C++ Programming HOW-TO

Al Dev (Alavoor Vasudevan) alavoor@yahoo.com v5.0, 4월 9일 2000

김 지희, 윤 주철 2000년 6월 15일

이 문서는 C++의 메모리 문제를 피하는 방법에 대해 다루며 C++언어를 올바르게 프로그램 할수 있도록 도와 주고자 한다. 이 문서의 정보는 모든 운영체제 - 리눅스, MS DOS, 애플 맥킨토시, 윈도우 95/NT, OS/2, IBM 운영체제들, VMS, Novell Netware, 유닉스의 모든 종류 (솔라리스, HPUX, AIX, SCO, Sinix, BSC등..), 그리고 "C++" 컴파일러를 지원하는 모든 다른 운영체제들 (다시 말해 이 지구상에 존재하는 거의 모든 운영체제들이라고 할 수 있다)- 에 적용될수 있다.

목 차

제 1 절 서론 **2**

1.1 현재 C++ 컴파일러들이 직면한 문제들 2

1.2 "C" 를 쓸것인가 "C++" 을 쓸것인가 아니면 자바를 쓸것인가? 3

제 2 절 다운로드 mychar **3**

제 3 절 mychar class의 사용 **3**

3.1 연산자들 4

3.2 함수들 5

3.3 기타 함수들 6

제 4 절 C++ Zap (Delete) 명령어 **7**

제 5 절 my_malloc 과 my_free의 사용 **8**

제 6 절 Debug 화일들 **9**

제 7 절 C++ Online 문서들 **9**

7.1 C++ 튜터리얼들 9

7.2 C++ 코딩 표준 10

7.3 C++ 퀵 레퍼런스 10

7.4 C++ 유즈넷 뉴스그룹들 10

제 8 절 메모리 도구들 **10**

제 9 절 관련 URL들 **10**

제 10 절 이 문서의 다른 포멧들 **11**

제 11 절 저작권 **12**

제 12 절 첨부 A example_mychar.cpp **12**

제 13 절 첨부 B mychar.h **15**

제 14 절 첨부 C mychar.cpp **18**

제 15 절 첨부 D my_malloc.cpp **33**

제 16 절 첨부 E my_malloc.h **44**

제 17 절 첨부 F debug.h **45**

제 18 절 첨부 G debug.cpp **46**

제 19 절 첨부 H Makefile **47**

제 1 절 서론

C++ 은 가장 널리 쓰이는 언어이고, 앞으론 자바의 등장과는 상관없이 오랜 기간동안 쓰일 것이다. C++ 은 매우 빠르게 실제로 자바보다 20내지 30배 더 빨리 실행된다. 자바는 "virtual engine"위에서 작동되는 번역 (interpreted)어이기때문에 매우 느리게 실행된다. 자바에서 메모리 관리는 자동이라서 프로그래머들은 메모리 얼로케이션 (allocation) 을 직접 다루지 않는다. 이 문서는 C++ 사용을 더욱 편하게 하기 위해 C++ 에서의 메모리 관리를 자동화하는 것을 시도한다. 자바의 좋은 점 메모리 얼로케이션이 자동적으로 처리된다는 것이다. 이 howto는 "C++"이 메모리 관리면에서 자바 언어와(를) "경쟁/모방"하는 것을 가능하게 할 것이다.

C++ 프로그래머들은 수동 메모리 얼로케이션때문에 많은 시간을 소비한다. 이 문서에 있는 정보는 여러분에게 디버깅하는 시간을 줄일 수 있는 더 나은 아이디어와 팁들을 제공할 것이다.

1.1 현재 C++ 컴파일러들이 직면한 문제들

C++ 은 C의 super-set 인 관계로 "C" 언어의 모든 안좋은 점들을 갖고 있다. 예를 들어, "C" 프로그래밍에서는 메모리 유출과 메모리 오버플로우가 흔하다. 왜냐면 다음과 같은 용례때문에

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

char * 와 strcpy의 사용은 "overflow" "fence past errors", 그리고 "memory leaks" 로 인한 지독한 메모리 문제들을 야기시킨다. 메모리 문제들은 디버그하기 매우 어렵고, 고치고 trouble-shoot하는데 많은 시간을 소모시킨다. 메모리 문제들은 프로그래머들의 생산성을 감소시킨다. 이 문서는 "C++"의 메모리 문제를 해소할 수 있는 여러가지 언급된 방법들을 통해 프로그래머들의 생산성을 증 시키는것을 돕는다. 메모리와 관련된 버그들은 크랙하기 매우 어렵고 심지어 숙련된 프로그래머들도 메모리와 관련된 문제점들을 디버그하는데 몇일, 몇주 혹은 몇달까지 걸리기도 한다. 여러개의 메모리 버그들은 몇개월동안 코드안에 "숨어"있을 것이고 그것들은 기대치 않았던 프로그램 충돌을 야기시킬 수 있다. char * 의 사용은 미국과 일본에서 디버깅 하는 시간과 프로그램의 실행정지로 매년 20억불의 손해를 야기시키고 있다. 여러분이 C++

에서 char * 를 사용하면, 특히 여러분의 프로그램이 백만줄 이상의 코드가 있다면 그것은 정말로 손해보는 일이다,

그래서 다음의 기술들이 "C" 언어의 약점을 극복하기 위해 다음의 기술들을 제안하는 것이다.

C++ 컴파일러들은 프로그래머들이 "char *" , "char[]" 데이터 타입과 strcpy, strcat, strncpy, strncat과 같은 함수들을 사용하는 것을 막아야한다고 제안한다. char *, char[] 와 같은 데이터타입들과 strcpy, strcat 같은 함수들은 C++의 사용에 있어서 반드시 완전히 금지되어야만 하는 해로운 것들이다!!

모든 C++ 프로그래머들은 char * and char[] 를 사용하는 대신에 이 문서에 나오는 'mychar class'와 STDLIB에 포함된 'string class' 를 반드시 사용해야 한다. 'mychar class' 는 생성자와 소멸자를 이용해서 메모리 관리를 자동화하고 ltrim, substring등등의 여러 함수들을 제공한다.

C++ 컴파일러에 있는 'string class' 도 또한 보자. string class 는 standard GNU C++ library 의 일부분이고 많은 string조작함수 제공한다. string class' 와 'mychar class' 를 쓰면 char * datatype을 쓸 필요가 없어진다. 또한, C++ 프로그래머들은 'malloc' 이나 'free'를 쓰는 대신 'new'와 'delete' 을 쓰도록 권장되어야 한다. 'mychar class' 는 char * 나 char [] 가 하는 모든 것들을 한다. 'mychar class' 는 완벽하게 char datatype을 대신해 쓰일 수 있다. 더구나 프로그래머들은 메모리 문제와 메모리 얼로케이션에 대해서 전혀 걱정할 필요가 없으니 얼마나 이로운 것인가.

GNU C++ 컴파일러는 char *, char[] datatypes 의 지원을 반드시 중지해야하며, char datatype을 이용한 구 프로그램을 컴파일하기 해서 컴파일러는 "-fchar-datatype" 라 불리는 옵션을 g++ 명령어에 더 추가로 제공해줘야 한다. 앞으로 2년안에 모든 C++ 프로그램들은 'mychar class' and 'string class' 를 쓰게 될것이고 char * and char[] 는 사라지게 될 것이다. 컴파일러는 안좋은 프로그래 습관을 방지하도록 노력해야한다.

1.2 "C" 를 쓸것인가 "C++" 을 쓸것인가 아니면 자바를 쓸것인가?

모든 응용프로그램이나 일반 목적의 프로그래밍을 위해서는 객체지향적인 언어인 "C++"을 이용해서 프로그래밍을 하는것이 추천된다. "C++"의 객체지향 기능을 충분히 유용할 수 있다. 하지만 -0이나 -03과 같은 컴파일러 optimizer옵션들은 C++의 속도를 증가시킬 수 있다.

요즘엔 "C" 언어가 운영체제, 디바이스 드라이버등을 개발하기위한 "시스템 프로그래밍"에 주로 쓰인다.

자바는 웹 브라우저내에서 실행되는 GUI을 개발하는데 더욱 적합한 플랫폼에 독립된 언어이지만 매우 느리게 실행된다. C++ and HTML, DHTML 과 더불어 Fast-CGI를 사용하는게 더 나은 성능을 발휘하기위해서는 C++ and HTML, DHTML 과 더불어 Fast-CGI를 사용하는게 바람직하다.

제 2 절 다운로드 mychar

모든 프로그램들과 예제들은 이 문서의 첨부에 있다. 여러분은 한개의 tar zip화일로 다운로드 받을 수 있다. mychar class, libraries, 그리고 예제 프로그램들은 다음에서 다운받을 수 있다.

- 여기" http://www.aldev.8m.com "가서 C++Programming howto.tar.gz file 를 클릭하시오.
- Mirror site : <http://aldev.webjump.com>

제 3 절 mychar class의 사용

'mychar class' 는 char 와 char * datatype의 완벽한 대체물이다. 여러분은 'mychar class' 를 char쓰는 것과 똑같은 쓰면서 더 많은 기능성을 얻을 수 있다. 여러분은 'libmychar.a' 를 include해야 하고 library 를 "C++" library들이 위치한 /usr/lib directory에 카피해야한다 ('libmychar.a'는 19 (Appendix H)에 있는 makefile에서 만들 수 있다). 'libmychar.a'을 사용하기 袖kV漫는 여러분의 프로그램을 다음과 같이 컴파일하시오.

```
g++ example.cpp -lmychar
```

다음 샘플코드를 보시오.

```
mychar aa;

aa = " Washington DC is the capital of USA ";

// You can use aa.val like a 'char *' variable in programs !!
for (unsigned long tmpii = 0; tmpii < aa.length(); tmpii++)
{
        fprintf(stdout, "aa.val[%ld]=%c ", tmpii, aa.val[tmpii]);
}

// Using pointers on 'char *' val ...
// Note: You must use a temporary local variable and assign the
// pointer to aa.val. If you directly use aa.val and when
// aa.val is incremented with aa.val++, then aa will go
// call destructor and later when aa.val is accessed that
// will cause core dump !!
for (char *tmpcc = aa.val; *tmpcc != 0; tmpcc++)
{
        // MUST use temporary variable tmpcc !! See note above.
        fprintf(stdout, "aa.val=%c ", *tmpcc);
}
```

mychar class 를 정의한 완전한 예제 프로그램 "example_mychar.cpp" 은 12 (Appendix A)에 있고 mychar class 는 13 (Appendix B)에 있다.

