

C++ for Scientific Computation

Rajeev Singh

Contents

1	Day 1: Basic input/output	5
1.1	aa_hello_world.c	5
1.2	ab_hello_world.cpp	6
1.3	ac_hello_world.cpp	7
1.4	ad_powers_of_integer.cpp	8
1.5	ae_powers_of_real.cpp	9
2	Day 2: Pointers/References, Arithmetic/Logical	10
2.1	af_pointer.cpp	10
2.2	ag_reference.cpp	11
2.3	ah_arithmetic_operators.cpp	12
2.4	ai_relational_logical.cpp	13
3	Day 3: Scope, Conditional, Loops	14
3.1	aj_blocks_scope.cpp	14
3.2	ak_scope.cpp	15
3.3	al_if_else.cpp	16
3.4	am_for_loop.cpp	17
3.5	an_while_loop.cpp	18
3.6	ao_do_while_loop.cpp	19
3.7	ap_break.cpp	20
3.8	aq_break_nested_loop.cpp	21
3.9	ar_break_all_loops.cpp	22
4	Day 4: Functions, Call by value/reference	23
4.1	as_function_square.cpp	23
4.2	at_function_factorial.cpp	24
4.3	au_function_call_by_value.cpp	25
4.4	av_function_call_by_reference.cpp	26
4.5	aw_function_call_by_reference_using_pointers.cpp	27
4.6	ax_function_multiple_return_values.cpp	28

5	Day 5: Functions- default args, function pointers; Arrays	29
5.1	ay_function_default_arguments.cpp	29
5.2	az_function_inline.cpp	30
5.3	ba_function_pointers.cpp	31
5.4	bb_function_pointers_as_arguments.cpp	32
5.5	bc_static_variables.cpp	33
5.6	bd_array.cpp	34
5.7	be_function_with_array_argument.cpp	35
5.8	bf_multidimensional_arrays.cpp	36
5.9	bg_array_and_pointer.cpp	37
6	Day 6: Dynamic memory, Multidimensional Array, BLAS	38
6.1	bh_dynamic_memory.cpp	38
6.2	bi_dynamic_array_size_input.cpp	39
6.3	bj_multidimensional_array_with_pointer.cpp	40
6.4	bk_multidimensional_dynamic_array.cpp	41
6.5	bl_multidimensional_dynamic_array_size_input.cpp	42
6.6	bm_multidimensional_array_with_mapping.cpp	43
7	Day 7: String, Advanced Datatypes, BLAS, Sparse Matrices	44
7.1	bn_strings.cpp	44
7.2	bo_typedef.cpp	45
7.3	bp_struct.cpp	46
7.4	bq_struct_pointer.cpp	48
7.5	br_struct_array.cpp	50
8	Day 9: Modules and Namespaces, Classes	52
8.1	bs_struct_with_functions.cpp	53
8.2	bt_struct_constructor_destructor.cpp	55
9	Day 10: Classes Continued	57
9.1	bu_struct_this_pointer.cpp	57
9.2	bv_copy_constructor.cpp	59
9.3	bw_default_copy_constructor.cpp	61
9.4	bx_struct_with_const_functions.cpp	63
9.5	by_struct_visibility.cpp	65
9.6	bz_safely_changing_size_of_array.cpp	67
10	Day 11: Classes Continued, Templates	69
10.1	ca_operator_overloading.cpp	69
10.2	cb_templates.cpp	71
11	Day 14: STL	72
11.1	cc_list.cpp	72
11.2	cd_vector.cpp	73
11.3	ce_valarray.cpp	74
11.4	cf_complex_numbers.cpp	75

11.5	cg_auto_pointer.cpp	76
12	Day 15: Boost	77
12.1	ch_boost_array.cpp	77
12.2	ci_boost_multi_array.cpp	78
12.3	cj_boost_mulptprecision_cpp_int.cpp	79
12.4	ck_boost_mulptprecision_gmp.cpp	80
12.5	cl_boost_mulptprecision_cpp_int_2.cpp	81
12.6	cm_boost_random_uniform.cpp	83
12.7	cn_boost_ublas_vector.cpp	84
12.8	co_boost_ublas_unit_vector.cpp	85
12.9	cp_boost_ublas_zero_vector.cpp	86
12.10	cq_boost_ublas_scalar_vector.cpp	87
12.11	cr_boost_ublas_sparse_vector_mapped.cpp	88
12.12	cs_boost_ublas_sparse_vector_compressed.cpp	89
12.13	ct_boost_ublas_sparse_vector_coordinate.cpp	90
13	Day 16: Boost Continued	91
13.1	cu_boost_ublas_vector_expressions_conj_etc.cpp	91
13.2	cv_boost_ublas_vector_expressions_binary.cpp	92
13.3	cw_boost_ublas_vector_expressions_outer_prod.cpp	93
13.4	cx_boost_ublas_vector_expressions_scalar_multi.cpp	94
13.5	cy_boost_ublas_vector_expressions_reductions.cpp	95
13.6	cz_boost_ublas_vector_expressions_inner_prod.cpp	96
13.7	da_boost_ublas_matrix_basic.cpp	97
13.8	db_boost_ublas_matrix_special.cpp	98
13.9	dc_boost_ublas_triangular_matrix.cpp	99
13.10	dd_boost_ublas_symmetric_matrix.cpp	100
13.11	de_boost_ublas_hermitian_matrix.cpp	101
13.12	df_boost_ublas_banded_matrix.cpp	102
13.13	dg_boost_ublas_sparse_matrix.cpp	103
13.14	dh_boost_ublas_matrix_expressions_conj_etc.cpp	104
13.15	di_boost_ublas_matrix_expressions_binary.cpp	105
13.16	dj_boost_ublas_matrix_expressions_scalar_multi.cpp	106
13.17	dk_boost_ublas_matrix_expressions_matrix_vector_multi.cpp	107
13.18	dl_boost_ublas_matrix_expressions_matrix_vector_triangular_solver.cpp	108
13.19	dm_boost_ublas_matrix_expressions_matrix_matrix_multi.cpp	109
13.20	dn_boost_ublas_matrix_expressions_matrix_matrix_triangular_solver.cpp	110
14	Day 17: Boost Continued	111
14.1	do_mpi.cpp	111
14.2	dp_boost_mpi_1.cpp	112
14.3	dq_boost_mpi_point_to_point.cpp	113
14.4	dr_boost_mpi_non_blocking.cpp	114
14.5	ds_boost_mpi_broadcast.cpp	115
14.6	dt_boost_mpi_gather.cpp	116

14.7	<code>du_boost_mpi_reduce.cpp</code>	117
14.8	<code>dv_boost_mpi_reduce_2.cpp</code>	118
14.9	<code>dw_boost_python_hello_world.cpp</code>	119
14.10	<code>dx_boost_python_exposing_classes.cpp</code>	120
15	Day 18: MTL	121
15.1	<code>dy_mtl_vector1.cpp</code>	121
15.2	<code>dz_mtl_vector2.cpp</code>	122
15.3	<code>ea_mtl_dense2D.cpp</code>	123
15.4	<code>eb_mtl_morton_dense.cpp</code>	124
15.5	<code>ec_mtl_compressed2D.cpp</code>	125
15.6	<code>ed_mtl_element_structure.cpp</code>	126
15.7	<code>ee_mtl_multi_vector.cpp</code>	128
15.8	<code>ef_mtl_insert.cpp</code>	129
15.9	<code>eg_mtl_insert_scope.cpp</code>	131
15.10	<code>eh_mtl_element_matrix.cpp</code>	132
15.11	<code>ei_mtl_insert_class_expensive.cpp</code>	134
15.12	<code>ej_mtl_insert_scope.cpp</code>	135
15.13	<code>ek_mtl_array_initialization.cpp</code>	137
15.14	<code>el_mtl_vector_expr.cpp</code>	138
15.15	<code>em_mtl_rich_vector_expr.cpp</code>	139
15.16	<code>en_mtl_matrix_expressions1.cpp</code>	140
15.17	<code>eo_mtl_matrix_expressions2.cpp</code>	141
15.18	<code>ep_mtl_matrix_expressions3.cpp</code>	142
15.19	<code>eq_mtl_matrix_vector1.cpp</code>	143
15.20	<code>er_mtl_matrix_vector2.cpp</code>	144

1 Day 1: Basic input/output

1.1 aa_hello_world.c

```
// C program to print "Hello World".  
//  
// Rajeev Singh  
// 2013-03-27  
  
#include <stdio.h>  
  
int main() {  
    printf("Hello World from C\n");  
    return 0;  
}
```

1.2 ab_hello_world.cpp

```
// C++ program to print "Hello World".
//
// Rajeev Singh
// 2013-03-27

#include <iostream>

int main() {
    std::cout << "Hello World from C++"; // << std::endl;
    return 0;
}
```

1.3 ac_hello_world.cpp

```
// C++ program to print "Hello World".  
//  
// Rajeev Singh  
// 2013-03-27  
  
#include <iostream>  
  
using namespace std;  
  
int main() {  
    cout << "Hello World from C++" << endl;  
    return 0;  
}
```

1.4 ad_powers_of_integer.cpp

```
// Program to calculate powers of given integer.
//
// Rajeev Singh
// 2013-03-27

#include <iostream>
#include <cmath>
using namespace std;

int main() {
    //int given_number;
    long int given_number;
    cout << "Enter an integer: ";
    cin >> given_number;

    cout << "Given number = " << given_number << endl
         << "Square      = " << pow(given_number,2) << endl
         << "Cube        = " << pow(given_number,3) << endl
         << "Forth power  = " << pow(given_number,4) << endl;

    return 0;
}
```


1.5 ae_powers_of_real.cpp

```
// Program to calculate powers of given integer.
//
// Rajeev Singh
// 2013-03-27

#include <iostream>
#include <cmath>
using namespace std;

int main() {
    double given_number;
    //long double given_number;
    cout << "Enter a real number : ";
    cin  >> given_number;

    cout << "Given number = " << given_number << endl
         << "Square       = " << pow(given_number,2) << endl
         << "Square root  = " << pow(given_number,1./2) << endl
         << "Cube        = " << pow(given_number,3) << endl
         << "Forth power  = " << pow(given_number,4) << endl;

    return 0;
}
```

2 Day 2: Pointers/References, Arithmetic/Logical

2.1 af_pointer.cpp

```
// Program to illustrate pointers.
//
// Rajeev Singh
// 2013-03-28

#include <iostream>
using namespace std;

int main() {
    int *np = NULL;
    int n = 10;

    cout << "Initial" << endl
         << "n    = " << n    << endl
         << "np   = " << np   << endl
         << "*np  = " << "since np is NULL, printing *np gives segmentation fault"
    << endl << endl;

    np = &n;
    cout << "After: np = &n" << endl
         << "n    = " << n    << endl
         << "np   = " << np   << endl
         << "*np  = " << *np  << endl << endl;

    *np = 22;
    cout << "After: *np = 22" << endl
         << "n    = " << n    << endl
         << "np   = " << np   << endl
         << "*np  = " << *np  << endl << endl;

    return 0;
}
```

