C++ Programming HOW-TO

Al Dev (Alavoor Vasudevan)

alavoor (at) yahoo.com

도운이

정지용

ways (at) gon.kaist.ac.kr

이 문서는 C++에 관한 URL과 C++ online 책에 대한 링크, C++ 프로그래밍 팁 등을 포괄적으로 제공한다. 이 문서는 C++에서 겪는 다양한 메모리 관련 문제를 해결해주는 Java와 비슷한 library도 제공한다. 이 library를 이용하면, java 소스코드를 C++로 컴파일 할 수도 있다. 이 문서는 "C++ 언어의 집"역할을 한다. 이 문서에서 제공되는 정보는 C++ 언어를 적절히 사용하고 모든 운영체제 - 리눅스, MS-DOS, BeOS, Apple Macintosh OS, Microsoft Windows 95/98/NT/2000, OS/2, IBM OSes (MVS, AS/400 등..), VAX VMS, Novell Netware, 유닉스-like OS들(Solaris, HPUX, AIX, SCO, Sinix, BSD 등..)과 기타 C++을 지원하는 모든 운영체제(아마도 세상의 거의 모든 운영체제가 이에 속할 것이다) - 에 적용하는 것을 도와줄 것이다.

고친 과정

고침 41.82002-03-31고친이 정지용

수정판 번역

고침 5.0 2000-06-15고친이 김지희, 윤주철

최초 번역

1. 서문

(이 문서의 최신판은 **http://www.milkywaygalaxy.freeservers.com** 에서 구할 수 있다)

이 문서의 목적은 C++에 관한 URL과 C++ online 책에 대한 링크, C++ 프로그래밍 팁 등을 포괄적으로 제공하는 것이다. 또한, 이 문서는 Java 스타일의 String class, string tokenizer, 메모리 함수등 일반적인 C++ 프로그램에서 널리 쓰일 수 있는 많은 함수들을 제공한다. C++ 과 Java는 오늘날 많은 소프트웨어 프로젝트에서 쓰이고 있다. 프로그래머는 C++과 Java를 번갈아가며 쓰게될 것이고, 이 Java 스타일의 class가 매우 유용함을 알게 될 것이다. 이 library와

C++ 표준 라이브러리의 사용법을 알려줄 다양한 예제가 제시될 것이다.

이 문서는 C++에 대한 교과서가 아니며, 이에 대해서는 이미 몇가지 좋은 on-line 책들이 있다. C++이 꽤 오랜시간동안 사용되어왔기 때문에, 매우 많은 수의 C++ 문서/글/튜토리얼이 인터넷상에 존재한다. 만약 당신이 C++을 처음 접하는 것이고, C++ 프로그램을 짜 본 적이 없다면, 13절장에 링크되어 있는 on-line C++ 책을 먼저 읽어보거나, Amazon (http://www.amazon.com) 이나 barnes (http://www.barnesnoble.com)과 같은 곳에서 C++ 책을 사 볼 것을 추천한다.

누군가가 말했듯 - **C/C++** 언어는 **OS**나 디바이스드라이버**,** 빠른 응답을 필요하는 **real-time** 프로그램등을 만드는 시스템 엔지니어나 쓰라고 하고**,** 당신은 **2002**년보다 컴퓨터가 몇백만배 빨라질 **2005**년을 생각하면 **Java**나 **PHP-scripting**을 써야한다**.** 하드웨어 는 점점 싸면서도, 빨라진다.

1.1. C++ 프로그램? C++ vs. Java/PHP

C++은 가장 강력한 언어들 중 하나이고, Java나 PHP-scripting 같은 것이 나타났지만, 앞으로도 오랜 시간동안 쓰이게 될 것이다. 실시간의 매우 빠른 응답을 필요하는 프로그램은 C나 C++을 쓴다. C++은 매우 빠르게동작하고, 실제로 Java보다 **10**배에서 **20**배 정도 빠르다 . Java는 C++의 "자손"이다. Java의 단 하나의 문제점은 바로 - **"Java** 는 느리다**!!"** . VM위에서 도는 Java 바이트코드는 컴파일된 실행코드보다 느리다. Java는 JIT(Just-In-Time) 컴파일러위에서 더 빠르게 돌지만, 여전히 C++보다는 느리다. 최적화 된 C/C++ 프로그램은 JIT 나 그 이전의 컴파일러로 컴파일 된 Java 코드보다 약 **3** 에서 **4**배 정도 빠르다!! 그렇다면, 왜 사람들이 Java를 쓰는가? 이는 Java가 순수한 객체지향을 지원하고, Java의 자동화된 메모리 관리로 인해 프로그래밍하기가 쉬우며, 프로그래머들이 직접 메모리 관리하기를 싫어하기 때문이다. 이 문서는 C++의 메모리 관리를 자동화하여 훨씬 사용하기 쉽게 하고자 했다. 여기서 나오는 library는 C++을 Java 만큼 쉽게 느끼게 해줄 것이고, C++이 Java와 경쟁할 수 있도록 해줄 것이다.

수동적인 메모리 관리를 위해 C++ 프로그램 디버깅의 대부분 시간이 소모된다. 이 문서는 디버깅 시간을 줄이기 위한 몇가지 아이디어와 팁을 줄 것이다.

언제 C++을 써야하고 언제 Java/PHP를 써야하는가?

아래와 같은 경우엔 C++을 써라:

- 실행속도와 성능이 매우 중요한 프로그램을 만들 때.
- 만드는 프로그램의 사용자 수가 많을 때. C++은 컴파일-링킹-디버깅 사이클이 필요하기 때문에, 프로그램 개발에 더 많은 시간이 소요된다. 따라서 사용자수가 충분히 많을 때에나 적당하다. 실행파일을 만들기 위해 많은 수의 object파일을 링크하는 것은 꽤 시간이 걸린다. (링크하는데 걸리는 시간을 줄이기 위해 archive나 라이브러리, 공유 라이브러리를 사용할 수도 있다.)
- C++프로그래밍 경험이 많을 때.

Java/PHP를 써야할 경우:

- (C/C++로 작성되는 것에 비해서) 실행속도와 성능이 중요하지 않을 때.
- 생산 비용을 낮추기 위해 - 컴파일-링크 사이클이 없기 때문에, Java/PHP는 C++보다 개발이 빠르다.
- 빠른 개발이 필요할 때.
- 코드 유지보수를 쉽게 하기 위해. C++ 을 유지보수하는 것이 Java나 PHP-scripting 보다 훨씬 어렵다.
- Java와 PHP-scripting은 미래이다, 하드웨어의 속도는 분자와 원자, 원자보다 작은 크기의 컴퓨터 도입과 함께 급등할 것이다. 미래의 컴퓨터는 오늘날 컴퓨터의 수조배의 성능을 가질 것이다. 미래에 하드웨어 성능이 진보함과 함께, Java나 PHP-script의 실행 성능은 중요치 않게 될 것이다. 오늘날 당신이 쓰는 컴퓨터(현재는 2002년이다.)는 엄청나게 느리고, 기어가고 있으며, 충분히 빠르지 못하다.

NOTE: Java 컴파일러 (JIT 나 다른 것들)에 많은 진보가 있었다. Java 프로그램은 GNU GCJ http://gcc.gnu.org/java로 컴파일 될 수 있다. GCJ는 간편하고, 최적화되어있으며, 진보적인 Java 프로그래밍 언어를 위한 컴파일러이다. GCJ는 Java source 코드를 머신 코드로 바로 컴파일 할 수도 있고, Java 바이트코드(class file)로 컴파일 할 수도 있다.

GCJ 정보:

- GNU GCJ 홈페이지http://gcc.gnu.org/java,
- GNU GCJ 의 Redhat RPM http://www.redhat.com/apps/download. 여기로 가서 "Find lastest RPMs" section에서 'gcc-java'와 'libgcj'로 search하면 된다.
- Redhat GCJ 설치 가이드 http://www.redhat.com/devnet/articles/gcj.pdf

1.2. 어떤 것을 선택할 것인가. Ada95, C, C++, Java 아니면 PHP?

언어의 선택은 어려운 일이다. 여기엔 너무나 많은 고려할 사항이 있다 - 개발자, 사람의 능력, 비용, 툴들, 정책 (국가의 정치 정책까지도), 사업가나 회사들에 대한 영향까지. 기술적인 이유로는 최적의 언어일지라도 단순히 정치적인 결정으로 인해 선택되지 못할 수도 있다.

David Wheeler의 언어 비교를 보자. Ada 비교 차트 (http://www.adahome.com/History/Steelman/steeltab.htm). Ada가 93%, Java는 72%, C++은 68% C는 53%을 각각 받았다. C++과 Java는 점수면에서는 비슷하다 (4% 차이). Ada의 개발 비용은 Stephen F. Zeigler (http://sw-eng.falls-church.va.us/AdaIC/docs/reports/cada/cada_art.html)에 따르면 C++의 반절이다. Ada95는 아래에서 구할 수 있다 -

- Ada 홈페이지 http://www.gnuada.org.
- Google Ada index (http://directory.google.com/Top/Computers/Programming/Languages/Ada)

C++ 컴파일러는 C 컴파일러보다 훨씬 복잡하고, C++은 C보다 약간 느리게 동작할 수 있다. C 컴파일러는 충분히 오랫동안 잘 쓰여져왔다.

몇몇 시스템에서, 당신은 생성된 코드를 최적화 하기 위해 몇가지 옵션을 쓸 수 있다.

오늘날, C는 주로 운영체제나 디바이스 드라이버, 빠르게 작동해야하는 프로그램을 쓰기 위해 로우레벨 시스템 프로그래밍에 쓰인다.

Note: 이 **HOWTO**에 제공되는 **String, StringBuffer, StringTokenizer class**를 이용하여 **C++** 코드를 **Java**와 완전히 똑같이 쓸 수 있다**.** 이 문서의 일부는 **C++**로 **Java class**를 흉내냄으로써 **C++**과 **Java**의 차이를 줄이고자 했다**. C++**과 **Java**를 왔다갔다하는 **Java** 프로그래머들은 이 **String class**를 좋아할 것이다**.**

만약 C++의 작성-컴파일-디버깅-컴파일 싸이클이 싫다면, web 개발이나 일반적이 프로그래밍에 쓰일 수 있는 PHP같은 script 언어를 알아보아라. PHP나 PERL같은 script언어는 빠른 어플리케이션 개발을 가능하게 한다. PHP는 몇가지 객체지향을 위한 특징도 갖고 있다. PHP HOWTO는 http://www.linuxdoc.org/HOWTO/PHP-HOWTO.html (한글번역 (http://kldp.org/HOWTO/html/PHP/PHP-HOWTO.html)) 에서 볼 수 있다.

1.3. 현재 C++ 컴파일러의 문제점

C++은 C를 포함하기 때문에, C의 *나쁜* 점들을 모두 갖고 있다. 메모리의 수동 할당과 해제는 지루하고, 에러를 만들어내기 일쑤이다. (9.3절 를 보라).

C 프로그래밍에서는 다음과 같은 것들로 인해 메모리 릭이나 오버플로우가 매우 흔하다.

```
Datatype  char * and char[]
String functions like strcpy, strcat, strncpy, strncat, etc..
Memory functions like malloc, realloc, strdup, etc..
```

char ***와 **strcpy의 사용은 **"**오버플로우**"**, **"**경계침범에러**(fence past errors)"**, **"**메모리 오염**(memory corruption)"**, **"**다른변수 침범**(step-on-others-toe)"** 이나 **"**메로리 릭**(memory leaks)"** 등의 끔찍한 메모리 문제를 일으킨다. 메모리 문제는 매우 디버깅이 힘들고, 따라서 고치기는데 많은 시간이 든다. 메모리 문제는 프로그래머의 생산성을 떨어뜨린다. 이 문서는 C++의 이러한 단점을 해결하기 위해 고안된 여러가지 방법들을 통해 프로그래머의 생산성을 높이는데 도움을 주고자 한다. 메모리 관련 버그는 잡기 힘들고, 경험많은 프로그래머들도 메모리 관련 문제를 고치는 데는 며칠에서 몇주가 걸린다. 메모리 버그는 몇달동안 코드 속에 숨어서 갑작스런 프로그램 정지를 일으킬 수 있다. **char *** 와 C/C++에서의 포인터 사용으로 인한 메모리 버그는 디버깅과 프로그램 정지로 인해 매년 20억 달러에 해당하는 시간의 소모를 일으킨다. 만약 C++에서 **char *** 과 포인터 를 사용한다면, 이는 매우 힘든 일이 될 것이다. 특히 프로그램의 크기가 10,000 줄 이상일 때.

따라서, 아래의 것들이 C-style에서의 문제점을 극복하기 위해 제안되었다. 앞에 나오는 것이 더 좋은 것이다.

1. 포인터 대신 레퍼런스를 사용한다.
2. (이 HOWTO에 주어진) Java 형식의 class를 사용하거나, C++ 표준라이브러리의 string class를 사용한다.

3. C++에서의 문자 포인터(char *) 사용은 String class를 사용하지 못할 때로 그 사용을 제한한다.
4. 만약 C++에서의 문자 포인터(char *)를 사용하고 싶지 않을 때는, extern 연계를 이용하는 (char *)를.

"C의 char *"를 사용하기 위해서는, C 프로그램을 다른 파일에 넣고, 연계명시 문 **extern "C"**를 이용하여 C++ 프로그램에 링크한다 -

```
extern "C" {
#include <some_c_header.h>
}

extern "C" {
    comp();
    some_c_function();
}
```

extern "C" 는 연계 명시이고, 양 중괄호로 둘러싸인 블록안의 모든 내용이 C++이 아닌 C의 연계 방법을 사용한다는 말이다.

'String 클래스**'**는 메모리 할당과 해제를 위해 생성자와 파괴자를 이용하고, **ltrim**, **substring** 등등과 같은 함수를 제공한다.

또한 관련된 7절 를 사용하는 C++ 컴파일러에서 찾아보아라. **string** 클래스는 표준 C++ 라이브러리의 일부이고, 여러가지 문자열 관련 함수를 제공한다.

C++ **'string** 클래스**'** 와 **'String** 클래스**'** 라이브러리가 많은 문자열 함수를 제공하기 때문에, 직접 문자열 함수를 쓰기 위해 문자 포인터를 사용할 필요성이 거의 없다. 또한, C++ 프로그래머는 항상 'malloc'이나 'free'대신 'new', 'delete'를 사용해야 한다.

두 문자열 클래스는 **char *** 나 **char []**가 할 수 있는 모든 일을 할 수 있다. 그리고 보태진 좋은 점은 메모리 문제나 메모리 할당에 대해 전혀 걱정할 필요가 없다는 것이다.

2. 추천할만한 C++ 컴파일러

ISO와 ANSI에 의해 채택된 현재의 C++ 표준은 1997년에 처음 완성되었다. 아직 모든 컴파일러가 이를 따르고 있지 않고, 모든 특징들이 다 지원되지는 않는 다는 것이다. - 표준에 맞는 컴파일러를 쓰는 것은 매우 중요하다.

2.1. MS Windows 2000/NT/95/98/ME/XP를 위한 컴파일러

MS Windows는 C++개발로 꽤 유명하기 때문에, 이 문서에 주어진 String class 라이브러가 잘 작동하고, Windows XP/2000/NT/95/98/ME 등 모든 버전에서 잘 작동한다. MS Windows를 위한 C++ 컴파일러 :

- GNU BloodShed http://www.bloodshed.net/devcpp.html
- Borland C++ compiler http://www.borland.com/bcppbuilder/freecompiler
- Microsoft Visual C++ compiler http://msdn.microsoft.com/visualc
- MSDOS C++ compiler http://www.delorie.com/djgpp

이 문서의 String 클래스는 위에 언급된 모든 컴파일러로 테스트 되었고, MS Visual C++ v6.0, Borland C++ v5.2, Borland C++ v5.5.1, Bloodshed 컴파일러에서 잘 동작한다.

2.2. UNIX와 기타 UNIX-like 시스템들의 컴파일러

GNU 세계에서는 GCC(GNU Compiler Collection)를 사용하는 것이 가장 좋은 선택이다. GCC는 대부분의 리눅스 배포판, FreeBSD, 기타 Unix 클론들에 들어있다. GCC 홈페이지는 http://gcc.gnu.org이다. 최신버전의 GCC(3.0)은 가장 표준을 잘 준수한 컴파일러 중 하나이다.

3. 여러가지 문자열관련 class

문자열 class는 프로그래밍에서 가장 중요한 것들 중 하나이고, 문자열 조정을 위해 매우 많이 쓰인다. 문자열 class는 여러가지가 있고, 물론 이들을 상속받음으로써 자신만의 문자열 class를 만들 수도 있다.

- 이 문서에 쓰여진 문자열 클래스는 23절를 보아라.
- 표준 C++ 라이브러리 string class (ANSI/ISO string class http://www.msoe.edu/eecs/cese/resources/stl/string.htm 와 http://www.sgi.com/tech/stl/basic_string.html
- Qt의 외부 라이브러리에 있는 Qt String class http://doc.trolltech.com/qstring.html , mirror : http://www.cs.berkeley.edu/~dmartin/qt/qstring.html
- 이들 중 맘에 드는 것이 없다면, 자신만의 문자열 클래스를 만들 수도 있다. 위에 언급된 하나 혹은 여러 class를 상속받아 만들 수도 있다.

3.1. Multiple Inheritance - 자신만의 문자열 class 만들기

위에 말한 것 같이, 하나 혹은 여러 class를 상속받아 자신만의 문자열 class를 만들 수도 있다. 여기서는 표준 C++ 라이브러리의 string class와 부록 A의 String class 를 상속받음으로써 다중 상속을 이용한 문자열 class를 만들어 볼 것이다.

우선 예제 파일 'string_multi.h'를 23절 에서 다운로드 받아라.

이 파일은 다음과 같다 :

```
// *****************************************************************
// String class와 표준 라이브러리의 "string" class를 상속받음으로써
// 직접 문자열 class를 만들어보는 예시를 위한 프로그램
// *****************************************************************

#ifndef __STRING_MULTI_H_ALDEV_
#define __STRING_MULTI_H_ALDEV_

#include <string>
#include "String.h"
#include "StringBuffer.h"

#ifdef NOT_MSWINDOWS
#else
using namespace std;  // MS Visual C++ compiler Version 6.0 에서 필요함.
#endif

// 중요! : C++에서는 생성자, 파괴자, 복사 연산자가 같이 상속되지 않는다.
// 따라서 만약 =, + 등의 연산자가 base class에 정의되어 있고, base
// class의 생성자를 이용한다면, 반드시 같은 역할을 하는 생성자를
// 상속받는 class에도 만들어주어야 한다.
// 아래에 주어진 mystring(), mystring(char [])를 보아라.
//
// 또한 atmpstr이 mystring으로 선언되었다고 할 때, atmpstr + mstr
// 과 같이 연산자를 쓸 때, 실제로 불리는 것은 atmpstr.operator+(mstr)이다.

class mystring:public String, string
{
public:
mystring():String() {}  // =, + 연산자를 위해 필요하다
mystring(char bb[]):String(bb) {}  // =, + 연산자를 위해 필요하다

mystring(char bb[], int start, int slength):String(bb, start, slength) {}
mystring(int bb):String(bb) {}  // + 연산자를 위해 필요하다
mystring(unsigned long bb):String(bb) {}  // + 연산자를 위해 필요하다
mystring(long bb):String(bb) {}  // + 연산자를 위해 필요하다
mystring(float bb):String(bb) {}  // + 연산자를 위해 필요하다
mystring(double bb):String(bb) {}  // + 연산자를 위해 필요하다
mystring(const String & rhs):String(rhs) {}  // + 연산자를 위해 필요
한 Copy Constructor
mystring(StringBuffer sb):String(sb) {}  // Java와의 호환을 위해
```

```
mystring(int bb, bool dummy):String(bb, dummy) {}  // String-
Buffer class를 위
해

int mystraa; // mystring의 최적화
private:
int mystrbb; // mystring의 최적화
};

#endif // __STRING_MULTI_H_ALDEV_
```

4. String class 다운받기

모든 프로그램과 예제는 이 문서의 부록에 주어진다. String class와 라이브러리, 예제 프로그램을 하나의 tar zip 압축파일로 묶어놓은 링크가 다음과 같다.

