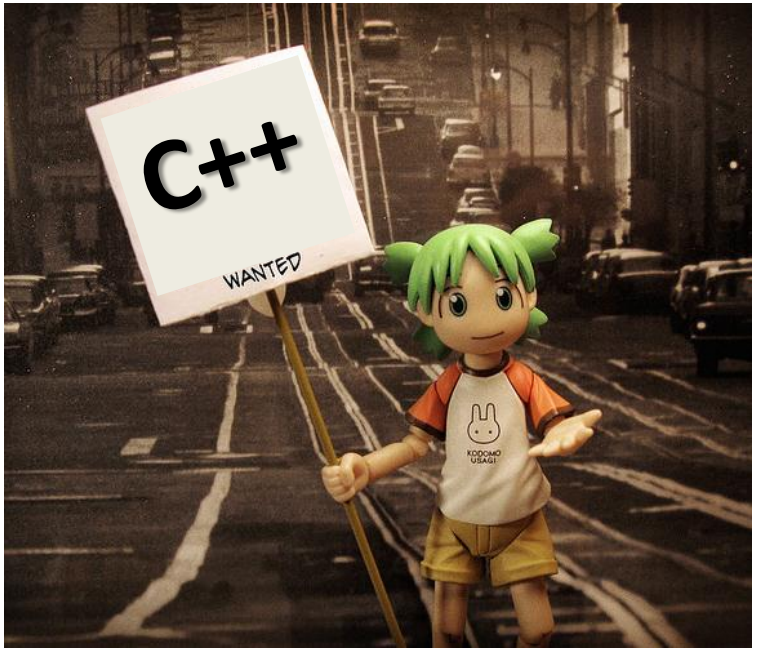
- Logging
- Task
- Lockfree
- SIMD
- ……

01.Accumulators
02.Any
03.Array
04.Asio
05.Assign
06.Bimap
07.Bind
08.Circular Buffer
09.Compressed Pair
10.Concept Check
11.Conversion
12.CRC
13.Date Time
14.Dynamic Bitset
15.Enable If
16.Exception
17.Filesystem
18.Flyweight
19.Foreach
20.Format

21.Function
22.Function Types
23.Fusion
24.GIL
25.Graph
26.Interprocess
27.Interval
28.Intrusive
29.IO State Server
30.Iostreams
31.Iterators
32.Lambda
33.Math
34.Member Function
35.MPL
36.Multi Array
37.Multi Index
38.Numeric Conversion
39.Operators
40.Optional

41.Parameter
42.Pointer Container
43.Pool
44.Preprocessor
45.Property Map
46.Proto
47.Python
48.Random
49.Range
50.Ref
51.Scope Exit
52.Serialization
53.Signals2
54.Smart Pointers
55.Spirit
56.Statechart
57.Static Assert
58.String Algo
59.Swap
60.System

61.Test
62.Thread
63.Timer
64.Tokenizer
65.Tribool
66.Tuple
67.Typeof
68.uBLAS
69.Units
70.Unordered
71.Utility
72.Variant
73.Wave
74.Xpressive