3.1 연산자들

'mychar class'는 다음과 같은 연산자들을 제공한다.

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

연산자들을 사용한 예제

```
mychar aa;
mychar 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 eqaul to 'Bill Clinton' " << endl;

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

3.2 함수들

'mychar class'는 다음고 같은 함수들을 제공한다.

- 현재 문자열 길이 **length()**
- 문자열 왼쪽 잘라내기. 처음에 나오는 공백-새줄, 텝-제거하기 **ltrim()**
- 문자열 오른쪽 잘라내기. 따라나오는 공백-새줄, 텝- 제거하기 **rtrim()**
- 처음과 나중에 나오는 공백 제거하기 **trim()**
- 따라나오는 새줄 제거하기 **chop()**
- 문자열을 대문자로 바꾸기 **to_upper()**
- 문자열을 소문자로 바꾸기 **to_lower()**
- 실수값들을 반올림하거나 내림하기 **roundf(float input_val, short precision)**
- double값들을 반올림하거나 내림하기 **roundd(double input_val, short precision)**
- start위치에서부터 문자열 substr과 일치하는 위치 찾아주기 **pos(char *substr, unsigned long start)**
- Explodes the string and returns the list in the list-head pointer explodeH **explode(char *seperator)**
- Implodes the strings in the list-head pointer explodeH and returns the mychar variable **implode(char *glue)**
- 리스트 헤드 포인터 explodeH의 문자열을 붙이고 mychar 변수반환 **join(char *glue)**
- 입력 문자열을 n번 반복 **repeat(char *input, unsigned int multiplier)**
- 문자열의 글자를 역순으로**reverse()**
- 문자열 'needle'을 건초더미 'val'에서 찾아 'str'로 바꾸기 **replace(char *needle, char *str)**
- Char를 번역 **str_tr(char *from, char *to)**
- 텍스트 문자열을 중앙에 **center(int length, char padchar = '')**
- 원래 문자열의 공백문자를 'number'개의 'padchar'로 바꾸어 준다. 맨 앞과 뒤의 공백은 항상 제거된다. 만약 'number'가 없거나 0이면, 문자열 안의 모든 공백이 제거된다.'number'의 기본값은 0이고 'padchar'의 기본값은 ''. **space(int number = 0, char padchar = ' ')**
- 'start'와 'end'를 포함한 모든 문자들로 이루어진 문자열 반환 **xrange(char start, char end)**
- 'list'에 들어 있는 모든 문자 제거. 'list'의 기본문자는 공백문자 ' ' **compress(char *list)**
- 'start' 위치에서부터 'length'개 문자를 문자열에서 제거. 만약 'statr'가 문자열 길이보다 더 길면 문자열은 변하지 않는다. **delstr(int start, int length)**

- 'start'에서 부터 'newstr'를 val에 삽입한다. 'newstr'은 'length'개 문자로 맞춰진다. 기본 'length'는 newstr의 문자열 길이 **insert(char *newstr, int start = 0, int length = 0, char padchar = ')**
- Val안의 가장 왼쪽에 있는 'length'개의 문자열을 반환. 문자열 왼쪽 줄맞추기에 빠른 방법 **left(int length = 0, char padchar = ' ')**
- Val안의 가장 오른쪽에 있는 'length'개의 문자열을 반환. 문자열 오른쪽 줄맞추기에 빠른 방법 **right(int length = 0, char padchar = ' ')**
- 'newstr'가 val안에 'start'에서 부터 덮어 씌여진다. 'newstr'은 'length'개의 문자로 맞춰진다. 기본 'length'은 newstr의 길이 **overlay(char *newstr, int start = 0, int length = 0, char padchar = ' ')**
- 문자열중 부분의 빼낸다. **substr(int start, int length = 0)**
- regx랑 처음 일치하는 곳을 찾는다. **at(char *regx)**
- Regx앞의 문자열을 반환 **before(char *regx)**
- Regxe뒤의 문자열을 반환 **after(char *regx)**
- 문자열이 NULL이면 참값 반환 **isnull()**
- 문자열을 NULL로 **clear()**

3.3 기타 함수들

기타 mychar 함수들을 여기에 모아두었다. 하지만 이것들을 사용하지는 마라. 대신 '+', '+=', '==' 등과 같은 연산자들을 사용하라. 이것들은 'mychar' class 'private'멤버들이다.

- Copy string **str_cpy(char *bb)**
- Long integer converted to string **str_cpy(unsigned long bb)**
- Integer converted to string **str_cpy(int bb)**
- Float converted to string **str_cpy(float bb)**
- String concatenate a char * **str_cat(char *bb)**
- String concatenate a int **str_cat(int bb)**
- String concatenate a int **str_cat(unsigned long bb)**
- String concatenate a float **str_cat(float bb)**
- Is equal to mychar ? **bool equalto(const mychar & rhs, bool type = false)**
- Is equal to char* ? **bool equalto(const char *rhs, bool type = false)**

예를 들어 정수를 문자열로 바꾸기 위해서는 다음과 같이 하라.

```
mychar  aa;

aa = 34;  //연산자 ' ='  는 int 을 string 로 바꾼다.
cout << "The value of aa is : " << aa.val << endl;

aa = 234.878; // 연산자 '=' 는 float 을 string로 바꾼다.
```

```
cout << "The value of aa is : " << aa.val << endl;

aa = 34 + 234.878;
cout << "The value of aa is : " << aa.val << endl;
// aa 는 '268.878' 로 된다.
// mychar를 cast 해야 한다.
aa = (mychar) 34 + " Honourable President Ronald Reagan " + 234.878;
cout << "The value of aa is : " << aa.val << endl;
// '34 Honourable President Ronald Reagan 234.878' 로 출력된다.
```

제 4 절 C++ Zap (Delete) 명령어

C++ 의 **delete** 과 **new** 명령어는 "C"의 malloc 과 free 함수보다 훨씬 낫다. malloc 과 free 대신에 new 와 zap (delete) 명령어를 가능한 많이 쓰도록 하자. delete명령어를 더 확실히 사용하기 위해서 Zap() 명령어를 만들자. zap() 명령어는 다음과 같이 정의될 수 있다.

```
/*
**완벽하고 튼튼한 매크로를 만들기 위해 do while 을 사용한다.
**예를 들어, 만약 "do-while"을 쓰지 않으면, 다음과 같이 될
**것이다
** if (bbint == 4)
**              aa = 0
** else
**              zap(aptr); //문제발생! aptr이 항상 NULL로 될 것이다
*/

#define zap(x) do { delete(x); x = NULL; } while (0)
```

자 이제 모든 포인터들이 널 포인터로 생성된다고 가정을 한다면 일련의 zap() 명령어로 다음과 같이 new로 생긴것들을 안전하게 삭제 할 수 있고 아직 new로 생기지 않은 것들을 지우지 않을 수 있다. 예를 들어,

```
zap(pFirstname);
zap(pFirstname); // no core dumps !! Because pFirstname is NULL now
zap(pFirstname); // no core dumps !! Because pFirstname is NULL now

zap(pLastname);
zap(pJobDescription);
```

이것은 전혀 신기한 것이 아니다. 이것은 단지 반복되는 코드를 세이브하고 더 읽기 좋게 만드는 것이다. zap() 명령어에서 typecast에 집착하지 마라 - 만약 zap() 명령어 위에서 뭔가 에러가 생기면 그건 어딘가에서 다른 에러가 있을 가능성이 크기 때문이다. 또한 5 (my_malloc()) , my_realloc() 그리고 my_free()는 더 깨끗하고 additional checks이 있기때문에 malloc(), realloc() and free(), 대신에 사용되어야 한다. 예를 들어 5 (my_malloc()) 과 my_free() 함수를 사용한 mychar.h" 보라. **주의:** 'new' 로 잡혀진 메모리을 free()로 풀거나 또눈 malloc으로 잡혀진 메모리를 'delete' 풀지마라. 이럴 경우 결과를 예측할 수 없다.

제 5 절 my_malloc 과 my_free의 사용

malloc 과 realloc을 피하고 가능한 한 new and zap(delete)을 많이 쓰도록 하자. 하지만 가끔은 "C++"에서 "C" 스타일의 메모리 얼로케이션을 써야할 필요가 있을 수도 있다. 함수 my_malloc() , my_realloc() and my_free()를 써라. 이 함수들은 allocations 과 initialisations을 적절히 하고 메모리 문제들을 방지한다. 또한 이 함수들은 (DEBUG 모드에서) 얼로케이트된 메모리를 추적하고 프로그램이 실행되기 전과 후의 총 메모리 사용을 프린트한다. 이것은 여러분에게 메모리 유출이 있는지의 여부를 알려준다. my_malloc 과 my_realloc 은 아래와 같이 정의된다. 이것은 조금 더 많은 메모리(SAFE_MEM = 5) 를 할당하고 공간을 초기화한다. 그리고 만약에 프로그램안에 이미 할당이 되어 있으면 할당이 되지 않는다. 'call_check(), remove_ptr()' 함수는 오직 DEBUG가 makefile에서 정의 되었을때에만 작동이 되고 이 함수들은 ((void)0) (즉 NULL for non-debug production release.) 에 assigned 된다. 그들은 사용된 총 메모리를 추적하는 것을 가능케 한다.

```
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 프로그램의 full implementation 을 위해서 my_malloc.cpp. 와 헤더 화일 my_malloc.h. 을 보라. my_malloc 과 my_free 사용예는 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 에서는 변수 자체가 전달되고, 정확한 my_realloc 이 불려지기때문에 (즉 적절한 데이타 타입 포인터가 반환되기때문에) 데이타타입을 변환할 필요가 없다는 것을 주지하라. my_realloc은 char*, int* 그리고 float*를 위한 중복된(overloaded) 함수들을 갖고 있다.

제 6 절 Debug 화일들

C++ 이나 C 프로그램을 디버그하려면 debug.h 화일을 include시키고 debug.h 함수를 연결시키도록 'Makefile' 안에서 DEBUG를 정의해라. '-DDEBUG' 를 삭제하면 debug 함수 calls는 ((void)0) i.e. NULL로 된다. 그러므로 프로젝트의 최종 production release 버전에 아무런 영향을 끼치지 않는다. 여러분은 프로그램에서 debug 함수들을 마음대로 사용할 수 있으며 production 실행화일의 크기를 증가시키지 않는다.

debug 루틴의 구현을 보려면18 (debug.cpp) 그리고 debug.h 와 debug 함수를 사용한 샘플을 보기위해서는 15 (my_malloc.cpp) 화일을 보라. 샘플 19 (Makefile) 을 보라.