- http://www.milkywaygalaxy.freeservers.com 로 가서 "Source code C++ Programming howto" (Milkyway Galaxy site (http://www.milkywaygalaxy.freeservers.com)) 를 눌러라.
- 미러 사이트는 다음과 같다 - angelfire (http://www.angelfire.com/country/aldev0), geocities (http://www.geocities.com/alavoor/index.html), virtualave (http://aldev0.virtualave.net), 50megs (http://aldev0.50megs.com), theglobe (http://members.theglobe.com/aldev1/index.html), NBCi (http://members.nbci.com/alavoor), Terrashare (http://aldev.terrashare.com), Fortunecity (http://members.fortunecity.com/aldev), Freewebsites (http://aldev.freewebsites.com), Tripod (http://members.tripod.lycos.com/aldev), Spree (http://members.spree.com/technology/aldev), Escalix (http://www.escalix.com/freepage/aldev), Httpcity (http://www.httpcity.com/aldev/index.html), Freeservers (http://aldev.freeservers.com).

4.1. 어떻게 저자의 String class를 믿을 수 있나?

혹 String class를 믿을 수 없을 수도 있다. 이를 해결하기 위해, 저자의 String class를 검증할 과학적인 방법이 있다. 현대 컴퓨터 공학자들은 소프트웨어를 검증하기 위해 머리가 아닌 CPU를 쓴다. 사람의 머리는 너무 느리기 때문에 테스트와 검증을 위해서는 컴퓨터의 힘을 빌리는 것이 좋다.

프로그램 example_String.cpp (http://www.milkywaygalaxy.freeservers.com) 으로 가서 'Source code for C++' 를 선택해라. (부록 A 에도 주어진다. 23절) 위 프로그램은 수백만번의 테스트를 자동으로 할 수 있는 테스트 모듈을 갖고 있다. String class에 이 테스트를 해본다면, String class가 견고하고 완벽한 프로그램임을 알 수 있을 것이다.

직접 **50000**반복으로 테스트 해보았더니 오류없이 잘 동작하였다. 또한, 어떠한 메모리 릭도 발견할 수 없었다. 테스트는 리눅스에서 /usr/bin/gtop, UNIX top 명령어,

KDEStart->System->KDE System Guard and KDEStart->System->Process management 등을 사용하여 cpu와 메모리를 체크하며 이루어졌다.

나는 반복횟수 를 천만번이나 그 이상으로 놓고 테스트 해보기를 추천한다. 반복횟수가 클수록 신뢰성은 더욱 높아질 것이다!! 테스트를 돌려놓고 밥을 먹으러 다녀오면 결과를 볼 수 있을 것이다!!

5. String class 사용하기

이 String class는 표준 C++ 라이브러리의 string class와 다르다는데 주의하라. 이 특별한 String class는 직접 만들어진 것이고, Java 프로그래머들이 C++을 쉽게 사용하도록 하기 위해 만들어졌다. 만약 당신이 C++과 더 익숙하다면 표준 C++ 라이브러리에 제공되는 진짜 string class 를 사용 하는 것이 좋다.

String class를 사용하기 위해, 23절의 "example_String.cpp" 예제 프로그램과 23절의 String class를 보아라.

'String class' 는 char와 char * 타입을 완벽하게 대신할 수 있다. **'String class'**를 char 처럼 사용할 수도 있고, 여러가지 다양한 기능도 사용할 수 있다. 23절 에 주어진 makefile에서 만들어지는 'libString.a'를 링크해야하고, C++ 라이브러리가 위치한 모든 곳의 라이브러리를 /usr/lib 나 /lib 디렉토리에 라이브러리를 복사해넣어야 한다. 'libString.a'를 사용하기 위해서는, 다음과 같이 컴파일하라.

```
g++ example.cpp -lString
```

다음에 주어진 예제 코드를 보라.

```
String aa;

aa = "Creating an Universe is very easy, similar to creat-
ing a baby human.";

// 프로그램에서 aa.val()을 'char *' 같이 사용할 수 있다.
for (unsigned long tmpii = 0; tmpii < aa.length(); tmpii++)
{
//fprintf(stdout, "aa.val()[%ld]=%c ", tmpii, aa.val()[tmpii]);
fprintf(stdout, "aa[%ld]=%c ", tmpii, aa[tmpii]);
}

// 실제로 'char *'로 사용하면..
for (char *tmpcc = aa.val(); *tmpcc != 0; tmpcc++)
{
fprintf(stdout, "aa.val()=%c ", *tmpcc);
}
```

5.1. 연산자

'String class' 는 아래와 같은 연산자를 제공한다 :-

- Equal to **==**
- Not equal to **!=**
- Assignment **=**
- Add to itself and Assignment **+=**
- String concatenation or addition **+**

연산자를 사용하는 예제를 보자.

```
String aa;
String bb("Bill Clinton");

aa = "put some value string";  // assignment operator
aa += "add some more"; // Add to itself and assign operator
aa = "My name is" + " Alavoor Vasudevan "; // string cat operator

if (bb == "Bill Clinton")  // boolean equal to operator
cout << "bb is equal to 'Bill Clinton' " << endl;

if (bb != "Al Gore")   // boolean 'not equal' to operator
cout << "bb is not equal to 'Al Gore'" << endl;
```

5.2. 함수

String class에서 제공되는 함수들은 Java 의 String class와 같은 이름을 갖는다. 함수 이름과 동작은 Java의 String class와 완전히똑같다. StringBuffer class역시 제공된다. 이들은 Java와 C++간의 포팅을 쉽게 할 것이다 (잘라내기 & 붙여넣기와 최소한의 코드 조정 만을 필요로 할 것이다). Java의 함수에 들어있는 코드를 C++의 멤버함수로 복사하기만 하면 될 것이고, 최소한의 변경만으로도 C++에서 잘 컴파일 될 것이다. 또다른 이점은 Java와 C++을 모두 사용하는 개발자들이 둘 간의 문법이나 함수 이름을 따로따로 기억할 필요가 없어진다는 것이다.

예를들어 integer를 문자열로 바꾸는 것을 보면,

```
String aa;

aa = 34;  // '=' 연산자가 int를 string으로 바꾼다.
cout << "The value of aa is : " << aa.val() << endl;

aa = 234.878;  //  '=' 연산자가 float를 string으로 바꾼다.
cout << "The value of aa is : " << aa.val() << endl;
```

```
aa = 34 + 234.878;
cout << "The value of aa is : " << aa.val() << endl;
// 출력은 '268.878'일 것이다.

// casting이 필요하다.
aa = (String) 34 + " Can create infinite number of uni-
verses!! " + 234.878;
cout << "The value of aa is : " << aa.val() << endl;
// 출력은 '34 Can create infinite number of universes!! 234.878'일 것이다.
```

String class의 함수이름에 대한 자세한 내용은 23절 를 참고해라. 같은 String.h파일이 다음 섹션에도 나올 것이다.

5.3. String class의 이름 바꾸기

5.3.1. Case 1: 단순한 이름 바꾸기

만약 String class의 이름이 맘에 들지 않는다면, **"typedef"** 를 이름을 바꾸기 위해 사용할 수 있다.

String.h를 include하는 모든 파일에 다음의 내용을 넣어라 :

```
// String이라는 이름이 맘에 들지 않는다면, 다음과 같이 바꾸어라.
typedef String StringSomethingElseIwant;

// 이제 코드는 다음과 같이 될 것이다.
int main()
{
StringSomethingElseIwant aa_renstr;
aa_renstr = "I renamed the String Class using typedef";

// .....
}
```

example_String.cpp (http://www.milkywaygalaxy.freeservers.com) 에 가서 'Source code for C++'을 선택해라.

5.3.2. Case 2: 이름이 겹칠 때

다른 같은 이름을 가진 class와 이름이 겹치는데, 두 class를 모두 사용하고 싶다면, 다음과 같은 방법을 써라. String.h를 include하는 모든 파일에 다음의 내용을 넣어라.

```
#define  String  String_somethingelse_which_I_want
#include "String.h"
#undef String
```

```
#include "ConflictingString.h"  // 이것도 String이란 class이다.

// 코드 내용..
int main()
{
String_somethingelse_which_I_want aa;
String bb; // 이것은 겹치는 String class이다.

aa = " some sample string";
bb = " another string abraka-dabraka";
.......
}
```

전처리기가 모든 String을 "String_somethingelse_which_I_want" 으로 바꿀 것이고, String의 정의를 없앨 것이다. undef 다음 부터는 "String" class를 정의하는 겹치는 string class header가 오게 된다.

6. String.h 파일

C++과 Java는 많은 소프트웨어 프로젝트에서 같이 쓰인다. C++과 Java를 왔다갔다하는 프로그래머들에게는 이 문자열 class가 매우 유용할 것이다.

C++ (혹은 다른 객체지향 언어)에서는, "class 데이터구조"(혹은 인터페이스) 만 읽으면 그 class를 사용할 수 있다. 인터페이스만 이해하면 되지, 인터페이스의 구현까지는 알 필요가 없는 것이다. String class의 경우, String.h 파일에 있는 String class만 읽고 이해하면 된다. String class를 쓰기 위해 구현(String.cpp)을 모두 읽을 필요는 없는 것이다. 객체지향 class들은 시간을 절약하게 해주고, 구현의 내용을 교묘하게 숨겨준다.

(객체지향인 Java에도 이와 같은 역할을 하여 구현 내용을 숨겨주는 **'interface'** 란 것이 있다.)

아래의 내용은 **String.h** 파일이고, 23절을 참고해라.

```
//
// Author : Al Dev  Email: alavoor[AT]yahoo.com
// string class나 String class를 써라.
//
// 메모리 릭을 막기 위해 - 문자 변수를 관리하기 위한 문자 class
// char[]나 char *보다는 String class나 string class를 써라.
//

#ifndef __STRING_H_ALDEV_
#define __STRING_H_ALDEV_

// 프로그램이 커질 수록 iostream을 사용하지 말아라.
#ifdef NOT_MSWINDOWS
```

```
#include <iostream>
#else
#include <iostream.h> // 하위호환성을 위해. C++ 표준은 .h가 없다.
#endif // NOT_MSWINDOWS

#include <stdio.h>   // File과 sprintf()를 위해
//#include <list.h> // list

// MS Windows 95 VC++과 Borland C++ 컴파일러인 경우 -
// d:\program files\CBuilder\include\examples\stdlib\list.cpp 와 include\list.h
// 을 보라.
//#include <list> // for list
//using namespace std;

const short INITIAL_SIZE =  50;
const short NUMBER_LENGTH = 300;
const int MAX_ISTREAM_SIZE = 2048;

//class StringBuffer;

// 나는 이 문자
열 class를 Linux (Redhat 7.1)와 MS Windows Borland C++ v5.2 (win32)
// 에서 컴파일 / 테스트 해보았다.
// 또한, MS Visual C++ compiler에서도 작동할 것이다.
class String
{
public:
String();
String(const char bb[]);  // + 연산자를 위해 필요
String(const char bb[], int start, int slength); // 문자들의 부분집합
String(int bb);  // + 연산자를 위해 필요
String(unsigned long bb);  // + 연산자를 위해 필요
String(long bb);  // + 연산자를 위해 필요
String(float bb);  // + 연산자를 위해 필요
String(double bb);  // + 연산자를 위해 필요
String(const String & rhs);  // + 연산자를 위해 필요한 copy constructor
//String(StringBuffer sb);  // Java와의 호환성을 위해
    // - 그러나 MS windows에서는
    // 컴파일되지 않고, core dump를 일으킨다.
String(int bb, bool dummy);  // StringBuffer class를 위해 필요
virtual ~String();  // virtual로 선언하여 상속받은 class의 소멸자가
    // 불리도록 한다.

char *val() {return sval;} // sval을 public으로 하는 것은 위험하므로

// Java의 String을 흉내낸 함수들
unsigned long length();
char charAt(int where);
void getChars(int sourceStart, int sourceEnd,
```

```
char target[], int targetStart);
char* toCharArray();
char* getBytes();

bool equals(String str2); // == 연산자를 참조하라
bool equals(char *str2); // == 연산자를 참조하라
bool equalsIgnoreCase(String str2);

bool regionMatches(int startIndex, String str2,
int str2StartIndex, int numChars);
bool regionMatches(bool ignoreCase, int startIndex,
String str2, int str2StartIndex, int numChars);

String toUpperCase();
String toLowerCase();

bool startsWith(String str2);
bool startsWith(char *str2);

bool endsWith(String str2);
bool endsWith(char *str2);

int compareTo(String str2);
int compareTo(char *str2);
int compareToIgnoreCase(String str2);
int compareToIgnoreCase(char *str2);

int indexOf(char ch, int startIndex = 0);
int indexOf(char *str2, int startIndex = 0);
int indexOf(String str2, int startIndex = 0);

int lastIndexOf(char ch, int startIndex = 0);
int lastIndexOf(char *str2, int startIndex = 0);
int lastIndexOf(String str2, int startIndex = 0);

String substring(int startIndex, int endIndex = 0);
String replace(char original, char replacement);
String replace(char *original, char *replacement);

String trim(); // 오버로딩 된 trim을 참조하라.

String concat(String str2);  // + 연산자를 참조
String concat(char *str2); // + 연산자를 참조
String concat(int bb);
String concat(unsigned long bb);
String concat(float bb);
String concat(double bb);

String reverse(); // 오버로딩 된 다른 reverse()를 참조
```

```
String deleteCharAt(int loc);
String deleteStr(int startIndex, int endIndex); // Java의 "delete()"

String valueOf(char ch)
{char aa[2]; aa[0]=ch; aa[1]=0; return String(aa);}
String valueOf(char chars[]){ return String(chars);}
String valueOf(char chars[], int startIndex, int numChars);
String valueOf(bool tf)
{if (tf) return String("true"); else return String("false");}
String valueOf(int num){ return String(num);}
String valueOf(long num){ return String(num);}
String valueOf(float num) {return String(num);}
String valueOf(double num) {return String(num);}

// 이 파일의 아래에 주어진 StringBuffer를 참고하라.

// ---- 여기까지 Java를 흉내낸 함수들  -----

//////////////////////////////////////////////////////
//  Java에는 없는 추가적인 함수들
//////////////////////////////////////////////////////
String ltrim();
void ltrim(bool dummy); // 직접적으로 object를 변화시킨다.
String rtrim();
void rtrim(bool dummy); //  직접적으로 object를 변화시킨다.
// chopall 참고.

void chopall(char ch='\n'); // 맨 뒤의 ch를 없앤다. rtrim 참고.
void chop(); // 맨 뒤의 문자를 없앤다.

void roundf(float input_val, short precision);
void decompose_float(long *integral, long *fraction);

void roundd(double input_val, short precision);
void decompose_double(long *integral, long *fraction);

void explode(char *separator); // token()과 오버로딩 된 explode()참조
String *explode(int & strcount, char separator = ' '); //  token()참조
void implode(char *glue);
void join(char *glue);
String repeat(char *input, unsigned int multiplier);
String tr(char *from, char *to); // character들을 바꾼다(translate).
String center(int padlength, char padchar = ' ');
String space(int number = 0, char padchar = ' ');
String xrange(char start, char end);
String compress(char *list = " ");
String left(int slength = 0, char padchar = ' ');
String right(int slength = 0, char padchar = ' ');
```

```
String overlay(char *newstr, int start = 0, int slength = 0, char pad-
char = ' ');

String at(char *regx); // regx의 첫번째 match
String before(char *regx); // regx 앞의 string
String after(char *regx); // regx 뒤의 string
String mid(int startIndex = 0, int length = 0);

bool isNull();
bool isInteger();
bool isInteger(int pos);
bool isNumeric();
bool isNumeric(int pos);
bool isEmpty();  // length() == 0 과 같은 상태
bool isUpperCase();
bool isUpperCase(int pos);
bool isLowerCase();
bool isLowerCase(int pos);
bool isWhiteSpace();
bool isWhiteSpace(int pos);
bool isBlackSpace();
bool isBlackSpace(int pos);
bool isAlpha();
bool isAlpha(int pos);
bool isAlphaNumeric();
bool isAlphaNumeric(int pos);
bool isPunct();
bool isPunct(int pos);
bool isPrintable();
bool isPrintable(int pos);
bool isHexDigit();
bool isHexDigit(int pos);
bool isCntrl();
bool isCntrl(int pos);
bool isGraph();
bool isGraph(int pos);

void clear();
int toInteger();
long parseLong();

double toDouble();
String token(char separator = ' '); // StringTokenizer와 explode()를 참조
String crypt(char *original, char *salt);
String getline(FILE *infp = stdin); // putline() 참조
//String getline(fstream *infp = stdin); // putline() 참조

void putline(FILE *outfp = stdout); // getline() 참조
//void putline(fstream *outfp = stdout); // getline() 참조
```

```
void swap(String aa, String bb); // aa를 bb로 바꾼다
String *sort(String aa[]);  // String의 array를 sort한다
String sort(int startIndex = 0, int length = 0);  // string 내
의 character들을 sort
int freq(char ch); // ch가 들어있는 횟수를 센다
void Format(const char *fmt, ...);
String replace (int startIndex, int endIndex, String str);

void substring(int startIndex, int endIndex, bool dummy); // object를 직
접 바꾼다
void reverse(bool dummy); // object를 직접 바꾼다
String deleteCharAt(int loc, bool dummy); // object를 직접 바꾼다
String deleteStr(int startIndex, int endIndex, bool dummy);
void trim(bool dummy); // object를 직접 바꾼다
String insert(int index, String str2);
String insert(int index, String str2, bool dummy); // object를 직접 바꾼다
String insert(int index, char ch);
String insert(int index, char ch, bool dummy); // object를 직접 바꾼다
String insert(char *newstr, int start = 0, int length = 0, char pad-
char = ' ');

String dump(); // od -c 와 같이 string을 dump한다.

// Java의 StringBuffer를 위해 필요한 것들
void ensureCapacity(int capacity);
void setLength(int len);
void setCharAt(int where, char ch); // charAt(), getCharAt() 참고

// Java의 Integer class, Long, Double class를 위해 필요
int parseInt(String ss) {return ss.toInteger();}
int parseInt(char *ss)
{String tmpstr(ss); return tmpstr.toInteger();}
long parseLong(String ss) {return ss.parseLong();}
long parseLong(char *ss)
{String tmpstr(ss); return tmpstr.parseLong();}
float floatValue() {return (float) toDouble(); }
double doubleValue() {return toDouble(); }
char * number2string(int bb);  // String(int) 참고
char * number2string(long bb);  // String(long) 참고
char * number2string(unsigned long bb);  // String(long) 참고
char * number2string(double bb);  // String(double) 참고

///////////////////////////////////////////////
//  겹치는 함수 이름들
///////////////////////////////////////////////
// char * c_str() // val() 을 대신사용
// bool find();  // regionMatches() 를 대신사용
// bool search();  // regionMatches() 를 대신사용
```

```
// bool matches(); // regionMatches() 를 대신사용
// int rindex(String str2, int startIndex = 0); lastIndexOf() 을 대신사용
// String blanks(int slength);  // repeat() 를 대신사용
// String append(String str2); // concat() 이나 + operator 을 대신사용
// String prepend(String str2);  // + operator을 대신사용 append()참고
// String split(char separator = ' ');  // token(), explode() 나 StringTok-
enizer class 를 대신사
용
bool contains(char *str2, int startIndex = 0); // indexOf() 를 대신사용
// void empty(); is_empty() 를 대신사용
// void vacuum(); clear() 를 대신사용
// void erase(); clear() 를 대신사용
// void zero(); clear() 를 대신사용
// bool is_float(); is_numeric(); 을 대신사용
// bool is_decimal(); is_numeric(); 을 대신사용
// bool is_Digit(); is_numeric(); 을 대신사용
// float float_value(); toDouble(); 을 대신사용
// float tofloat(); toDouble(); 을 대신사용
// double double_value(); toDouble(); 을 대신사용
// double numeric_value(); toDouble(); 을 대신사용
// int int_value(); toInteger() 를 대신사용
// int tonumber(); toInteger() 를 대신사용
// String get(); substring() 이나 val() 을 대신 사용. 그러나 Java 스타일
의 substring이 더 좋다
// String getFrom(); substring() 이나 val() 을 대신 사용. 그러나 Java 스타일
의 substring이 더 좋다
// String head(int len); substring(0, len) 을 대신사용
// String tail(int len); substring(length()-len, length()) 를 대신사용
// String cut(); deleteCharAt() 이나 deleteStr() 을 대신사용
// String cutFrom(); deleteCharAt() 이나 deleteStr() 을 대신사용
// String paste(); insert() 를 대신사용
// String fill(); replace() 를 대신사용
// char firstChar(); // substring(0, 1); 을 대신사용
// char lastChar(); // substring(length()-1, length()); 를 대신사용
// String findNext(); token(), explode() 이나 StringTokenizer class 를 대신
사용

// begin();  iterator. operator [ii]를 대신사용
// end();  iterator. operator [ii]를 대신사용
// copy();  assignment =  연산다를 대신 사용, String aa = bb;
// clone();  assignment =  연산자를 대신 사용, String aa = bb;
// void putCharAt(int where, char ch); setCharAt() 을 대신사용
// void replaceCharAt(int where, char ch); setCharAt() 을 대신사용
// char getCharAt(int where); CharAt() 을 대신사용
// void parseArgs(int where, char ch); StringTokensizer class, token() 이
나 explode() 를 대신사용
// void truncate(); trim(), rtrim(), chop() 이나 chopall() 을 대신사용
// 숫자를 string으로 변
환 : notostring(), int2str, long2str은 number2string()을 사용
```

```
// 연산자들...
String operator+ (const String & rhs);
friend String operator+ (const String & lhs, const String & rhs);

String& operator+= (const String & rhs); // 레퍼런스를 이용하면 더 빠를 것이다.
String& operator= (const String & rhs); // 레퍼런스를 이용하면 더 빠를 것이다.
bool operator== (const String & rhs); // 레퍼런스를 이용하면 더 빠를 것이다.
bool operator== (const char *rhs);
bool operator!= (const String & rhs);
bool operator!= (const char *rhs);
char operator [] (unsigned long Index) const;
char& operator [] (unsigned long Index);
friend ostream &  operator<< (ostream & Out, const String & str2);
friend istream &  operator>> (istream & In, String & str2);

bool String::operator< (const char *rhs) const; // map & vector 를 위한 유용
한 method
bool String::operator< (const String & rhs) const; // map & vector 를 위
한 유용한 method

//do later: static list<String>  explodeH;  // list head

protected:
char *sval; // sval을 public으로 하는 것은 위험하다.
void verifyIndex(unsigned long index) const; // Win32에서의  warning때문
에 inline이 아니다.
void verifyIndex(unsigned long index, char *aa) const;// Win32에서
의  warning때문에 inline이 아니다.

void _str_cat(char bb[]);
void _str_cat(int bb);
void _str_cat(unsigned long bb);
void _str_cat(float bb);

void _str_cpy(char bb[]);
void _str_cpy(int bb); // itoa
void _str_cpy(unsigned long bb);
void _str_cpy(float bb); // itof

private:
// Note: 모든 private 변수와 함수는 _ (밑줄)로 시작한다.

//static String *_global_String; // add 연산에서 필요
//inline void _free_glob(String **aa);

bool _equalto(const String & rhs, bool type = false);
bool _equalto(const char *rhs, bool type = false);
String *_pString;  // 내부에서 사용하는 임시 포인터
```

```
char *_pNumber2String;  // 내부에서 사용하는 임시 포인터
inline void _allocpString();
inline void _allocpNumber2String();
inline void Common2AllCstrs();
inline void _reverse();
inline void _deleteCharAt(int loc);
inline void _deleteStr(int startIndex, int endIndex);
inline void _trim();
inline void _ltrim();
inline void _rtrim();
inline void _substring(int startIndex, int endIndex);
void _roundno(double input_dbl, float input_flt, short preci-
sion, bool type);
};

// 전역변수는 String.cpp 에서 정의된다

#endif // __STRING_H_ALDEV_
```