2.2 ag_reference.cpp

```
// Program to illustrate the use of references (special pointers).
//
// Rajeev Singh
// 2013-03-28

#include <iostream>
using namespace std;

int main() {
    int    n = 5;
    int & r = n;
    int    m;

    cout << "Initial" << endl
         << "n = " << n << endl
         << "r = " << r << endl
         << "m = " << m << endl << endl;

    m = r + 3;      // m == n + 3
    cout << "After: m = r + 3" << endl
         << "n = " << n << endl
         << "r = " << r << endl
         << "m = " << m << endl << endl;

    r = m;          // r still points to n and n == m
    cout << "After: r = m" << endl
         << "n = " << n << endl
         << "r = " << r << endl
         << "m = " << m << endl << endl;

    m = 0;          // r and n are unchanged
    cout << "After: m = 0" << endl
         << "n = " << n << endl
         << "r = " << r << endl
         << "m = " << m << endl << endl;

    int & s = m;
    r = s;          // r still points to n and n == m (== 0)
    cout << "After: r = s where s is new reference to m" << endl
         << "n = " << n << endl
         << "r = " << r << endl
         << "m = " << m << endl << endl;

    return 0;
}
```

2.3 ah_arithmetic_operators.cpp

```
// Program to illustrate basic arithmetic operators.
//
// Rajeev Singh
// 2013-03-28

#include <iostream>
using namespace std;

int main() {
    int m = 100,
        n = 200;

    cout << "Initial" << endl
         << "m = " << m << endl
         << "n = " << n << endl
         << "m + n = " << m + n << endl
         << "m - n = " << m - n << endl
         << "m * n = " << m * n << endl
         << "m / n = " << m / n << endl
         << "m % n = " << m % n << endl << endl;

    //m = m + 200;
    m += 200;      // both this commands are same
    cout << "After: m += 200" << endl
         << "m = " << m << endl
         << "n = " << n << endl
         << "m + n = " << m + n << endl
         << "m - n = " << m - n << endl
         << "m * n = " << m * n << endl
         << "m / n = " << m / n << endl
         << "m % n = " << m % n << endl << endl;

    m++;
    cout << "After: m++" << endl
         << "m = " << m << endl
         << "n = " << n << endl
         << "m + n = " << m + n << endl
         << "m - n = " << m - n << endl
         << "m * n = " << m * n << endl
         << "m / n = " << m / n << endl
         << "m % n = " << m % n << endl << endl;

    return 0;
}
```

2.4 ai_relational_logical.cpp

```
// program to illustrate logical and relational operators.
//
// Rajeev Singh
// 2013-03-28

#include <iostream>
using namespace std;

int main() {
    int    x = 2;
    int    y = 4;
    int    z = 4;
    bool    b;

    cout << "x = " << x << endl
         << "y = " << y << endl
         << "z = " << z << endl << endl;

    // z == 4 is not tested
    b = ( x == 2 && y == 3 && z == 4 );
    cout << "b = ( x == 2 && y == 3 && z == 4 )" << endl
         << "b = " << b << endl << endl;

    // only x == 2 is tested
    b = ( x == 2 || y == 3 || z == 4 );
    cout << "b = ( x == 2 || y == 3 || z == 4 )" << endl
         << "b = " << b << endl << endl;

    // correct, since x != 0 in "y/x"
    b = ( x != 0 && y/x > 1 );
    cout << "b = ( x != 0 && y/x > 1 )" << endl
         << "b = " << b << endl << endl;

    return 0;
}
```

3 Day 3: Scope, Conditional, Loops

3.1 aj_blocks_scope.cpp

```
// program to illustrate blocks.
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    { // block 1
        int n1 = 1;
        double f1 = 0.0;
        cout << "Block 1 " << endl;
        cout << "n1 = " << n1 << endl;
        cout << "f1 = " << f1 << endl;
    }

    { // block 2
        int n1 = 2;
        // n1 has value 2 in this block
        cout << "Block 2 " << endl;
        cout << "n1 = " << n1 << endl;

        //int n1 = 5; // ERROR
    }

    return 0;
}
```

3.2 ak_scope.cpp

```
// program to illustrate scope of variables
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    { // block 1
        int m, n1 = 1;
        { // block 1.1
            int n2 = 2;
            { // block 1.1.1
                m = n1 + n2; // evaluates to m = 3
                cout << "Block 1.1.1: m = " << m << endl;
            }
        }

        { // block 1.2
            int n2 = 3;
            m = n1 + n2; // evaluates to m = 4
            cout << "Block 1.2 : m = " << m << endl;
        }
    }

    return 0;
}
```

3.3 al_if_else.cpp

```
// program to illustrate conditional structure
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int n = 1;

    if ( n > 0 )
    {
        n = n / n;
    }

    if ( n < 0 ) {
        n += 5; // NOTE: trivial block!
        cout << "hello " << n << endl;
    }
    else if ( n %2 == 0 ) {
        n += 1;
        cout << "hello " << n << endl;
    }
    else {
        n -= 6;
        cout << "hello " << n << endl;
    }

    cout << "n = " << n << endl;

    return 0;
}
```


3.4 am_for_loop.cpp

```
// program to illustrate for loop
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int n = 1;

    for (int i=1; i<10; i++) {
        if (i>5) {
            n *= i;
            cout << "n = " << n << endl;
        }
    }

    return 0;
}
```

3.5 an_while_loop.cpp

```
// program to illustrate while loop
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int n = 1;
    int i = 1;

    while (i < 10) {
        n *= i;
        i++;
        cout << "n = " << n << endl;
    }

    return 0;
}
```

3.6 ao_do_while_loop.cpp

```
// program to illustrate do-while loop
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int n = 1;
    int i = 100;

    do {
        n *= i;
        i++;
        cout << "n = " << n << endl;
    } while (i < 10);

    return 0;
}
```

3.7 ap_break.cpp

```
// program to illustrate use of break
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int n = 1;

    for ( int i = 1; i < 20; i++ ) {
        // avoid overflow
        if ( n > 21474836 )
            break;
        n *= i;
        cout << "n = " << n << endl;
    }

    return 0;
}
```

3.8 aq_break_nested_loop.cpp

```
// program to illustrate behavior of break in nested loops
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    for ( int i = 1; i < 20; i++ ) {
        int n = 1;
        for ( int j = 1; j < i; j++ ) {
            if ( n > 21474836 )
                break;
            n *= j;
        }

        cout << "n = " << n << endl;
    }

    return 0;
}
```

3.9 ar_break_all_loops.cpp

```
// program to illustrate breaking all nested loops
//
// Rajeev Singh
// 2013-03-29

#include <iostream>
using namespace std;

int main() {
    int flag = 0;
    for ( int i = 1; i < 20; i++ ) {
        int n = 1;
        for ( int j = 1; j < i; j++ ) {
            if ( n > 21474836 ) {
                flag = 1;
                break;
            }
            n *= j;
        }
        if (flag == 1)
            break;

        cout << "n = " << n << endl;
    }

    return 0;
}
```

4 Day 4: Functions, Call by value/reference

4.1 as_function_square.cpp

```
// program to illustrate defining function
//
// Rajeev Singh
// 2013-03-31

#include <iostream>
using namespace std;

double
square (const double x) {
    return x*x;
}

int main() {
    double a = 2.5;

    cout << "a    = " << a << endl
          << "a^2 = " << square(a) << endl;

    return 0;
}
```

4.2 at_function_factorial.cpp

```
// program to illustrate defining the factorial function
//
// Rajeev Singh
// 2013-03-31

#include <iostream>
using namespace std;

int
factorial (const int n) {
    int f = 1;
    for (int i = 1; i <= n; i++ )
        f *= i;

    return f;
}

int main() {
    int m = 10;

    cout << "m   = " << m << endl
         << "m!  = " << factorial(m) << endl;

    return 0;
}
```


4.3 au_function_call_by_value.cpp

```
// program to illustrate call by value feature
//
// Rajeev Singh
// 2013-03-31

#include <iostream>
using namespace std;

int
f (int n) {
    n = 10;
    return n;
}

int main() {
    int m = 5;

    cout << "m before calling = " << m << endl;
    cout << "function output   = " << f(m) << endl;
    cout << "m after calling   = " << m << endl;

    return 0;
}
```

4.4 av_function_call_by_reference.cpp

```
// program to illustrate call by reference feature
//
// Rajeev Singh
// 2013-03-31

#include <iostream>
using namespace std;

int
f (int & n) {
    n = 10;
    return n;
}

int main() {
    int m = 5;

    cout << "m before calling = " << m << endl;
    cout << "funtion output    = " << f(m) << endl;
    cout << "m after calling   = " << m << endl;

    return 0;
}
```

4.5 aw_function_call_by_reference_using_pointers.cpp

```
// program to illustrate call by reference feature using general  
// pointers  
//  
// Rajeev Singh  
// 2013-03-31  
  
#include <iostream>  
using namespace std;  
  
int  
f (int * n) {  
    *n = 10;  
    return *n;  
}  
  
int main() {  
    int m = 5;  
  
    cout << "m before calling = " << m << endl;  
    cout << "funtion output    = " << f(&m) << endl;  
    cout << "m after calling   = " << m << endl;  
  
    return 0;  
}
```

4.6 ax_function_multiple_return_values.cpp

```
// program to illustrate funtions with multiple return values using
// call by reference
//
// Rajeev Singh
// 2013-03-31

#include <iostream>
using namespace std;

void
min_max ( const int n1, const int n2,
          int & min, int & max );

int main() {
    int m1, m2, min, max;

    cout << "Enter two integers :";
    cin >> m1 >> m2;

    min_max(m1, m2, min, max);
    cout << "m1          = " << m1 << endl
         << "m2          = " << m2 << endl << endl
         << "min(m1,m2) = " << min << endl
         << "max(m1,m2) = " << max << endl;

    return 0;
}

void
min_max ( const int n1, const int n2,
          int & min, int & max ) {
    if ( n1 < n2 ) {
        min = n1;
        max = n2;
    }
    else {
        min = n2;
        max = n1;
    }
}
```

5 Day 5: Functions- default args, function pointers; Arrays

5.1 ay_function_default_arguments.cpp

```
// program to illustrate function with default arguments
//
// Rajeev Singh
// 2013-04-01

#include <iostream>
using namespace std;

double
square (const double x = 10.0) {
    return x*x;
}

int main() {
    double a = 2.5;

    cout << "a          = " << a << endl;
    cout << "a^2        = " << square(a) << endl;
    cout << "square() = " << square() << endl;

    return 0;
}
```