제 7 절 C++ Online 문서들

다음의 C++ 사이트들을 방문해 보라. :-

- C++ Crash-proof site <http://www.troubleshooters.com/codecorn/crashprf.htm>
- C++ Memory site <http://www.troubleshooters.com/codecorn/memleak.htm>

인터넷에는 수많은 C++ 관련 문서들이 있다. Yahoo, Lycos, Infoseek, 그리고 Excite 와 같인 서치엔진에 가서 **'C++ tutorials' 'C++ references' 'C++ books'** 을 키워드로 타입을 하면 찾을 수 있다. Advanced search 를 클릭해서 서치 결과의 분류를 좁힐 수도 있고 어휘나 구절을 정확히 써서 서치할 수도 있다.

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

7.1 C++ 튜터리얼들

인터넷에 사용가능한 많은 온라인 튜터리얼들이 있다. 서치 엔진에서 'C++ tutorials' 를 타입해서 찾을 수 있다.

7.2 C++ 코딩 표준

다음과 같은 C++ 코딩 URL들을 가 보자.

- C++ coding standard <http://www.cs.umd.edu/users/cml/cstyle/CppCodingStandard.html>
- Coding standards from Possibility <http://www.possibility.com/Cpp/CppCodingStandard.html>
- Coding standards 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>

7.3 C++ 퀵 레퍼런스

서치엔진에서 'C++ Reference' 를 타입해서 찾을 수 있다.

7.4 C++ 유즈넷 뉴스그룹들

- C++ newsgroups : <comp.lang.c++.announce>
- C++ newsgroups : <comp.lang.c++.*>

제 8 절 메모리 도구들

다음의 메모리 디버깅 도구들을 사용할 수 있다.

- 리눅스 contrib cdrom에 대해서는 mem_test*.rpm 팩키지를 보면 된다.
- 리눅스 시디롬에 대해서는 ElectricFence*.rpm 팩키지를 보면 된다.
- Purify Tool from Rational Software Corp <http://www.rational.com>
- Insure++ Tool from Parasoft Corp <http://www.parasoft.com>
- Linux Tools at <http://www.xnet.com/~blatura/linapp6.html##tools>
- Yahoo, Lycos, Excite, 혹은 Mamma.com 과 같은 인터넷 서치엔진에서 "Linux memory debugging tools"을 키워드로 타입을 해서 찾아보면 된다

제 9 절 관련 URL들

C, C++ 과 관련된 다음의 인터넷 사이트를 방문해 보자.

- Vim color text editor for C++, C <http://metalab.unc.edu/LDP/HOWTO/Vim-HOWTO.html>
- C++ Beautifier HOWTO <http://metalab.unc.edu/LDP/HOWTO/C-C++Beautifier-HOWTO.html>
- CVS HOWTO for C++ programs <http://metalab.unc.edu/LDP/HOWTO/CVS-HOWTO.html>
- Linux goodies main site <http://www.aldev.8m.com>
- Linux goodies mirror site <http://aldev.webjump.com>

제 10 절 이 문서의 다른 포멧들

이 문서는 10여종의 다른 포멧 -DVI, 포스트스크립트, 레이텍스, LyX, GNU-info, HTML, RTF(Rich Text Format), Plain-text, Unix man pages 그리고 SGML - 으로 나왔다.

- 이 HOWTO 문서는 HTML, DVI, Postscript 혹은 SGML 포멧으로 된 한개의 tar화일로 다음에서 볼 수 있다. <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/other-formats/> or <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO/other-formats/>
- Plain text 포멧은: <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> or <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO>
- 불어나 독일어, 스페인어, 중국어 일본어와 같은 외국어로 번역된 것은 <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> or <ftp://metalab.unc.edu/pub/Linux/docs/HOWTO> 에서 볼 수 있다. 번역을 도와 주고자 하는 사람은 누구나 환영한다.

이문서는 "SGML" 툴을 이용해서 쓰여졌으며, "SGML" 툴은 The document is written using a tool called "SGML tool" which can be got from - <http://www.xs4all.nl/~cg/sgmltools/> 에서 구할 수 있다. 소스를 컴파일 하면 다음과 같은 명령어를 보게 될 것이다.

- sgml2html C++Programming-HOWTO.sgml (to generate html file)
- sgml2rtf C++Programming-HOWTO.sgml (to generate RTF file)
- sgml2latex C++Programming-HOWTO.sgml (to generate latex file)

이 문서는 다음에 있다.

- <http://metalab.unc.edu/LDP/HOWTO/C++Programming-HOWTO.html>

또한 다음의 미러 사이트에서 이 문서를 찾을 수 있다.

- <http://www.caldera.com/LDP/HOWTO/C++Programming-HOWTO.html>
- <http://www.WGS.com/LDP/HOWTO/C++Programming-HOWTO.html>
- <http://www.cc.gatech.edu/linux/LDP/HOWTO/C++Programming-HOWTO.html>
- <http://www.redhat.com/linux-info/ldp/HOWTO/C++Programming-HOWTO.html>
- (네트워크 주소면에서) 가까운 다른 미러 사이트들은 <http://metalab.unc.edu/LDP/hmirrors.html> 에서 찾을 수 있다. 사이트를 선택하고 /LDP/HOWTO/C++Programming-HOWTO.html 디렉토리로 가면 된다.

이 문서를 dvi 포멧으로 보기위해서는 xdvi 프로그램을 사용하면 된다. xdvi 프로그램은 레드헷 리눅스의 tetex-xdvi*.rpm 팩키지 안에 있으며 제어판|응용프로그램(Applications) | Publishing | TeX 메뉴버튼을 통해서 찾을 수 있다.

```
dvi 문서를 읽기 위해서는 xdvi -geometry 80x90 howto.dvi 라는
명령어를 주어야 한다.
그리고 마우스로 윈도우의 크기를 조절하고 xdvi의 메인 페이지를 보면
된다.
위아래로 움직이기 위해서는 확살표 키나 페이지 업, 페이지 다운등의
키를 사용하거나
'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' 의 글자키를 위로 올리거나
아래로 중앙으로 혹은 다음 페이지나 이
전 페이지로 이동하기 위해서 사용할 수도 있다.
expert 메뉴를 없애려면 'x' 를 누르면 된다.
```

'gv' (ghostview) 나 'ghostscript' 라는 프로그램을 이용해서 포스트스크립트 화일을 읽을 수 있다. 고스트스크립트 프로그램과 gv 프로그램은 각각 레드헷 리눅스의 ghostscript*.rpm 팩키지에 있고 gv 프로그램은 gv*.rpm 팩키지에 있고, ControlPanel | Applications | Graphics 메뉴버튼을 통해서 위치를 찾아낼 수 있다. gv 프로그램은 고스트스크립트보다 사용자가 편리하도록 만들어져 있다. 고스트스크립트와 gv는 OS/2나 윈도우 95 그리고 NT와 같은 다른 플렛폼에서도 사용 가능하다.

- Get ghostscript for Windows 95, OS/2, and for all OSes from <http://www.cs.wisc.edu/~ghost>

```
포스트스크립트 문서를 읽기위해서는
 gv howto.ps 라는 명령어를 쓴다.
고스트스크립트를 사용하기 위해서는
 ghostscript howto.ps 를 쓴다.
```

HTML 포멧의 문서는 넷스케이프 네비게이터나 마이크로소프트 인터넷 익스플로어, 레드헷 바론 웹 브라우저 혹은 다른 웹 브라우저를 써서 읽을 수 있다 latex, LyX 문서는 LyX a "X-Windows" front end to latex 을 이용해 읽을 수 있다.

제 11 절 저작권

저작권 조항은 LDP (리눅스 문서 프로젝트) 로서 GNU/GPL 이다. LDP 는 하나의 GNU/GPL프로젝트이다. 추가로 요구되는 것들로서 저자의 이름, 이메일 주소, 그리고 모든 복사본들에 이 저작권 조항에 대한 부분을 넣어주어야 한다. 이 문서를 조금이라도 바꾸거나 뭔 추가하게 되면 이 문서의 모든 저자들에게 그것을 알려주기 바란다.