6.1. StringBuffer.h

C++ 과 Java는 많은 소프트웨어 프로젝트에서 동시에 쓰인다. C++과 Java를 왔다갔다하는 프로그래머에게 이 stringbuffer class는 매우 유용할 것이다.

```
//
// Author : Al Dev  Email: alavoor[AT]yahoo.com
//

#ifndef __STRINGBUFFER_H_ALDEV_
#define __STRINGBUFFER_H_ALDEV_

// Java의 StringBuffer 를 모방한 것
// 이 class는 Java의 code를 최소한의 수정만으로도
// C++에서 동작하도록 하기 위해 만들어졌다.
// Note: C++로 코딩하는 동안은 이 StringBuffer
// class를 *쓰지 말아라*.
// 이 class는 오직 Java 코드를 cut/paste하는 경우를
// 위해서 쓰여진 것이다.
class StringBuffer: public String
{
public:
StringBuffer();
~StringBuffer();
StringBuffer(char *aa);
StringBuffer(int size);
StringBuffer(String str);
```

```
int capacity();
StringBuffer append(String str2);
// operator + 참조
//{ *this += str2; return *this;} // 이 code는 core dump를 일으킨다

StringBuffer append(char *str2);
StringBuffer append(int bb);
StringBuffer append(unsigned long bb) ;
StringBuffer append(float bb) ;
StringBuffer append(double bb) ;

StringBuffer insert(int index, String str2);
StringBuffer insert(int index, char ch);

StringBuffer reverse();

// Java의 "delete()"에 해당. (delete는 C++의 keyword이므로 사용하지 못한다)
StringBuffer deleteStr(int startIndex, int endIndex);
StringBuffer deleteCharAt(int loc);

StringBuffer substring(int startIndex, int endIndex = 0);
void assign(char *str);

private:
StringBuffer *_pStringBuffer;
inline void allocpStringBuffer();
inline void Common2AllCstrs();
};

#endif // __STRINGBUFFER_H_ALDEV_
```

6.2. StringTokenizer.h

C++ 과 Java는 많은 소프트웨어 프로젝트에서 동시에 쓰인다. C++과 Java를 왔다갔다하는 프로그래머에게 이 stringtokenizer class는 매우 유용할 것이다.

```
//
// Author : Al Dev  Email: alavoor[AT]yahoo.com
//

#ifndef __STRINGTOKENIZER_H_ALDEV_
#define __STRINGTOKENIZER_H_ALDEV_

// Java의 StringBuffer 를 모방한 것
// Java의 코드를 C++에서 혹은 그 반대로 컴파일하는 것이
// 가능하게 만들어졌다.
```

```
class StringTokenizer: public String
{
public:
StringTokenizer(String str);
StringTokenizer(String str, String delimiters);
StringTokenizer(String str, String delimiters, bool delimAsToken);
~StringTokenizer();

int  countTokens();
bool  hasMoreElements();
bool  hasMoreTokens();
String  nextElement(); // Java에서는 "Object"type을 return한다.
String  nextToken();
String  nextToken(String delimiters);
private:
int CurrentPosition; // 현재 string에서의 index
int TotalTokens;
int RemainingTokens;
char * ListOfDl; // delimiters(구분자)들의 list
char *  WorkStr; // 임시 작업 string
char *  OrigStr; // 지나온 original string
bool   DlFlag;  // 구분자인지를 나타내는 flag
inline  void vPrepWorkStr(char *delimiters = NULL);
};

#endif // __STRINGTOKENIZER_H_ALDEV_
```

7. 표준 C++ 라이브러리 string class

위에 언급된 String class (S가 대문자인 것에 주의!)는 Java를 사용하는 사람들을 위한 것인 반면, 표준 C++ 라이브러리에서 제공되는 "진짜" string class를 주목할 필요가 있다.

string class는 C에서의 가장 큰 문제점 중 하나인 문자배열의 단점을 극복하기 위해 만들어졌다. 문자배열이 무척 빠르긴 하지만, 많은 단점을 갖고 있다. 문자배열은 많은 버그의 원인이고, 이를 parsing하는 일은 굉장히 귀찮은 일이다.

string class는 문자열을 파싱하고, 조정하는데 필요한 좋은 인터페이스를 제공하고, STL과도 호환가능하다. 즉, 모든 STL의 알고리즘을 사용할 수 있다. 실제로 문자열은 **vector<char>** (문자들을 위한 container 혹은 진보된 문자배열) 로 취급될 수 있다.

다음의 사이트에서 참고할만한 것들을 얻을 수 있다:

- SGI STL 기초_string reference: http://www.sgi.com/tech/stl/basic_string.html.

7.1. 예제로 살펴보는 string

string을 만드는 것은 쉽다. Creating a string is easy:

```
#include <string>
#include <iostream>

using namespace std;

int main()
{
    string str("Hello World!"); // 혹은 string str = "Hello World!";
    cout << str << endl;
}
```

이 코드는 "**str**' 란 string을 만들고, "**Hello World!**' 라는 내용을 넣을 것이다. 그리고 cout을 사용하여 표준출력(stdout)으로 출력할 것이다.

(이제부터 모든 헤더와 namespace 관련부분은 생략할 것이다.)

문자열의 부분을 구하는 것 역시 쉽다 :

```
string str("Hello Universe!");
string start = str.substr(0, 5);
string end = str.substr(5);
```

여기서는 첫 6글자를 string "**start**"에, 나머지는 "**end**"에 들어갈 것이다.

문자열의 길이를 얻기 위해서는, 아래와 같이 하면 된다.

```
string str("How long is this string?");
cout << "Length of string is: " << str.size() << endl;
```

혹은 이와 정확히 똑같은 역할을 하는 **length()** 를 써도 된다.

7.2. 문자열을 찾기

문자열을 찾는 것은 문자배열보다 훨씬 쉽다. string class는 문자열을 찾는데 효율적인 멤버함수들을 제공한다. 모든 멤버함수는 string::size_type 을 return한다.

표 **1.** 문자열 검색 멤버 함수

멤버 함수	작동
find()	주어진 부분문자열이 처음으로 나타나는 곳을 찾는다

find_first_of()	find()와 같으나 주어진 문자가 처음으로 나타나는 위치를 찾는다
find_last_of()	find first of()와 같으나, 주어진 문자가 마지막으로 나타나는 위치를 찾는다
find_first_not_of()	find first of() 과 같으나 주어진 문자가 아닌 첫 문자가 처음으로 나타나는 위치를 찾는다
find_last_not_of()	find last of() 과 같으나 주어진 문자가 아닌 문자를 찾는다.
rfind()	find()와 같으나, 찾는 방향이 반대이다. (뒤쪽부터 찾는다)

가장 흔한 상황은 문자열을 찾는 것이고, 이는 **find()** 함수를 사용하면 된다.

```
string str("Hello, can you find Ben?");
string::size_type position = str.find("Ben");
cout << "First occurence of Ben was found at: " << position << endl;
```

이 코드는 '**Ben**'에 대해 대소문자를 구별하는 검색을 하고, 시작위치를 '**position**'에 string::size_type 타입으로 넣는다. 리턴하는 값이 int가 아니라 특별히 고안된 string::size_type 타입이라는데 주의하라.

find_first_of() 함수는 실제적인 예가 필요할 것이다. 아래와 같은 상황을 보자.

```
string s = "C++ is an impressive language.";
string::size_type pos = s.find_first_of(" .");

while (pos != string::npos) {
    cout << "Found space or dot at: " << pos << endl;
    pos = s.find_first_of(" .", pos + 1);
}
```

find_first_of()함수를 쓰면, 우리는 첫번째 인자의 모든 문자를 찾게 되고, 따라서 여기서는 스페이스(' ') 혹은 점('.')을 찾게 된다.

프로그램을 컴파일해서 어떻게 출력되는지 보아라.

7.3. string tokenizer

문자열을 가지고 자주 하게되는 작업 중 하나는, 어떤 구분자를 가지고 토큰들로 나누는 것이다 (tokenize). tokenizer는 문자열을 find()를 계속 부르는 일 없이 쉽게 조그만 조각들로 쪼갤 수 있도록 해준다. C에서는, 아마도 문자 배열에 대해 **strtok()** 란 함수를 썼을 것이지만, 문자열에 대해서는 이러한 함수가 없다. 따라서 직접 이런 함수를 만들어야 겠지만, 몇가지 해결책이 있다.

The advanced tokenizer:

```
void Tokenize(const string& str,
                      vector<string>& tokens,
                      const string& delimiters = " ")
{
    // 맨 첫 글자가 구분자인 경우 무시
    string::size_type lastPos = str.find_first_not_of(delimiters, 0);
    // 구분자가 아닌 첫 글자를 찾는다
    string::size_type pos     = str.find_first_of(delimiters, lastPos);

    while (string::npos != pos || string::npos != lastPos)
    {
        // token을 찾았으니 vector에 추가한다
        tokens.push_back(str.substr(lastPos, pos - lastPos));
        // 구분자를 뛰어넘는다.  "not_of"에 주의하라
        lastPos = str.find_first_not_of(delimiters, pos);
        // 다음 구분자가 아닌 글자를 찾는다
        pos = str.find_first_of(delimiters, lastPos);
    }
}
```

tokenizer는 다음과 같이 쓰일 수 있다.

```
#include <string>
#include <algorithm>
#include <vector>

using namespace std;

int main()
{
    vector<string> tokens;

    string str("Split me up! Word1 Word2 Word3.");

    Tokenize(str, tokens);

    copy(tokens.begin(), tokens.end(), os-
tream_iterator<string>(cout, ", "));
}
```

위의 코드는 Tokenize 함수를 사용하는 예로서, 첫번째 인자인 **str**를 쪼갠다. 그리고 우리가 세 번째 인자를 주지 않았기 때문에 디폴트로 설정된 " "(spacebar)를 구분자로 사용한다. 그리고 모든 element는 **tokens** 벡터에 들어가게 될 것이다.

마지막으로 표준 출력에 벡터 전체를 copy()함으로써 벡터의 내용을 화면으로 볼 수 있을 것이다.

또다른 접근 방법은 stringstream을 사용하는 것이다. C++에서 stream은 특수한 기능이 하나 있는데, 이는 공백(whitespace)를 만날 때까지 읽기를 계속한다는 것이다. 따라서 아래의 코드는 공백을 기준으로 문자열을 나누고자 할 때 잘 동작할 것이다.

```
#include <vector>
#include <string>
#include <sstream>

using namespace std;

int main()
{
    string str("Split me by whitespaces");
    string buf; // 버퍼 string
    stringstream ss(str); // string을 stream에 넣는다

    vector<string> tokens; // word들을 넣을 vector

    while (ss >> buf)
        tokens.push_back(buf);
}
```

이제 stringstream은 출력 연산자(>>)를 사용 하여 문자열을 **buf** 에 공백을 만날 때마다 넣는다. **buf**는 이를 차례대로 벡터에 push_back() 한다. 그리고 이제 **tokens** 벡터는 **str** 에 들어있는 모든 단어를 갖게 된다.

8. File class

C++ 과 Java는 많은 소프트웨어 프로젝트에서 동시에 쓰인다. C++과 Java를 왔다갔다하는 프로그래머에게 이 File class는 매우 유용할 것이다.

운영체제의 파일을 관리할 때, File class가 필요할 것이다. 이 class는 Java의 File class를 모방한 것으로, C++ 프로그래밍에서 매우 유용할 것이다. C++에서 이 class를 사용하면, 파일이 존재하는지(**exists()**), 디렉토리가 존재하는지(**exists()**), 파일의 길이는 어떤지(**length()**) 등을 알 수 있다.

이 class가 표준 C++ 라이브러리에는 없는 좋은 기능들을 갖고 있다는 데 주목해라. 그러나 여러가지 다른 동작을 해주어야 하는 fstreams(iostreams)과 헷갈리지는 말아라.

- C++ File class 는 다음에서 구할 수 있다 File.h http://www.angelfire.com/country/aldev0/cpphowto/File.h 와 File.cpp File.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++'를 click 하라.
- Java: Java.io.File class http://java.sun.com/j2se/1.3/docs/api/java/io/File.html
- Java의 File Class 레퍼런스 http://unicornsrest.org/reference/java/qref11/java.io.File.html

9. C++ 에서의 메모리 할당

C에서는, 메모리의 할당과 해제를 위해 malloc()과 free()를 비롯한 malloc()계열의 함수를 쓰지만, 다들 단점을 갖고 있다. 그래서 C++ 은 메모리를 다루기 위한 연산자들을 도입했고, 이들은 **new** 와 **delete**이다. 이 연산자들은 실행시에 힙(heap - 혹은 자유 공간)으로부터 메모리를 할당, 해제한다.

C++에서는 정말로 꼭 malloc()이나 free()만을 써야하는 상황이 아니라면 언제나 **new** 와 **delete**를 써야한다. 그러나 주의할 점은, 이 두 가지를 섞어서 쓰면 안된다는 것이다. malloc()으로 얻은 메모리를 delete로 해제할 수는 없고, 반대로 new로 얻은 메모리를 free()시킬 수도 없다.

9.1. C++ Zap (Delete) 함수

C++에서의 **delete** 와 **new** 연산자는 C의 malloc, free보다 낫다. 따라서 malloc과 free 대신 new와 zap(delete)를 쓰도록 하는 것이 좋다.

delete 연산자가 좀 더 깔끔하게 사용되게 하기위해 다음과 같은 Zap() inline 함수를 만들자. 다음과 같이 zap()을 정의하자.

```
// x가 NULL인지 체크하기 위해 assert를 사용하였다.
// 이는 프로그램의 "논리적" 에러를 미리 잡아내기 위한 것이다.
// delete가 NULL인 경우에도 잘 동작하긴 하지만, assert를
// 사용함으로써 좀 더 일찍 에러를 잡아낼 수 있다.

// Zap을 template을 사용하여 정의하자.
// delete대신 zap을 사용하면 더 깔끔할 것이다.
template <class T>
inline void zap(T & x)
{
{assert(x != NULL);}
delete x;
x = NULL;
}

// C++에 두 가지 delete 연산자의 용법이 있는 이유는 C++ 에게
// 한 객체에 대한 포인터와 객체의 배열에 대한 포인터를 구별하도록
// 말해주는 방법이 필요하기 때문이다.
```

```
// delete연산자는 프로그래머에게 "[]"를 쓰게함으로써 이를 구별한다.
// 따라서 우리는 포인터의 배열을 지우기 위한 zaparr 함수를 다음과 같이 정의할 수 있다
template <class T>
inline void zaparr(T & x)
{
 {assert(x != NULL);}
     delete [] x;
     x = NULL;
}
```

zap()함수는 포인터를 delete시키고 NULL로 세팅한다. 이는 똑같은 delete 포인터에 대해 여러번의 zap()이 불려서 프로그램이 망가지는 것을 방지한다. 다음의 zap_example()함수를 보아라. example_String.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++'을 클릭해라.

```
//  example_String.cpp에서 zap_example()를 보라.
zap(pFirstname);
//zap(pFirstname); // pFirstname이 NULL이므로 코어 덤프가 일어나지 않는다.
//zap(pFirstname); // pFirstname이 NULL이므로 코어 덤프가 일어나지 않는다.

zap(pLastname);
zap(pJobDescription);

int *iiarray = new int[10];
zaparr(iiarray);
```

뭐 특별한 것이 있는 것은 아니고, 이것은 단지 반복적인 코드를 줄이고 타이핑하는 시간을 아껴주며 프로그램을 좀 더 읽기 좋게 만들어주는 것 뿐이다. C++ 프로그래머들은 자주 delete한 pointer를 NULL로 세팅하는 것을 잊는다. 그리고 이는 코어덤프와 오작동으로 이어질 수 있다. zap()은 이러한 문제를 자동으로 처리해준다. zap()에 타입 캐스팅을 할 필요는 없다. 만약 위 zap()함수에서 에러가 난다면, 다른 데서 시작된 에러일 것이다.