5.2 az_function_inline.cpp

```
/* program to illustrate inline functions
 *
 * WARNING: do not inline functions with large bodies. it can cause
 *          the final executable to be very large in size and decrease
 *          performance.
 *
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

inline double
square (const double x = 10.0) {
    return x*x;
}

int main() {
    double a = 2.5;

    cout << "a          = " << a << endl;
    cout << "a^2        = " << square(a) << endl;
    cout << "square() = " << square() << endl;

    return 0;
}
```

5.3 ba_function_pointers.cpp

```
/* program to illustrate function pointers
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

double
square (const double x) {
    return x*x;
}

int main() {
    double a = 2.5;
    double (* pf) (const double x);
    pf = square;

    cout << "a          = " << a << endl;
    cout << "square(a) = " << square(a) << endl;
    cout << "pf(a)      = " << pf(a) << endl;

    return 0;
}
```

5.4 bb_function_pointers_as_arguments.cpp

```
/* program to illustrate function pointers as arguments
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

double
square (const double x) {
    return x*x;
}

double
cube (const double x) {
    return x*x*x;
}

double
f ( double ( * func ) ( const double x ),
    const double x ) {
    return func( x );
}

int main() {
    double a = 2.5;

    cout << "a                = " << a << endl;
    cout << "f(square, a) = " << f(square, a) << endl;
    cout << "f(cube, a) = " << f(cube, a) << endl;

    return 0;
}
```


5.5 bc_static_variables.cpp

```
/* program to illustrate static variables
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

double
f ( const double x, long & cnt ) {
    static long counter = 0; // allocated and initialised
                           // once per program

    cnt = ++counter;
    return 2.0*x*x - x;
}

int main() {
    long cnt = 0;

    for ( double x = -10; x <= 10.0; x += 0.1 )
        f( x, cnt );

    cout << "num times f called = " << cnt << endl;

    return 0;
}
```

5.6 bd_array.cpp

```
/* program to illustrate array
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

int main() {
    double f[5];

    for ( int i = 0; i < 5; i++ )
        f[i] = 2*i;

    cout << "f = " << f << endl;
    for ( int i = 0; i < 5; i++ )
        cout << "f[" << i << "] = " << f[i] << endl;

    cout << "f[5] = " << f[5] << endl; // bug but program still compiles
    // if you lucky such bugs will be detected by segmentation fault

    return 0;
}
```

5.7 be_function_with_array_argument.cpp

```
/* program to illustrate arrays as function arguments
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

void
copy ( const double x[3], double y[3] ) {
    for ( int i = 0; i < 3; i++ )
        y[i] = x[i];
}

void
add ( const double x[3], double y[3] ) {
    for ( int i = 0; i < 3; i++ )
        y[i] += x[i];
}

int main() {
    double a[3],
           b[] = {0, 0, 0}; // b is automaticall of size 3

    for ( int i = 0; i < 3; i++ )
        a[i] = 2*i;

    cout << "Intial a and b:" << endl;
    for ( int i = 0; i < 3; i++ )
        cout << "a[" << i << "] = " << a[i]
              << " b[" << i << "] = " << b[i]<< endl;

    copy( a, b );
    cout << endl << "After calling copy funtion:" << endl;
    for ( int i = 0; i < 3; i++ )
        cout << "a[" << i << "] = " << a[i]
              << " b[" << i << "] = " << b[i]<< endl;

    add( a, b );
    cout << endl << "After calling sum funtion:" << endl;
    for ( int i = 0; i < 3; i++ )
        cout << "a[" << i << "] = " << a[i]
              << " b[" << i << "] = " << b[i]<< endl;

    return 0;
}
```

5.8 bf_multidimensional_arrays.cpp

```
/* program to illustrate multidimensional arrays
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

void
mulvec ( const double M[3][3],
         const double x[3],
         double y[3] ) {
    for ( int i = 0; i < 3; i++ ) {
        y[i] = 0.0;

        for ( int j = 0; j < 3; j++ )
            y[i] += M[i][j] * x[j];
    }
}

int main() {
    double M[3][3],
           x[3], y[3];

    for ( int i = 0; i < 3; i++ ) {
        x[i] = 2*i;
        for ( int j = 0; j < 3; j++ )
            M[i][j] = 3*i+j;
    }

    mulvec(M, x, y);

    cout << "M:" << endl;
    for ( int i = 0; i < 3; i++ ) {
        for ( int j = 0; j < 3; j++ )
            cout << " " << M[i][j];
        cout << endl;
    }

    cout << "x:" << endl;
    for ( int j = 0; j < 3; j++ )
        cout << " " << x[j] << endl;

    cout << "y = M*x:" << endl;
    for ( int j = 0; j < 3; j++ )
        cout << " " << y[j] << endl;

    return 0;
}
```

5.9 bg_array_and_pointer.cpp

```
/* program to illustrate pointers as arrays
 *
 * in C/C++ there is NO distinction between a pointer and an array.
 *
 * Rajeev Singh
 * 2013-04-01
 *
 */

#include <iostream>
using namespace std;

int main() {
    int    n[5] = { 2, 3, 5, 7, 11 };
    int * p    = n;
    int * q    = &n[1];

    cout << "n:" << endl;
    for ( int j = 0; j < 5; j++ )
        cout << " " << n[j] << endl;

    cout << "p:" << endl;
    for ( int j = 0; j < 5; j++ )
        cout << " " << p[j] << endl;

    cout << "q:" << endl;
    for ( int j = 0; j < 5; j++ )
        cout << " " << q[j] << endl;

    return 0;
}
```

6 Day 6: Dynamic memory, Multidimensional Array, BLAS

6.1 bh_dynamic_memory.cpp

```
/* program to illustrate dynamic memory
 *
 * this example shows the C++ way of doing the job. this will not work
 * for C.
 *
 * Rajeev Singh
 * 2013-04-02
 *
 */

#include <iostream>
using namespace std;

int main() {
    int    n = 10;
    double * v = new double[n];

    for ( int i = 0; i < n; i++ )
        v[i] = double( i*i );

    cout << "n = " << n << endl;
    cout << "v:" << endl;
    for ( int j = 0; j < n; j++ )
        cout << " " << v[j] << endl;

    delete[] v;
    return 0;
}
```

6.2 bi_dynamic_array_size_input.cpp

```
/* program to illustrate dynamic memory
 *
 * Rajeev Singh
 * 2013-04-02
 *
 */

#include <iostream>
using namespace std;

int main() {
    int n;

    cout << "Enter the size of the array: ";
    cin >> n;

    double * v = new double[n];

    for ( int i = 0; i < n; i++ )
        v[i] = double( i*i );

    cout << "n = " << n << endl;
    cout << "v:" << endl;
    for ( int j = 0; j < n; j++ )
        cout << " " << v[j] << endl;

    delete[] v;
    return 0;
}
```

6.3 bj_multidimensional_array_with_pointer.cpp

```
/* program to illustrate dynamic memory
 *
 * Rajeev Singh
 * 2013-04-02
 *
 */

#include <iostream>
using namespace std;

int main() {
    int n1[4], n2[4], n3[4], n4[4];
    int * p1 = n1;           // p1 -> pointer
    int * p2 = n2;
    int * p3 = n3;
    int * p4 = n4;

    int *p[4] = {p1, p2, p3, p4}; // p -> pointer of pointers

    for (int i = 0; i < 4; i++ )
        for (int j = 0; j < 4; j++ )
            p[i][j] = 4*i+j;

    cout << "p:" << endl;
    for (int i = 0; i < 4; i++ ) {
        for (int j = 0; j < 4; j++ )
            cout << " " << p[i][j];
        cout << endl;
    }

    return 0;
}
```


6.4 bk_multidimensional_dynamic_array.cpp

```
/* program to illustrate dynamic memory
 *
 * Rajeev Singh
 * 2013-04-02
 *
 */

#include <iostream>
using namespace std;

int main() {
    int * p1 = new int[4];
    int * p2 = new int[4];
    int * p3 = new int[4];
    int * p4 = new int[4];

    int **p = new int*[4];

    p[0] = p1;
    p[1] = p2;
    p[2] = p3;
    p[3] = p4;

    for (int i = 0; i < 4; i++ )
        for (int j = 0; j < 4; j++ )
            p[i][j] = 4*i+j;

    cout << "p:" << endl;
    for (int i = 0; i < 4; i++ ) {
        for (int j = 0; j < 4; j++ )
            cout << " " << p[i][j];
        cout << endl;
    }

    return 0;
}
```

6.5 bl_multidimensional_dynamic_array_size_input.cpp

```
/* program to illustrate dynamic memory
 *
 * Rajeev Singh
 * 2013-04-02
 *
 */

#include <iostream>
using namespace std;

int main() {
    int m, n;
    cout << "Enter the size of the matrix: ";
    cin >> m >> n;

    int **p = new int*[m];

    for (int i = 0; i < m; i++ )
        p[i] = new int[n];

    for (int i = 0; i < m; i++ )
        for (int j = 0; j < n; j++ )
            p[i][j] = n*i+j;

    cout << "p:" << endl;
    for (int i = 0; i < m; i++ ) {
        for (int j = 0; j < n; j++ )
            cout << " " << p[i][j];
        cout << endl;
    }

    return 0;
}
```

6.6 bm_multidimensional_array_with_mapping.cpp

```
/* program to illustrate dynamic memory
 *
 * NOTE: using pointer of pointers can be slower than using mappings
 * for big arrays for the following reason:
 *
 * pointer of pointers -> two access to RAM to get an element
 * mapping              -> single access to RAM for the same
 *
 * accessing RAM is much more expensive than simple integer
 * multiplication and addition
 *
 * Rajeev Singh
 * 2013-04-02
 */

#include <iostream>
using namespace std;

int main() {
    int m, n;
    cout << "Enter the size of the matrix: ";
    cin >> m >> n;

    int *p = new int[m*n];

    for (int i = 0; i < m; i++ )
        for (int j = 0; j < n; j++ )
            p[n*i+j] = n*i+j;

    cout << "p:" << endl;
    for (int i = 0; i < m; i++ ) {
        for (int j = 0; j < n; j++ )
            cout << " " << p[n*i+j];
        cout << endl;
    }

    return 0;
}
```

GO TO THE TALK TO DISCUSS BLAS

7 Day 7: String, Advanced Datatypes, BLAS, Sparse Matrices

7.1 bn_strings.cpp

```
/* program to illustrate strings as array of characters
 *
 * Rajeev Singh
 * 2013-04-03
 *
 */

#include <iostream>
using namespace std;

int main() {
    char str1[] = { 'S', 't', 'r', 'i', 'n', 'g', '\0' };
    char str2[] = "String"; // '\0' is appended automatically
    char str3[] = "This is a very long \
string";

    cout << str1 << endl;
    cout << str2 << endl;
    cout << str3 << endl;

    return 0;
}
```

7.2 bo_typedef.cpp

```
/* program to illustrate renaming datatypes using typedef
 *
 * Rajeev Singh
 * 2013-04-03
 *
 */

#include <iostream>
using namespace std;

int main() {
    typedef char * string_t;
    string_t str2 = "String"; // '\0' is appended automatically
    string_t str3 = "This is a very long \
string";

    cout << str2 << endl;
    cout << str3 << endl;

    return 0;
}
```