제 12 절 첨부 A example_mychar.cpp

모든 프로그램들은 Download mychar 로 부터 하나의 tar.gz 화일로 다운로드 받을 수 있다. 이 화일을 얻으려면 웹브라우저에서 'Text' 타입으로 저장을 하면 된다.

```
//*****************************************************************
//저작권은 GNU/GPL 이지만 추가로 모든 복제본에 저자의 이름과 이메일을
//포함시켜주어야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************
// 컴파일 하려면( libmychar.a가 현재 디렉토리에 있다고
// 가정하고 )
//              g++ example_mychar.cpp -L. -lmychar

#include <stdlib.h>  // for putenv
#include "mychar.h"

////////////////////////////////////////////////
//mychar 사용 예제 프로그램
////////////////////////////////////////////////

int main(int argc, char **argv)
{
        char p_name[1024];
        sprintf(p_name, "PROGRAM_NAME=%s", argv[0]);
```

```
putenv(p_name);
print_total_memsize(); // in the beginning
mychar aa, bb;

//bb.str_cpy("  bbSTRing  ");
bb = "   bbSTRing   ";

//+ 연산자 테스트하기
//aa + " rhs "; //  여기서는 어떤 출력도 갖지 않게 될것이다 !!
//반드시 다음줄에 나오는 것 처럼 직접 fprintf 를 사용해야 한다.
fprintf(stdout, "\n0) aa.val is :%sEOF\n", (aa + " my rhs " ).val);

//=연산자 테스트하기
aa = " lhs " ;
fprintf(stdout, "0-1) With operator= aa.val is :%sEOF\n", aa.val);

//+ 연산자 테스트 하기
//aa + " rhs "; //  여기서는 어떤 출력도 갖지 않게 될것이다 !!
//반드시 다음줄에 나오는 것 처럼 직접 fprintf 를 사용해야 한다.
fprintf(stdout, "\n0) With lsh operator+, aa.val is :%sEOF\n", (" my lhs " + aa ).val);

//aa.str_cpy(bb.val);
aa = bb;
aa.to_upper();
fprintf(stdout, "1) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa.to_lower();
fprintf(stdout, "2) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa.ltrim();
fprintf(stdout, "3) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa.rtrim();
fprintf(stdout, "4) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa.trim();
fprintf(stdout, "5) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa = aa + " testing newlines \n\n\n\n";
aa.chop();
fprintf(stdout, "5-1) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa = aa + " rhs ";
fprintf(stdout, "6) aa.val is :%sEOF\n", aa.val);
```

```
aa = bb;
aa = " lhs " + aa;
fprintf(stdout, "7) aa.val is :%sEOF\n", aa.val);

// 숫자 더하기 예제
//aa = (mychar) 9989 + "kkk" + 33 ;
aa = 9999;
fprintf(stdout, "7-1) aa.val is :%sEOF\n", aa.val);

aa = bb;
aa = " lhs " + aa + " rhs " + " 9989 " + " 33 ";
fprintf(stdout, "8) aa.val is :%sEOF\n", aa.val);

aa = " AA value ";
aa = bb + "alkja " + " 99djd " ;
fprintf(stdout, "9) aa.val is :%sEOF\n", aa.val);

aa = " AA value ";
aa = (mychar) "alkja " + " 99djd " ;
fprintf(stdout, "10) aa.val is :%sEOF\n", aa.val);

aa = " AA value ";
aa += (mychar) " al dev test kkk... " + " al2 slkj" + " al3333 ";
fprintf(stdout, "11) aa.val is :%sEOF\n", aa.val);

aa = " AA value ";
aa = aa + " add aa " + aa + aa + aa + " 1111 " + " 2222 " + aa + aa + aa + " 3333 ";
fprintf(stdout, "12) aa.val is :%sEOF\n", aa.val);

aa = "12345678";
aa.reverse();
fprintf(stdout, "13) aa.val is :%sEOF\n", aa.val);

aa = " AA value ";
aa = aa + " add aa " + aa + 1111 +" "+ 2222 + " " + 3.344 + aa;
fprintf(stdout, "14) aa.val is :%sEOF\n", aa.val);

aa.roundd(123456.0123456789012345, 13);
fprintf(stdout, "15) double aa.val is :%sEOF\n", aa.val);

aa.roundf(123456.0123456789, 13);
fprintf(stdout, "16) float aa.val is :%sEOF\n", aa.val);

// Test equal to operators
aa = " AA value ";
mychar cc(" AA value ");
if (aa == cc)
        fprintf(stdout, "\naa=%s and cc=%s are equal!!\n", aa.val, cc.val);
else
```

```
                fprintf(stdout, "\naa=%s and cc=%s are NOT equal!!\n", aa.val, cc.val);
        cc = "CC";
        if (aa == cc)
                fprintf(stdout, "\naa=%s and cc=%s are equal!!\n", aa.val, cc.val);
        else
                fprintf(stdout, "\naa=%s and cc=%s are NOT equal!!\n", aa.val, cc.val);
        if (aa == " AA value ")
                fprintf(stdout, "\naa=%s and string are equal!!\n", aa.val);
        else
                fprintf(stdout, "\naa=%s and string are NOT equal!!\n", aa.val);
        if (aa == " AA valuexxx ")
                fprintf(stdout, "\naa=%s and string are equal!!\n", aa.val);
        else
                fprintf(stdout, "\naa=%s and string are NOT equal!!\n", aa.val);

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

        //'char *' val ...에 포인터 사용하기
        fprintf(stdout, "\n ");

        for (char *tmpcc = aa.val; *tmpcc != 0; tmpcc++)
        {
                // MUST use temporary var tmpcc !! See note above.
                fprintf(stdout, "aa.val=%c ", *tmpcc);
        }
        fprintf(stdout, "\n");

        print_total_memsize(); // in the end
        exit(0);
}
```

제 13 절 첨부 B mychar.h

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

//메모리유출을 방지하기위해서는 - a char class to manage character
```

```
//variables
//char[] 혹은 char *대신  mychar 나 string class 를 사용하는 것이 좋다.


#ifndef __MYCHAR_H_
#define __MYCHAR_H_

//#include <iostream> // iostream은 프로그램이 커지니까 사용하지 않는다.
//#include <stdlib.h> //free() 와malloc() 를 쓰기위해
#include <string.h> // for strcpy()
#include <ctype.h> // for isspace()
#include <stdio.h> // for sprintf()
#include <list.h> // for sprintf()
#include <math.h> // for modf(), rint()

#include "my_malloc.h"
#include "debug.h" // debug_(name, value)  debug2_(name, value, LOG_YES)

const short INITIAL_SIZE =      50;
const short NUMBER_LENGTH = 70;

// 최소한의 함수와 변수를 가진 작은 클래스 (class)
// This class to be kept small... 이 클래스는 계속 작게 유지될것이다.
class mychar
{
        public:
                mychar();
                mychar(char bb[]);  // needed by operator+
                mychar(int bb);  // needed by operator+
                mychar(unsigned long bb);  // needed by operator+
                mychar(float bb);  // needed by operator+
                mychar(double bb);  // needed by operator+
                mychar(const mychar & rhs);  // Copy Constructor needed by operator+
                ~mychar();

                char *val;

                unsigned long length() { return strlen(val); }

                void ltrim();
                void rtrim();
                void trim();
                void chop();

                void to_upper();
                void to_lower();

                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);

        long pos(char substr[], unsigned long start);

        void explode(char *seperator);
        void implode(char *glue);
        void join(char *glue);
        void repeat(char *input, unsigned int multiplier);
        void reverse();
        void replace(char *needle, char *str);
        void str_tr(char *from, char *to);
        void center(int length, char padchar = ' ');
        void space(int number = 0, char padchar = ' ');
        void xrange(char start, char end);
        void compress(char *list);
        void delstr(int start, int length);
        void insert(char *newstr, int start = 0, int length = 0, char padchar = ' ');
        void left(int length = 0, char padchar = ' ');
        void right(int length = 0, char padchar = ' ');
        void overlay(char *newstr, int start = 0, int length = 0, char padchar = ' ');
        mychar substr(int start, int length = 0);

        mychar at(char *regx); // matches first match of regx
        mychar before(char *regx); // returns string before regx
        mychar after(char *regx); // returns string after regx

        bool isnull();
        void clear();

        // All Operators ...
        mychar operator+ (const mychar & rhs);
        friend mychar operator+ (const mychar & lhs, const mychar & rhs);

        mychar& operator+= (const mychar & rhs); // using reference will be faster
        mychar& operator= (const mychar & rhs); // using reference will be faster
        bool operator== (const mychar & rhs); // using reference will be faster
        bool operator== (const char *rhs);
        bool operator!= (const mychar & rhs);
        bool operator!= (const char *rhs);

        static  list<mychar>            explodeH;  // list head

private:
        //static mychar *global_mychar; // for use in add operator
        //inline void free_glob(mychar **aa);
        void str_cpy(char bb[]);
        void str_cpy(int bb); // itoa
        void str_cpy(unsigned long bb);
        void str_cpy(float bb); // itof
```

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

                bool equalto(const mychar & rhs, bool type = false);
                bool equalto(const char *rhs, bool type = false);
};
// Global variables are defined in mychar.cpp

#endif // __MYCHAR_H_
```

제 14 절 첨부 C mychar.cpp

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

//문자열 클래스나 이 클래스를 사용한다.
//
//메모리유출을 방지하기위해서는 - a char class to manage character
//variables
//char[] 혹은 char *대신  mychar 나 string class 를 사용하는 것이 좋다.

// 다음과 같이 컴파일 할 것
//              g++ mychar.cpp

#include "mychar.h"

// Global variables ....
//mychar *mychar::global_mychar = NULL; // global var
list<mychar>            mychar::explodeH;

mychar::mychar()
{
        debug_("In cstr()", "ok");
        val = (char *) my_malloc(sizeof(char)* INITIAL_SIZE);
}

mychar::mychar(char *bb)
{
        unsigned long tmpii = strlen(bb);
        val = (char *) my_malloc(sizeof(char)* tmpii);
```

```
        strncpy(val, bb, tmpii);
        val[tmpii] = '\0';

        //debug_("In cstr(char *bb) bb", bb);
        //debug_("In cstr(char *bb) val", val);
        #ifdef DEBUG
                //fprintf(stderr, "\nAddress of val=%x\n", & val);
                //fprintf(stderr, "\nAddress of this-pointer=%x\n", this);
        #endif // DEBUG
}

mychar::mychar(int bb)
{
        val = (char *) my_malloc(NUMBER_LENGTH); // integers 70 digits max
        sprintf(val, "%d", bb);
}

mychar::mychar(unsigned long bb)
{
        val = (char *) my_malloc(NUMBER_LENGTH); // long 70 digits max
        sprintf(val, "%lu", bb);
}

mychar::mychar(float bb)
{
        val = (char *) my_malloc(NUMBER_LENGTH); // float 70 digits max
        sprintf(val, "%f", bb);
}

mychar::mychar(double bb)
{
        val = (char *) my_malloc(NUMBER_LENGTH); // double 70 digits max
        sprintf(val, "%f", bb);
}

// Copy Constructor needed by operator +
mychar::mychar(const mychar & rhs)
{
        // Do a deep-copy instead of compiler's default shallow copy copy-cstr
        debug_("In copy-cstr()", "ok");
        unsigned long tmpii = strlen(rhs.val);
        val = (char *) my_malloc(sizeof(char)* tmpii);
        strncpy(val, rhs.val, tmpii);
        val[tmpii] = '\0';
}

mychar::~mychar()
{
        //debug_("In dstr val", val);
        #ifdef DEBUG
```

```
                //fprintf(stderr, "\nAddress of val=%x\n", & val);
                //fprintf(stderr, "\nAddress of this-pointer=%x\n", this);
        #endif // DEBUG
        my_free(val);
        //delete [] val;
        val = NULL;
}

// MUST use pointer-to-pointer **aa, otherwise the argument
// is NOT freed !!
/*
inline void mychar::free_glob(mychar **aa)
{
        debug_("called free_glob()", "ok" );
        if (*aa != NULL)  // (*aa != NULL)
        {
                debug_("*aa is not null", "ok");
                delete *aa;
                *aa = NULL;
        }
        //else
                debug_("*aa is null", "ok");

        //if (*aa == NULL)
        debug_("*aa set to null", "ok");
}
*/

// Explodes the string and returns the list in
// the list-head pointer explodeH
void mychar::explode(char *seperator)
{
        char *aa = NULL, *bb = NULL;
        aa = (char *) my_malloc(length());
        for (bb = strtok(aa, seperator); bb != NULL; bb = strtok(NULL, seperator) )
        {
                mychar *tmp = new mychar(bb);
                mychar::explodeH.insert(mychar::explodeH.end(), *tmp);
        }
        my_free(aa);

        list<mychar>::iterator iter1; // see file include/g++/stl_list.h
        debug_("Before checking explode..", "ok");
        if (mychar::explodeH.empty() == true )
        {
                debug_("List is empty!!", "ok");
        }

        for (iter1 = mychar::explodeH.begin(); iter1 != mychar::explodeH.end(); iter1++)
        {
```