하나의 컨테이너에

int 타입의 데이터를

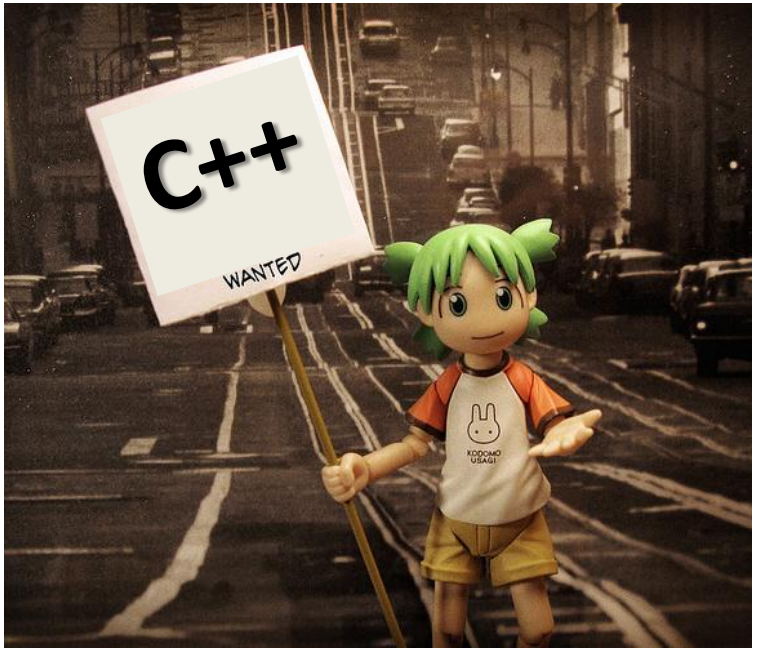
float 타입의 데이터를

string 타입의 데이터를

유저 정의 타입(구조체, 클래스) 데이터를

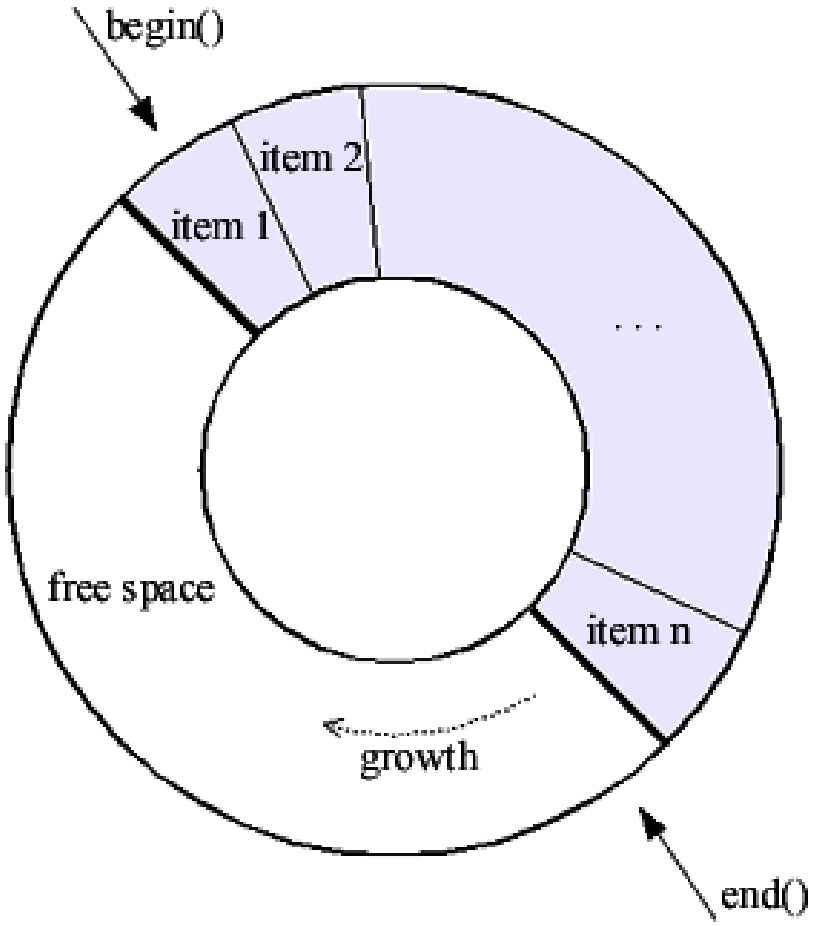
담고 싶어요!!!

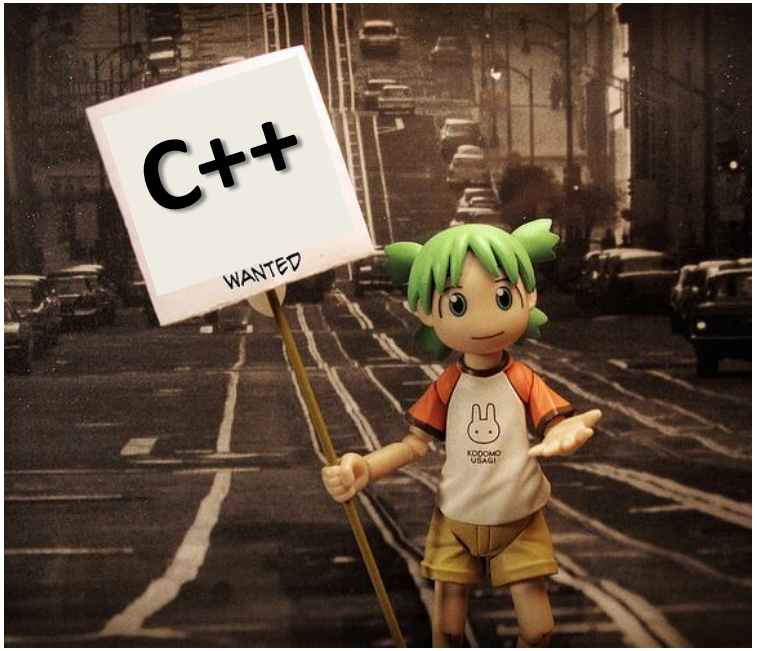
Any



원형 버퍼가 필요한데 만들어야 하나..?

Circular Buffer


begin()
item 2
item 1
. . .
free space
item n
growth
end()



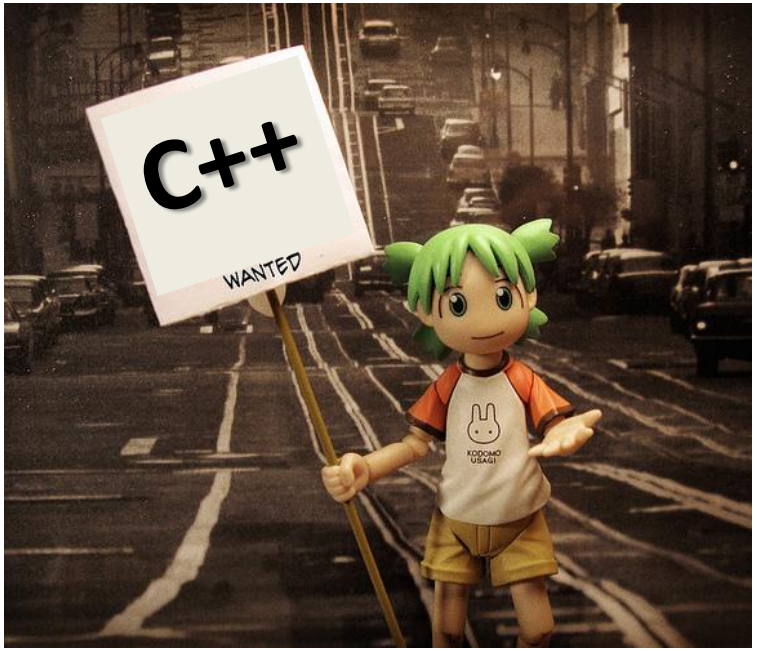
숫자를 문자로 바꾸어야 하는데....