7.3 bp_struct.cpp

```
/* program to illustrate defining new datatypes using struct
 *
 * Rajeev Singh
 * 2013-04-03
 *
 */

#include <iostream>
using namespace std;

typedef double real_t;

struct vector_t {
    size_t size;
    real_t * coeffs;
};

void
add_vec ( const vector_t & x,
          const vector_t & y,
          vector_t &      z ) {
    for ( int i = 0; i < x.size; i++ )
        z.coeffs[i] = x.coeffs[i] + y.coeffs[i];
}

int main() {
    int      n = 10;
    vector_t a, b, c;
    a.size = n;
    b.size = n;
    c.size = n;

    a.coeffs = new real_t[n];
    b.coeffs = new real_t[n];
    c.coeffs = new real_t[n];

    for ( int i = 0; i < n; i++ ) {
        a.coeffs[i] = i;
        b.coeffs[i] = 2*i;
    }

    add_vec( a, b, c );

    cout << "a:" << endl;
    cout << "a.size = " << a.size << endl;
    cout << "a.coeffs:" << endl;
    for ( int i = 0; i < n; i++ )
        cout << "      " << a.coeffs[i] << endl;

    cout << "b:" << endl;
    cout << "b.size = " << b.size << endl;
    cout << "b.coeffs:" << endl;
```

```

    for ( int i = 0; i < n; i++ )
        cout << "          " << b.coeffs[i] << endl;

    cout << "c:" << endl;
    cout << "c.size = " << c.size << endl;
    cout << "c.coeffs:" << endl;
    for ( int i = 0; i < n; i++ )
        cout << "          " << c.coeffs[i] << endl;

    delete[] a.coeffs;
    delete[] b.coeffs;
    delete[] c.coeffs;
    return 0;
}

```

7.4 bq_struct_pointer.cpp

```
/* program to illustrate using pointers to struct
 *
 * Rajeev Singh
 * 2013-04-03
 *
 */

#include <iostream>
using namespace std;

typedef double real_t;

struct vector_t {
    size_t size;
    real_t * coeffs;
};

void
add_vec ( const vector_t * x,
          const vector_t * y,
          vector_t *      z ) {
    for ( int i = 0; i < x->size; i++ )
        z->coeffs[i] = x->coeffs[i] + y->coeffs[i];
}

int main() {
    int      n = 10;
    vector_t * a = new vector_t,
              * b = new vector_t,
              * c = new vector_t;

    a->size = n;
    b->size = n;
    c->size = n;

    a->coeffs = new real_t[n];
    b->coeffs = new real_t[n];
    c->coeffs = new real_t[n];

    for ( int i = 0; i < n; i++ ) {
        a->coeffs[i] = i;
        b->coeffs[i] = 2*i;
    }

    add_vec( a, b, c );

    cout << "a:" << endl;
    cout << "a->size = " << a->size << endl;
    cout << "a->coeffs:" << endl;
    for ( int i = 0; i < n; i++ )
        cout << "      " << a->coeffs[i] << endl;
```



```

cout << "b:" << endl;
cout << "b->size = " << b->size << endl;
cout << "b->coeffs:" << endl;
for ( int i = 0; i < n; i++ )
    cout << "          " << b->coeffs[i] << endl;

cout << "c:" << endl;
cout << "c->size = " << c->size << endl;
cout << "c->coeffs:" << endl;
for ( int i = 0; i < n; i++ )
    cout << "          " << c->coeffs[i] << endl;

delete[] a->coeffs;
delete[] b->coeffs;
delete[] c->coeffs;

delete a;
delete b;
delete c;

return 0;
}

```

7.5 br_struct_array.cpp

```
/* program to illustrate using struct array
 *
 * Rajeev Singh
 * 2013-04-03
 *
 */

#include <iostream>
using namespace std;

typedef double real_t;

struct vector_t {
    size_t size;
    real_t * coeffs;
};

void
add_vec ( const vector_t * x,
          const vector_t * y,
          vector_t * z ) {
    for ( int i = 0; i < x->size; i++ )
        z->coeffs[i] = x->coeffs[i] + y->coeffs[i];
}

int main() {
    int n = 10;
    vector_t * a = new vector_t[3];

    for ( int i = 0; i < 3; i++ ) {
        a[i].size = n;
        a[i].coeffs = new real_t[n];
    }

    for ( int i = 0; i < n; i++ ) {
        a[0].coeffs[i] = i;
        a[1].coeffs[i] = 2*i;
    }

    add_vec( &a[0], &a[1], &a[2] );

    cout << "a[0]:" << endl;
    cout << "a[0].size = " << a[0].size << endl;
    cout << "a[0].coeffs:" << endl;
    for ( int i = 0; i < n; i++ )
        cout << "          " << a[0].coeffs[i] << endl;

    cout << "a[1]:" << endl;
    cout << "a[1].size = " << a[1].size << endl;
    cout << "a[1].coeffs:" << endl;
    for ( int i = 0; i < n; i++ )
        cout << "          " << a[1].coeffs[i] << endl;
```

```

cout << "a[2]:" << endl;
cout << "a[2].size = " << a[2].size << endl;
cout << "a[2].coeffs:" << endl;
for ( int i = 0; i < n; i++ )
    cout << "          " << a[2].coeffs[i] << endl;

for ( int i = 0; i < 3; i++ )
    delete[] a[i].coeffs;

delete[] a;

return 0;
}

```

GO TO THE TALK TO DISCUSS BLAS AND SPARSE MATRICES

8 Day 9: Modules and Namespaces, Classes

GO TO THE TALK TO DISCUSS MODULES AND NAMESPACES

8.1 bs_struct_with_functions.cpp

```
/* program to illustrate using struct with functions as members
 *
 * Rajeev Singh
 * 2013-04-05
 *
 */

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
    size_t size;
    real_t * coeffs;

    void init ( const unsigned n );
    void del   ();
    void fill  ( const real_t f );
    void scale ( const real_t f );
    void print ();
};

int main() {
    vector_t x;

    x.init( 10 );
    x.print();
    x.fill( 1.0 );
    x.print();
    x.scale( 5.0 );
    x.print();
    x.del();

    return 0;
}

void vector_t::init (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}

void vector_t::del () {
    delete[] coeffs;
}

void vector_t::fill ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] = f;
}

void vector_t::scale ( const real_t f ) {
    for (int i = 0; i < size; i++ )
```

```

        coeffs[i] *= f;
    }

    void vector_t::print () {
        cout << "size = " << size << endl;
        cout << "coeffs:" << endl;
        for (int i = 0; i < size; i++ )
            cout << "          " << coeffs[i] << endl;
    }

```

8.2 bt_struct_constructor_destructor.cpp

```
/* program to illustrate using struct with special functions for  
 * construction and destruction of objects  
 *  
 * Rajeev Singh  
 * 2013-04-05  
 *  
 */  
  
#include <iostream>  
using namespace std;  
typedef double real_t;  
  
struct vector_t {  
    size_t size;  
    real_t * coeffs;  
  
    vector_t ( const unsigned n );  
    ~vector_t ();  
    void fill ( const real_t f );  
    void scale ( const real_t f );  
    void print ();  
};  
  
int main() {  
    vector_t x(10);  
  
    x.print();  
    x.fill( 1.0 );  
    x.print();  
    x.scale( 5.0 );  
    x.print();  
  
    return 0;  
}  
  
vector_t::vector_t (const unsigned n ) {  
    size = n;  
    coeffs = new real_t[n];  
}  
  
vector_t::~~vector_t () {  
    delete[] coeffs;  
}  
  
void vector_t::fill ( const real_t f ) {  
    for (int i = 0; i < size; i++ )  
        coeffs[i] = f;  
}  
  
void vector_t::scale ( const real_t f ) {  
    for (int i = 0; i < size; i++ )  
        coeffs[i] *= f;  
}
```

```
}  
  
void vector_t::print () {  
    cout << "size = " << size << endl;  
    cout << "coeffs:" << endl;  
    for (int i = 0; i < size; i++ )  
        cout << "          " << coeffs[i] << endl;  
}
```


9 Day 10: Classes Continued

9.1 bu_struct_this_pointer.cpp

```
/* program to illustrate the use of this pointer  
 * members  
 *  
 * Rajeev Singh  
 * 2013-04-07  
 *  
 */  
  
#include <iostream>  
using namespace std;  
typedef double real_t;  
  
struct vector_t {  
private:  
    size_t size;  
    real_t * coeffs;  
  
public:  
    vector_t ( const unsigned n );  
    ~vector_t ();  
    void fill ( const real_t f );  
    void scale ( const real_t f );  
    void add ( const real_t alpha, const vector_t & a );  
    void print () const;  
};  
  
int main() {  
    vector_t x(10), y(10);  
  
    x.fill( 1.0 );  
    y.fill( 2.0 );  
    cout << "x:" << endl;  
    x.print();  
    cout << "y:" << endl;  
    y.print();  
    x.add( 10.0, y );  
    cout << "x:" << endl;  
    x.print();  
    cout << "y:" << endl;  
    y.print();  
  
    return 0;  
}  
  
vector_t::vector_t (const unsigned n ) {  
    size = n;  
    coeffs = new real_t[n];  
}  
  
vector_t::~~vector_t () {
```

```

        delete[] coeffs;
    }

    void vector_t::fill ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] = f;
    }

    void vector_t::scale ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] *= f;
    }

    void vector_t::add ( const real_t alpha, const vector_t & a ) {
        for (int i = 0; i < this->size; i++ )
            this->coeffs[i] += alpha * a.coeffs[i];
    }

    void vector_t::print () const {
        cout << "size = " << size << endl;
        cout << "coeffs:" << endl;
        for (int i = 0; i < size; i++ )
            cout << "        " << coeffs[i] << endl;
    }

```

9.2 bv_copy_constructor.cpp

```
/* program to illustrate the use of copy constructor
* members
*
* Rajeev Singh
* 2013-04-07
*
*/

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
private:
    size_t size;
    real_t * coeffs;

public:
    vector_t ( const unsigned n );
    vector_t ( const vector_t & a );
    ~vector_t ();
    void fill ( const real_t f );
    void scale ( const real_t f );
    void add ( const real_t alpha, const vector_t & a );
    void print () const;
};

int main() {
    vector_t x(10);

    x.fill( 1.0 );
    cout << "x:" << endl;
    x.print();

    vector_t y(x);
    cout << "y:" << endl;
    y.print();

    x.scale( 5.0 );
    cout << "x:" << endl;
    x.print();
    cout << "y:" << endl;
    y.print();

    return 0;
}

vector_t::vector_t (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}
```

```

vector_t::vector_t (const vector_t & a ) {
    this->size = a.size;
    this->coeffs = new real_t[a.size];
    for (int i = 0; i < a.size; i++ )
        this->coeffs[i] = a.coeffs[i];
}

vector_t::~~vector_t () {
    delete[] coeffs;
}

void vector_t::fill ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] = f;
}

void vector_t::scale ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] *= f;
}

void vector_t::add ( const real_t alpha, const vector_t & a ) {
    for (int i = 0; i < this->size; i++ )
        this->coeffs[i] += alpha * a.coeffs[i];
}

void vector_t::print () const {
    cout << "size = " << size << endl;
    cout << "coeffs:" << endl;
    for (int i = 0; i < size; i++ )
        cout << "        " << coeffs[i] << endl;
}