```
                if (iter1 == NULL)
                {
                        debug_("Iterator iter1 is NULL!!", "ok" );
                        break;
                }
                debug_("(*iter1).val", (*iter1).val);
        }
}

// Implodes the strings in the list-head
// pointer explodeH and returns the mychar class
void mychar::implode(char *glue)
{
}

// Joins the strings in the list-head
// pointer explodeH and returns the mychar class
void mychar::join(char *glue)
{
        implode(glue);
}

// Repeat the input string n times
void mychar::repeat(char *input, unsigned int multiplier)
{
        // For example -
        // repeat("1", 4) returns "1111"
        if (!input) // input == NULL
        {
                val[0] = 0;
                return;
        }

        val = (char *) my_malloc(strlen(input) * multiplier);
        for (unsigned int tmpii = 0; tmpii < multiplier; tmpii++)
        {
                strcat(val, input);
        }
}

// Reverse the string
void mychar::reverse()
{
        // For example -
        //              reverse() on "12345" returns "54321"
        char aa;
        unsigned long tot_len = length();
        unsigned long midpoint = tot_len / 2;
        for (unsigned long tmpjj = 0; tmpjj < midpoint; tmpjj++)
        {
```

```
                aa = val[tmpjj];  // temporary storage var
                val[tmpjj] = val[tot_len - tmpjj - 1];  // swap the values
                val[tot_len - tmpjj - 1] = aa; // swap the values
        }
}

// Replace all occurences of string 'needle' with 'str' in the haystack 'val'
void mychar::replace(char *needle, char *str)
{
        // For example -
        //              replace("AAA", "BB") on val = "some AAA and AAACC"
        //              reurns val = "some BB and BBCC"
}

// Translate certain chars
void mychar::str_tr(char *from, char *to)
{
        // For e.g ("abcd", "ABC") translates all occurences of each
        // character in 'from' to corresponding character in 'to'
}

// Center the text
void center(int length, char padchar = ' ')
{
        // For example -
        //              center(10, '*') on val="aa" returns "****aa****"
        //              center(10) on val="aa" returns "    aa    "
        // The result is a string of 'length' characters with val centered in it.
}

// Formats the original string by placing <number> of <padchar> characters
// between each set of blank-delimited words. Leading and Trailing blanks
// are always removed. If <number> is omitted or is 0, then all spaces are
// in the string are removed. The default number is 0 and
// default padchar ' '
void space(int number, char padchar = ' ')
{
        // For example -
        //              space(3) on val = "I do not know"
        //                              will return "I   do   not   know"
        //              space(1, '_') on val = "A deep black space"
        //                              will return "A_deep_black_space"
        //              space() on val = "I   know     this"
        //                              will return "Iknowthis"
}

// The result is string comprised of all characters between
// and including <start> and <end>
void xrange(char start, char end)
{
```

```
        // For example -
        //      xrange('a', 'j') returns val = "abcdefghij"
        //      xrange(1, 8) returns val = "12345678"
}

// Removes any characters contained in <list>. The default character
// for <list> is a blank ' '
void compress(char *list)
{
        // For example -
        //      compress("$,%") on val = "$1,934" returns "1934"
        //      compress() on val = "call me alavoor vasudevan" returns "callmealavoorvasudevan"
}

// Deletes a portion of string of <length> characters from <start> position.
// If start is greater than the string length then string is unchanged.
void delstr(int start, int length)
{
        // For example -
        //      delstr(3,3) on val = 'pokemon' returns 'poon'
}

// The <newstr> in inserted into val beginning at <start>. The <newstr> will
// be padded or truncated to <length> characters. The default <length> is
// string length of newstr
void insert(char *newstr, int start = 0, int length = 0, char padchar = ' ')
{
        // For example -
        //      insert("something new", 4, 20, '*') on val = "old thing"
        //              returns "old something new*******thing"
}

// The result is string of <length> chars madeup of leftmost chars in val.
// Quick way to left justify a string.
void left(int length = 0, char padchar = ' ')
{
        // For example -
        //      left(10) on val = "Wig" returns "Wig      "
        //      left(4) on val = "Wighat" returns "Wigh"
}

// The result is string of <length> chars madeup of rightmost chars in val.
// Quick way to right justify a string.
void right(int length = 0, char padchar = ' ')
{
        // For example -
        //      right(10) on val = "never stop to saying" returns " to saying"
        //      right(4) on val = "Wighat" returns "ghat"
        //      right(6) on val = "4.50" returns "  4.50"
}
```

```
// The <newstr> in overlayed into val beginning at <start>. The <newstr> will
// be padded or truncated to <length> characters. The default <length> is
// string length of newstr
void overlay(char *newstr, int start = 0, int length = 0, char padchar = ' ')
{
        // For example -
        //      overlay("12345678", 4, 10, '*') on val = "oldthing is very bad"
        //              returns "old12345678**ery bad"
}

// sub string
mychar mychar::substr(int start, int length = 0)
{
        if (!length) // length == 0
                return(mychar(& val[start-1]) );
        else
        {
                mychar tmp = mychar(& val[start-1]);
                tmp.val[length-1] = 0;
                return(tmp);
        }
}

// If string is literrally equal to .. or not equal to
// If type is false then it is ==
bool mychar::equalto(const mychar & rhs, bool type = false)
{
        if (type == false) // test for ==
        {
                if (strlen(rhs.val) == length())
                {
                        if (!strncmp(rhs.val, val, length())) //  == 0
                                return true;
                        else
                                return false;
                }
                else
                        return false;
        }
        else // test for !=
        {
                if (strlen(rhs.val) != length())
                {
                        if (!strncmp(rhs.val, val, length())) //  == 0
                                return true;
                        else
                                return false;
                }
                else
```

```
                        return false;
        }
}

// If string is literrally equal to .. or not equal to
// If type is false then it is ==
bool mychar::equalto(const char *rhs, bool type = false)
{
        if (type == false) // test for ==
        {
                if (strlen(rhs) == length())
                {
                        if (!strncmp(rhs, val, length())) //  == 0
                                return true;
                        else
                                return false;
                }
                else
                        return false;
        }
        else // test for !=
        {
                if (strlen(rhs) != length())
                {
                        if (!strncmp(rhs, val, length())) //  == 0
                                return true;
                        else
                                return false;
                }
                else
                        return false;
        }
}

// find position, matching substr beginning from start..
long mychar::pos(char *substr, unsigned long start)
{
        char * tok;
        long res = -1;

        if ( !isnull() && (start < strlen(val) ) )
        {
                tok = strstr(val + start, substr);
                if (tok == NULL)
                        res = -1;
                else
                        res = (long) (tok - val);
        }
        return res;
}
```

```
bool mychar::isnull()
{
        if (val[0] == '\0')
                return true;
        else
        {
                if (val == NULL)
                        return true;
                else
                        return false;
        }
}

void mychar::clear()
{
        val = (char *) my_realloc(val, 10);
        val[0] = '\0';
}

// Remove trailing new-lines
void mychar::chop()
{
        unsigned long tmpii = strlen(val) - 1 ;
        for (; tmpii >= 0; tmpii--)
        {
                if (val[tmpii] == '\n')
                        val[tmpii] = 0;
                else
                        break;
        }
}

void mychar::ltrim()
{
        // May cause problems in my_realloc since
        // location of bb will be destroyed !!
        char *bb = val;

        if (bb == NULL)
                return;

        while (isspace(*bb))
                bb++;
        debug_("bb", bb);

        if (bb != NULL && bb != val)
        {
                debug_("doing string copy", "done");
                //str_cpy(bb); // causes problems in my_realloc and bb is getting destroyed!!
```

```
                strcpy(val, bb); // strcpy is ok since val space is > bb space
        }
        else
                debug_("Not doing string copy", "done");
}

void mychar::rtrim()
{
        for (long tmpii = strlen(val) - 1 ; tmpii >= 0; tmpii--)
        {
                if ( isspace(val[tmpii]) )
                        val[tmpii] = '\0';
                else
                        break;
        }
}

void mychar::trim()
{
        rtrim();
        ltrim();
}

void mychar::to_lower()
{
        for (long tmpii = strlen(val); tmpii >= 0; tmpii--)
        {
                val[tmpii] = tolower(val[tmpii]);
        }
}

// Use for rounding off fractions digits of floats
// Rounds-off floats with given precision and then
// stores the result into mychar's val field
// Also returns the result as a char *
void mychar::roundf(float input_val, short precision)
{
        float   integ_flt, deci_flt;
        const   short MAX_PREC = 4;

        debug_("In roundf", "ok");

        if (precision > MAX_PREC) // this is the max reliable precision
                precision = MAX_PREC;

        // get the integral and decimal parts of the float value..
        deci_flt = modff(input_val, & integ_flt);

        for (int tmpzz = 0; tmpzz < precision; tmpzz++)
        {
```

```
                debug_("deci_flt", deci_flt);
                deci_flt *= 10;
        }
        debug_("deci_flt", deci_flt);

        unsigned long deci_int = (unsigned long) ( rint(deci_flt) );

        val = (char *) my_malloc(NUMBER_LENGTH); // float 70 digits max

        if (deci_int > 999) // (MAX_PREC) digits
                sprintf(val, "%lu.%lu", (unsigned long) integ_flt, deci_int);
        else
        if (deci_int > 99) // (MAX_PREC - 1) digits
                sprintf(val, "%lu.0%lu", (unsigned long) integ_flt, deci_int);
        else
        if (deci_int > 9) // (MAX_PREC - 2) digits
                sprintf(val, "%lu.00%lu", (unsigned long) integ_flt, deci_int);
        else
                sprintf(val, "%lu.00000%lu", (unsigned long) integ_flt, deci_int);
}

void mychar::roundd(double input_val, short precision)
{
        double  integ_flt, deci_flt;
        const   short MAX_PREC = 6;

        if (precision > MAX_PREC) // this is the max reliable precision
                precision = MAX_PREC;

        debug_("In roundd", "ok");
        // get the integral and decimal parts of the double value..
        deci_flt = modf(input_val, & integ_flt);

        for (int tmpzz = 0; tmpzz < precision; tmpzz++)
        {
                debug_("deci_flt", deci_flt);
                deci_flt *= 10;
        }
        debug_("deci_flt", deci_flt);

        val = (char *) my_malloc(NUMBER_LENGTH); // double 70 digits max

        unsigned long deci_int = (unsigned long) ( rint(deci_flt) );

        if (deci_int > 99999) // (MAX_PREC) digits
                sprintf(val, "%lu.%lu", (unsigned long) integ_flt, deci_int);
        else
        if (deci_int > 9999) // (MAX_PREC - 1) digits
                sprintf(val, "%lu.0%lu", (unsigned long) integ_flt, deci_int);
        else
```