문자를 숫자로 바꾸어야 하는데....

클래스나 구조체를 숫자나 문자로 바꾸어야 하는데.....

그리고 당근 안전하게!!!

lexical_cast



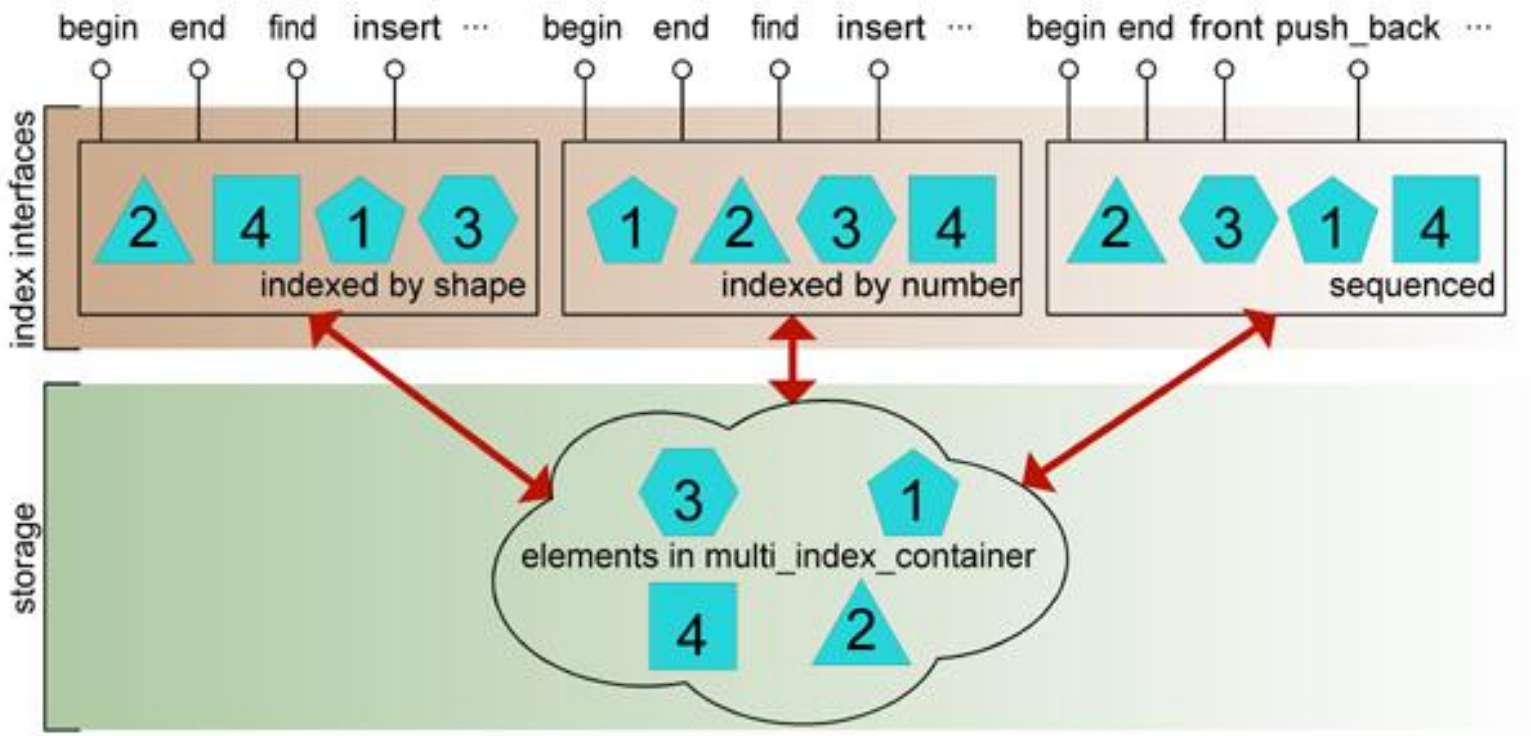
컨테이너에...

플레이어 식별 번호를 Key 값으로 저장,

플레이어 이름을 Key 값으로 저장...