```

9.3 bw_default_copy_constructor.cpp

```
/* program to illustrate the problem with the default copy constructor
 * members
 *
 * Rajeev Singh
 * 2013-04-07
 *
 */

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
private:
    size_t size;
    real_t * coeffs;

public:
    vector_t ( const unsigned n );
    //~vector_t ();
    void fill ( const real_t f );
    void scale ( const real_t f );
    void add ( const real_t alpha, const vector_t & a );
    void print () const;
};

int main() {
    vector_t x(10);

    x.fill( 1.0 );
    cout << "x:" << endl;
    x.print();

    vector_t y(x);
    cout << "y:" << endl;
    y.print();

    x.scale( 5.0 );
    cout << "x:" << endl;
    x.print();
    cout << "y:" << endl;
    y.print();

    return 0;
}

vector_t::vector_t (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}

/*
```

```

vector_t::~~vector_t () {
    delete[] coeffs;
}
*/

void vector_t::fill ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] = f;
}

void vector_t::scale ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] *= f;
}

void vector_t::add ( const real_t alpha, const vector_t & a ) {
    for (int i = 0; i < this->size; i++ )
        this->coeffs[i] += alpha * a.coeffs[i];
}

void vector_t::print () const {
    cout << "size = " << size << endl;
    cout << "coeffs:" << endl;
    for (int i = 0; i < size; i++ )
        cout << "        " << coeffs[i] << endl;
}

```

9.4 bx_struct_with_const_functions.cpp

```
/* program to illustrate using struct with const functions
*/
* Rajeev Singh
* 2013-04-05
*/

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
    size_t size;
    real_t * coeffs;

    vector_t ( const unsigned n );
    ~vector_t ();
    void fill ( const real_t f );
    void scale ( const real_t f );
    void print () const;
};

int main() {
    const vector_t x(10);

    x.print();
    //x.fill( 1.0 );    // error
    //x.print();
    //x.scale( 5.0 );   // error
    //x.print();

    return 0;
}

vector_t::vector_t (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}

vector_t::~~vector_t () {
    delete[] coeffs;
}

void vector_t::fill ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] = f;
}

void vector_t::scale ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] *= f;
}
```

```
void vector_t::print () const {  
    cout << "size = " << size << endl;  
    cout << "coeffs:" << endl;  
    for (int i = 0; i < size; i++ )  
        cout << "          " << coeffs[i] << endl;  
}
```


9.5 by_struct_visibility.cpp

```
/* program to illustrate using struct with different visibility for
* members
* 
* Rajeev Singh
* 2013-04-05
* 
*/

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
private:
    size_t size;
    real_t * coeffs;

public:
    vector_t ( const unsigned n );
    ~vector_t ();
    void fill ( const real_t f );
    void scale ( const real_t f );
    void print () const;
};

int main() {
    vector_t x(10);

    x.print();
    x.fill( 1.0 );
    x.print();
    x.scale( 5.0 );
    x.print();

    //cout << x.size << endl; // error

    return 0;
}

vector_t::vector_t (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}

vector_t::~~vector_t () {
    delete[] coeffs;
}

void vector_t::fill ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] = f;
}
```

```

void vector_t::scale ( const real_t f ) {
    for (int i = 0; i < size; i++ )
        coeffs[i] *= f;
}

void vector_t::print () const {
    cout << "size = " << size << endl;
    cout << "coeffs:" << endl;
    for (int i = 0; i < size; i++ )
        cout << "          " << coeffs[i] << endl;
}

```

9.6 bz_safely_changing_size_of_array.cpp

```
/* program to illustrate using visibility to change the size of array
 * safely
 *
 * Rajeev Singh
 * 2013-04-07
 *
 */

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
private:
    size_t size;
    real_t * coeffs;

public:
    vector_t ( const unsigned n );
    ~vector_t ();
    int get_size ();
    void set_size ( const unsigned n );
    void fill ( const real_t f );
    void scale ( const real_t f );
    void print () const;
};

int main() {
    vector_t x(10);

    x.print();
    x.fill( 1.0 );
    x.print();
    x.scale( 5.0 );
    x.print();
    cout << "Size of x = " << x.get_size() << endl;
    x.set_size( 4 );
    x.print();

    return 0;
}

vector_t::vector_t (const unsigned n ) {
    size = n;
    coeffs = new real_t[n];
}

vector_t::~~vector_t () {
    delete[] coeffs;
}

int vector_t::get_size () {
```

```

        return size;
    }

    void vector_t::set_size ( const unsigned n ) {
        if (size != n) {
            size = n;
            delete coeffs; // delete the old data
            coeffs = new real_t[n];
        }
    }

    void vector_t::fill ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] = f;
    }

    void vector_t::scale ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] *= f;
    }

    void vector_t::print () const {
        cout << "size = " << size << endl;
        cout << "coeffs:" << endl;
        for (int i = 0; i < size; i++ )
            cout << "        " << coeffs[i] << endl;
    }

```

10 Day 11: Classes Continued, Templates

10.1 ca_operator_overloading.cpp

```
/* program to illustrate operator overloading
 * safely
 *
 * Rajeev Singh
 * 2013-04-08
 *
 */

#include <iostream>
using namespace std;
typedef double real_t;

struct vector_t {
private:
    size_t size;
    real_t * coeffs;

public:
    vector_t ( const unsigned n );
    ~vector_t ();
    int get_size ();
    void set_size ( const unsigned n );
    void fill ( const real_t f );
    void scale ( const real_t f );
    void print () const;

    vector_t & operator += (const vector_t & a) {
        for (size_t i = 0; i < size; i++ )
            coeffs[i] += a.coeffs[i];
        return *this;
    }

    vector_t & operator -= (const vector_t & a) {
        for (size_t i = 0; i < size; i++ )
            coeffs[i] -= a.coeffs[i];
        return *this;
    }
};

int main() {
    vector_t x(4), y(4);

    x.fill( 3.0 );
    y.fill( 1.0 );
    x.print();
    y.print();

    x += y;
    x.print();
    y.print();
}
```

```

        return 0;
    }

    vector_t::vector_t (const unsigned n ) {
        size = n;
        coeffs = new real_t[n];
    }

    vector_t::~~vector_t () {
        delete[] coeffs;
    }

    int  vector_t::get_size () {
        return size;
    }

    void vector_t::set_size ( const unsigned n ) {
        if (size != n) {
            size = n;
            delete coeffs; // delete the old data
            coeffs = new real_t[n];
        }
    }

    void vector_t::fill ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] = f;
    }

    void vector_t::scale ( const real_t f ) {
        for (int i = 0; i < size; i++ )
            coeffs[i] *= f;
    }

    void vector_t::print () const {
        cout << "size = " << size << endl;
        cout << "coeffs:" << endl;
        for (int i = 0; i < size; i++ )
            cout << "        " << coeffs[i] << endl;
    }

```

10.2 cb_templates.cpp

```
/* program to illustrate generic programming using templates  
 * safely  
 *  
 * Rajeev Singh  
 * 2013-04-08  
 *  
 */  
  
#include <iostream>  
using namespace std;  
  
template <typename T>  
T square(const T f) {  
    return f*f;  
}  
  
int main() {  
    double x = square( 2.1 );  
    int     m = square( 2 );  
  
    cout << "x = : " << x << endl;  
    cout << "m = : " << m << endl;  
  
    return 0;  
}
```

GO TO THE TALK TO DISCUSS BLAS AND TEMPLATES

11 Day 14: STL

OPEN <http://www.cplusplus.com/reference/>

11.1 cc_list.cpp

```
/* program to illustrate lists from STL
 *
 * Rajeev Singh
 * 2013-04-11
 *
 */

#include <iostream>
#include <list>

using namespace std;

int main() {
    list< int > ilist;

    ilist.push_front( 1 );
    ilist.push_front( 2 );
    ilist.push_back( 3 );
    ilist.push_back( 4 );

    for ( list<int>::iterator it = ilist.begin(); it != ilist.end(); it++)
        cout << *it << endl;

    int sum = 0;

    while ( ! ilist.empty() ) {
        sum += ilist.front();
        ilist.pop_front();
    }

    cout << "Sum of the list = " << sum << endl;
    return 0;
}
```


11.2 cd_vector.cpp

```
/* program to illustrate vector from STL
 *
 * Rajeev Singh
 * 2013-04-11
 *
 */

#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector< int > ivector;

    ivector.push_back( 1 );
    ivector.push_back( 2 );
    ivector.push_back( 3 );
    ivector.push_back( 4 );

    for ( vector<int>::iterator it = ivector.begin(); it != ivector.end(); it++)
        cout << *it << endl;

    cout << endl;
    for ( int i = 0; i < ivector.size(); i++ )
        cout << ivector[i] << endl;

    int sum = 0;

    while ( ! ivector.empty() ) {
        sum += ivector.back();
        ivector.pop_back();
    }

    cout << "Sum of the vector = " << sum << endl;
    return 0;
}
```

11.3 ce_valarray.cpp

```
/* program to illustrate valarray from STL
 *
 * Example taken from:
 * http://www.cplusplus.com/reference/valarray/valarray/operators/
 *
 * Rajeev Singh
 * 2013-04-11
 *
 */

// valarray operators example
#include <iostream>
#include <valarray>
using namespace std;

void print_all( valarray<int> & foo, valarray<int> & bar ) {
    cout << endl << "foo:  " << "bar:" << endl;
    for (int i = 0; i < foo.size(); i++ )
        cout << foo[i] << "      " << bar[i] << endl;
}

int main () {
    int init[] = {10,20,30,40};

                                //      foo:              bar:

    valarray<int> foo (init, 4); // 10 20 30 40
    valarray<int> bar (25,4);    // 10 20 30 40      25 25 25 25
    print_all(foo, bar);

    bar += foo;                  // 10 20 30 40      35 45 55 65
    print_all(foo, bar);

    foo = bar + 10;              // 45 55 65 75      35 45 55 65
    print_all(foo, bar);

    foo -= 10;                  // 35 45 55 65      35 45 55 65
    print_all(foo, bar);

    valarray<bool> comp = (foo==bar);

    if ( comp.min() == true )
        cout << "They are equal.\n";
    else
        cout << "They are not equal.\n";

    return 0;
}
```