```
        if (deci_int > 999) // (MAX_PREC - 2) digits
                sprintf(val, "%lu.00%lu", (unsigned long) integ_flt, deci_int);
        else
        if (deci_int > 99) // (MAX_PREC - 3) digits
                sprintf(val, "%lu.000%lu", (unsigned long) integ_flt, deci_int);
        else
        if (deci_int > 9) // (MAX_PREC - 4) digits
                sprintf(val, "%lu.0000%lu", (unsigned long) integ_flt, deci_int);
        else // (MAX_PREC - 5) digits
                sprintf(val, "%lu.00000%lu", (unsigned long) integ_flt, deci_int);
}

void mychar::to_upper()
{
        for (long tmpii = strlen(val); tmpii >= 0; tmpii--)
        {
                val[tmpii] = toupper(val[tmpii]);
        }
}

void mychar::str_cpy(char bb[])
{
        debug_("In str_cpy bb", bb);
        if (bb == NULL)
        {
                val[0] = '\0';
                return;
        }

        unsigned long tmpii = strlen(bb);

        if (tmpii == 0)
        {
                val[0] = '\0';
                return;
        }

        debug_("In str_cpy tmpii", tmpii);
        debug_("In str_cpy val", val);
        val = (char *) my_realloc(val, tmpii);
        //val = new char [tmpii + SAFE_MEM_2];
        debug_("In str_cpy bb", bb);

        strncpy(val, bb, tmpii);
        debug_("In str_cpy val", val);
        val[tmpii] = '\0';
        debug_("In str_cpy val", val);
}

void mychar::str_cpy(int bb)
```

```
{
        char tmpaa[100];
        sprintf(tmpaa, "%d", bb);
        str_cpy(tmpaa);
}

void mychar::str_cpy(unsigned long bb)
{
        char tmpaa[100];
        sprintf(tmpaa, "%ld", bb);
        str_cpy(tmpaa);
}

void mychar::str_cpy(float bb)
{
        char tmpaa[100];
        sprintf(tmpaa, "%f", bb);
        str_cpy(tmpaa);
}

void mychar::str_cat(char bb[])
{
        unsigned long tmpjj = strlen(bb), tmpii = strlen(val);
        val = (char *) my_realloc(val, tmpii + tmpjj);
        debug_("val in str_cat() ", val);
        strncat(val, bb, tmpjj);
}

void mychar::str_cat(int bb)
{
        char tmpaa[100];
        sprintf(tmpaa, "%d", bb);

        unsigned long tmpjj = strlen(tmpaa), tmpii = strlen(val);
        val = (char *) my_realloc(val, tmpii + tmpjj);
        strncat(val, tmpaa, tmpjj);
}

void mychar::str_cat(unsigned long bb)
{
        char tmpaa[100];
        sprintf(tmpaa, "%ld", bb);

        unsigned long tmpjj = strlen(tmpaa), tmpii = strlen(val);
        val = (char *) my_realloc(val, tmpii + tmpjj);
        strncat(val, tmpaa, tmpjj);
}

void mychar::str_cat(float bb)
{
```

```
        char tmpaa[100];
        sprintf(tmpaa, "%f", bb);

        unsigned long tmpjj = strlen(tmpaa), tmpii = strlen(val);
        val = (char *) my_realloc(val, tmpii + tmpjj);
        strncat(val, tmpaa, tmpjj);
}

mychar operator+ (const mychar & lhs, const mychar & rhs)
{
        /*******************************************************/
        // Note : For adding two char strings, first cast mychar
        // as in -
        //aa = (mychar) "alkja " + " 99djd " ;
        /*******************************************************/

        mychar tmp(lhs);
        tmp.str_cat(rhs.val);
        return(tmp);

        /*
        if (mychar::global_mychar == NULL)
        {
                mychar::global_mychar = new mychar;
                mychar::global_mychar->str_cpy(lhs.val);
                mychar::global_mychar->str_cat(rhs.val);
                //return *mychar::global_mychar;
                return mychar(mychar::global_mychar->val);
        }
        */
        /*
        else
        if (mychar::global_mychar1 == NULL)
        {
                debug_("1)global", "ok" );
                mychar::global_mychar1 = new mychar;
                mychar::global_mychar1->str_cpy(lhs.val);
                mychar::global_mychar1->str_cat(rhs.val);
                return *mychar::global_mychar1;
        }
        */
        /*
        else
        {
                fprintf(stderr, "\nError: cannot alloc global_mychar\n");
                exit(-1);
        }
        */

        /*
```

```
        mychar *aa = new mychar;
        aa->str_cpy(lhs.val);
        aa->str_cat(rhs.val);
        return *aa;
        */
}

mychar mychar::operator+ (const mychar & rhs)
{
        mychar tmp(*this);
        tmp.str_cat(rhs.val);
        debug_("rhs.val in operator+", rhs.val );
        debug_("tmp.val in operator+", tmp.val );
        return (tmp);
}

// Using reference will be faster in = operator
mychar& mychar:: operator= ( const mychar& rhs )
{
        if (& rhs == this)
        {
                debug_("Fatal Error: In operator(=). rhs is == to 'this pointer'!!", "ok" );
                return *this;
        }

        this->str_cpy(rhs.val);
        debug_("rhs value", rhs.val );

        // Free global vars memory
        //free_glob(& mychar::global_mychar);
        //if (mychar::global_mychar == NULL)
                //fprintf(stderr, "\nglobal_mychar is freed!\n");

        //return (mychar(*this));
        return *this;
}

// Using reference will be faster in = operator
mychar& mychar::operator+= (const mychar & rhs)
{
        /*******************************************************/
        // Note : For adding two char strings, first cast mychar
        // as in -
        //aa += (mychar) "cccc" + "dddd";
        /*******************************************************/

        if (& rhs == this)
        {
                debug_("Fatal error: In operator+= rhs is equals 'this' ptr", "ok");
                return *this;
```

```
        }
        this->str_cat(rhs.val);
        return *this;
        //return (mychar(*this));
}

bool mychar::operator== (const mychar & rhs)
{
        return(equalto(rhs.val));
}

bool mychar::operator== (const char *rhs)
{
        return(equalto(rhs));
}

bool mychar::operator!= (const mychar & rhs)
{
        return(equalto(rhs.val, true));
}