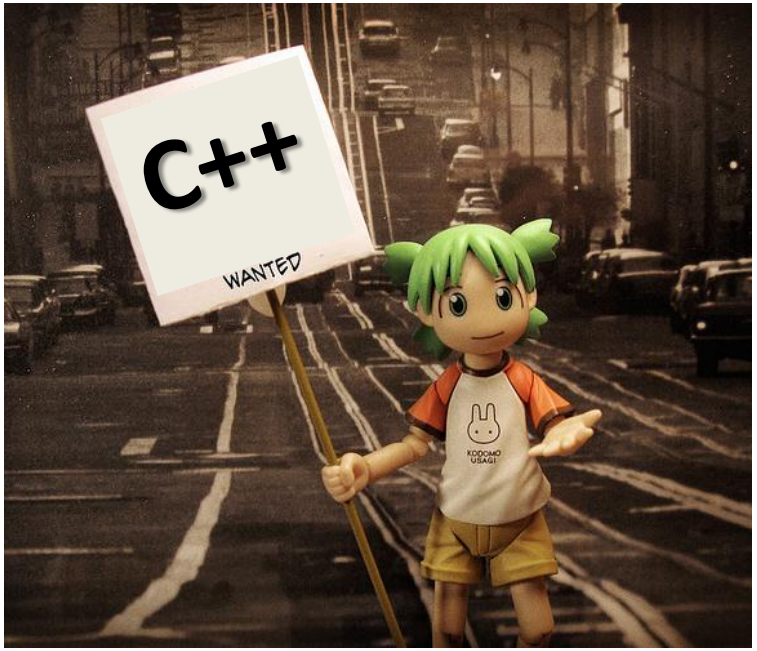
그런데 하나의 컨테이너만 사용했으면
좋겠는데....

multi_index



http://www.boost.org/doc/libs/1_47_0/libs/multi_index/doc/tutorial/index.html

C++ 11



디렉토리 생성/삭제...

디렉토리에 있는 파일 검색...

어떻게 하더라.....

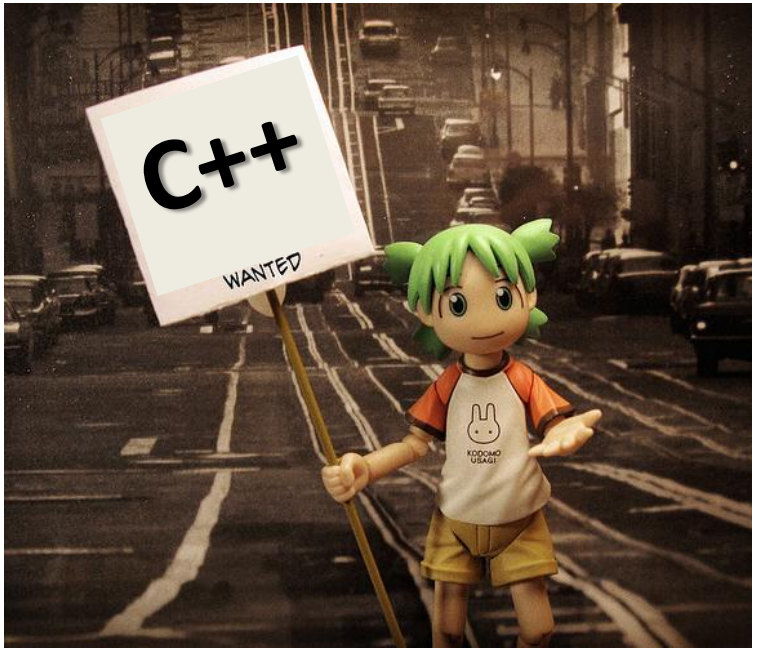
윈도우하고 리눅스의 API는 명령어가 서로 다르던데..

filesystem

```
using namespace boost::filesystem;

remove_all(“my_dir”);                    // 디렉토리 안의 모든 파일 삭제
create_directory(“my_dir”);              // 디렉토리 생성

ofstream file(“my_dir/a.txt”);           // 파일 읽기
file << “test\n";
file.close();

if (!exists(“my_dir/a.txt”)) {           // 파일이 있는지 조사
  std::cout << “파일이 없습니다“ << std::endl;
}
```

https://sites.google.com/site/boostjp/study_meeting/study1

C++ 11



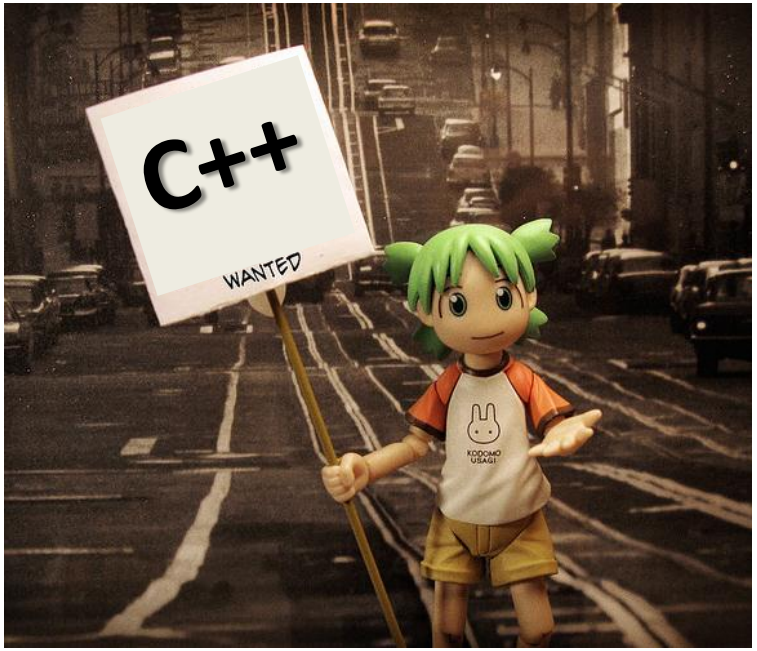
C++로 날짜 계산을 간단하게...