11.4 cf_complex_numbers.cpp

```
/* program to illustrate complex numbers from STL
 *
 * Rajeev Singh
 * 2013-04-11
 *
 */

#include <iostream>
#include <complex>
using namespace std;

int main () {
    complex< float > c1;
    c1.real() = 1.0;
    c1.imag() = -2.0;
    cout << "c1 = " << c1 << endl << endl;

    complex< double > I ( 0.0, 1.0 );
    complex< double > r ( 5.0 );
    complex< double > z;
    complex< double > i = I;

    cout << "I = " << I << endl;
    cout << "r = " << r << endl;
    cout << "z = " << z << endl;
    cout << "i = " << i << endl;
    cout << endl;
    cout << " sqrt( r + i ) = " << sqrt( r + i ) << endl;
    cout << " sin( r + i ) = " << sin( r + i ) << endl;

    return 0;
}
```

11.5 cg_auto_pointer.cpp

GO TO THE TALK TO DISCUSS THE NEED FOR AUTO POINTER

```
/* program to illustrate the use of auto-pointers from STL
*/
* Rajeev Singh
* 2013-04-11
*/

#include <iostream>
#include <memory>
using namespace std;

int main () {
    {
        double    x[100];
        double * y = new double[100];
    } // "x" is deallocated, but not "y"

    {
        double    x[100];
        auto_ptr< double > y( new double[100] );
    } // both "x" and "y" are deallocated

    cout << "done" << endl;

    return 0;
}
```

12 Day 15: Boost

OPEN Boost Document

12.1 ch_boost_array.cpp

```
/* program to illustrate the use of boost::array
*/
* Rajeev Singh
* 2013-04-12
*/

#include <iostream>
#include <boost/array.hpp>
using namespace std;
using namespace boost;

void print_all( array<int,4> & foo, array<int,4> & bar ) {
    cout << endl << "foo:  " << "bar:" << endl;
    for (int i = 0; i < foo.size(); i++ )
        cout << foo[i] << "      " << bar[i] << endl;
}

int main () {
                                //      foo:      bar:

    array<int,4> foo = {10,20,30,40};           // 10 20 30 40
    array<int,4> bar = {25} ;                   // 10 20 30 40    25 25 25 25
    print_all(foo, bar);

    //foo += bar; // doesn't work as '+' is not overloaded
    // not really sure what is the use of it

    return 0;
}
```

12.2 ci_boost_multi_array.cpp

```
/* program to illustrate the use of boost::multi_array
 *
 * example taken from the documentation of Boost.MultiArray
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <iostream>
#include "boost/multi_array.hpp"
#include <cassert>

using namespace std;

int
main () {
    // Create a 3D array that is 3 x 4 x 2
    typedef boost::multi_array<double, 3> array_type;
    typedef array_type::index index;
    array_type A(boost::extents[3][4][2]);

    // Assign values to the elements
    int values = 0;
    for(index i = 0; i != 3; ++i)
        for(index j = 0; j != 4; ++j)
            for(index k = 0; k != 2; ++k)
                A[i][j][k] = values++;

    // Verify values
    int verify = 0;
    for(index i = 0; i != 3; ++i) {
        for(index j = 0; j != 4; ++j) {
            for(index k = 0; k != 2; ++k) {
                assert(A[i][j][k] == verify++);
                cout << A[i][j][k] << " ";
            } // k
            cout << endl;
        } // j
        cout << endl;
    } // i

    return 0;
}
```

12.3 cj_boost_mulptrecision_cpp_int.cpp

```
/* program to illustrate the use of boost::multiprecision
*/
* example taken from boost document
*/
* Rajeev Singh
* 2013-04-12
*/

#include <boost/multiprecision/cpp_int.hpp>

using namespace boost::multiprecision;

int main () {
    int128_t v = 1;

    // Do some fixed precision arithmetic:
    for(unsigned i = 1; i <= 20; ++i)
        v *= i;

    std::cout << v << std::endl; // prints 20!

    // Repeat at arbitrary precision:
    cpp_int u = 1;
    for(unsigned i = 1; i <= 100; ++i)
        u *= i;

    std::cout << u << std::endl; // prints 100!

    return 0;
}
```

12.4 ck_boost_multiprecision_gmp.cpp

```
/* program to illustrate the use of boost::multiprecision
 *
 * example taken from boost document
 *
 * compile command:
 *   g++ -I /home/rajeev/software/general/boost_1_53_0/
 *   ck_boost_multiprecision_gmp.cpp -lgmp
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/multiprecision/gmp.hpp>

using namespace boost::multiprecision;

int main () {
    mpz_int v = 1;

    // Do some arithmetic:
    for(unsigned i = 1; i <= 1000; ++i)
        v *= i;

    std::cout << v << std::endl; // prints 1000!

    // Access the underlying representation:
    mpz_t z;
    mpz_init(z);
    mpz_set(z, v.backend().data());

    return 0;
}
```


12.5 cl_boost_mulptrecision_cpp_int_2.cpp

```
/* program to illustrate the use of boost::multiprecision
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/multiprecision/cpp_int.hpp>
#include <iostream>
#include <iomanip>
#include <vector>

void print_factorials()
{
    using boost::multiprecision::cpp_int;
    //
    // Print all the factorials that will fit inside a 128-bit integer.
    //
    // Begin by building a big table of factorials, once we know just how
    // large the largest is, we'll be able to "pretty format" the results.
    //
    // Calculate the largest number that will fit inside 128 bits, we could
    // also have used numeric_limits<int128_t>::max() for this value:
    cpp_int limit = (cpp_int(1) << 128) - 1;
    //
    // Our table of values:
    std::vector<cpp_int> results;
    //
    // Initial values:
    unsigned i = 1;
    cpp_int factorial = 1;
    //
    // Cycle through the factorials till we reach the limit:
    while(factorial < limit)
    {
        results.push_back(factorial);
        ++i;
        factorial *= i;
    }
    //
    // Lets see how many digits the largest factorial was:
    unsigned digits = results.back().str().size();
    //
    // Now print them out, using right justification, while we're at it
    // we'll indicate the limit of each integer type, so begin by defining
    // the limits for 16, 32, 64 etc bit integers:
    cpp_int limits[] = {
        (cpp_int(1) << 16) - 1,
        (cpp_int(1) << 32) - 1,
        (cpp_int(1) << 64) - 1,
```

```

        (cpp_int(1) << 128) - 1,
    };
    std::string bit_counts[] = { "16", "32", "64", "128" };
    unsigned current_limit = 0;
    for(unsigned j = 0; j < results.size(); ++j)
    {
        if(limits[current_limit] < results[j])
        {
            std::string message = "Limit of " + bit_counts[current_limit] + " bit integers reached.
            std::cout << std::setfill('.') << std::setw(digits+1) << std::right << message << endl;
            ++current_limit;
        }
        std::cout << std::setw(digits + 1) << std::right << results[j] << std::endl;
    }
}

int main() {
    print_factorials();
    return 0;
}

```

12.6 cm_boost_random_uniform.cpp

```
/* program to illustrate the use of boost::random
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <iostream>
#include <boost/random/mercenne_twister.hpp>
#include <boost/random/uniform_real.hpp>

using namespace std;

int main (void) {
    boost::random::mt19937 generator;
    boost::uniform_real<> uni_dist(0,1);

    int i, j;

    for (i = 0; i < 100; i++)
        cout << uni_dist(generator) << endl;

    return 0;
}
```

12.7 cn_boost_ublas_vector.cpp

```
/* program to illustrate the use of boost::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v (3);
    for (unsigned i = 0; i < v.size (); ++ i)
        v (i) = i;
    std::cout << v << std::endl;
}
```

12.8 co_boost_ublas_unit_vector.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    for (int i = 0; i < 3; ++ i) {
        unit_vector<double> v (3, i);
        std::cout << v << std::endl;
    }
}
```

12.9 cp_boost_ublas_zero_vector.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    zero_vector<double> v (3);
    std::cout << v << std::endl;
}
```

12.10 cq_boost_ublas_scalar_vector.cpp

```
/* program to illustrate the use of boost::numeric::ublas  
 *  
 * example taken from boost document  
 *  
 * Rajeev Singh  
 * 2013-04-12  
 *  
 */  
  
#include <boost/numeric/ublas/vector.hpp>  
#include <boost/numeric/ublas/io.hpp>  
  
int main () {  
    using namespace boost::numeric::ublas;  
    scalar_vector<double> v (3);  
    std::cout << v << std::endl;  
}
```

12.11 cr_boost_ublas_sparse_vector_mapped.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    mapped_vector<double> v (6, 3);
    for (unsigned i = 0; i < v.size ()/2; ++ i)
        v (2*i) = i+10;
    std::cout << v << std::endl;
}
```


12.12 cs_boost_ublas_sparse_vector_compressed.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    compressed_vector<double> v (6, 3);
    for (unsigned i = 0; i < v.size ()/2; ++ i)
        v (2*i) = i+10;
    std::cout << v << std::endl;
}
```

12.13 ct_boost_ublas_sparse_vector_coordinate.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-12
 *
 */

#include <boost/numeric/ublas/vector_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    coordinate_vector<double> v (6, 3);
    for (unsigned i = 0; i < v.size ()/2; ++ i)
        v (2*i) = i+10;
    std::cout << v << std::endl;
}
```

13 Day 16: Boost Continued

OPEN Boost Document

13.1 cu_boost_ublas_vector_expressions_conj_etc.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<std::complex<double> > v (3);
    for (unsigned i = 0; i < v.size (); ++ i)
        v (i) = std::complex<double> (i, i);

    std::cout << - v << std::endl;
    std::cout << conj (v) << std::endl;
    std::cout << real (v) << std::endl;
    std::cout << imag (v) << std::endl;
    std::cout << trans (v) << std::endl;
    std::cout << herm (v) << std::endl;
}
```

13.2 cv_boost_ublas_vector_expressions_binary.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v1 (3), v2 (3);
    for (unsigned i = 0; i < std::min (v1.size (), v2.size ()); ++ i)
        v1 (i) = v2 (i) = i;

    std::cout << v1 + v2 << std::endl;
    std::cout << v1 - v2 << std::endl;
}
```

13.3 cw_boost_ublas_vector_expressions_outer_prod.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v1 (3), v2 (3);
    for (unsigned i = 0; i < std::min (v1.size (), v2.size ()); ++ i)
        v1 (i) = v2 (i) = i;

    std::cout << outer_prod (v1, v2) << std::endl;
}
```

13.4 cx_boost_ublas_vector_expressions_scalar_multi.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/vector.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v (3);
    for (unsigned i = 0; i < v.size (); ++ i)
        v (i) = i;

    std::cout << 2.0 * v << std::endl;
    std::cout << v * 2.0 << std::endl;
}
```

13.5 cy_boost_ublas_vector_expressions_reductions.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/vector.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v (3);
    for (unsigned i = 0; i < v.size (); ++ i)
        v (i) = i;

    std::cout << sum (v) << std::endl;
    std::cout << norm_1 (v) << std::endl;
    std::cout << norm_2 (v) << std::endl;
    std::cout << norm_inf (v) << std::endl;
    std::cout << index_norm_inf (v) << std::endl;
}
```