또한 9.2절 , my_realloc() 과 my_free() 이 malloc(), realloc() 그리고 free() 대신 쓰여야 한다. 이들은 훨씬 깔끔하고, 여러가지 체크도 해준다. 예를들어, 9.2절 과 my_free() 함수를 사용하는 "String.h" 파일을 보라.

주의 **:** 'new'로 할당된 메모리를 해제하기 위해 free()를 쓰거나, malloc()으로 할당된 메모리를 해제하기 위해 'delete'를 쓰지 말아라. 그렇지 않으면 결과를 예측할 수 없는 에러에 빠질 것이다.

example_String.cpp (http://www.milkywaygalaxy.freeservers.com) 에서 'Source code of C++' 를 클릭한다음, zap함수의 예를 보아라.

9.2. my_malloc 과 my_free 의 사용

malloc과 realloc 을 최대한 사용하지 말고, **new** 와 **9.1**절**(delete)**을 사용해라. 그러나 때로는 C++에서 C 스타일의 메모리 할당을 사용해야 할 필요도 있다. 이 때는 **my_malloc()** , **my_realloc()** , **my_free()** 을 사용해라. 이 함수들은 적절한 할당과 초기화를 해주고, 메모리 문제를 예방해준다. 또한 이 함수들은 DEBUG모드에서 메모리 할당을 추적해주고, 프로그램 실행 전후에 총 메모리 사용량을 표시해준다. 이는 메모리 릭이 있는지를 알려줄 것이다.

my_malloc 과 my_realloc은 다음과 같이 정의되었다. 이는 약간의 메모리를 더 할당해서 (SAFE_MEM = 5) 초기화시키고, 메모리를 할당할 수 없으면 프로그램을 종료한다. 'call_check(), remove_ptr()' 함수는 DEBUG_MEM 가 makefile에서 ((void)0) (이는 NULL을 의미한다)으로 지정되어있을 때에만 작동한다. 이는 총 메모리 사용량을 추적할 수 있게 해준다.

```
void *local_my_malloc(size_t size, char fname[], int lineno)
{
size_t  tmpii = size + SAFE_MEM;
void *aa = NULL;
aa = (void *) malloc(tmpii);
if (aa == NULL)
raise_error_exit(MALLOC, VOID_TYPE, fname, lineno);
memset(aa, 0, tmpii);
call_check(aa, tmpii, fname, lineno);
return aa;
}

char *local_my_realloc(char *aa, size_t size, char fname[], int lineno)
{
remove_ptr(aa, fname, lineno);
unsigned long tmpjj = 0;
if (aa) // aa !=  NULL
tmpjj = strlen(aa);
unsigned long tmpqq = size + SAFE_MEM;
size_t  tmpii = sizeof (char) * (tmpqq);
aa = (char *) realloc(aa, tmpii);
if (aa == NULL)
raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);

// do not memset memset(aa, 0, tmpii);
aa[tmpqq-1] = 0;
unsigned long kk = tmpjj;
if (tmpjj > tmpqq)
kk = tmpqq;
for ( ; kk < tmpqq; kk++)
aa[kk] = 0;
call_check(aa, tmpii, fname, lineno);
return aa;
}
```

my_malloc 의 모든 구현을 보려면 23절 에서 23절 의 헤더파일을 보면 된다.

my_malloc 과 my_free 를 쓰는 예는 다음과 같다.

```
char  *aa;
int   *bb;
float *cc;
aa = (char *) my_malloc(sizeof(char)* 214);
bb = (int *) my_malloc(sizeof(int) * 10);
cc = (float *) my_malloc(sizeof(int) * 20);

aa = my_realloc(aa, sizeof(char) * 34);
bb = my_realloc(bb, sizeof(int) * 14);
cc = my_realloc(cc, sizeof(float) * 10);
```

my_realloc 에서 data type을 cast 할 필요가 없는 것에 주의해라. 이는 인자로 받은 변수의 타입에 맞춰서 리턴값을 보내기 때문이다. The my_realloc 함수는 char *, int *, float * 타입으로 오버로딩 되어있다.

9.3. C++ 에서의 가바지 콜렉터

C/C++에서 가비지 콜렉션은 표준에서 지원되지 않고, 따라서 메모리를 직접 할당, 해제하는 것이 어렵고 복잡하며 에러를 내기 쉽다. 가비지 콜렉션(**GC:Garbage Collection**) 은 구현하는 방법이 여러가지가 있고, 각 프로그램마다 적용될 수 있는 방법이 다르기 때문에 C++ 표준의 일부가 될 수 없었다. 전산학자들은 많은 GC 알고리즘을 개발했고, 이들은 각 문제분야에서만 적용될 수 있는 것들이었다. 즉, 모든 일반적인 문제에 적용될 수 있는 하나의 범용 GC알고리즘은 없다. 따라서 GC는 C++ 표준에 들어가지 못했다. 따라서 언제나 하는 일에 맞는 C++ 라이브러리를 많은 라이브러리들 중에서 고를 수 있다.

다음 C++ 가비지 콜렉션(Garbage Collection) (http://www.xanalys.com/software_tools/mm/glossary/g.html#garbage.collection) 사이트와 메모리 관리 (http://www.xanalys.com/software_tools/mm) 사이트를 가보아라.

10. 포인터가 문제이다

포인터는 일반적인 프로그램에서 꼭 필요한 것은 아니다. Java와 같은 현대 언어에서는 포인터가 없다 (Java는 내부적으로만 포인터를 사용한다). 포인터는 프로그램을 어지럽고 읽기 힘들게 만든다.

최대한 포인터의 사용을 피하고, 대신 레퍼런스를 사용해라. 포인터는 정말 문제가 많고, 포인터 없이 프로그램을 쓰는 게 가능하다. 포인터는 레퍼런스를 쓸 수 없는 곳에서만 써야한다.

레퍼런스 는 별칭(alias)이다. 레퍼런스를 만들면, 이는 다른 객체(혹은 대상)에 다른 이름을 주는 것이다. 그 순간부터 레퍼런스는 대상의 다른 이름으로서 돌아가고, 레퍼런스에 행하는 모든 연산이 그 대상에 실제로 적용된다.

레퍼런스의 문법 **:** 타입을 선언할 때, 뒤에 레퍼런스 연산자 (&) 를 붙임으로써 레퍼런스를 선언할 수 있다. 레퍼런스는 반드시 만들어 질 때 초기화 되어야 한다. 다음의 예를 보자 -

```
int weight;
int & rweight = weight;

DOG aa;
DOG & rDogRef = aa;
```

레퍼런스를 사용할 때 지킬 것 -

- 객체에 대한 다른 이름을 주고자 할 때 레퍼런스를 사용해라.
- 모든 레퍼런스는 초기화되어야 한다.
- 프로그램의 높은 효율과 퍼포먼스를 위해 레퍼런스를 사용해라
- 레퍼런스와 포인터를 보호하기 위해 가능한경우면 언제나 **const**를 사용해라.

레퍼런스를 사용할 때 하지 말아야 할 것 -

- 중요 **:** NULL인 객체에 대해 레퍼런스를 쓰지 말아라.
- 포인터의 주소를 나타내는 &와 레퍼런스 연산자를 헷갈리지 마라. 레퍼런스 연산자는 오직 선언부 (위에 나와있는 레퍼런스 사용법 참조) 에서만 쓰인다.
- 레퍼런스에 새로 값을 지정하려(즉, 변경하려) 하지 마라.
- 레퍼런스를 쓸 수 있다면 포인터를 쓰지 마라.
- 지역변수에 대한 레퍼런스를 리턴하지 마라.
- 레퍼런스가 스코프가 벗어난 변수를 가리키도록 하지 마라.

11. 디버깅

정확한 버그의 원인을 알아내는 것은 꽤나 성가신 일이지만, 여기에도 몇가지 테크닉이 있다.

- 표준출력으로 프린트하여 - 프로그램이 간단한 경우, 몇몇 변수들의 값을 프린트해보고, 어떤 값인지 본다 - 무엇이 잘못되었는지 찾기
- 디버거를 이용하기. 디버거는 breakpoint를 설정하고, 실행도중에 코드를 추적해볼 수 있도록 해준다. 대부분의 IDE는 디버거가 같이 있다. GNU 시스템의 경우, gdb가 있다.
- 컴파일러에서 지원하는 옵션들을 사용하여 보다 많은 경고(warning)을 볼 수 있도록 해라. 예를들어 g++의 경우, -Wall 옵션을 사용해라.

디버깅에 도움이 되는 사이트 :

- Unix 환경에서 C, C++ 디버깅하기 http://www.liacs.nl/~jdassen/onderwijs/stuva/debug/debug.html
- MPatrol - 유용한 메모리 디버깅 툴 : http://www.cbmamiga.demon.co.uk/mpatrol
- NJAMD - 역시 유용한 메모리 디버깅 툴 : http://sourceforge.net/projects/njamd/
- LeakTracer - 메모리 릭을 찾는 간단하면서도 유용한 툴 : http://www.andreasen.org/LeakTracer/

11.1. 디버깅 파일

C++이나 C 프로그램을 디버깅하려면 23절 의 파일을 인클루드하고, 'Makefile'에 DEBUG_STR, DEBUG_PRT , DEBUG&_MEM을 디파인 해서 debug.h 의 함수들로 추적할 수 있도록 한다. '-DDEBUG_STR' 등을 없앨 때, 디버깅 함수들은 ((void)0) (NULL을 의미)으로 세팅될 것이다. 따라서 최종 결과물에는 아무런 영향을 주지 않는다. 이 디버깅 함수들은 프로그램에 다양하게 적용될 수 있을 것이고, 결과물의 크기를 증가시키지는 않을 것이다.

debug 루틴의 구현을 위해서는 23절 를 참조해라.

또한, 23절 에서 debug.h 와 디버깅 함수들을 이용한 예제 프로그램을 보아라.

23절 의 예를 보라.

12. C++을 위한 IDE(Integrated Development Enviroment)와 에디터

C++로 프로그래밍을 할 때, 에디터나 IDE를 사용하는 것이 좋다. 대부분의 프로그래머는 자신이 좋아하는 것들을 갖게 마련이고, 어떤 것이 좋은지에 대해 거의 종교적인 믿음을 갖는다.

너는 내장 에디터와 컴파일러, 문서들과 기타 등등으로 모두 포함하는 IDE (Intergrated Development Environment : 통합 개발환경)를 사용할 수도 있다. 또는 몇몇 사람들이 그러는 것처럼, 단순한 에디터만을 사용할 수도 있다.

12.1. IDE 들

C++ 개발을 위한 다음과 같은 IDE (Integrated Development Environment) tool이 있다.

- 최고로 인정받는 Dev-C++은 win32와 Linux에서 동작하는 만능 IDE이다. 이는 GCC와 Mingw, Cygwin을 컴파일러와 라이브러리로 사용한다. 이는

http://www.bloodshed.net/devcpp.html 와 미러 사이트 (http://sourceforge.net/projects/dev-cpp) 에서 구할 수 있다.

- KDE KDevelop Kdevelop (http://www.kdevelop.org)
- Blatura site C++ Tools (http://home.xnet.com/~blatura/linapp6.html#tools)
- Amulet Amulet (http://www.cs.cmu.edu/afs/cs.cmu.edu/project/amulet/www)
- App Dev suite Angoss (http://www.angoss.com/sw/home.htm)
- Make replacement Brass (http://wsd.iitb.fhg.de/~kir/brashome/)
- S/W product metrics CCC (http://www.fste.ac.cowan.edu.au/~tlittlef)
- Project mgmt, edit, compile, debug C-Forge (http://www.codeforge.com)
- Dev environment Code Crusader (http://www.cco.caltech.edu/~jafl/jx/progs.html)
- Graphic gdb Code Medic (http://www.cco.caltech.edu/~glenn/medic/)
- Code analysis CodeWizard (http://www.parasoft.com/wizard/index.html)
- Gen HTML, LaTex for C++ cod Doc C++ (http://www.zib.de/Visual/software/doc++/index.html)
- GUI toolkit openGL Ftk (http://fltk.easysw.com/)
- C++ and Java IDE GLG IDE (http://www.genlogic.com)
- HP IDE HP Eloquence (http://www.hp-eloquence.com/sales/info.html)
- IDE C++, Java, Pascal RHIDE (http://www.tu-chemnitz.de/~sho/rho/rhide.html)
- IDE for C++, Java SNiff (http://www.takefive.com/)
- IDE for C++, Java Wipeout (http://www.softwarebuero.de/wipeout-eng.html)
- X-based dev env XWPE (http://www.rpi.edu/~payned/xwpe)

12.2. 에디터

IDE의 문제점은 이에 같이 들어있는 에디터의 기능이 매우 떨어진다는 것이다. 따라서 많은 사람들은 파워풀한 에디터를 원했고, 컴파일러와 함께 사용한다.

파워풀한 에디터로는 vim과 emacs를 들 수 있다. 둘 다 모두 많은 플랫폼에서 동작하고, 효율성을 높여줄 신택스 하이라이팅을 비롯한 여러가지 기능들을 갖고 있다.

다른 것으로는 UltraEdit(win32 only) 와 EditPlus(win32 only)가 있다.

- Vim 사이트 http://vim.sourceforge.net/
- Vim HOWTO http://www.linuxdoc.org/LDP/HOWTO/Vim-HOWTO.html
- Emacs 사이트 http://www.gnu.org/software/emacs/
- EditPlus for Windows 사이트 http://www.editplus.com/

- UltraEdit for Windows 사이트 http://www.ultraedit.com/

12.3. 다른 참고할만한 내용

- C++ Beautifier HOWTO http://www.linuxdoc.org/LDP/HOWTO/C-C++Beautifier-HOWTO.html (한글번역 (http://kldp.org/HOWTO/html/C-C++Beautifier/C-C++Beautifier-HOWTO.html))
- C++ 프로그램을 위한 버전관리 시스템 HOWTO (CVS HOWTO) http://www.linuxdoc.org/LDP/HOWTO/CVS-HOWTO.html
- 유용한 리눅스 사이트 http://www.milkywaygalaxy.freeservers.com 와 미러 사이트 - angelfire (http://www.angelfire.com/country/aldev0), geocities (http://www.geocities.com/alavoor/index.html), virtualave (http://aldev0.virtualave.net), 50megs (http://aldev0.50megs.com), theglobe (http://members.theglobe.com/aldev1/index.html), NBCi (http://members.nbci.com/alavoor), Terrashare (http://aldev.terrashare.com), Fortunecity (http://members.fortunecity.com/aldev), Freewebsites (http://aldev.freewebsites.com), Tripod (http://members.tripod.lycos.com/aldev), Spree (http://members.spree.com/technology/aldev), Escalix (http://www.escalix.com/freepage/aldev), Httpcity (http://www.httpcity.com/aldev/index.html), Freeservers (http://aldev.freeservers.com).

13. C++ Online 텍스트와 문서들

C++에 대한 수백만의 온라인 문서/텍스트/참고자료 가이드 등이 존재한다. 이는 C++이 매우 오랫동안 쓰이고 있기 때문이다. 아마 Google, Yahoo, Lycos, Excite 등의 인터넷 검색엔진을 사용하면 도움이 될 것이다.

- "C++ Annotations" 온라인 북 사이트:Annotations (http://www.icce.rug.nl/documents/cplusplus)
- "Teach Yourself C++ in 21 days" 온라인 사이트 Teach C++ (http://guides.oernii.sk/c++/index.htm)
- C++ Textbook by Bruce Eckel Thinking in C++ (http://www.mindview.net/Books/DownloadSites)
- C++ Open books: Panorama (http://www.softpanorama.org/Bookshelf/cpp.shtml) Open Books를 클릭하면 된다.
- "Who's Afraid of C++?" online textbook: Steveheller (http://www.steveheller.com/whos)
- "Introduction to Object Oriented Programming" an ebook C++ OOP (http://www.gnacademy.org/uu-gna/text/cc/Tutorial/tutorial.html)
- C++ in Hypertext C++ Hypertext (http://cs.nmhu.edu/personal/curtis/cs1htmlfiles/Cs1text.htm)

- Object Oriented Systems OOP article (http://www.javaworld.com/javaworld/jw-07-1999/jw-07-toolbox_p.html)
- C++ Language Reference from cplusplus.com http://www.cplusplus.com/ref
- C++ Documentation from cplusplus.com http://www.cplusplus.com
- Common C++ Pitfalls to be avoided http://www.horstmann.com/cpp/pitfalls.html
- Porting C++ to Java PortingC (http://www.ibm.com/java/education/portingc)
- C/C++ Journals UtahJournals (http://www.math.utah.edu/pub/tex/bib/toc/cccuj.html)
- Yahoo C++ category site CCyahoo (http://dir.yahoo.com/Computers_and_Internet/Programming_Languages/C_and_C__/C__)
- C Library Reference Guidec_guide (http://www.acm.uiuc.edu/webmonkeys/book/c_guide/index.html)
- Online textbooks C++/Java FreeLib (http://www.informit.com/free_library)
- "C++ In Action" by Bartosz Milewski at http://www.relisoft.com/book/index.htm
- 어떻게 코드를 쓰면 안되는가에 대한 재미있는 예제. "How to write unmaintainable code" http://mindprod.com/unmain.html

C++ 프로그래머를 위해 유용할 Java 책들 :

- Great Web reference site WebRef (http://www.webreference.com/index2.html)
- Many Java books JBooks (http://freewarejava.com/books/index.shtml)
- Intro to Java V3.0 JavaNotes (http://math.hws.edu/javanotes) mirror JavaNotes (http://matrix.mvhs.fuhsd.org/~deruiter/javanotes3/contents.html)
- Web Library: http://www.itlibrary.com
- Thinking in Java: Thinking Java (http://www.mindview.net/Books/DownloadSites)
- John Hopkins Univ - Java resources Hall (http://www.apl.jhu.edu/~hall/java)
- online Java tutorial Chortle (http://chortle.ccsu.ctstateu.edu/cs151/cs151java.html)
- Practical guide for Java SunBooks (http://java.sun.com/docs/books/tutorial/index.html)
- Java Soton (http://journals.ecs.soton.ac.uk/java/tutorial/TOC.html)

13.1. C++ 사이트들

다음의 C++ 사이트들을 방문해보라 :-

- C++ STL basic_string class 문서 - http://www.sgi.com/tech/stl/basic_string.html.
- 17절 섹션을 참고해라.

- C++ Crash-proof site http://www.troubleshooters.com/codecorn/crashprf.htm
- C++ Memory sitehttp://www.troubleshooters.com/codecorn/memleak.htm
- GNU Main site http://www.gnu.org and gnu C++ site at http://gcc.gnu.org
- AS University C++ Standard String class http://www.eas.asu.edu/~cse200/outline
- Java JString for C++ http://www.mike95.com/c_plusplus/classes/JString/JString_cpp.asp
- C++ Language Reference http://www.msoe.edu/~tritt/cpplang.html
- C++ Program examples and samples http://www.msoe.edu/~tritt/cpp/examples.html
- Neil's C++ stuff http://www.cyclone7.com/cpp

인터넷에는 어마어마하게 많은 C++ 문서들이 있다. Google, Yahoo, Lycos, Infoseek, Excite 같은 검색엔진에 가서 다음의 키워드를 넣어보아라. **'C++ tutorials' 'C++ references' 'C++ books' Advanced** 를 클릭해서 **search by exact phrase**를 선택함으로써 더 정확한 결과를 얻을 수 있을 것이다.

- http://www.google.com
- http://www.yahoo.com
- http://www.lycos.com
- http://www.infoseek.com
- http://www.excite.com
- http://www.mamma.com

13.2. C++ Tutorials

인터넷에는 많은 수의 온라인 튜토리얼이 있다. 검색엔진에서 'C++ tutorials'로 검색을 해봐라.