C++로 시스템의 Tick 타임이나 고행상도 시간을 측정하고 싶은데.....

chrono

```
#include <boost/chrono.hpp>
#include <cmath>

int main()
{
   boost::chrono::system_clock::time_point start =
                                        boost::chrono::system_clock::now();

   for ( long i = 0; i < 10000000; ++i ) {
     std::sqrt( 123.456L ); // burn some time
   }

   boost::chrono::duration<double> sec =
                        boost::chrono::system_clock::now() - start;

   std::cout << "took " << sec.count() << " seconds\n";
   return 0;
}
```

C++ 11



멀티 코어 시대...

공짜 점심은 없어졌다고 하는데...

쓰레드를 어떻게 사용하지 ?

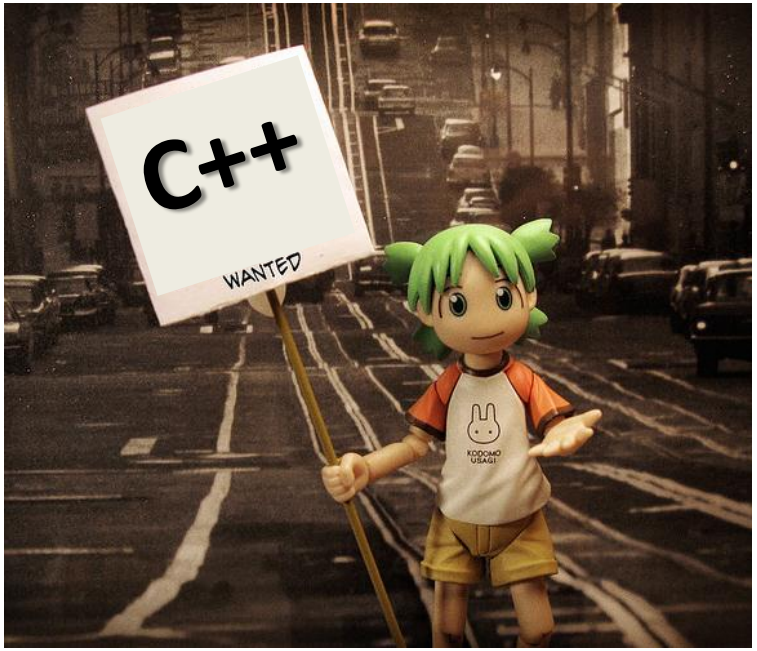
헐...윈도우와 리눅스의 쓰레드 프로그래밍 방식이 완전 다르네....

thread

```
void hello()
{
    cout << "Hello Concurrent World" << endl;
}


int main()
{
   boost::thread t(hello);
   t.join();
}
```

https://sites.google.com/site/boostjp/study_meeting/study1

C++ 11

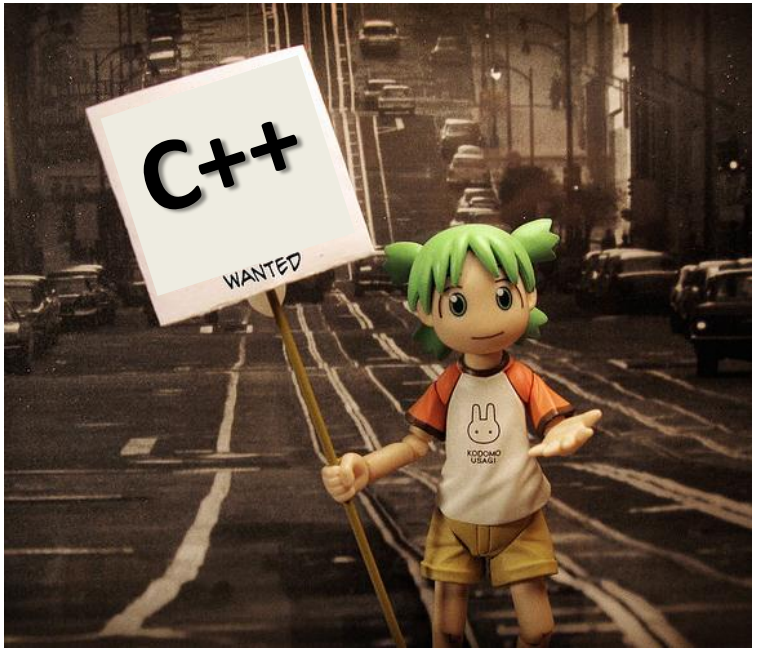


컨테이너의 범위를 더 쉽게 지정.....

range

```
template <class R, class T>
typename boost::range_iterator<R>::typefind(R& r, T x)
{
    return std::find(boost::begin(r), boost::end(r), x);
}

std::vector<int> v;
int ar[3];

std::vector<int>::iteratorit = find(v, 3);  // 컨테이너
int*p = find(ar, 3);                        // 배열
```

https://sites.google.com/site/boostjp/study_meeting/study1



다양한 네트웍 기능을 사용하는
프로그램을 만들어야 하는데 싶다...