13.6 cz_boost_ublas_vector_expressions_inner_prod.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/vector.hpp>

int main () {
    using namespace boost::numeric::ublas;
    vector<double> v1 (3), v2 (3);
    for (unsigned i = 0; i < std::min (v1.size (), v2.size ()); ++ i)
        v1 (i) = v2 (i) = i;

    std::cout << inner_prod (v1, v2) << std::endl;
}
```


13.7 da_boost_ublas_matrix_basic.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m (3, 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;
    std::cout << m << std::endl;
}
```

13.8 db_boost_ublas_matrix_special.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    {
        identity_matrix<double> m (3);
        std::cout << "Identity:" << std::endl;
        std::cout << m << std::endl;
    }

    {
        zero_matrix<double> m (3, 3);
        std::cout << "Zero:" << std::endl;
        std::cout << m << std::endl;
    }

    {
        scalar_matrix<double> m (3, 3);
        std::cout << "Scalar:" << std::endl;
        std::cout << m << std::endl;
    }
}
```

13.9 dc_boost_ublas_triangular_matrix.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/triangular.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    triangular_matrix<double, lower> m1 (3, 3);
    for (unsigned i = 0; i < m1.size1 (); ++ i)
        for (unsigned j = 0; j <= i; ++ j)
            m1 (i, j) = 3 * i + j;
    std::cout << m1 << std::endl;
    triangular_matrix<double, upper> mu (3, 3);
    for (unsigned i = 0; i < mu.size1 (); ++ i)
        for (unsigned j = i; j < mu.size2 (); ++ j)
            mu (i, j) = 3 * i + j;
    std::cout << mu << std::endl;
}
```

13.10 dd_boost_ublas_symmetric_matrix.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/symmetric.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    symmetric_matrix<double, lower> ml (3, 3);
    for (unsigned i = 0; i < ml.size1 (); ++ i)
        for (unsigned j = 0; j <= i; ++ j)
            ml (i, j) = 3 * i + j;
    std::cout << ml << std::endl;
    symmetric_matrix<double, upper> mu (3, 3);
    for (unsigned i = 0; i < mu.size1 (); ++ i)
        for (unsigned j = i; j < mu.size2 (); ++ j)
            mu (i, j) = 3 * i + j;
    std::cout << mu << std::endl;
}
```

13.11 de_boost_ublas_hermitian_matrix.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/hermitian.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    hermitian_matrix<std::complex<double>, lower> m1 (3, 3);
    for (unsigned i = 0; i < m1.size1 (); ++ i) {
        for (unsigned j = 0; j < i; ++ j)
            m1 (i, j) = std::complex<double> (3 * i + j, 3 * i + j);
        m1 (i, i) = std::complex<double> (4 * i, 0);
    }
    std::cout << m1 << std::endl;
    hermitian_matrix<std::complex<double>, upper> mu (3, 3);
    for (unsigned i = 0; i < mu.size1 (); ++ i) {
        mu (i, i) = std::complex<double> (4 * i, 0);
        for (unsigned j = i + 1; j < mu.size2 (); ++ j)
            mu (i, j) = std::complex<double> (3 * i + j, 3 * i + j);
    }
    std::cout << mu << std::endl;
}
```

13.12 df_boost_ublas_banded_matrix.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/banded.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    banded_matrix<double> m (3, 3, 1, 1);
    for (signed i = 0; i < signed (m.size1 ()); ++ i)
        for (signed j = std::max (i - 1, 0); j < std::min (i + 2, signed (m.size2 ()))
            m (i, j) = 3 * i + j;
    std::cout << m << std::endl;
}
```

13.13 dg_boost_ublas_sparse_matrix.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix_sparse.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    {
        mapped_matrix<double> m (3, 3, 3 * 3);
        for (unsigned i = 0; i < m.size1 (); ++ i)
            for (unsigned j = 0; j < m.size2 (); ++ j)
                m (i, j) = 3 * i + j;
        std::cout << m << std::endl;
    }
    {
        compressed_matrix<double> m (3, 3, 3 * 3);
        for (unsigned i = 0; i < m.size1 (); ++ i)
            for (unsigned j = 0; j < m.size2 (); ++ j)
                m (i, j) = 3 * i + j;
        std::cout << m << std::endl;
    }
    {
        coordinate_matrix<double> m (3, 3, 3 * 3);
        for (unsigned i = 0; i < m.size1 (); ++ i)
            for (unsigned j = 0; j < m.size2 (); ++ j)
                m (i, j) = 3 * i + j;
        std::cout << m << std::endl;
    }
}
```

13.14 dh_boost_ublas_matrix_expressions_conj_etc.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<std::complex<double>> > m (3, 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = std::complex<double> (3 * i + j, 3 * i + j);

    std::cout << - m << std::endl;
    std::cout << conj (m) << std::endl;
    std::cout << real (m) << std::endl;
    std::cout << imag (m) << std::endl;
    std::cout << trans (m) << std::endl;
    std::cout << herm (m) << std::endl;
}
```


13.15 di_boost_ublas_matrix_expressions_binary.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m1 (3, 3), m2 (3, 3);
    for (unsigned i = 0; i < std::min (m1.size1 (), m2.size1 ()); ++ i)
        for (unsigned j = 0; j < std::min (m1.size2 (), m2.size2 ()); ++ j)
            m1 (i, j) = m2 (i, j) = 3 * i + j;

    std::cout << m1 + m2 << std::endl;
    std::cout << m1 - m2 << std::endl;
}
```

13.16 dj_boost_ublas_matrix_expressions_scalar_multi.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m (3, 3);
    for (unsigned i = 0; i < m.size1 (); ++ i)
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;

    std::cout << 2.0 * m << std::endl;
    std::cout << m * 2.0 << std::endl;
}
```

13.17 dk_boost_ublas_matrix_expressions_matrix_vector_multi.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m (3, 3);
    vector<double> v (3);
    for (unsigned i = 0; i < std::min (m.size1 (), v.size ()); ++ i) {
        for (unsigned j = 0; j < m.size2 (); ++ j)
            m (i, j) = 3 * i + j;
        v (i) = i;
    }

    std::cout << prod (m, v) << std::endl;
    std::cout << prod (v, m) << std::endl;
}
```

13.18 dl_boost_ublas_matrix_expressions_matrix_vector_triangular_solver.

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/triangular.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m (3, 3);
    vector<double> v (3);
    for (unsigned i = 0; i < std::min (m.size1 (), v.size ()); ++ i) {
        for (unsigned j = 0; j <= i; ++ j)
            m (i, j) = 3 * i + j + 1;
        v (i) = i;
    }

    std::cout << solve (m, v, lower_tag ()) << std::endl;
    std::cout << solve (v, m, lower_tag ()) << std::endl;
}
```

13.19 dm_boost_ublas_matrix_expressions_matrix_matrix_multi.cpp

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/matrix.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m1 (3, 3), m2 (3, 3);
    for (unsigned i = 0; i < std::min (m1.size1 (), m2.size1 ()); ++ i)
        for (unsigned j = 0; j < std::min (m1.size2 (), m2.size2 ()); ++ j)
            m1 (i, j) = m2 (i, j) = 3 * i + j;

    std::cout << prod (m1, m2) << std::endl;
}
```

13.20 dn_boost_ublas_matrix_expressions_matrix_matrix_triangular_solver

```
/* program to illustrate the use of boost::numeric::ublas
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-14
 *
 */

#include <boost/numeric/ublas/triangular.hpp>
#include <boost/numeric/ublas/io.hpp>

int main () {
    using namespace boost::numeric::ublas;
    matrix<double> m1 (3, 3), m2 (3, 3);
    for (unsigned i = 0; i < std::min (m1.size1 (), m2.size1 ()); ++ i)
        for (unsigned j = 0; j <= i; ++ j)
            m1 (i, j) = m2 (i, j) = 3 * i + j + 1;

    std::cout << solve (m1, m2, lower_tag ()) << std::endl;
}
```

14 Day 17: Boost Continued

OPEN Boost Document

14.1 do_mpi.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <mpi.h>
#include <iostream>

int main(int argc, char* argv[])
{
    MPI_Init(&argc, &argv);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0) {
        int value = 17;
        int result = MPI_Send(&value, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        if (result == MPI_SUCCESS)
            std::cout << "Rank 0 OK!" << std::endl;
    } else if (rank == 1) {
        int value;
        int result = MPI_Recv(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
                             MPI_STATUS_IGNORE);
        if (result == MPI_SUCCESS && value == 17)
            std::cout << "Rank 1 OK!" << std::endl;
    }
    MPI_Finalize();
    return 0;
}
```

14.2 dp_boost_mpi_1.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * compile command:
 * mpic++ -I /home/rajeev/software/general/boost_1_53_0
 * dp_boost_mpi_1.cpp -lboost_mpi
 *
 * Rajeev Singh
 * 2013-04-15
 */

#include <boost/mpi/environment.hpp>
#include <boost/mpi/communicator.hpp>
#include <iostream>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;
    std::cout << "I am process " << world.rank() << " of " << world.size()
                << "." << std::endl;
    return 0;
}
```


14.3 dq_boost_mpi_point_to_point.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * compile command:
 * mpic++ dq_boost_mpi_point_to_point.cpp -lboost_mpi
 * -lboost_serialization
 *
 * Rajeev Singh
 * 2013-04-15
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <string>
#include <boost/serialization/string.hpp>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    if (world.rank() == 0) {
        world.send(1, 0, std::string("Hello"));
        std::string msg;
        world.recv(1, 1, msg);
        std::cout << msg << "!" << std::endl;
    } else {
        std::string msg;
        world.recv(0, 0, msg);
        std::cout << msg << ", ";
        std::cout.flush();
        world.send(0, 1, std::string("world"));
    }

    return 0;
}
```

14.4 dr_boost_mpi_non_blocking.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <string>
#include <boost/serialization/string.hpp>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    if (world.rank() == 0) {
        mpi::request reqs[2];
        std::string msg, out_msg = "Hello";
        reqs[0] = world.isend(1, 0, out_msg);
        reqs[1] = world.irecv(1, 1, msg);
        mpi::wait_all(reqs, reqs + 2);
        std::cout << msg << "!" << std::endl;
    } else {
        mpi::request reqs[2];
        std::string msg, out_msg = "world";
        reqs[0] = world.isend(0, 1, out_msg);
        reqs[1] = world.irecv(0, 0, msg);
        mpi::wait_all(reqs, reqs + 2);
        std::cout << msg << ", ";
    }

    return 0;
}
```