- C++ Tutorial http://www.xploiter.com/programming/c/index.shtml
- Cplusplus.com Tutorial http://www.cplusplus.com/doc/tutorial
- C++ Tutorial IISc, India
  http://www.csa.iisc.ernet.in/Documentation/Tutorials/StyleGuides/c++-style.html
- C++ Tutorial Brown Univ http://wilma.cs.brown.edu/courses/cs032/resources/C++tutorial.html
- C++ Tutorial http://home.msuiit.edu.ph/~ddd/tutorials/cpp/cpplist.htm
- C++ Tutorial IOstreams http://osiris.sunderland.ac.uk/~cs0pdu/pub/com365/Sched3/iocpp.html

13.3. 유용한 링크들

- Bird's eye view of C++ URLs (153개의 URL link가 있다) http://www.enteract.com/~bradapp/links/cplusplus-links.html
- This URL: http://www.snippets.org 360개의 파일에 해당하는 C code들이 있다.
- Mathtools http://www.mathtools.net 는 모든 과학, 공학적 필요에 의한 기술적인 컴퓨팅 포탈이다. 이 포탈사이트는 무료이고, 2만여개의 유용한 기술적인 컴퓨터 프로그래머들의 링크를 가지고 있고, C/C++, Java, Excel, MATLAB, Fortran 등을 다루고 있다.

13.4. C++ Quick-Reference

검색엔진에서 'C++ Reference' 로 검색을 해보아라.

- C++ quick ref http://www.cs.jcu.edu.au/~david/C++SYNTAX.html
- C++ Standard Library Quick Reference http://www.halpernwightsoftware.com/stdlib-scratch/quickref.html
- C++ STL from halper http://www.halpernwightsoftware.com/stdlib-scratch/quickref.html

13.5. C++ Usenet 뉴스그룹

- C++ newsgroups : comp.lang.c++.announce (news:comp.lang.c++.announce)
- C++ newsgroups : comp.lang.c++.* (news:comp.lang.c++.*)
- C++ newsgroups : http://marshall-cline.home.att.net/cpp-faq-lite

13.6. Java 형태의 API

C++에서 Java 형태의 API를 제공하는 다음 사이트들을 가보아라.

- Java utils in C++ http://www.pulsar.org/users/ej/archive/oop
- PhD Thesis book Java API in C++ http://www.pulsar.org/archive/phd/ejphd

14. C++ 코딩 관습

코딩 관습은 프로그램의 가독성과 유지보수를 위해 매우 중요한 요소이다. 또한 프로그래머의 생산성을 크게 향상시킨다. 이는 좋은 코딩 훈련을 위해 필요하다. 아래의 내용은 class 정의에 있어 제안된 것이다.

- 모든 public 변수들은 **mFooVar**과 같이 **m** 으로 시작해야 한다. **m** 은 **member**를 의미한다.
- 모든 protected 변수들은 **mtFooVar** 와 같이 **mt** 로 시작해야하고, 메쏘드는 **tFooNum()** 와 같이 t로 시작해야 한다. **t** 는 **protected**를 의미한다.
- 모든 private 변수들은 **mvFooVar**와 같이 **mv** 로 시작해야하고, 메쏘드들은 **vFooLone()** 와 같이 v로 시작해야 한다. **v** 는 **private** 을 의미한다.
- 모든 public, protected, private 변수이름들은 m 다음에는 **mFooVar** 의 F같이 대문자로 시작해야한다.
- 모든 포인터 변수들은 다음과 같이 p로 시작해야 한다.
  - Public 변수 **mpFooVar** 과 메쏘드 FooNum()
  - Protected 변수 **mtpFooVar** 와 메쏘드 tFooNum()
  - Private 변수 **mvpFooVar** 와 메쏘드 vFooNum()

세계적으로 일관된 C++ 코딩 관습은 보다 프로그래밍을 잘 할 수 있도록 도와줄 것이다.

아래에 주어진 예제 코드에서 **t** 는 **protected**를, **v** 는 **private**를, **m** 는 **member-variable** 를, **p** 는 **pointer**를 의미한다.

```
class SomeFunMuncho
{
public:
int mTempZimboniMacho; // OOP에서는 오직 임시 변수들만 public이어야 한다.
float *mpTempArrayNumbers;
int HandleError();
float getBonyBox();  // 변수에 접근하기 위한 함수
float setBonyBox();  // 변수에 접근하기 위한 함수

protected:
float mtBonyBox;
int *mtpBonyHands;
char *tHandsFull();
int tGetNumbers();
private:
float mvJustDoIt;
char mvFirstName[30];
int *mvpTotalValue;
char *vSubmitBars();
int vGetNumbers();
};
```

프로그램이 수백만 라인으로 커지게 되면, 위와 같은 관습을 매우 좋아하게 될 것이다. 단순히 **mvFirstName** 라는 변수이름을 본 것만으로도, 이것이 클래스의 멤버이고, private 변수라는 것을 알 수 있으므로 코드의 가독성은 올라가게 된다.

다음의 C++ 코딩 표준 URL들을 방문해보아라.

- C++ FAQ Lite - Coding standards http://www.parashift.com/c++-faq-lite/coding-standards.html
- Rice university coding standard http://www.cs.rice.edu/~dwallach/CPlusPlusStyle.html
- Identifiers to avoid in C++ Programs http://oakroadsystems.com/tech/cppredef.htm
- Coding standards from Possibility http://www.possibility.com/Cpp/CppCodingStandard.html and mirror site (http://www.cs.umd.edu/users/cml/cstyle/CppCodingStandard.html)
- Coding standards for Java and C++ from Ambysoft http://www.ambysoft.com/JavaCodingStandards.html
- Rules and recommendations http://www.cs.umd.edu/users/cml/cstyle/
- Indent and annotate http://www.cs.umd.edu/users/cml/cstyle/indhill-annot.html
- Elemental rules http://www.cs.umd.edu/users/cml/cstyle/Ellemtel-rules.html
- C++ style doc http://www.cs.umd.edu/users/cml/cstyle/Wildfire-C++Style.html
- C++ Coding Standards by Brett Scolcum http://www.skypoint.com/~slocum/prog/cppstds.html
- Logikos C++ Coding Standards http://www.logikos.com/standards/cpp_std.html
- NRad C++ coding standards http://cadswes.colorado.edu/~billo/standards/nrad
- BEJUG C++ coding standards http://www.meurrens.org/ip-Links/java/joodcs/ToddHoff.html
- Arctic Labs coding standards http://www.arcticlabs.com/codingstandards

See also

- For rapid navigation with ctags Vim color text editor (http://www.linuxdoc.org/LDP/HOWTO/Vim-HOWTO.html)
- To improve productivity see C++ Beautifier HOWTO (http://www.linuxdoc.org/LDP/HOWTO/C-C++Beautifier-HOWTO.html)

15. C++ 스크립트 언어

C++의 큰 단점은 조그만 변경을 할 때마다 항상 컴파일과 링크를 다시 해주어야 실행파일을 만들 수 있다는 것이다. 컴파일/링크/디버깅 사이클은 많은 시간이 걸리고 생산적이지 못하다. 현대의 CPU와 RAM은 매우 빠르고 싸지고 있으므로, 어떤 때는 하드웨어에 많은 돈을 투자하고, 개발을 위해서는 script 언어를 쓰는 것이 좋을 수도 있다.

15.1. PIKE & PHP (C/C++ Scripting Languages)

PHP나 PIKE 같은 스크립트 언어는 링킹과 재컴파일 과정을 없앴고, 따라서 개발 과정을 단축시킬 수 있다.

메모리(RAM) 가격이 떨어지고, CPU 속도가 올라감에 따라 PHP나 PIKE같은 스크립트 언어의 인기는 폭발할 것이다. PHP나 PIKE는 객체지향과 C/C++을 닮은 문법으로 인해 가장 널리 쓰이는 스크립트 언어가 될 것이다.

PHP나 Pike C++ 스크립트 언어를 씀으로써 프로그래밍의 생산성은 다섯 배 나 증가할 것이다. 그리고 PHP나 PIKE는 '개념의 증명'을 위해 유용하고, 프로토타입을 빠르게 만들어낼 수 있다.

PHP는 일반적인 프로그래밍이나 웹 프로그래밍에서 매우 인기를 얻고 있다. PHP는 가까운 미래에 가장 널리 쓰이는 스크립트 언어가 될 것이다. PHP 는 http://www.linuxdoc.org/HOWTO/PHP-HOWTO.html (한글번역 (http://kldp.org/HOWTO/html/PHP/PHP-HOWTO.html)) 에 있다.

The Pike 는 http://pike.roxen.com 와 http://www.roxen.com 에 있다.

Roxen 웹서버는 순수하게 Pike로 쓰여져서, 얼마나 Pike가 강력한지 보여주고 있다. Pike는 몇몇 작업에 대해서는 Java보다 빠르게 돌아가고, 메모리도 상당히 효율적으로 사용한다.

15.2. SoftIntegration Ch (C/C++ 스크립트 언어)

만약 상용 스크립트 언어를 원한다면, SoftIntegration corporation (http://www.softintegration.com) 에서 'Ch scripting'을 보아라.

Ch라 불리는 스크립트 언어 환경은 C를 포함하고, 높은 수준의 확장들과 C++을 비롯한 다른 언어들의 특징들을 포함한다. 따라서 한번 배우기만 하면 거의 모든 종류의 프로그래밍에 사용할 수 있다. 이 C호환 스크립트 언어 환경은 여러 플랫폼에서 한 프로그램을 이식가능하게 하는 중간 역할을 한다. 이식가능한 Ch 코드는 인터넷과 인트라넷을 통해 슈퍼컴퓨터부터 웍스테이션, PC, 팜(Palm), PDA 를 비롯 컴퓨터의 범주에 속하지 않는 CNC 머신, 로봇, TV, 냉장고 등 으로 퍼져 어디서나 안전하게 돌아갈 수 있다.

15.3. PHP (C++ Scripting Language)

PHP는 웹문서를 처리하는 스크립트 언어로 매우 빠르게 진화했고, 객체지향을 지원한다. PHP는 'class'라는 키워드를 가지고 객체지향 스크립팅을 구현하고자 했다. 아마도 가까운 시일내에 PHP는 객체 지향 프로젝트를 위한 강력한 스크립트 언어로 빠르게 진화할 것이다. 가까운 미래에 PHP는 웹프로그래밍 뿐 아니라 일반적인 어플리케이션 프로그래밍에도 쓰이게 될 것이다. 웹과 일반 어플리케이션에 다른 언어를 쓰기 보다는 그냥 PHP만 쓰면 될 것이기 때문이다. PHP HOWTO : http://www.linuxdoc.org/HOWTO/PHP-HOWTO.html (한글번역 (http://kldp.org/HOWTO/html/PHP/PHP-HOWTO.html).

16. Templates

Template 는 코드 재사용을 쉽게 만들어 generic 프로그래밍을 가능하게 하는 C++의 특징이다.

아래와 같은 간단한 예를 보자 :

```
#include <string>
#include <iostream>

void printstring(const std::string& str) {
    std::cout << str << std::endl;
}

int main()
{
    std::string str("Hello World");
    printstring(str);
}
```

printstring() 는 **std::string** 를 첫번째 인자로 받는다. 따라서 이는 오직 string만 프린트 할 수 있고, 문자배열(char array)을 프린트 하기 위해서는 함수를 오버로딩시키든가 새로운 이름의 함수를 만들어야 한다.

이는 함수의 구현이 중복되므로 좋지 않은 것이고, 유지보수하기가 힘들어지게 된다.

template을 쓰면 우리는 코드를 재사용가능하게 만들 수 있다. 아래와 같은 함수를 보라 :

```
template<typename T>
void print(const T& var) {
    std::cout << var << std::endl;
}
```

컴파일러는 우리가 무슨 타입을 넘겨주든지 알아서 **print** 함수의 코드를 자동으로 만들어 줄 것이다. 이것이 template의 중요한 장점이다. Java는 template이 없고, 따라서 Java에서의 generic 프로그래밍과 코드 재 사용은 더 힘들다.

레퍼런스 :

- http://babbage.cs.qc.edu/STL_Docs/templates.htm 미러 : http://www.mike95.com/c_plusplus/tutorial/templates
- 이것은 #pragma template에 대해 알려준다 : - http://www.dgp.toronto.edu/people/JamesStewart/270/9697f/notes/Nov25-tut.html
- 매우 좋은 site: http://www.cplusplus.com/doc/tutorial/tut5-1.html http://www.cplusplus.com/doc/tutorial

- C++의 검증을 위해 : http://examware.com 에 가서 "Tutorials"를 클릭하고 C/C++ 버튼을 누르시오.
- C++ Open books: http://www.softpanorama.org/Lang/cpp.shtml tutorial을 클릭
- Templates tutorial : http://www.infosys.tuwien.ac.at/Research/Component/tutorial/prwmain.htm

17. STL References

STL에 관한 다음 사이트들을 방문해보라 :

- iterator에 대한 매우 좋은 설명 http://www.cs.trinity.edu/~joldham/1321/lectures/iterators/
- Intro to STL SGI http://www.sgi.com/tech/stl/stl_introduction.html
- Mumits STL 초보 가이드 (약간 오래된 것) http://www.xraylith.wisc.edu/~khan/software/stl/STL.newbie.html
- ObjectSpace 예제 : ObjectSpace는 300개가 넘는 예제를 가지고 있고, 따라서 초보자에게 아주 좋은 출발점을 제시해준다. ftp://butler.hpl.hp.com/stl/examples.zip
- Joseph Y. Laurino's STL page. http://weber.u.washington.edu/~bytewave/bytewave_stl.html
- Marian Corcoran's STL FAQ. ftp://butler.hpl.hp.com/stl/stl.faq

STL tutorials:

- Phil Ottewell's STL Tutorial - http://www.yrl.co.uk/~phil/stl/stl.htmlx
- 좋지만 오래된 문서 - http://www.decompile.com/html/tut.html 미러 : http://mip.ups-tlse.fr/~grundman/stl-tutorial/tutorial.html
- The Code Project, C++/STL/MFC 에 대한 소개 http://www.codeproject.com/cpp/stlintroduction.asp
- C++ Standard Template Library, another great tutorial, by Mark Sebern http://www.msoe.edu/eecs/cese/resources/stl/index.htm
- Technical University Vienna by Johannes Weidl http://dnaugler.cs.semo.edu/tutorials/stl mirror http://www.infosys.tuwien.ac.at/Research/Component/tutorial/prwmain.htm

Main STL sites:

- C++ STL from SGI http://www.sgi.com/tech/stl
- C++ STL from RPI univ http://www.cs.rpi.edu/projects/STL/htdocs/stl.html

- C++ STL site ODP for STL (http://dmoz.org/Computers/Programming/Languages/C++/Class_Libraries/STL) 와 미러 (http://dir.lycos.com/Computers/Programming/Languages/C%2B%2B/Class_Libraries/STL)
- STL for C++ Programmers http://userwww.econ.hvu.nl/~ammeraal/stlcpp.html
- C++ STL from halper http://www.halpernwightsoftware.com/stdlib-scratch/quickref.html

17.1. STL 개요

STL은 프로그래머에게 몇가지 유용한 데이터구조와 알고리즘을 제공한다. 이는 다음과 같은 것들이 있다.

- 컨테이너. 두 가지 타입이 있다 :
  - 순차적(Sequential). 여기에는 vector, list, deque 등이 있다.
  - 정렬된 조합(Associative). 여기에는 set, map, multiset, multimap 이 있다.
- Iterator. 컨테이너의 내용을 살펴볼 수 있게 해주는 포인터 같은 것들이다.
- 일반적인(generic) 알고리즘들. STL은 컨테이너 타입에 대해 동작하는 여러가지 효과적으로 구현된 표준 알고리즘들 (예를들어 find, sort, merge 등)이 있다. (몇몇 container들은 이 중 일부를 특별한 목적으로 멤버함수로 갖고 있다)
- Function obejct. function object는 operator()의 정의를 제공하는 class의 instance이다. 이는 이 object들을 함수 같이 사용할 수 있다는 것이다.
- Adaptors. STL은 다음과 같은 것들을 제공한다.
  - Container adaptor는 vector를 stack을 만들기 위한 기초로 사용할 수 있게 해준다.
  - function adaptor 는 이미 존재하는 function object로부터 새로운 function object를 만들 수 있게 해준다.
- Allocators. 모든 STL 컨테이너 class는 프로그램이 사용하는 메모리 정보를 갖고 있기 위한 allocator class를 사용한다. 하지만 나는 이 부분은 생략할 것이다.

앞으로 vector, list, set 그리고 map 컨테이너의 사용법을 살펴볼 것이다. 이들을 사용하기 위해서는 내가 STL iterator에 대해 말할 수 있도록 iterator를 쓸 줄 알아야 할 것이다. 또 set과 map 컨테이너를 사용한다는 것은 내가 function object에 대해 뭔가 설명할 수 있도록 간단한 function object가 있어야 한다는 것이다. STL이 지원하는 알고리즘에 대해서는 간단히 설명할 것이고, adoptor는 언급하지 않을 것이다.

몇몇 함수 인자의 타입에 대해서 이름이 바뀔 수 있다. 예를들어 대부분의 int 타입 인자들은 실제로는 size_type이라는 type을 갖고 이것이 적절한 기본 타입으로 tyepdef되는 형태에 의해 쓰인다. 만약 여러 함수들의 실제 인자 타입을 알고싶다면 작업하는 것에 대한 문서나 헤더파일을 참고해라.

STL에서 제공되는 몇가지 유틸리티 class들이 있는데, 이 중 제일 중요한 것은 pair class이다. 이는 다음과 같이 정의되어있다.

```
template<class T1, class T2>
class pair {
public:
    T1 first;
    T2 second;
    pair(const T1& a, const T2& b) : first(a), second(b) {}

};
```

그리고 쉽게 pair를 만들도록 다음과 같은 make_pair 함수가 제공된다 :

```
pair<T1,T2> make_pair(const T1& f, const T2&,s)
```

또한 ==와 < 연산자도 있다. 이 template class에는 복잡한 것이 없고 그냥 사용하면 된다. 이를 이용하기 위해서는 #include 로 <utility>를 include하면 된다. pair는 여러곳에서 쓰일 수 있는데, 특히 set과 map class에서 많이 나타난다.

17.2. 헤더 파일

STL을 사용하기 위해서는 적절하게 헤더파일을 #include 해주어야 한다. 만약 컴파일러가 표준에 맞지 않는다면 약간 다를 수도 있지만, 표준에 맞는 컴파일러 (g++ 같은)는 다음과 같이 하면 된다 :

- **<vector>** : 벡터 타입
- **<list>** : 리스트 타입
- **<set>** : 집합(set) 타입
- **<map>** : 맵 타입
- **<algorithm>** : 일반적인 알고리즘 들

표준 C++ 라이브러리는 **.h** 를 뒤에 붙이지 않는 다는 것에 주의해라. 만약 옛버전의 혹은 좋지 않은 컴파일러를 사용하는데, 위와 같이 해서 include가 되지 않는다면 **.h**를 붙여서 시도해보아라. 하지만 그보다는 새로운 컴파일러를 구하는 게 더 나을 것이다.

17.3. 컨테이너 class 인터페이스

컨테이너 class들은 서로 같은 이름을 갖는 멤버함수를 많이 갖는다. 이 함수들은 모든 class에 대해 똑같은 (혹은 매우 비슷한) 인터페이스를 제공한다 (그러나 물론 그 내부 구현은

다를 것이다). 아래의 표는 우리가 살펴볼 함수들을 나열한 것이다. 별표는 그 컨테이너 타입이 그 이름의 멤버 함수를 제공한다는 것이다.