고성능 네트웍 프로그램을 만들어야
하는데....

멀티 플랫폼이며, 당연 해당 플랫폼의
고성능 IO를 사용해야 하는데....

Asio

Boost.Asio란?

- Boost 라이브러리의 일부
- Asynchronous I/O (비동기 입출력)
- I/O와 같이 시간이 걸리는 처리를 OS의 비동기 기능과 스레드를 사용하여 처리
- 보통 네트워크 라이브러리로 알려져 있다
  그러나 파일 입출력이나 시리얼 입출력에서도 사용
- 멀티 플랫폼 지원

믿을 수 있나?

- 신뢰성이 높음
- 한국의 몇몇 온라인 게임에서 이미 사용 중
- 한국의 모 대형 IT 회사의 내부 네트웍 라이브러리 표준이 Boost.Asio로 정해져 있음

OS 플랫폼 별 구현

- Linux Kernel 2.4
  select를 사용하므로 FD_SIZE 크기를 넘지 못함

- Linux Kerner 2.6
  epoll을 사용

- FreeBSD, Mac OS X
  Kqueue를 사용

- Solaris
  /dev/poll을 사용

- Windows(Windows 2000 이후)
  Overlapped I/O와 I/O Completion을 사용

Boost.Asio에는 뭐가 있을까?

- boost::asio::io_service 가장 중요
- ip::tcp::socket (http에도 사용)
- ip::udp::socket
- ip::icmp::socket (ping 등에 사용)
- ssl::context (Open SSL이 필요)
- serial_port
- boost::deadline_timer

BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
socket descriptor - int (POSIX) or SOCKET (Windows)	For TCP: ip::tcp::socket, ip::tcp::acceptor For UDP: ip::udp::socket basic_socket, basic_stream_socket, basic_datagram_socket, basic_raw_socket
in_addr, in6_addr	ip::address, ip::address_v4, ip::address_v6
sockaddr_in, sockaddr_in6	For TCP: ip::tcp::endpoint For UDP: ip::udp::endpoint ip::basic_endpoint
accept()	For TCP: ip::tcp::acceptor::accept() basic_socket_acceptor::accept()
bind()	For TCP: ip::tcp::acceptor::bind(), ip::tcp::socket::bind() For UDP: ip::udp::socket::bind() basic_socket::bind()

BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
close()	For TCP: ip::tcp::acceptor::close(), ip::tcp::socket::close() For UDP: ip::udp::socket::close() basic_socket::close()
connect()	For TCP: ip::tcp::socket::connect() For UDP: ip::udp::socket::connect() basic_socket::connect()
getaddrinfo(), gethostbyaddr(), gethostbyname(), getnameinfo(), getservbyname(), getservbyport()	For TCP: ip::tcp::resolver::resolve(), ip::tcp::resolver::async_resolve() For UDP: ip::udp::resolver::resolve(), ip::udp::resolver::async_resolve() ip::basic_resolver::resolve(), ip::basic_resolver::async_resolve()
gethostname()	ip::host_name()

BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
getpeername()	For TCP: ip::tcp::socket::remote_endpoint() For UDP: ip::udp::socket::remote_endpoint() basic_socket::remote_endpoint()
getsockname()	For TCP: ip::tcp::acceptor::local_endpoint(), ip::tcp::socket::local_endpoint() For UDP: ip::udp::socket::local_endpoint() basic_socket::local_endpoint()
getsockopt()	For TCP: ip::tcp::acceptor::get_option(), ip::tcp::socket::get_option() For UDP: ip::udp::socket::get_option() basic_socket::get_option()
inet_addr(), inet_aton(), inet_pton()	ip::address::from_string(), ip::address_v4::from_string(), ip_address_v6::from_string()

BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
inet_ntoa(), inet_ntop()	ip::address::to_string(), ip::address_v4::to_string(), ip_address_v6::to_string()
ioctl()	For TCP: ip::tcp::socket::io_control() For UDP: ip::udp::socket::io_control() basic_socket::io_control()
listen()	For TCP: ip::tcp::acceptor::listen() basic_socket_acceptor::listen()
poll(), select(), pselect()	io_service::run(), io_service::run_one(), io_service::poll(), io_service::poll_one() Note: in conjunction with asynchronous operations.

BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
readv(), recv(), read()	For TCP: ip::tcp::socket::read_some(), ip::tcp::socket::async_read_some(), ip::tcp::socket::receive(), ip::tcp::socket::async_receive() For UDP: ip::udp::socket::receive(), ip::udp::socket::async_receive() basic_stream_socket::read_some(), basic_stream_socket::async_read_some(), basic_stream_socket::receive(), basic_stream_socket::async_receive(), basic_datagram_socket::receive(), basic_datagram_socket::async_receive()
recvfrom()	For UDP: ip::udp::socket::receive_from(), ip::udp::socket::async_receive_from() basic_datagram_socket::receive_from(), basic_datagram_socket::async_receive_from()

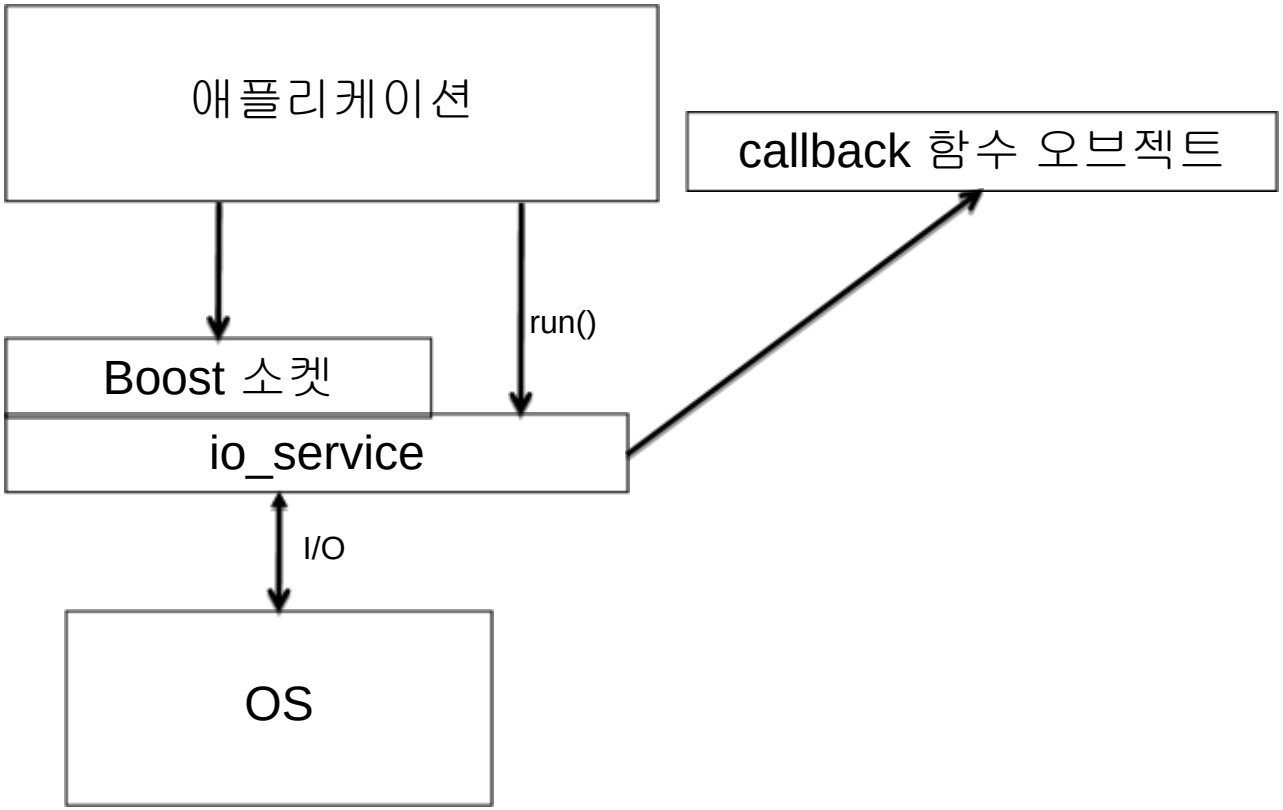
BSD Socket - Boost.Asio

BSD Socket API	Boost.Asio
send(), write(), writev()	For TCP: ip::tcp::socket::write_some(), ip::tcp::socket::async_write_some(), ip::tcp::socket::send(), ip::tcp::socket::async_send() For UDP: ip::udp::socket::send(), ip::udp::socket::async_send() basic_stream_socket::write_some(), basic_stream_socket::async_write_some(), basic_stream_socket::send(), basic_stream_socket::async_send(), basic_datagram_socket::send(), basic_datagram_socket::async_send()
sendto()	For UDP: ip::udp::socket::send_to(), ip::udp::socket::async_send_to() basic_datagram_socket::send_to(), basic_datagram_socket::async_send_to()
setsockopt()	For TCP: ip::tcp::acceptor::set_option(), ip::tcp::socket::set_option() For UDP: ip::udp::socket::set_option() basic_socket::set_option()