bool mychar::operator!= (const char *rhs)
{
        return(equalto(rhs, true));
}
```

제 15 절 첨부 D my_malloc.cpp

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

/*
**      In your main() function put these lines -
                char p_name[1024];
                sprintf(p_name, "PROGRAM_NAME=%s", argv[0]);
                putenv(p_name);
                print_total_memsize(); // in the beginning
                ......
                ......
                print_total_memsize(); // in the end
*/

#include <stdio.h>
```

```
#include <alloc.h>  // for c++ -- malloc, alloc etc...
#include <stdlib.h>  // malloc, alloc..
#include <time.h>  // strftime, localtime, ...
#include <list.h>  // strftime, localtime, ...  see file include/g++/stl_list.h
//#include <debug.h> // debug_("a", a);  debug2_("a", a, true);

#include "my_malloc.h"

const short SAFE_MEM = 10;
const short DATE_MAX_SIZE = 200;

const short MALLOC = 1;
const short REALLOC     = 2;

const short VOID_TYPE =         1;
const short CHAR_TYPE =         2;
const short SHORT_TYPE =        3;
const short INT_TYPE =          4;
const short LONG_TYPE =         5;
const short FLOAT_TYPE =        6;
const short DOUBLE_TYPE =       7;

const char LOG_FILE[30] = "memory_error.log";

// Uncomment this line to debug total mem size allocated...
//#define DEBUG_MEM  "debug_memory_sizes_allocated"

static void raise_error_exit(short mtype, short datatype, char fname[], int lineno);

#ifdef DEBUG
class MemCheck
{
        public:
                MemCheck(void *aptr, size_t amem_size, char fname[], int lineno);
                void    *ptr;
                size_t  mem_size;
                static  list<MemCheck>          mcH;  // list head
                static  unsigned long           total_memsize;  // total memory allocated
};

// Global variables ....
list<MemCheck>          MemCheck::mcH;
unsigned long           MemCheck::total_memsize = 0;

MemCheck::MemCheck(void *aptr, size_t amem_size, char fname[], int lineno)
{
        char func_name[100];
        FILE    *ferr = NULL;
        sprintf(func_name, "MemCheck() - File: %s Line: %d", fname, lineno);
```

```
ferr = fopen(LOG_FILE, "a");
if (ferr == NULL)
{
        fprintf(stdout, "\nWarning: Cannot open file %s\n", LOG_FILE);
        fprintf(stderr, "\nWarning: Cannot open file %s\n", LOG_FILE);
        #ifdef DEBUG_MEM
                exit(-1);
        #else
                return;
        #endif
}

// Search if the pointer already exists in the list...
bool does_exist = false;
list<MemCheck>::iterator iter1; // see file include/g++/stl_list.h
//fprintf(ferr, "\n%s Before checking.. !!\n", func_name);
if (MemCheck::mcH.empty() == true )
{
        //fprintf(ferr, "\n%s List is empty!!\n", func_name);
}
for (iter1 = MemCheck::mcH.begin(); iter1 != MemCheck::mcH.end(); iter1++)
{
        if (iter1 == NULL)
        {
                fprintf(ferr, "\n%s Iterator iter1 is NULL!!\n", func_name);
                break;
        }
        if ( ((*iter1).ptr) == aptr)
        {
                does_exist = true;
                fprintf(ferr, "\n%s Already exists!!\n", func_name);
                fprintf(ferr, "\n%s Fatal Error exiting now ....!!\n", func_name);
                #ifdef DEBUG_MEM
                        exit(-1); //------------------------------------------------------------
                #else
                        return;
                #endif
                // Now change the mem size to new values...
                // For total size - Remove old size and add new size
                //fprintf(ferr, "\n%s total_memsize = %lu\n", func_name, (*iter1).total_memsize
                //fprintf(ferr, "\n%s mem_size = %u\n", func_name, (*iter1).mem_size);
                //fprintf(ferr, "\n%s amem_size = %u\n", func_name, amem_size);
                (*iter1).total_memsize = (*iter1).total_memsize + amem_size;
                if ((*iter1).total_memsize > 0 )
                {
                        if ((*iter1).total_memsize >= (*iter1).mem_size )
                                (*iter1).total_memsize = (*iter1).total_memsize - (*iter1).mem_
                        else
                        {
                                fprintf(ferr, "\n\n%s total_memsize is less than mem_size!!", f
```

```
                                        fprintf(ferr, "\n%s total_memsize = %lu", func_name, (*iter1).t
                                        fprintf(ferr, "\n%s mem_size = %u", func_name, (*iter1).mem_siz
                                        fprintf(ferr, "\n%s amem_size = %u\n", func_name, amem_size);
                                }
                        }
                        (*iter1).mem_size = amem_size;
                }
        }

        // The pointer aptr does not exist in the list, so append it now...
        if (does_exist == false)
        {
                //fprintf(ferr, "\n%s aptr Not found\n", func_name);
                ptr = aptr;
                mem_size = amem_size;
                MemCheck::total_memsize += amem_size;
                MemCheck::mcH.insert(MemCheck::mcH.end(), *this);
        }
        fclose(ferr);
}

static inline void call_check(void *aa, size_t tmpii, char fname[], int lineno)
{
        MemCheck bb(aa, tmpii, fname, lineno);
        if (& bb);  // a dummy statement to avoid compiler warning msg.
}

static inline void remove_ptr(void *aa, char fname[], int lineno)
{
        char    func_name[100];
        if (aa == NULL)
                return;

        sprintf(func_name, "remove_ptr() - File: %s Line: %d", fname, lineno);
        FILE *ferr = NULL;
        ferr = fopen(LOG_FILE, "a");
        if (ferr == NULL)
        {
                fprintf(stdout, "\nWarning: Cannot open file %s\n", LOG_FILE);
                fprintf(stderr, "\nWarning: Cannot open file %s\n", LOG_FILE);
                #ifdef DEBUG_MEM
                        exit(-1);
                #else
                        return;
                #endif
        }

        bool does_exist = false;
        if (MemCheck::mcH.empty() == true)
        {
```

```
                //fprintf(ferr, "\n%s List is empty!!\n", func_name);
                //fclose(ferr);
                //return;
        }
        list<MemCheck>::iterator iter1; // see file include/g++/stl_list.h
        for (iter1 = MemCheck::mcH.begin(); iter1 != MemCheck::mcH.end(); iter1++)
        {
                if (iter1 == NULL)
                {
                        fprintf(ferr, "\n%s Iterator iter1 is NULL!!\n", func_name);
                        break;
                }
                if ( ((*iter1).ptr) == aa)
                {
                        does_exist = true;
                        // Now change the mem size to new values...
                        // For total size - Remove old size
                        //fprintf(ferr, "\n%s total_memsize = %lu\n", func_name, (*iter1).total_memsize
                        //fprintf(ferr, "\n%s mem_size = %u\n", func_name, (*iter1).mem_size);
                        if ((*iter1).total_memsize > 0 )
                        {
                                if ((*iter1).total_memsize >= (*iter1).mem_size )
                                        (*iter1).total_memsize = (*iter1).total_memsize - (*iter1).mem_
                                else
                                {
                                        fprintf(ferr, "\n\n%s total_memsize is less than mem_size!!", fu
                                        fprintf(ferr, "\n%s total_memsize = %lu", func_name, (*iter1).to
                                        fprintf(ferr, "\n%s mem_size = %u\n", func_name, (*iter1).mem_s
                                }
                        }
                        MemCheck::mcH.erase(iter1);
                        break;  // must break to avoid infinite looping
                }
        }
        if (does_exist == false)
        {
                //fprintf(ferr, "\n%s Fatal Error: - You did not allocate memory!! \n", func_name);
                //fprintf(ferr, "\n%s The value passed is %s\n", func_name, (char *) aa);
        }
        else
                //fprintf(ferr, "\n%s found\n", func_name);
        fclose(ferr);
}

static inline void call_free_check(void *aa, char *fname, int lineno)
{
        char func_name[100];
        sprintf(func_name, "call_free_check() - File: %s Line: %d", fname, lineno);

        FILE *ferr = NULL;
```

```
ferr = fopen(LOG_FILE, "a");
if (ferr == NULL)
{
        fprintf(stdout, "\nWarning: Cannot open file %s\n", LOG_FILE);
        fprintf(stderr, "\nWarning: Cannot open file %s\n", LOG_FILE);
        #ifdef DEBUG_MEM
                exit(-1);
        #else
                return;
        #endif
}

bool does_exist = false;
list<MemCheck>::iterator iter1; // see file include/g++/stl_list.h
for (iter1 = MemCheck::mcH.begin(); iter1 != MemCheck::mcH.end(); iter1++)
{
        if (iter1 == NULL)
        {
                fprintf(ferr, "\n%s Iterator iter1 is NULL!!\n", func_name);
                break;
        }
        if ( ((*iter1).ptr) == aa)
        {
                does_exist = true;
                //fprintf(ferr, "\n%s iter1.mem_size = %u\n", func_name, (*iter1).mem_size);
                //fprintf(ferr, "\n%s Total memory allocated = %lu\n",  func_name, (*iter1).tot
                if ((*iter1).total_memsize > 0 )
                {
                        if ((*iter1).total_memsize >= (*iter1).mem_size )
                                (*iter1).total_memsize = (*iter1).total_memsize - (*iter1).mem_
                        else
                        {
                                fprintf(ferr, "\n\n%s total_memsize is less than mem_size!!", f
                                fprintf(ferr, "\n%s total_memsize = %lu", func_name, (*iter1).t
                                fprintf(ferr, "\n%s mem_size = %u", func_name, (*iter1).mem_siz
                        }
                }
                MemCheck::mcH.erase(iter1);
                break;  // must break to avoid infinite looping
        }
}
if (does_exist == false)
{
        fprintf(ferr, "\n%s Fatal Error: free() - You did not allocate memory!!\n",
                        func_name);
        //fprintf(ferr, "\n%s The value passed is %s\n", func_name, (char *) aa);
        fclose(ferr);
        #ifdef DEBUG_MEM
                exit(-1);
        #else
```

```
                        return;
                #endif
        }
        else
        {
                //fprintf(ferr, "\n%s found\n", func_name);
        }
        fclose(ferr);
}

void local_print_total_memsize(char *fname, int lineno)
{
        char func_name[100];
        sprintf(func_name, "local_print_total_memsize() - %s Line: %d", fname, lineno);

        FILE *ferr = NULL;
        ferr = fopen(LOG_FILE, "a");
        if (ferr == NULL)
        {
                fprintf(stdout, "\nWarning: Cannot open file %s\n", LOG_FILE);
                fprintf(stderr, "\nWarning: Cannot open file %s\n", LOG_FILE);
                #ifdef DEBUG_MEM
                        exit(-1);
                #else
                        return;
                #endif
        }

        fprintf(ferr, "\n%s Total memory MemCheck::total_memsize = %lu\n",  func_name, MemCheck::total_
        fclose(ferr);
}
#else  //------------> DEBUG

void local_print_total_memsize(char *fname, int lineno)
{
        // This function is available whether debug or no-debug...
}

#endif // DEBUG

void local_my_free(void *aa, char fname[], int lineno)
{
        if (aa == NULL)
                return;
        call_free_check(aa, fname, lineno);
        free(aa);
        aa = NULL;
}

// size_t is type-defed unsigned long
```

```
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;
}

// size_t is type-defed unsigned long
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;
}

// size_t is type-defed unsigned long
short *local_my_realloc(short *aa, size_t size, char fname[], int lineno)
{
        remove_ptr(aa, fname, lineno);
        unsigned long tmpqq = size + SAFE_MEM;
        size_t  tmpii = sizeof (short) * (tmpqq);
        aa = (short *) realloc(aa, tmpii);
        if (aa == NULL)
                raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);
        // do not memset!! memset(aa, 0, tmpii);
        // Not for numbers!! aa[tmpqq-1] = 0;
        call_check(aa, tmpii, fname, lineno);
        return aa;
}
```

```
// size_t is type-defed unsigned long
int *local_my_realloc(int *aa, size_t size, char fname[], int lineno)
{
        remove_ptr(aa, fname, lineno);
        unsigned long tmpqq = size + SAFE_MEM;
        size_t  tmpii = sizeof (int) * (tmpqq);
        aa = (int *) realloc(aa, tmpii);
        if (aa == NULL)
                raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);
        // do not memset!! memset(aa, 0, tmpii);
        // Not for numbers!! aa[tmpqq-1] = 0;
        call_check(aa, tmpii, fname, lineno);
        return aa;
}

// size_t is type-defed unsigned long
long *local_my_realloc(long *aa, size_t size, char fname[], int lineno)
{
        remove_ptr(aa, fname, lineno);
        unsigned long tmpqq = size + SAFE_MEM;
        size_t  tmpii = sizeof (long) * (tmpqq);
        aa = (long *) realloc(aa, tmpii);
        if (aa == NULL)
                raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);
        // do not memset!! memset(aa, 0, tmpii);
        // Not for numbers!! aa[tmpqq-1] = 0;
        call_check(aa, tmpii, fname, lineno);
        return aa;
}

// size_t is type-defed unsigned long
float *local_my_realloc(float *aa, size_t size, char fname[], int lineno)
{
        remove_ptr(aa, fname, lineno);
        unsigned long tmpqq = size + SAFE_MEM;
        size_t  tmpii = sizeof (float) * (tmpqq);
        aa = (float *) realloc(aa, tmpii);
        if (aa == NULL)
                raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);
        // do not memset!! memset(aa, 0, tmpii);
        // Not for numbers!! aa[tmpqq-1] = 0;
        call_check(aa, tmpii, fname, lineno);
        return aa;
}

// size_t is type-defed unsigned long
double *local_my_realloc(double *aa, size_t size, char fname[], int lineno)
{
        remove_ptr(aa, fname, lineno);
```

```
        unsigned long tmpqq = size + SAFE_MEM;
        size_t  tmpii = sizeof (double) * (tmpqq);
        aa = (double *) realloc(aa, tmpii);
        if (aa == NULL)
                raise_error_exit(REALLOC, CHAR_TYPE, fname, lineno);
        // do not memset!! memset(aa, 0, tmpii);
        // Not for numbers!! aa[tmpqq-1] = 0;
        call_check(aa, tmpii, fname, lineno);
        return aa;
}

static void raise_error_exit(short mtype, short datatype, char fname[], int lineno)
{
        if (mtype == MALLOC)
        {
                fprintf(stdout, "\nFatal Error: malloc() failed!!");
                fprintf(stderr, "\nFatal Error: malloc() failed!!");
        }
        else
        if (mtype == REALLOC)
        {
                fprintf(stdout, "\nFatal Error: realloc() failed!!");
                fprintf(stderr, "\nFatal Error: realloc() failed!!");
        }
        else
        {
                fprintf(stdout, "\nFatal Error: mtype not supplied!!");
                fprintf(stderr, "\nFatal Error: mtype not supplied!!");
                exit(-1);
        }

        // Get current date-time and print time stamp in error file...
        char date_str[DATE_MAX_SIZE + SAFE_MEM];
        time_t tt;
        tt = time(NULL);
        struct tm *ct = NULL;
        ct = localtime(& tt); // time() in secs since Epoch 1 Jan 1970
        if (ct == NULL)
        {
                fprintf(stdout, "\nWarning: Could not find the local time, localtime() failed\n");
                fprintf(stderr, "\nWarning: Could not find the local time, localtime() failed\n");
        }
        else
                strftime(date_str, DATE_MAX_SIZE , "%C", ct);

        FILE *ferr = NULL;
        char    filename[100];
        strcpy(filename, LOG_FILE);
        ferr = fopen(filename, "a");
        if (ferr == NULL)
```

```
{
        fprintf(stdout, "\nWarning: Cannot open file %s\n", filename);
        fprintf(stderr, "\nWarning: Cannot open file %s\n", filename);
}
else
{
        // ************************************************
        // ******* Do putenv in the main() function *********
        //              char p_name[1024];
        //              sprintf(p_name, "PROGRAM_NAME=%s", argv[0]);
        //              putenv(p_name);
        // ************************************************
        char    program_name[200+SAFE_MEM];
        if (getenv("PROGRAM_NAME") == NULL)
        {
                fprintf(ferr, "\n%sWarning: You did not putenv() PROGRAM_NAME env variable in ma
                                date_str);
                program_name[0] = 0;
        }
        else
                strncpy(program_name, getenv("PROGRAM_NAME"), 200);

        if (mtype == MALLOC)
                fprintf(ferr, "\n%s: %s - Fatal Error - my_malloc() failed.", date_str, program_
        else
        if (mtype == REALLOC)
        {
                fprintf(ferr, "\n%s: %s - Fatal Error - my_realloc() failed.", date_str, progra
                char dtype[50];
                switch(datatype)
                {
                        case VOID_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case CHAR_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case SHORT_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case INT_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case LONG_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case FLOAT_TYPE:
                                strcpy(dtype, "char*");
                                break;
                        case DOUBLE_TYPE:
```

```
                                        strcpy(dtype, "char*");
                                        break;
                                default:
                                        strcpy(dtype, "none*");
                                        break;
                        }
                        fprintf(ferr, "\n%s %s - Fatal Error: %s realloc() failed!!", date_str, program
                }

                fprintf(ferr, "\n%s %s - Very severe error condition. Exiting application now....",
                                        date_str, program_name);
                fclose(ferr);
        }

        exit(-1);
}
```