표 **2.** 컨테이너 **Class** 인터페이스

연산자/함수명	목적	vector	list	set	map
==	비교	*	*	*	*
<	비교	*	*	*	*
begin	iterator	*	*	*	*
end	iterator	*	*	*	*
size	원소의 수	*	*	*	*
empty	비었는지	*	*	*	*
front	첫번째 원소	*	*		
back	마지막 원소	*	*		
[ ]	원소 접근 및 변경	*			*
insert	원소(들) 추가	*	*	*	*
push_back	맨 뒤에 원소 추가	*	*		
push_front	맨 앞에 원소 추가		*		
erase	원소(들) 삭제	*	*	*	*
pop_back	맨 뒤 원소 삭제	*	*		
pop_front	맨 앞 원소 삭제		*		

만약 아래의 내용중 의문가는 부분이 있으면 (아마도 잇을 것이다), 조그만 테스트 프로그램을 하나 짜서 어떻게 돌아가는 지 알아볼 수 있을 것이다.

17.4. 벡터 : Vectors

벡터는 C++의 배열과 비슷한, 하지만 이를 발전시킨 컨테이너이다. 특히, 벡터는 선언시에 얼마나 벡터가 커야할지를 알 필요가 없고, **push_back** 함수를 이용하여 언제나 새로운 원소를 추가할 수 있다. (사실 **insert** 함수가 어디에든 새 원소들을 넣을 수 있게 해주지만, 이는 매우 비효율적이다. 만약 이를 자주 해야한다면 list를 대신 사용하는 것을 고려해보아라.)

17.4.1. 벡터 만들기

벡터는 class template이므로, 선언시에 벡터가 갖게 될 객체의 타입을 선언해주어야 한다. 예를들어 다음과 같다.

```
vector<int> v1;
vector<string> v2;
vector<FiniteAutomaton> v3;
```

위 내용은 v1을 int 값을 갖는 벡터로, v2를 string을 갖는 벡터로, v3를 FiniteAutomaton (아마도 미리 이런 타입이 선언되었을 것이다) 를 갖고 있는 벡터로 선언한다. 이 선언들은 벡터의 크기에 대해 전혀 알려주지 않고 (구현에 따라 기본 벡터 사이즈가 있을 것이다.) , 필요에 따라 얼마든지 늘려서 쓸 수 있다.

그러나 다음과 같이 선언함으로써 초기 크기를 정해줄 수도 있다.

```
vector<char> v4(26);
```

이는 v4가 문자(char)의 벡터가 되고, 처음에는 26개의 글자를 가질 수 있다는 것이다. 또한, 벡터 안에 들어가는 수들을 초기화 하는 방법도 있는데, 이는 다음과 같다.

```
vector<float> v5(100,1.0);
```

위 선언은 **v5** 가 100개의 1.0으로 초기화 된 실수값을 갖는 벡터임을 선언한다.

17.4.2. 벡터를 체크하기

한번 벡터를 만든 후에는, **size** 함수를 써서 현재 벡터의 크기를 알아낼 수 있다. 이 함수는 아무 인자 없이 벡터에 들어있는 원소 수를 나타내는 integer를 리턴한다. (엄밀하게 말하자면 **size_type** 타입이 리턴되지만, 이것이 바로 integer로 바뀌어진다) 그렇다면 다음의 작은 프로그램으로 무엇이 출력될까?

```
<vector-size.cpp>=
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> v1;
    vector<int> v2(10);
    vector<int> v3(10,7);

    cout << "v1.size() returns " << v1.size() << endl;
    cout << "v2.size() returns " << v2.size() << endl;
    cout << "v3.size() returns " << v3.size() << endl;
}
```

벡터가 비었는지를 체크하기 위해서는, empty 함수를 쓰면 된다. 이것도 역시 아무 인자 없이 boolean 값을 리턴하는데, 비어있으면 true를, 비어있지 않으면 false를 리턴한다. 그렇다면 다음의 프로그램은 무엇을 프린트할까 (true는 1로, false는 0으로 프린트된다)?

```
<vector-empty.cpp>=
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> v1;
    vector<int> v2(10);
    vector<int> v3(10,7);

    cout << "v1.empty() has value " << v1.empty() << endl;
    cout << "v2.empty() has value " << v2.empty() << endl;
    cout << "v3.empty() has value " << v3.empty() << endl;
}
```

17.4.3. 벡터의 원소에 접근하기

벡터의 원소는 []연산자를 사용해서 접근할 수 있다. 따라서 모든 원소를 프린트하려면 다음과 같이 하면 된다.

```
vector<int> v;
// ...
for (int i=0; i<v.size(); i++)
     cout << v[i];
```

(이는 원래의 배열을 사용하는 것과 매우 비슷하다).

또한, [] 연산자는 벡터 원소의 값을 바꾸기 위해서도 쓰일 수 있다.

```
vector<int> v;
// ...
for (int i=0; i<v.size(); i++)
      v[i] = 2*i;
```

front 함수는 벡터의 첫번째 원소를 리턴한다.

```
vector<char> v(10,'a');
// ...
char ch = v.front();
```

또한, front를 이용해서 첫번째 원소의 값을 바꿀 수도 있다.

```
vector<char> v(10,'a');
// ...
v.front() = 'b';
```

back 함수는 front와 같은 역할을 하지만, 벡터의 맨 마지막 원소를 리턴하는 것이 다르다.

```
vector<char> v(10,'z');
// ...
char last = v.back();
v.back() = 'a';
```

[]를 사용하는 간단한 예제를 보자.

```
<vector-access.cpp>=
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    vector<int> v1(5);
    int x;
    cout << "Enter 5 integers (seperated by spaces):" << endl;
    for (int i=0; i<5; i++)
          cin >> v1[i];
    cout << "You entered:" << endl;
    for (int i=0; i<5; i++)
          cout << v1[i] << ' ';
    cout << endl;
}
```

17.4.4. 벡터의 원소를 추가 / 삭제하기

위에 언급된 [] 외에도, 벡터의 원소에 접근하거나 바꿀 수 있는 방법이 몇가지 더 있다.

- push_back은 새로운 원소를 벡터의 끝에 더할 것이다.
- pop_back은 벡터의 끝에서 원소를 하나 없앨 것이다.
- insert 는 하나 또는 여러개의 원소를 벡터의 원하는 위치에 삽입할 것이다.
- erase는 하나 또는 여러개의 원소를 원하는 위치에서 없앨 것이다.

그런데 insert나 erase는 벡터에서 오버헤드가 큰 연산임에 주의하라. 만약 insert나 erase를 써야한다면, 벡터 대신 list 데이터구조를 사용하는 것이 더 효율적일 것이다.

```
<vector-mod.cpp>=
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    vector<int> v;

    for (int i=0; i<10; i++) v.push_back(i);
    cout << "Vector initialised to:" << endl;
    for (int i=0; i<10; i++) cout << v[i] << ' ' ;
    cout << endl;

    for (int i=0; i<3; i++) v.pop_back();
    cout << "Vector length now: " << v.size() << endl;
    cout << "It contains:" << endl;
    for (int i=0; i<v.size(); i++) cout << v[i] << ' ';
    cout << endl;

    int a1[5];
    for (int i=0; i<5; i++) a1[i] = 100;

    v.insert(& v[3], & a1[0],& a1[3]);
    cout << "Vector now contains:" << endl;
    for (int i=0; i<v.size(); i++) cout << v[i] << ' ';
    cout << endl;

    v.erase(& v[4],& v[7]);
    cout << "Vector now contains:" << endl;
    for (int i=0; i<v.size(); i++) cout << v[i] << ' ';
    cout << endl;
}
```

위의 예에서는 벡터 v가 선언된 후, push_back을 사용하여 초기화 되었다. 그리고 pop_back으로 뒤의 몇 원소가 없어졌고, 배열이 하나 만들어져서 그 내용이 insert를 사용해서 v에 삽입되었다. 마지막으로 몇몇 원소들을 지우기 위해 erase가 사용되었다. 위에 사용된 함수들은 다음과 같은 인자들을 받는다.

- push_back : vector에 들어가는 것과 같은 타입의 인자를 하나 받는다.
- pop_back : 인자를 받지 않는다. 그리고 빈 벡터에 대해 pop_back을 하면 안된다.
- insert 는 세 가지 형태로 쓰인다.

- insert(pos, T& x) : 원소 x 하나를 pos가 가리키는 위치에 삽입한다.
- insert(pos, start, end) : 다른 컨테이너 안의 내용을 pos가 가리키는 위치에 삽입한다.
- 삽입되는 원소들은 start에서 시작해서, end를 만날 때까지 (end가 가리키는 것은 들어가지 않는다) 이다.
- insert(pos, int rep, T& x) : rep 개의 x값을 pos 위치에 삽입한다. (같은 값을 여러번 삽입)

위의 코드에 나와있듯이, pos가 가리키는 포지션 값은 원소가 삽입될 곳의 주소여야 한다. 마찬가지로 start와 end도 주소값이어야 한다. (사실 이것은 이들이 iterator이기 때문이다. 이에 대해서는 다음 장에서 더 살펴볼 것이다.)

- erase는 두 가지 형태로 쓰인다 (pos, start와 end는 insert와 같은 형식을 갖는다)

  - erase(pos) : pos가 가리키는 위치의 원소를 없앤다.
  - erase(start,end) : start에서 end까지(end는 포함하지 않음)에 해당하는 원소들을 없앤다.

17.4.5. Vector Iterator

벡터 v의 원소들을 차례대로 보는 가장 쉬운 방법은 위에 한 방법같이 하는 것이다.

```
for (int i=0; i<v.size(); i++) { ... v[i] ... }
```

또다른 방법은 바로 iterator를 이용하는 것이다. iterator는 컨테이너의 포인터라고 생각하면 된다. 따라서 이를 증가시키면서 원소를 하나씩 접근하는 것이 가능하다. 벡터가 아닌 컨테이너의 경우는 iterator가 원소를 차례대로 접근할 수 있는 유일한 방법이다.

type T의 원소를 갖고 있는 벡터의 경우 :

```
vector<T> v;
```

iterator는 다음과 같이 선언된다.

```
vector<T>::iterator i;
```

이러한 iterator는 begin()이나 end()같은 함수에 의해 리턴되는 값으로 만들어진다. 같은 타입의 iterator들은 == 나 != 로 비교가능하고, ++을 이용한 증가나 *를 이용한 참조 등이 가능하다. [이 외에도 벡터 iterator는 더 많은 연산자를 갖고 있다. 이에 대해서는 다음 장을 참고해라].

다음은 iterator를 어떻게 벡터와 사용하는 지에 대한 예제이다.

```
<vector-iterator.cpp>=
#include <iostream>
#include <vector>
```

```
using namespace std;

int main()
{
    vector<int> v(10);

    int j = 1;

    vector<int>::iterator i;

    // v를 1에서 10까지의 정수로 채운다.
    i = v.begin();
    while (i != v.end())
    {
        *i = j;
        j++;
        i++;
    }

    // v의 각 원소를 제곱한다.
    for (i=v.begin(); i!=v.end(); i++) *i = (*i) * (*i);

    // v의 내용을 출력한다.
    cout << "The vector v contains: ";
    for (i=v.begin(); i!=v.end(); i++) cout << *i << ' ';
    cout << endl;

}
```

*i 가 등호의 왼쪽(LHS)에서는 값을 변경하기 위해, 오른쪽(RHS)에서는 값을 참조하기 위해 쓰인 것에 주목해라.

17.4.6. 벡터의 비교

두 개의 벡터를 ==와 <를 이용해서 비교할 수 있다. ==는 양 쪽의 벡터가 같은 수의 원소를 갖고 대응되는 각원소들이 모두 같을 때 true를 리턴할 것이다. <은 두 벡터의 원소들을 차례대로 사전순서(lexicographical order)대로 비교한다. 예를들어 v1과 v2를 비교한다고 해보자 (v1 < v2). i=0이라 할 때, v1[i] < v2[i] 이면 true를 리턴하고, v1[i] > v2[i] 이면 false를 리턴한다. 만약 둘이 같으면 i를 증가시킨다 (즉, 다음 원소로 넘어간다). 만약 v1의 끝이 v2가 끝나기 전에 나타났다면 (즉, v1의 원소의 개수가 더 작고, v1이 v2의 앞부분과 내용이 같을 때) true를 리턴하고, 그렇지 않으면 false를 리턴한다. 다음의 예를 보자.

```
(1,2,3,4) < (5,6,7,8,9,10) 는 false.
(1,2,3) < (1,2,3,4) 는 true
(1,2,3,4) < (1,2,3) 는 false
```

```
(0,1,2,3) < (1,2,3) 는 true
```

아래의 코드는 위에서 세번째 예를 보여준다.

```
<vector-comp.cpp>=
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    vector<int> v1;
    vector<int> v2;
    for (int i=0; i<4; i++) v1.push_back(i+1);
    for (int i=0; i<3; i++) v2.push_back(i+1);

    cout << "v1: ";
    for (int i=0; i<v1.size(); i++) cout << v1[i] << ' ';
    cout << endl;

    cout << "v2: ";
    for (int i=0; i<v2.size(); i++) cout << v2[i] << ' ';
    cout << endl;

    cout << "v1 < v2 is: " << (v1<v2 ? "true" : "false") << endl;
}
```

<= 와 >= 역시 예상하는 대로 동작할 것이다.

17.5. Iterator 와 STL

17절장을 봐라.

17.6. 리스트

17절장을 봐라.

17.7. 집합(Set)

set 컨테이너 타입은 벡터같이 인덱스를 통해 원소에 접근하는 것이 아니라, 원소를 직접 저장하고 뺄 수 있도록 해준다. set 컨테이너는 서로 다른 원소들만을 갖는 수학적인 집합과 같이 동작한다. 그러나, 수학적인 집합과는 다르게, 집합 안의 원소들은 (사용자가 지정하는) 어

떤 순서 대로 저장되게 된다. 실제로 이것은 set 컨테이너로 수학적인 집합을 구현하는 데 있어 작은 제한일 뿐이고, 이렇게 함으로써 순서가 없는 것보다 많은 연산에서 더 효율적이 될 수 있다.

17.7.1. Set을 만들기

set 컨테이너를 만들기 위해서는 두 가지 template 인자가 필요하다 - 이는 set이 갖게 될 원소들의 타입과 두 원소를 비교할 수 있는 비교함수 function object의 타입이다.

```
set<T, Compare> s;
```

(**set < T > s**와 같은 선언도 가능해야한다. 이는 두번째 인자로서 디폴트 template 인자인 less < T >를 사용한다. 하지만 많은 C++ 컴파일러 (g++포함)가 기본 template 인자를 지원하지 못하고 있다.)

간단한 타입 **T** 에 대해서는 **less < T >** function object를 쓸 수도 있다. ("function object"가 무엇인가 하는 고민은 할필요 없다.) 예를들어 아래와 같이 선언하면 된다.

```
set<int, less<int> > s1;
set<double, less<double> > s2;
set<char, less<char> > s3;
set<string, less<string> > s4;
```

(선언할 때 뒤쪽의 > 두 개가 space로 띄어져 있음에 주의하라. 이는 compiler가 >를 쉬프트 연산자(>>) 와 구별하기 위해 꼭 필요한 것이다.) 각각의 경우 function object들은 각각의 타입에 맞게 <를 사용할 것이다. (이는 각각 **int, double, char, string** 타입이다.)

아래의 코드는 정수(int)의 set을 선언하고, **insert** 메쏘드를 사용하여 정수를 몇개 추가한다. 그리고 set을 차례대로 보면서 원소들을 출력한다. 재미있는 것은 추가하는 순서가 어떤 순서이든지 set의 내용은 정렬된 상태로 출력된다는 것이다.

```
<set-construct1.cpp>=
#include <iostream>
#include <set>

using namespace std;

int main()
{
    set<int, less<int> > s;
    set<int, less<int> >::iterator i;

    s.insert(4);
    s.insert(0);
    s.insert(-9);
    s.insert(7);
    s.insert(-2);
```

```
    s.insert(4);
    s.insert(2);

    cout << "The set contains the elements: ";
    for (i=s.begin(); i!=s.end(); i++) cout << *i << ' ';
    cout << endl;
}
```

4가 두번 추가되었음에도 불구하고, 한번밖에 나오지 않는 것에 주의해라. 이는 집합이기 때문에 당연한 것이다.

17.7.2. Function Objects란 무엇인가?

C++의 멋진 특징 중 하나는 연산자의 오버로딩이다. 따라서 새로 만들어진 class에 대해 + 가 어떤 의미든지 갖도록 할 수 있다. 그런데, C++에서 오버로드 할 수 있는 연산자 중 함수 호출 연산자인 ()가 있고, 이는 class의 인스턴스가 함수와 같이 동작할 수 있도록 해줄 수 있다. 이것이 function object이다.

간단한 예제를 보자.

```
<function-object.cpp>=
#include <iostream>

using namespace std;

template<class T>
class square {
public:
    T operator()(T x) { return x*x; }
};
// 이는 *가 정의되는 어떤 T에 대해서든지 쓰일 수 있다.

int main()
{
    // function object를 만든다.
    square<double> f1;
    square<int> f2;

    // 이를 사용한다.
    cout << "5.1^2 = " << f1(5.1) << endl;
    cout << "100^2 = " << f2(100) << endl;

    // 아래의 내용은 컴파일 에러를 출력할 것이다.
    // cout << "100.1^2 = " << f2(100.1) << endl;
}
```

function object는 STL의 몇몇 부분, 특히 set과 map에서 많이 쓰인다.

function object가 필요한 경우를 생각해보자. 아래의 내용을 만족하는 **comp**라는 것을 생각해보자.

1. 만약 comp(x,y), comp(y,z)가 true이면, comp(x,z)도 역시 true이다.
2. comp(x,x)는 언제나 false이다.

어떤 x,y에 대해 comp(x,y)와 comp(y,x)가 false이면 x와 y는 같은 객체이다.

이는 숫자에서 미만관계 (<)를 나타낸다. 위에서 쓰인 less < T > function object 는 type T에 대해 < 연산자로 정의되어 있다. 즉, 다음과 같다.

```
template<class T>
struct less {
  bool operator()(T x, T y) { return x<y; }
}
```

(진짜 정의는 레퍼런스를 사용하고, 적절한 const 선언을 사용하며 binary_function template class를 상속받는다.)

이는 만약 T가 < 연산자를 그 타입에 대해 정의해놓았다면, T 타입의 집합을 선언할 때, 비교를 위한 것으로 less < T > 를 사용할 수 있다는 것이다. 만약 < 연산자가 하고자 하는 것과 맞지 않을 수도 있다. 이럴 때는 다른 예가 있다. 이는 < 연산자를 이용하여 간단한 class를 만들고, 다른 방식의 비교를 하는 function object를 만든다. 오버로딩 된 <와 () 연산자가 STL과 잘 돌아가기 위해서는 const 를 적당히 써줘야 한다는 것에 주의하라.

```
<set-construct2.cpp>=
#include <iostream>
#include <set>

using namespace std;

// 이 class는 두 개의 멤버 변수를 갖는다.
// 오버로딩된 <은 멤버 f1값을 갖고 두 class를 비교한다.
class myClass {
private:
    int f1;
    char f2;
public:
    myClass(int a, char b) : f1(a), f2(b) {}
    int field1() const { return f1; }
    char field2() const { return f2; }
    bool operator<(myClass y) const
    { return (f1<y.field1()); }
};

// 이 function object는 멤버 f2의 값을 기초로
// myClass 타입의 객체들을 비교한다.
```

```
class comp_myClass {
public:
    bool operator()(myClass c1, myClass c2) const
    { return (c1.field2() < c2.field2()); }
};

int main()
{
    set<myClass, less<myClass> > s1;
    set<myClass, less<myClass> >::iterator i;
    set<myClass, comp_myClass> s2;
    set<myClass, comp_myClass>::iterator j;

    s1.insert(myClass(1,'a'));
    s2.insert(myClass(1,'a'));
    s1.insert(myClass(1,'b'));
    s2.insert(myClass(1,'b'));
    s1.insert(myClass(2,'a'));
    s2.insert(myClass(2,'a'));

    cout << "Set s1 contains: ";
    for (i=s1.begin(); i!=s1.end(); i++)
    {
        cout << "(" << (*i).field1() << ","
                << (*i).field2() << ")" << ' ';
    }
    cout << endl;

    cout << "Set s2 contains: ";
    for (j=s2.begin(); j!=s2.end(); j++)
    {
        cout << "(" << (*j).field1() << ","
                << (*j).field2() << ")" << ' ';
    }
    cout << endl;
}
```

(1,a)와 (2,a)를 가진 집합 s1은 f1을 기준으로 비교를 한다. 따라서 (1,a)와 (1,b)는 같은 원소로 취급된다. (1,a)와 (1,b)를 가진 집합 s2는 f2를 기준으로 비교를 하기 때문에 (1,a)와 (2,a)가 같은 원소로 취급된다.