BSD Socket - Boost.Asio

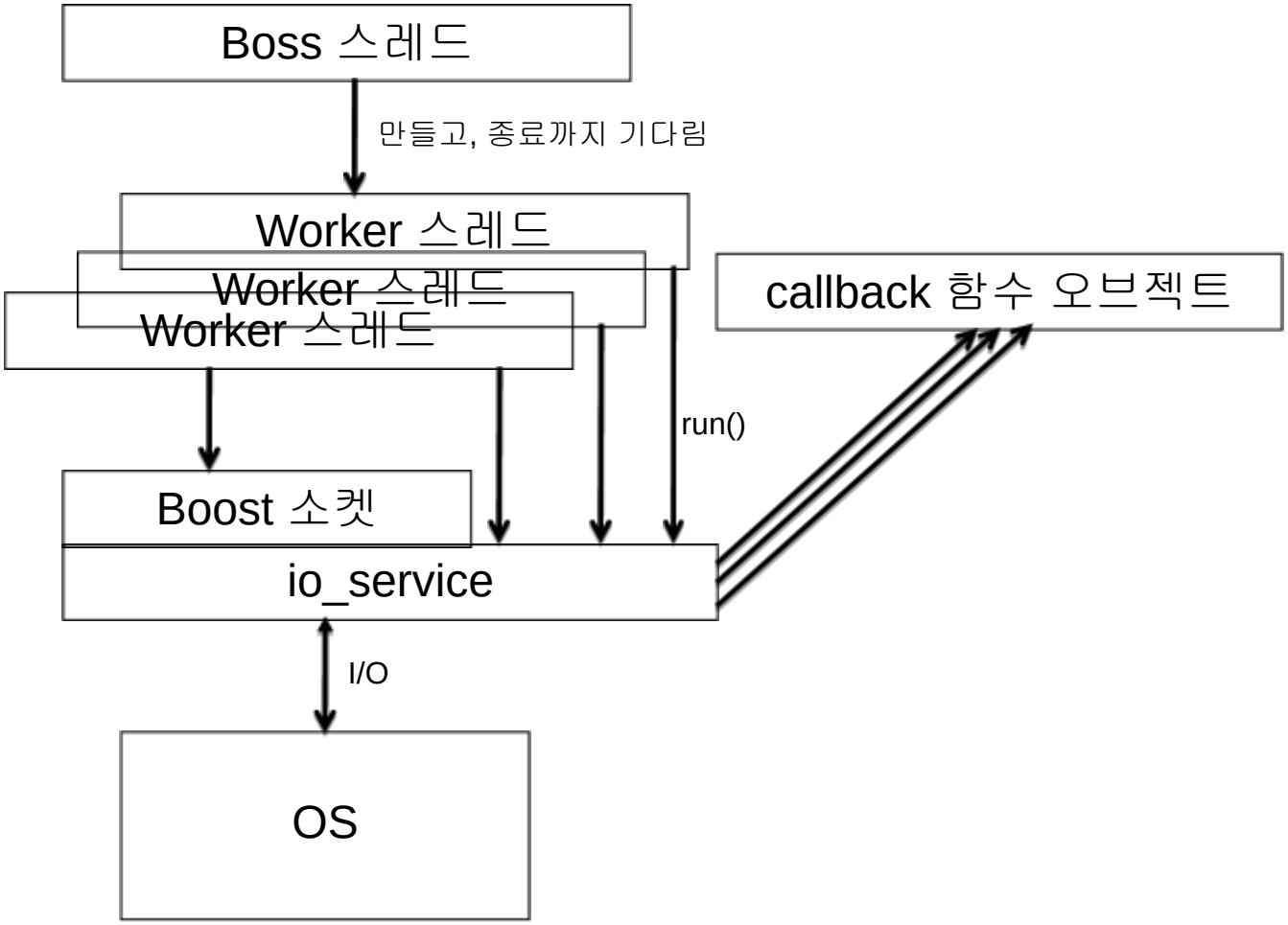
BSD Socket API	Boost.Asio
shutdown()	For TCP: ip::tcp::socket::shutdown() For UDP: ip::udp::socket::shutdown() basic_socket::shutdown()
sockatmark()	For TCP: ip::tcp::socket::at_mark() basic_socket::at_mark()
socket()	For TCP: ip::tcp::acceptor::open(), ip::tcp::socket::open() For UDP: ip::udp::socket::open() basic_socket::open()
socketpair()	local::connect_pair() Note: POSIX operating systems only.

Asio의 비동기 모델 - 스레드 모델



참조 : http://d.hatena.ne.jp/Softgels/20090304/1236123151

Asio의 비동기 모델 - 멀티 스레드 모델



참조 : http://d.hatena.ne.jp/Softgels/20090304/1236123151

Non-Boost Asio

- Boost 라이브러리와 독립된 Asio
- Boost 라이브러리의 Lib 파일을 사용하지 않고 싶을 때
- 라이브러리 빌드 없이 헤더 파일 추가로만 사용 가능
  그래도 Boost 라이브러리의 헤더 파일 추가는 필요
- http://think-async.com/Asio

Boost.Asio의 가치(서버 프로그래머)

- 서버 프로그래머의 능력과 경험에 따라 다름
- 경험이 적거나 신뢰 있는 네트웍 라이브러리 구축을 원할 때는 추천
- 멀티 플랫폼용 네트웍 라이브러리를 원할 때도 추천
- 이미 신뢰 받는 네트웍 라이브러리를 만든 경험이 있고, 멀티 플랫폼을 고려하지 않는다면.....

참고

C++11 Features in Visual C++ 11
http://blogs.msdn.com/b/vcblog/archive/2011/09/12/10209291.aspx

Boost 라이브러리 공식 홈페이지
http://www.boost.org/

Boostpro
http://www.boostpro.com/download/

Boost e-Book : The Boost C++ Libraries
http://en.highscore.de/cpp/boost/

boostcon / 2011 Presentations
https://github.com/boostcon/2011_presentations

Boost Asio 간단한 예제
http://javawork.egloos.com/1813629