제 16 절 첨부 E my_malloc.h

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

/*
**      In your main() function put -
                char p_name[1024];
                sprintf(p_name, "PROGRAM_NAME=%s", argv[0]);
                putenv(p_name);
                print_total_memsize(); // in the beginning
                ......
                ......
                print_total_memsize(); // in the end
*/

/* Use zap instead of delete as this will be very clean!!
** Use do while to make it robust and bullet-proof macro
*/
#define zap(x) do { if (x) { delete(x); x = 0; } } while (0)

void *local_my_malloc(size_t size, char fname[], int lineno);

char *local_my_realloc(char *aa, size_t size, char fname[], int lineno);
short *local_my_realloc(short *aa, size_t size, char fname[], int lineno);
void local_my_free(void *aa, char fname[], int lineno);
```

```
void local_print_total_memsize(char fname[], int lineno);

#define my_free(NM) (void) (local_my_free(NM, __FILE__, __LINE__))
#define my_malloc(SZ) (local_my_malloc(SZ, __FILE__, __LINE__))
#define my_realloc(NM, SZ) (local_my_realloc(NM, SZ, __FILE__, __LINE__))
#define print_total_memsize() (void) (local_print_total_memsize(__FILE__, __LINE__))

#ifdef DEBUG  //------------> DEBUG
#else  //------------> DEBUG
#define call_check(AA, BB, CC, DD) ((void) 0)
#define call_free_check(AA, BB, CC) ((void) 0)
#define remove_ptr(AA, CC, DD) ((void) 0)
#endif //------------> DEBUG
```

제 17 절 첨부 F debug.h

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

#define print_log(AA, BB, CC, DD, EE) ((void) 0)

#ifdef DEBUG

#include <iostream>
#include <string>
//#include <assert.h>  // assert() macro which is also used for debugging

const bool LOG_YES = true;  // print output to log file
const bool LOG_NO = false;  // Do not print output to log file

// Debugging code
// Use debug2_ to output result to a log file

#define debug_(NM, VL) (void) ( local_dbg(NM, VL, __FILE__, __LINE__) )
#define debug2_(NM, VL, LOG_FILE) (void) ( local_dbg(NM, VL, __FILE__, __LINE__, LOG_FILE) )

void local_dbg(char name[], char value[], char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], string value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], int value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], unsigned long value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], float value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], double value, char fname[], int lineno, bool logfile= false);
```

```
#else  //--------> else

#define debug_(NM, VL) ((void) 0)
#define debug2_(NM, VL, LOG_FILE) ((void) 0)

#endif // DEBUG
```

제 18 절 첨부 G debug.cpp

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
//*****************************************************************
//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
//포함시켜야 한다.
//저자: Al Dev  이메일: alavoor@yahoo.com
//*****************************************************************

#ifdef DEBUG  // ONLY if DEBUG is defined then these functions below are needed

#include "debug.h"
//#include "log.h"

// Variable value[] can be char, string, int, unsigned long, float, etc...

void local_dbg(char name[], char value[], char fname[], int lineno, bool logfile) {
        if (value == NULL)
                return;
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %s\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], string value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %s\n", fname, lineno, name, value.c_str());
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], int value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %d\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], unsigned int value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %u\n", fname, lineno, name, value);
```

```
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], long value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %d\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], unsigned long value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %u\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], short value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %d\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], unsigned short value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %u\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], float value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %f\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

void local_dbg(char name[], double value, char fname[], int lineno, bool logfile) {
        if (logfile == true)
                print_log("\nDebug %s : Line: %d %s is = %f\n", fname, lineno, name, value);
        else
                cout << "\nDebug " << fname << ": Line: " << lineno << " " << name << " is = " << value

// You add many more here - value can be a class, ENUM, datetime, etc...


#endif // DEBUG
```

제 19 절 첨부 H Makefile

Download mychar 에서 모든 프로그램들을 하나의 tar.gz 화일로 다운로드받을 수 있다. 이 화일을 받으려면 웹브라우저에서 이 화일을 'Text' 타입으로 저장한다.

```
#//*****************************************************************
#//저작권은 GNU/GPL 에 있지만 저자의 이름과 이메일은 모든 복사본에
#//포함시켜야 한다.
#//저자: Al Dev  이메일: alavoor@yahoo.com
#//*****************************************************************

.SUFFIXES: .pc .cpp .c .o

CC=gcc
CXX=g++

MAKEMAKE=mm
LIBRARY=libmychar.a
DEST=/home/myname/lib

# To build the library, and main test program  uncomment line below :-
#MYCFLAGS=-O -Wall

# To test without debug trace uncomment line below:-
#MYCFLAGS=-g3 -Wall

# To enable 'full debug ' tracing uncomment line below:-
MYCFLAGS=-g3 -DDEBUG -Wall

#PURIFY=purify -best-effort

SRCS=my_malloc.cpp mychar.cpp debug.cpp example_mychar.cpp
HDR=my_malloc.h  mychar.h debug.h
OBJS=my_malloc.o mychar.o debug.o example_mychar.o
EXE=mychar

# For generating makefile dependencies..
SHELL=/bin/sh

CPPFLAGS=$(MYCFLAGS) $(OS_DEFINES)
CFLAGS=$(MYCFLAGS) $(OS_DEFINES)

#
# If the libmychar.a is in the current
# directory then use -L. (dash L dot)
MYLIBDIR=-L$(MY_DIR)/libmy -L.

ALLLDFLAGS= $(LDFLAGS)  $(MYLIBDIR)

COMMONLIBS=-lstdc++ -lm
MYLIBS=-lmychar
LIBS=$(COMMONLIBS)  $(MYLIBS)

all: $(LIBRARY) $(EXE)
```

```
$(MAKEMAKE):
        @rm -f $(MAKEMAKE)
        $(PURIFY) $(CXX) -M  $(INCLUDE) $(CPPFLAGS) *.cpp > $(MAKEMAKE)

$(EXE): $(OBJS)
        @echo "Creating a executable "
        $(PURIFY) $(CC) -o $(EXE) $(OBJS) $(ALLLDFLAGS) $(LIBS)

$(LIBRARY): $(OBJS)
        @echo "\n**********************************************"
        @echo "   Loading $(LIBRARY) ... to $(DEST)"
        @echo "**********************************************"
        @ar cru $(LIBRARY) $(OBJS)
        @echo "\n "

.cpp.o: $(SRCS) $(HDR)
#       @echo "Creating a object files from " $*.cpp " files "
        $(PURIFY) $(CXX) -c  $(INCLUDE) $(CPPFLAGS) $*.cpp

.c.o: $(SRCS) $(HDR)
#       @echo "Creating a object files from " $*.c " files "
        $(PURIFY) $(CC) -c $(INCLUDE) $(CFLAGS) $*.c

clean:
        rm -f *.o *.log *~ *.log.old *.pid core err a.out lib*.a afiedt.buf
        rm -f $(EXE)
        rm -f $(MAKEMAKE)

#%.d: %.c
#       @echo "Generating the dependency file *.d from *.c"
#       $(SHELL) -ec '$(CC) -M $(CPPFLAGS) $< | sed '\''s/$*.o/& $@/g'\'' > $@'
#%.d: %.cpp
#       @echo "Generating the dependency file *.d from *.cpp"
#       $(SHELL) -ec '$(CC) -M $(CPPFLAGS) $< | sed '\''s/$*.o/& $@/g'\'' > $@'

# Must include all the c flags for -M option
#$(MAKEMAKE):
#       @echo "Generating the dependency file *.d from *.cpp"
#       $(CXX) -M  $(INCLUDE) $(CPPFLAGS) *.cpp > $(MAKEMAKE)

include $(MAKEMAKE)
#include $(SRCS:.cpp=.d)
#include $(SRCS:.c=.d)
```