17.7.3. 출력하기

위의 예에서 집합의 내용을 출력하는 것은 별로 좋지 않다. 아래의 헤더파일은 **operator<<**을 오버로딩하는 간단한 표현을 갖고 있다. 이는 간단한 원소 타입을 갖는 작은 집합에서는 잘 동작한다.

```
<printset.h>=
#ifndef _PRINTSET_H
#define _PRINTSET_H

#include <iostream>
#include <set>

template<class T, class Comp>
std::ostream& operator<<(std::ostream& os, const std::set<T,Comp>& s)
{
    std::set<T,Comp>::iterator iter = s.begin();
    int sz = s.size();
    int cnt = 0;

    os << "{";
    while (cnt < sz-1)
    {
        os << *iter << ",";
        iter++;
        cnt++;
    }
    if (sz != 0) os << *iter;
    os << "}";

    return os;
}
#endif
```

여기서 출력을 위해 사용한 << 용법은 집합의 원소들이 << 연산자를 사용할 수 있도록 정의되어있다고 가정한 것이다. 그래서 이를 콤마(,)로 구분하고 대괄호로 둘러싸서 출력되도록 한 것이다. 이는 다음 예에서도 사용될 것이다.

17.7.4. 원소의 수 구하기

집합이 공집합인지는 empty() 메쏘드를 사용하여 알 수 있다. 집합에 몇개의 원소가 들어있는지는 size() 메쏘드를 사용하여 알 수 있다. 이들은 인자없이 불려서 각각 true 나 false 혹은 정수(int)를 리턴한다.

```
<set-size.cpp>=
#include <iostream>
#include <set>
#include "printset.h"

using namespace std;

int main()
{
```

```
    set<int, less<int> > s;

    cout << "The set s is  "
            << (s.empty() ? "empty." : "non-empty.") << endl;
    cout << "It has " << s.size() << "elements." << endl;

    cout << "Now adding some elements... " << endl;

    s.insert(1);
    s.insert(6);
    s.insert(7);
    s.insert(-7);
    s.insert(5);
    s.insert(2);
    s.insert(1);
    s.insert(6);

    cout << "The set s is now
            << (s.empty() ? "empty." : "non-empty.") << endl;
    cout << "It has " << s.size() << "elements." << endl;
    cout << "s = " << s << endl;
}
```

17.7.5. 집합이 서로 같은지 검사하기

두 집합이 서로 같은지는 == 연산자를 사용하여 검사할 수 있다. 이는 T::operator== 를 사용하여 각 원소를 차례대로 검사함으로써 이루어진다.

```
<set-equality.cpp>=
#include <iostream>
#include <set>
#include "printset.h"

using namespace std;

int main()
{
    set<int, less<int> > s1, s2 ,s3;

    for (int i=0; i<10; i++)
    {
        s1.insert(i);
        s2.insert(2*i);
        s3.insert(i);
    }

    cout << "s1 = " << s1 << endl;
```

```
    cout << "s2 = " << s2 << endl;
    cout << "s3 = " << s3 << endl;
    cout << "s1==s2 is: " << (s1==s2 ? true. : false.) << endl;
    cout << "s1==s3 is: " << (s1==s3 ? true. : false.) << endl;
}
```

또한, 두 집합을 <으로 비교하는 것도 가능하다. s1 < s2 는 s1이 사전순서로(lexicographically) s2보다 작으면 true, 그렇지 않으면 false이다.

17.7.6. 원소를 추가하거나 삭제하기

집합에 원소를 추가하는 것은 insert 메쏘드 (위에 사용한 것과 같이)를, 삭제하는 것은 erase 메쏘드를 통해 이루어진다.

타입 T의 원소들을 갖고 있는 집합의 경우, 다음과 같이 이루어진다:

- **pair < iterator, bool> insert(T& x)**. 이는 표준 insert 함수이다. 리턴값은 무시할수도 있고, 성공적으로 추가했는지를 알기 위해 사용할 수도 있다 (같은 원소가 이미 집합에 있을 경우 실패한다). 만약 추가가 성공했다면, bool 값은 true이고, iterator는 금방 추가된 원소를 가리키게 될 것이다. 만약 원소가 이미 존재하는 것이라면, bool 값은 false이고, iterator는 이미 있는 값이 동일한 원소를 가리키게 될 것이다.
- **iterator insert(iterator position, T& x)**. 이 insert 함수는 인자로서 추가하고자 하는 원소 외에 iterator를 받는데, 이는 추가할 위치를 찾기 시작할 iterator이다. 리턴되는 iterator는 위와 마찬가지로 새로 추가된 원소나 이미 존재하는 같은 값의 원소이다.
- **int erase(T& x)**. 이 erase함수는 지우고자 하는 원소를 인자로 받아 만약 그 원소가 존재하면 지우고서 1을 리턴하고, 없으면 0을 리턴한다.
- **void erase(iterator position)**. 이 erase함수는 특정 원소를 가리키는 iterator를 인자로 받아 그 원소를 지운다.
- **void erase(iterator first, iterator last)**. 이 erase함수는 두 iterator를 인자로 받아 [first,last] 범위의 모든 원소를 지운다.

아래의 예는 위 함수들의 사용법을 보여준다.

```
<set-add-delete.cpp>=
#include <iostream>
#include <set>
#include "printset.h"

using namespace std;

int main()
{
    set<int, less<int> > s1;
```

```
// 표준적인 방식으로 원소를 추가한다.
s1.insert(1);
s1.insert(2);
s1.insert(-2);

// 특정위치에 원소 삽입
s1.insert(s1.end(), 3);
s1.insert(s1.begin(), -3);
s1.insert((s1.begin()++)++, 0);

cout << "s1 = " << s1 << endl;

// 성공적으로 추가되었는지 체크
pair<set<int, less<int> >::iterator,bool> x = s1.insert(4);
cout << "Insertion of 4 " << (x.second ? worked. : failed.)
        << endl;
x = s1.insert(0);
cout << "Insertion of 0 " << (x.second ? worked. : failed.)
        << endl;

// insert에서 리턴된 iterator를 두번째 형태의 insert의 인자로
// 사용할 수 있다.
cout << "Inserting 10, 8 and 7." << endl;
s1.insert(10);
x=s1.insert(7);
s1.insert(x.first, 8);

cout << "s1 = " << s1 << endl;

// 몇 원소들을 지운다.
cout << "Removal of 0 " << (s1.erase(0) ? worked. : failed.)
        << endl;
cout << "Removal of 5 " << (s1.erase(5) ? worked. : failed.)
        << endl;

// 원소를 찾아서, 지운다. (find 함수는 다음 장을 참조)
cout << "Searching for 7." << endl;
set<int,less<int> >::iterator e = s1.find(7);
cout << "Removing 7." << endl;
s1.erase(e);

cout << "s1 = " << s1 << endl;

// 마지막으로 모든 원소를 지운다.
cout << "Removing all elements from s1." << endl;
s1.erase(s1.begin(), s1.end());
cout << "s1 = " << s1 << endl;
cout << "s1 is now " << (s1.empty() ? empty. : non-empty.)
        << endl;
```

```
}
```

17.7.7. 원소를 찾기

어떤 원소가 집합에 있는지 체크해주는 두가지 함수가 있다.

- **iterator find(T& x)**. 이 함수는 집합에 원소 x가 존재하는지 찾는다. 만약 찾으면 이를 가리키는 iterator를 리턴하고, 없으면 end()를 리턴한다.
- **int count(T& x)**. 이함수는 존재하면 1을, 없으면 0을 리턴한다. (multiset에서의 count함수는 같은 원소가 여러개 있을 수 있으므로 리턴값이 1보다 더 클 수도 있다. count라는 말의 뜻을 생각해보라!)

find의 사용법은 이미 위에 보인 적이 있다. 우리는 count를 이용하여 간단한 template기반의 집합에 속하는지 test하는 함수를 만들 수 있다. (이는 인자 x에 대한 레퍼런스를 제공한다.)

```
<setmember.h>=
#ifndef _SETMEMBER_H
#define _SETMEMBER_H
#include <set>

template<class T, class Comp>
bool member(T x, std::set<T,Comp>& s)
{
 return (s.count(x)==1 ? true : false);
}
#endif

// 이는 다음과 같이 쓰일 수 있다.

<set-membership.cpp>=
#include <iostream>
#include <set>
#include "printset.h"
#include "setmember.h"

using namespace std;

int main()
{
    set<int, less<int> > s;
    for (int i= 0; i<10; i++) s.insert(i);
    cout << "s = " << s << endl;
    cout << "1 is " << (member(1,s) ?  : not) << " a member of s "
            <<  endl;
    cout << "10 is " << (member(10,s) ?  : not) << " a member of s "
```

```
            <<  endl;
}
```

17.7.8. 집합 연산

STL은 부분집합, 합집합, 교집합, 차집합, 대칭차집합(XOR) 등의 집합연산을 generic 알고리즘으로 제공한다. 이 함수들을 이용하기 위해서는 algo.h를 include 해야한다. (아래의 내용중 iter는 적절한 iterator를 의미한다).

- bool includes(iter f1,iter l1,iter f2,iter l2).

  위 함수는 [f2,l2] 범위에 있는 것들이 [f1,l1] 안의 것들을 포함하는 지를 체크한다. 만약 포함하면 true를, 그렇지 않으면 false를 리턴한다. 따라서 한 집합이 다른 집합을 포함하는 지를 보려면, 다음과 같이 하면 된다.

  includes(s1.begin(), s1.end(), s2.begin(), s2.end())

  The includes function checks the truth of 3#3 (that is of 4#4). 이 함수는 집합이 < 연산자를 이용해 정렬되었다고 본다. 만약, <이 아닌 다른 연산자 가 사용되었다면, 이(function object)를 마지막 인자로서 추가로 넘겨주면 된다.

- iter set_union(iter f1,iter l1,iter f2,iter l2,iter result).

  이는 [f1,l1]과 [f2,l2] 범위에 있는 집합들의 합집합을 만든다. 인자로 주는 result 값은 새로 만들어진 합집합의 첫 인자를 가리키는 iterator이다. 리턴값은 새로운 집합의 끝(end)를 가리키는 iterator이다.

result 인자가 iterator란 말은, 다음과 같은 식으로 set_union을 사용하면 안된다는 것이다.

```
set<int, less<int> > s1, s2, s3;
// s1 과 s2의 원소를 가지고 합집합을 만든다.
// (그러나 이런 식으로는 동작하지 않음)
set_union(s1.begin(), s1.end(),
          s2.begin(), s2.end(),
          s3.begin());
```

그 이유는 begin()과 end()가 집합이나 맵에 사용될 때는 상수 input iterator가 되기 때문이다. 이러한 iterator는 집합의 원소를 읽기 위해서는 사용될 수 있지만, 값을 쓸 수는 없다. (또한 만약 값을 쓸 수 있게 한다면 집합의 순서를 망가뜨릴 수 있는 위험이 있기 때문이기도 하다)

해결책은 set_type의 insert iterator를 사용하는 것이다. 이는 (*i)=value 같은 불가능한 구문을 s.insert(i,value)의 형태로 쓸 수 있게 해준다. (여기서 s는 iterator i가 가리키는 집합이다. 이는 다음과 같이 쓰인다.

```
// 편의를 위해 Typedef를 사용
typedef set<int, less<int> > intSet;
```

```
intSet s1, s2, s3;
// s1과 s2에 몇 원소를 추가.
// 그리고 합집합을 구한다.
set_union(s1.begin(), s1.end(),
          s2.begin(), s2.end(),
          insert_iterator<intSet>(s3,s3.begin()) );
```

이제 위에 나오는 것들을 종합적으로 사용하는 예제를 보자.

```
<set-theory.cpp>=
#include <iostream>
#include <set>
#include <algorithm>
#include <iterator>
#include "printset.h"

using namespace std;

int main()
{
    typedef set<int, less<int> > intSet;

    intSet s1, s2, s3, s4;

    for (int i=0; i<10; i++)
    { s1.insert(i);
        s2.insert(i+4);
    }
    for (int i=0; i<5; i++) s3.insert(i);

    cout << "s1 = " << s1 << endl;
    cout << "s2 = " << s2 << endl;
    cout << "s3 = " << s3 << endl;

    // s1이 s2의 부분집합인가?
    bool test = includes(s2.begin(),s2.end(),s1.begin(),s1.end());
    cout << "s1 subset of s2 is " << (test ? true. : false.) << endl;

    // s3가 s1의 부분집합인가?
    test = includes(s1.begin(),s1.end(),s3.begin(),s3.end());
    cout << "s3 subset of s1 is " << (test ? true. : false.) << endl;

    // s1과 s2의 합집합.
    set_union(s1.begin(), s1.end(), s2.begin(), s2.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s1 union s2 = " << s4 << endl;

    // s4를 지우고, s1과 s2의 교집합을 구한다.
```

```
    // ( 만약 s4를 지우지 않으면 원래 s4에 들어있는 것들도
    // 같이 들어가게 될 것이다. )
    s4.erase(s4.begin(),s4.end());
    set_intersection(s1.begin(), s1.end(), s2.begin(), s2.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s1 intersection s2 = " << s4 << endl;

    // 차집합
    s4.erase(s4.begin(),s4.end());
    set_difference(s1.begin(), s1.end(), s2.begin(), s2.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s1 minus s2 = " << s4 << endl;

    // 차집합은 대칭적이지 않다. (즉, A-B != B-A)
    s4.erase(s4.begin(),s4.end());
    set_difference(s2.begin(), s2.end(), s1.begin(), s1.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s2 minus s1 = " << s4 << endl;

    // 대칭차집합
    s4.erase(s4.begin(),s4.end());
    set_symmetric_difference(s1.begin(), s1.end(), s2.begin(), s2.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s1 symmetric_difference  s2 = " << s4 << endl;

    // 대칭차집합은 대칭적이다. (즉, commutative)
    s4.erase(s4.begin(),s4.end());
    set_symmetric_difference(s2.begin(), s2.end(), s1.begin(), s1.end(),
            insert_iterator<intSet>(s4,s4.begin()) );
    cout << "s2 symmetric_difference  s1 = " << s4 << endl;
}
```

17.8. 맵

17절장을 보아라.

17.9. STL 알고리즘

17절장을 보아라.

18. C++에서의 쓰레드

- IBM pthread 가이드, 개념, API 레퍼런스 http://www.as400.ibm.com/developer/threads/uguide/document.htm 미러사이트 : IBM main site (http://java.icmc.sc.usp.br/library/books/ibm_pthreads/document.htm)
- C++의 QpThread 라이브러리는 system level 쓰레드(POSIX thread) 위에서 쓰레드와 Unix signal을 위한 C++ 객체지향구조를 제공한다. http://lin.fsid.cvut.cz/~kra/index.html
- ThreadJack 은 플랫폼에 상관없는 C++ class library로서, Java와 유사한 멀티쓰레드 프로그래밍 모델을 제공한다. http://www.esm.co.jp/divisions/open-sys/ThreadJack/index-e.html 그리고 다음의 주소에서 받을 수 있다. http://www.esm.co.jp/divisions/open-sys/ThreadJack/source-dl-e.html
- APE는 "APE Portable Environment"의 약자로, UNIX (pthread)와 Win32 API를 기반으로 어디서나 쓸 수 있는 쓰레드 기반 서버를 제공하는 C++ class 라이브러리이다. APE는 쓰레드, 소켓, 파일제어, 동기화 도구 등의 추상 class 를 제공한다. APE의 목적은 C++을 이용하여 작고 간단한 프로젝트에서도 실제적이면서도 편리한 쓰레드기반 서버를 만드는 것이다. 따라서 단순함과 낮은 오버헤드가 중요하다. http://www.voxilla.org/projects/projape.html
- Portabale Thread Lib http://www.media.osaka-cu.ac.jp/~k-abe/PTL
- Thread-Recyling in C++ http://www.sigs.de/html/kuhlmann.html

18.1. 쓰레드 튜토리얼

- 튜토리얼 전체를 하나의 파일로 아래에서 받을 수 있다. http://www.milkywaygalaxy.freeservers.com "Source code c++ Programming howto"를 클릭하라.
- 쓰레드 튜토리얼은 http://www.math.arizona.edu/swig/pthreads/threads.html에 있다.
- HERT 튜토리얼은 http://www.hert.org/docs/tutorials에 있고, 쓰레드에 대해 찾아보려면 여기로 가라.
- linuxjournal (http://www2.linuxjournal.com/lj-issues/issue61/3138.html)의 쓰레드 소개글.
- North Arizona Univ NAU (http://www.cse.nau.edu/~mc8/Thread/Contents.html)
- POSIX threads Acctcom multi-thread (http://users.actcom.co.il/~choo/lupg/tutorials/multi-thread/multi-thread.html)

18.2. C++에서 쓰레드 class 디자인하기

이 장은 Ryan Teixeira (mailto:ryte@geocities.com) 에 의해 쓰여졌고, 그 문서는 여기 (http://www.geocities.com/SiliconValley/Heights/6038/dthreads.html)에 있다. .

18.2.1. 소개

멀티 쓰레드 프로그래밍은 점점 인기를 얻고있다. 이 장은 쓰레드를 지원하는 C++ class의 디자인을 보여줄 것이다. mutex나 세마포어같은 쓰레드 프로그래밍의 몇몇 측면은 여기서 논의되지 않는다. 또한 쓰레드의 관리를 위한 시스템 콜들은 일반적인 형태로 나타내었다.

18.2.2. 쓰레드에 대한 간단한 소개

쓰레드를 이해하기 위해서는, 한꺼번에 돌아가는 여러 프로그램을 생각해야한다. 또한, 이 프로그램들이 똑같은 전역변수와 함수들에 접근한다고 생각해보아라. 이 프로그램들은 실에 비유될 수 있고, 그래서 쓰레드라고 불린다. 중요한 차이점이 있다면, 각각의 쓰레드는 다른 쓰레드가 진행하는 것을 기다릴 필요가 없다는 것이다. 모든 쓰레드가 동시에 진행된다. 비유를 하자면, 이들은 육상선수와 같이 아무도 다른 선수를 기다리지 않는다. 각자 자신의 속도로 진행되는 것이다.

왜 쓰레드를 사용하냐고 물어본다면, 쓰레드는 종종 어플리케이션의 성능을 향상시킬 수 있고, 구현하는게 까다롭지 않다. 즉, 조그만 투자로 큰 효과를 볼 수 있는 것이다. 이미지를 서비스하는 이미지 서버 프로그램을 생각해보아라. 이 프로그램은 다른 프로그램으로부터 이미지에 대한 요청을 받는다. 그러면 이 이미지를 데이터베이스에서 찾아 요청을 보낸 프로그램에게 다시 보내준다. 만약 서버가 하나의 쓰레드로 만들어졌다면, 한번에 하나의 프로그램만 요청을 보낼 수 있을 것이다. 만약 프로그램이 이미지를 찾거나 보내주는 중이라면 다른 요청을 처리할 수 없을 것이다. 물론 이러한 시스템을 쓰레드를 이용하지 않고도 만들 수 있지만, 쓰레드를 쓰면, 여러개의 요청을 아주 자연스럽게 처리할 수 있게 된다. 간단한 접근 방법은 하나의 요청당 하나의 쓰레드를 만드는 것이다. 메인 쓰레드는 요청에 따라 쓰레드를 만들어 주기만 하면 된다. 그러면 새로 만들어진 쓰레드가 요청하는 프로그램과 대화하면서 서비스를 해주면 된다. 이미지를 찾아서 보낸 후에는 쓰레드가 스스로 종료하면 된다. 이렇게 하면 하나의 요청을 서비스 하는 도중에도 다른 요청을 받을 수 있는 유연한 시스템이 될 것이다.

18.2.3. 기본적인 접근방법

쓰레드를 만들기 위해서는, 쓰레드의 시작점이 될 함수를 명시해야 한다. 운영체제 레벨에서는, 이것이 일반적인 함수이다. 그런데 C++의 class 멤버함수는 시작함수가 될 수 없기 때문에 약간의 트릭을 써야한다. 하지만, 클래스의 static 멤버함수는 가능하다. 이것이 우리가 시작함수로 이용할 것이다. static 멤버함수는 C++ 객체의 this 포인터를 사용할 수 없다. 이들은 오직 static 데이터만 접근할 수 있다. 다행히도 방법이 있다. 쓰레드의 시작점 함수는 인자로 void *를 갖게 되는데, 이를 쓰레드 안에서 어떤 타입으로나 casting 해서 쓸 수 있다. 우리는 이를 static 함수에 this 를 넘겨주기 위해 쓸 것이다. 따라서 static 함수는 이를 casting 하여 static이 아닌 함수를 부르기 위해 쓸 수 있다.

18.2.4. 구현

우리는 약간 제한된 기능을 갖는 쓰레드 class를 만들 것이다. 실제 쓰레드는 이 class가 하는 것보다 훨씬 많은 일들을 할 수 있다.

```
class Thread
{
   public:
      Thread();
      int Start(void * arg);
   protected:
      int Run(void * arg);
      static void * EntryPoint(void*);
      virtual void Setup();
      virtual void Execute(void*);
      void * Arg() const {return Arg_;}
      void Arg(void* a){Arg_ = a;}
   private:
      THREADID ThreadId_;
      void * Arg_;

};

Thread::Thread() {}

int Thread::Start(void * arg)
{
   Arg(arg); // user 데이터를 저장함.
   int code = thread_create(Thread::EntryPoint, this, & ThreadId_);
   return code;
}

int Thread::Run(void * arg)
{
   Setup();
   Execute( arg );
}

/*static */
void * Thread::EntryPoint(void * pthis)
{
   Thread * pt = (Thread*)pthis;
   pthis->Run( Arg() );
}

virtual void Thread::Setup()
{
        // Setup에 해당하는 일들
}
```

```
virtual void Thread::Execute(void* arg)
{
        // 실행할 내용
}
```

우리가 쓰레드를 C++ 객체로 사용하고자 한다는 것을 이해하는 것이 중요하다. 각각의 객체는 하나의 쓰레드에 대한 인터페이스를 제공한다. 쓰레드와 객체는 다르다. 객체는 쓰레드 없이 존재할 수 있다. 이 구현에서, 쓰레드 자체는 Start 함수가 불릴 때까지 존재하지 않는다.

여기서 user의 인자를 class에 저장한다는데 주의해라. 이는 쓰레드가 시작될 때까지 임시로 이를 저장할 공간이 필요하기 때문이다. 운영체제 쓰레드는 인자를 하나 넘길 수 있게 해주지만, 우리는 this 때문에 이를 직접 넘겨줄 수 없다. 그래서 우리는 인자를 잠시 class에 저장했다가 함수가 시작될 때 다시 꺼내서 넘겨주게 된다.

Thread(); 생성자이다.

int Start(void * arg); 이 함수는 쓰레드를 만들고, 이를 시작하게 해준다. 이 인자는 쓰레드에 데이터를 넘겨주기 위해 사용되고, Start()는 운영체제의 쓰레드 생성 함수를 부름으로써 쓰레드를 만든다.

int Run(void * arg); 이 함수는 건드리면 안되는 함수이다.

static void * EntryPoint(void * pthis); 이 함수는 쓰레드의 시작 점 역할을 한다. 이 함수는 단순히 pthis를 Thread *로 casting해서 Run 함수를 불러준다.

virtual void Setup(); 이 함수는 쓰레드가 만들어진 후, 실행이 시작되기 전에 불려진다. 이 함수를 override 할 때는, 부모 class의 Setup()를 부르는 것을 기억하라.

virtual void Execute(void *); 하고자 하는 일을 위해 이 함수를 override해라.

18.2.5. Thread Class 사용하기

thread class를 사용하기 위해서는, 새로운 class를 만들어야 한다. 그리고 만들고자 하는 기능을 위해 Execute()를 override하면 된다. 또한, Execute가 불리기 전의 초기화를 위해 Setup()을 override할 수도 있다. 만약, Setup()을 override한다면, 부모 class의 Setup()을 부르는 것을 기억하라.

18.2.6. 결론

이 장은 C++로 쓰레드 class의 구현을 살펴보았다. 물론 이것은 단순한 접근방법이고, 더 좋은 디자인을 위한 기초로 쓰일 수 있을 것이다.

만약 코멘트나 제안하고 싶은 것이 있으면, 메일을 써주기 바란다. Ryan Teixeira (mailto:ryte@geocities.com)

19. C++ 유틸리티들

C++ 유틸리티를 위한 다음 사이트를 방문해보라.

- Portable C++ utilities from http://www.boost.org. Boost 웹사이트는 이식 가능한 C++ 소스 라이브러리를 제공한다. 중요한 점은 이 라이브러리가 C++ 표준 라이브러리와 함께 잘 동작한다는 것이다. 이 목적은 "쓰일 수 있는 실제적인 것"을 만들자는 것으로, 실제 구현에 참고사항이될 수 있다. 따라서 Boost 라이브러리는 최종 표준화에 적절할 것이다.
- The smart pointer library (http://www.boost.org/libs/smart_ptr/index.htm) 는 다섯가지 smart pointer class template를 제공한다. smart pointer는 C++의 new 표현으로 할당되는 동적 메모리의 관리를 쉽게 해준다. 추가로 scoped_ptr는 다른 방법들로 할당되는 동적 메모리 관리를 쉽게 해준다.
- C++ Binary File I/O http://www.angelfire.com/country/aldev0/cpphowto/cpp_BinaryFileIO.html
- Portability Guide http://www.angelfire.com/country/aldev0/cpphowto/cpp_PortabilityGuide.html
- Snippets collections of C++ routines http://www.angelfire.com/country/aldev0/cpphowto/cpp_Snippets.html 와 snippets site (http://www.strangecreations.com/library/snippets)
- escape ISB for C++ - 프로그램을 어떻게 개발하고, 분산처리, 객체기반 어플리케이션을 Netscape Internet Service Broker를 사용하는 윈도우와 유닉스에서 어떻게 개발하는지에 대한 정보를 제공한다. http://docs.iplanet.com/docs/manuals/enterprise/cpluspg/contents.htm
- Common C++ http://www.voxilla.org/projects/projape.html
- Large List of free C++ libs http://www.thefreecountry.com/developercity/freelib.html
- C++ Tools http://development.freeservers.com
- C++ Tools CUJ http://www.cuj.com/code
- C++libs Univ of vaasa http://garbo.uwasa.fi/pc/c-lang.html

19.1. 메모리 툴

아래의 메모리 디버깅 툴들을 사용하라.

- **"MPatrol"** 은 강력한 메모리 디버깅 툴이다. 웹사이트 : http://www.cbmamiga.demon.co.uk/mpatrol , 그리고 http://www.rpmfind.net 에 가서 'mpatrol' 로 찾아보아라. 만약, 리눅스를 사용하고 있다면, mpatrol*.src.rpm 파일을 rpmfind.net에서 다운로드 받아라. mpatrol*.src.rpm을 최신버전으로 업데이트 하기 위해서는, 옛날 mpatrol.spec 파일을 사용하여 최신의 mpatrol*.tar.gz을 새로운 소스패키지(SRPM)으로 만들 수 있다.

- 리눅스 배포판 씨디롬에서 mem_test*.rpm 패키지를 보아라. 그리고 http://www.rpmfind.net 으로 가서 mem_test 로 찾아보아라.
- 리눅스 배포판 씨디롬에서 ElectricFence*.rpm 패키지를 보아라. 그리고 http://www.rpmfind.net 으로 가서 electricfence로 찾아보아라.
- Purify Tool from Rational Software Corp http://www.rational.com
- Insure++ Tool from Parasoft Corp http://www.parasoft.com
- Linux Tools : http://www.xnet.com/~blatura/linapp6.html#tools
- Google, Yahoo, Lycos, Excite, Mamma.com 같은 서치엔진으로 가서 "Linux memory debugging tools"라는 키워드로 찾아보아라.

20. 이 문서의 다른 포맷

이 문서는 14가지 포맷으로 배포된다. - DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU-info, HTML, RTF(Rich Text Format), Plain-text, Unix man pages, 하나의 HTML파일, SGML (linuxdoc format), SGML (Docbook format), MS WinHelp 포맷.

이 howto 문서는

- http://www.linuxdoc.org 에서 HOWTOs를 누르고, 웹브라우저에서 CTRL+f 나 ALT+f를 이용해 howto 이름을 넣어서 찾을 수 있다.

또한, 다음 미러사이트에서도 찾을 수 있다 -

- http://www.caldera.com/LDP/HOWTO
- http://www.linux.ucla.edu/LDP
- http://www.cc.gatech.edu/linux/LDP
- http://www.redhat.com/mirrors/LDP
- 또는 다음 페이지에서 가까운 미러사이트를 찾아볼 수도 있다. http://www.linuxdoc.org/mirrors.html site를 하나 골라서 /LDP/HOWTO/xxxx-HOWTO.html을 찾아가면 된다.

- 이 문서의 HTML, DVI, Postscript 혹은 SGML 파일의 tar-ball을 다음 주소에서 받을 수 있다. ftp://www.linuxdoc.org/pub/linux/docs/HOWTO/other-formats/ 와 http://www.linuxdoc.org/docs.html#howto
- Plain text 포맷은 : ftp://www.linuxdoc.org/pub/linux/docs/HOWTO 와 http://www.linuxdoc.org/docs.html#howto

- 하나의 HTML 파일은 : http://www.linuxdoc.org/docs.html#howto 하나의 HTML 파일은 다음의 명령으로 만들어질 수 있다. (sgml2html의 man page를 보라) sgml2html -split 0 xxxxhowto.sgml
- 프랑스어, 독일어, 스페인어, 중국어, 일본어 등의 번역은 ftp://www.linuxdoc.org/pub/linux/docs/HOWTO 와 http://www.linuxdoc.org/docs.html#howto 에서 찾을 수 있다. 또 다른 언어로의 번역또한 환영한다.

이 문서는http://www.sgmltools.org에서 찾을 수 있는 "SGML-Tools"로 쓰여졌다. 소스를 컴파일하려면 다음과 같이 하면 된다.

- sgml2html xxxxhowto.sgml (html 만들기)
- sgml2html -split 0 xxxxhowto.sgml (하나의 html file로 만들기)
- sgml2rtf xxxxhowto.sgml (RTF file만들기)
- sgml2latex xxxxhowto.sgml (latex file만들기)

20.1. Acrobat PDF 포맷

PDF 파일은 acrobat의 **distill**이나 **Ghostscript**를 사용해서 Postscript 파일로부터 만들 수 있다. 그리고 Postscript 파일은 LaTex 파일로부터 만들어지는 DVI파일로 만들 수 있다. distill 소프트웨어는 http://www.adobe.com. 에서 받을 수 있다. 아래의 예제를 보라.

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
bash$ man dvips
bash$ dvips -o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

혹은 **ps2pdf** 같은 Ghostsciprt 명령어를 사용할 수도 있다. ps2pdf는 Adobe의 Acrobat Distiller와 거의 같은 역할을 한다 (즉, PostScript 파일을 PDF로 바꾼다) **ps2pdf** 는 Ghostscript를 실행시키고, **pdfwrite**라는 특별한 "출력장치"를 사용하는 작은 스크립트이다. ps2pdf를 사용하기 위해서는 pdfwrite 장치가 Ghostscript 컴파일 시에 Makefile에 포함되어있어야 한다. 자세한 내용은 해당 문서를 보도록 해라.

20.2. linuxdoc 을 Docbook 포맷으로 바꾸기

이 문서는 linuxdoc SGML 포맷으로 쓰여졌다. Docbook SGML 포맷은 linuxdoc 포맷을 포함하고, 더 많은 기능을 갖고 있다. linuxdoc은 간단하고 사용하기 쉽다. linuxdoc SGML파일을 Docbook SGML로 바꾸려면 **ld2db.sh** 프로그램과 몇몇 perl 스크립트를 사용해라. ld2db에서

얻어지는 것은 100% 깨끗한 것이 아니고, **clean_ld2db.pl** perl 스크립트를 사용할 필요성이 생긴다. 또한, 직접 문서에서 몇몇을 고쳐야할 수도 있다.

- ld2db 프로그램은 http://www.dcs.gla.ac.uk/~rrt/docbook.html 이나 Milkyway Galaxy site (http://www.milkywaygalaxy.freeservers.com)에서 받으면 된다.
- cleanup_ld2db.pl perl script 는 Milkyway Galaxy site (http://www.milkywaygalaxy.freeservers.com)에서 받으면 된다.

ld2db.sh 는 100% 깨끗하지 않아서, 다음 과같이 실행하려면 에러가 날 것이다.

```
bash$ ld2db.sh file-linuxdoc.sgml db.sgml
bash$ cleanup.pl db.sgml > db_clean.sgml
bash$ gvim db_clean.sgml
bash$ docbook2html db.sgml
```

또한, perl 스크립트를 돌린 후에 몇몇 작은 에러를 고치고 싶기도 할 것이다. 예를들어, </Para>를 매 <Listitem>마다 넣을 필요가 있을 수도 있다.

20.3. MS WinHelp 포맷으로 바꾸기

SGML Howto를 Microsoft Windows Help 파일로 바꿀 수도 있다. 먼저, sgml을 다음과 같이 html로 바꿔라.

```
bash$ sgml2html xxxxhowto.sgml     (html 파일 만들기)
bash$ sgml2html -split 0   xxxxhowto.sgml (하나의 html 파일로 만들기)
```

그리고 HtmlToHlp (http://Javadocs.planetmirror.com/htmltohlpe.html)이라는 툴을 쓰면 된다. 또한, winhelp 파일을 만들기 위해 sgml2rtf를 쓴다음, RTF파일을 쓸 수도 있다.

20.4. 여러가지 포맷의 문서를 읽기

dvi 포맷의 문서를 보기 위해서는, xdvi 프로그램을 사용해라. xdvi 프로그램은 Redhat 리눅스의 경우 ControlPanel | Applications | Publishing | Tex menu 버튼에 있고, tetex-xdvi*.rpm 패키지에 들어있다. dvi 문서를 읽으려면 다음과 같은 명령을 쓰면 된다.

```
xdvi -geometry 80x90 howto.dvi
man xdvi
```

그리고 윈도우 크기를 마우스로 조정 한다. 이리저리 살표보기 위해서는 화살표키나 PageUp, PageDown키, 그리고 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' 키 등을 위, 아래 중앙으로 움직이거나 다음페이지, 이전페이지 등으로 넘기기 위해 쓸 수 있다. expert 메뉴를 끄기 위해서는 'x'를 누르면 된다.

Postscript 파일을 읽기 위해서는 'gv'(ghostview)프로그램이나 'ghostscript'를 쓰면 된다. ghostscript 프로그램은 ghostscript*.rpm 패키지에 있고, gv 프로그램은 gv*.rpm 패키지에 들어

있다. 이들은 ControlPanel | Applications | Graphics menu 버튼에 있다. gv 프로그램이 ghostscript보다 훨씬 사용하기 편하다. ghostscript와 gv 는 윈도우나 OS/2 등 다른 플랫폼에서도 사용가능하다.

- 윈도우, OS/2등 모든 OS를 위한 ghostsciprt는 다음 사이트에 있다. http://www.cs.wisc.edu/~ghost

Postscript 문서를 읽기 위해서는 다음과 같이 하면 된다 -

```
gv howto.ps
ghostscript howto.ps
```

HTML 포멧 문서는 Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser 나 다른 웹브라우저로 읽으면 된다.

latex, LyX 는 latex의 X-Window 프론트엔드인 LyX 로 읽으면 된다.

21. 다른 언어로의 번역

- 폴란드어로의 번역은 http://strony.wp.pl/wp/chq/c/howto/book1.htm에 있다. (Darek Ostolski 에게 감사~ : Darek Ostolski (mailto:chq[at]wp.pl))
- 프랑스어나 독일어, 스페인어, 중국어, 일본어 등으로의 번역은 ftp://www.linuxdoc.org/pub/linux/docs/HOWTO 나 http://www.linuxdoc.org/docs.html#howto에서 찾으면 된다.

다른 언어로의 번역은 환영한다.

22. Copyright

Copyright는 LDP(리눅스 문서화 프로젝트)에 따라 GNU/GPL이다. LDP는 GNU/GPL 프로젝트 이다. 부가적인 요구사항은 저자의 이름과 이메일주소, 이 Copyright 를 언제나 포함시켜달라는 것이다. 만약 변경이나 추가할 사항이 있으면 이 문서의 저자들에게 알려주기 바란다. 이 문서에 언급된 회사이름이나 상표는 각각 그 소유주의 것이다. Copyright policy is GNU/GPL as per LDP (Linux Documentation project). LDP is a GNU/GPL project. Additional requests are that you retain the author's name, email address and this copyright notice on all the copies. If you make any changes or additions to this document then you please intimate all the authors of this document. Brand names mentioned in this document are property of their respective owners.

23. 부록 A String 프로그램 파일

모든 프로그램 파일을 하나의 **tar.gz** 으로 4절에서 받을 수 있다. 그리고 압축을 다음과 같이 풀면 된다.

```
bash$ man tar
bash$ tar ztvf C++Programming-HOWTO.tar.gz
이는 압축된 파일들을 보여줄 것이다.

bash$ tar zxvf C++Programming-HOWTO.tar.gz
이는 압축을 실제로 풀 것이다.
```

- 헤더파일을 먼저 읽고, 예제 cpp파일을 보아라.
  - String.h http://www.angelfire.com/country/aldev0/cpphowto/String.h
  - StringBuffer.h http://www.angelfire.com/country/aldev0/cpphowto/StringBuffer.h
  - StringTokenizer.h http://www.angelfire.com/country/aldev0/cpphowto/StringTokenizer.h
  - StringRW.h http://www.angelfire.com/country/aldev0/cpphowto/StringRW.h
  - string_multi.h http://www.angelfire.com/country/aldev0/cpphowto/string_multi.h
  - example_String.cpp example_String.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'을 클릭해라.
- 파일 관리 class. length() 함수만 구현되었다.
  - File.h http://www.angelfire.com/country/aldev0/cpphowto/File.h
  - File.cpp File.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭하라.
- zap() 함수의 구현은 아래 링크에 있다.
  - my_malloc.h http://www.angelfire.com/country/aldev0/cpphowto/my_malloc.h
  - my_malloc.cpp my_malloc.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.
- String class의 구현.
  - String.cpp String.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.
  - StringTokenizer.cpp StringTokenizer.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.
  - StringBuffer.cpp StringBuffer.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.

- StringRW.cpp StringRW.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.

- 디버깅 도구들 ..
  - debug.h http://www.angelfire.com/country/aldev0/cpphowto/debug.h
  - debug.cpp debug.cpp (http://www.milkywaygalaxy.freeservers.com) 'Source code of C++ howto'를 클릭해라.
  - Makefile.unx http://www.angelfire.com/country/aldev0/cpphowto/Makefile.unx

- String class의 작동을 테스트하기 위한 예시용 자바 파일.
  - string.java http://www.angelfire.com/country/aldev0/cpphowto/string.java