14.5 ds_boost_mpi_broadcast.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <string>
#include <boost/serialization/string.hpp>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    std::string value;
    if (world.rank() == 0) {
        value = "Hello, World!";
    }

    broadcast(world, value, 0);

    std::cout << "Process #" << world.rank() << " says " << value
               << std::endl;
    return 0;
}
```

14.6 dt_boost_mpi_gather.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <vector>
#include <cstdlib>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    std::srand(time(0) + world.rank());
    int my_number = std::rand();
    if (world.rank() == 0) {
        std::vector<int> all_numbers;
        gather(world, my_number, all_numbers, 0);
        for (int proc = 0; proc < world.size(); ++proc)
            std::cout << "Process #" << proc << " thought of "
                      << all_numbers[proc] << std::endl;
    } else {
        gather(world, my_number, 0);
    }

    return 0;
}
```

14.7 du_boost_mpi_reduce.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <cstdlib>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    std::srand(time(0) + world.rank());
    int my_number = std::rand();

    if (world.rank() == 0) {
        int minimum;
        reduce(world, my_number, minimum, mpi::minimum<int>(), 0);
        std::cout << "The minimum value is " << minimum << std::endl;
    } else {
        reduce(world, my_number, mpi::minimum<int>(), 0);
    }

    return 0;
}
```

14.8 dv_boost_mpi_reduce_2.cpp

```
/* program to illustrate the use of boost::mpi
 *
 * example taken from boost document
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/mpi.hpp>
#include <iostream>
#include <string>
#include <functional>
#include <boost/serialization/string.hpp>
namespace mpi = boost::mpi;

int main(int argc, char* argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    std::string names[10] = { "zero ", "one ", "two ", "three ",
                              "four ", "five ", "six ", "seven ",
                              "eight ", "nine " };

    std::string result;
    reduce(world,
           world.rank() < 10? names[world.rank()]
                               : std::string("many "),
           result, std::plus<std::string>(), 0);

    if (world.rank() == 0)
        std::cout << "The result is " << result << std::endl;

    return 0;
}
```

14.9 dw_boost_python_hello_world.cpp

```
/* program to illustrate the use of boost::python
 *
 * example taken from boost document
 *
 * compile command:
 * g++ -shared -fPIC -o a.so dw_boost_python_hello_world.cpp -I
 * /usr/include/python2.7 -lpython2.7 -lboost_python
 *
 * Rajeev Singh
 * 2013-04-15
 */

#include <Python.h>
#include <boost/python.hpp>

char const* greet()
{
    return "hello, world";
}

BOOST_PYTHON_MODULE(a)
{
    using namespace boost::python;
    def("greet", greet);
}
```

14.10 dx_boost_python_exposing_classes.cpp

```
/* program to illustrate the use of boost::python
 *
 * example taken from boost document
 *
 * compile command:
 * g++ -shared -fPIC -o a.so dx_boost_python_exposing_classes.cpp -I
 * /usr/include/python2.7 -lpython2.7 -lboost_python
 *
 *
 * Rajeev Singh
 * 2013-04-15
 *
 */

#include <boost/python.hpp>
using namespace boost::python;

struct World
{
    void set(std::string msg) { this->msg = msg; }
    std::string greet() { return msg; }
    std::string msg;
};

BOOST_PYTHON_MODULE(a)
{
    class_<World>("World")
        .def("greet", &World::greet)
        .def("set", &World::set)
        ;
}
```


15 Day 18: MTL

Open MTL Document

15.1 dy_mtl_vector1.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: vector1.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    // Define dense vector of doubles with 10 elements all set to 0.0.
    dense_vector<double> v(10, 0.0);

    // Set element 7 to 3.0.
    v[7]= 3.0;

    std::cout << "v is " << v << "\n";
    return 0;
}
```

15.2 dz_mtl_vector2.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: vector2.cpp

#include <complex>
#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    // Define dense vector of complex with 7 elements.
    dense_vector<std::complex<float>, mtl::vector::parameters<tag::row_major> >
    v(7);

    // Set all elements to 3+2i
    v = std::complex<float>(3.0, 2.0);
    std::cout << "v is " << v << "\n";

    // Set all elements to 5+0i
    v = 5.0;
    std::cout << "v is " << v << "\n";

    v = 6;
    std::cout << "v is " << v << "\n";

    return 0;
}
```

15.3 ea_mtl_dense2D.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: dense2D.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    // A is a row-major matrix
    dense2D<double> A(10, 10);

    // Matrices are not initialized by default
    A= 0.0;

    // Assign a value to a matrix element
    A(2, 3)= 7.0;

    // You can also use a more C-like notation
    A[2][4]= 3.0;

    std::cout << "A is \n" << A << "\n";

    // B is a column-major matrix
    dense2D<float, matrix::parameters<tag::col_major> > B(10, 10);

    // Assign the identity matrix times 3 to B
    B= 3;
    std::cout << "B is \n" << B << "\n";

    return 0;
}
```

15.4 eb_mtl_morton_dense.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: morton_dense.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    // Z-order matrix
    morton_dense<double, recursion::morton_z_mask> A(10, 10);

    A= 0;
    A(2, 3)= 7.0;
    A[2][4]= 3.0;
    std::cout << "A is \n" << A << "\n";

    // B is an N-order matrix with column-major 4x4 blocks, see paper
    morton_dense<float, recursion::doppled_4_col_mask> B(10, 10);

    // Assign the identity matrix times 3 to B
    B= 3;
    std::cout << "B is \n" << B << "\n";

    return 0;
}
```

15.5 ec_mtl_compressed2D.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: compressed2D.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    // CRS matrix
    compressed2D<double>    A(12, 12);

    // Laplace operator discretized on a 3x4 grid
    matrix::laplacian_setup(A, 3, 4);
    std::cout << "A is \n" << A;

    // Element access is allowed for reading
    std::cout << "A[3][2] is " << A[3][2] << "\n\n";

    // CCS matrix
    compressed2D<float, matrix::parameters<tag::col_major> > B(10, 10);

    // Assign the identity matrix times 3 to B
    B = 3;
    std::cout << "B is \n" << B << "\n";

    return 0;
}
```

15.6 ed_mtl_element_structure.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

using namespace std;
int main(int, char**)
{
    typedef double value_type;
    typedef int size_type;
    const int nb_elements= 6, nb_nodes= 12;

    value_type array[][4]= {{2, 3, 4, 5},
                           {4, 10, 13, 16},
                           {6, 25, 38, 46},
                           {8, 32, 77, 100}};
    mtl::dense2D<value_type> E_mat(array);

    std::cout<<"E_mat=\n"<< E_mat <<"\n";

    mtl::matrix::element_structure<value_type> A;

    typedef mtl::matrix::element<value_type> element_type;
    element_type* elements = new element_type[nb_elements];

    mtl::dense_vector<size_type> index_a(4, 0),
                                index_b(4, 0),
                                index_c(4, 0),
                                index_d(4, 0),
                                index_e(4, 0),
                                index_f(4, 0);

    // construct nodes for every element
    index_a[0]= 0; index_a[1]= 1; index_a[2]= 4; index_a[3]= 5;
    index_b= index_a + 1;
    index_c= index_a + 2;
    index_d= index_a + 4;
    index_e= index_a + 5;
    index_f= index_a + 6;

    //construct the 6 elements from the example grid
    element_type a(0, index_a, E_mat);
    element_type b(1, index_b, E_mat);
    element_type c(2, index_c, E_mat);
    element_type d(3, index_d, E_mat);
    element_type e(4, index_e, E_mat);
```

```

element_type f(5, index_f, E_mat);

//construct neighborhood information for each element
a.add_neighbors(&b, &d, &e);
b.add_neighbors(&a, &c, &d, &e, &f);
c.add_neighbors(&a, &b, &e);
d.add_neighbors(&a, &b, &e);
e.add_neighbors(&a, &b, &c, &d, &f);
f.add_neighbors(&b, &c, &e);

std::cout<< "a=" << a << "\n";

//construct array of elements
elements[0]=a;
elements[1]=b;
elements[2]=c;
elements[3]=d;
elements[4]=e;
elements[5]=f;

//construct element_structure from the 6 single elements
A.consume(nb_elements, nb_nodes, elements);
mtl::dense_vector<value_type> x(nb_nodes, 1.0), test(A * x);

std::cout<< "test="<< test << "\n";

return 0;
}

```

15.7 ee_mtl_multi_vector.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: multi_vector.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    typedef dense_vector<double>    Vector;

    Vector          v(2, 3.4), w(3, 2.5);
    mtl::multi_vector<Vector>      A(2, 3);
    dense2D<double>                B(2,2), C(3,2), D(3,3);

    // Initialize matrices
    A= 3.0; B= 4.0; C= 5.0; D= 6.0;

    // vector= multi_vector * vector
    v= A * w;
    std::cout << "v = " << v << std::endl;

    // vector= transposed multi_vector * vector
    w= trans(A) * v;
    std::cout << "w = " << w << std::endl;

    // vector= matrix * vector
    v= B * A.vector(1);
    std::cout << "v = " << v << std::endl;

    // vector= matrix * vector
    A.vector(0)= B * A.vector(1);
    std::cout << "A.vector(0) = " << A.vector(0) << std::endl;

    // Orthogonalize multi_vector
    orth(A);
    std::cout << "A = " << A << std::endl;

    return 0;
}
```


15.8 ef_mtl_insert.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: insert.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

using namespace mtl;

template <typename Matrix>
void fill(Matrix& m)
{
    // Matrices are not initialized by default
    m = 0.0;

    // Create inserter for matrix m
    matrix::inserter<Matrix> ins(m);

    // Insert value in m[0][0]
    ins[0][0] << 2.0;
    ins[1][2] << 0.5;
    ins[2][1] << 3.0;

    // Destructor of ins sets final state of m
}

template <typename Matrix>
void modify(Matrix& m)
{
    // Type of m's elements
    typedef typename Collection<Matrix>::value_type value_type;

    // Create inserter for matrix m
    // Existing values are not overwritten but inserted
    matrix::inserter<Matrix, update_plus<value_type> > ins(m, 3);

    // Increment value in m[0][0]
    ins[0][0] << 1.0;

    // Elements that doesn't exist (in sparse matrices) are inserted
    ins[1][1] << 2.5;
    ins[2][1] << 1.0;
    ins[2][2] << 4.0;

    // Destructor of ins sets final state of m
}
```

```

int main(int, char**)
{
    // Matrices of different types
    compressed2D<double>          A(3, 3);
    dense2D<double>               B(3, 3);
    morton_dense<float, morton_mask> C(3, 3);

    // Fill the matrices generically
    fill(A); fill(B); fill(C);
    std::cout << "A is \n" << A << "\nB is \n" << B << "\nC is \n" << C;

    // Modify the matrices generically
    modify(A); modify(B); modify(C);
    std::cout << "\n\nAfter modification:\nA is \n" << A
              << "\nB is \n" << B << "\nC is \n" << C;

    return 0;
}

```

15.9 eg_mtl_insert_scope.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// Filename: insert_scope.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

using namespace mtl;

int main(int, char**)
{
    compressed2D<double>          A(3, 3);
    {
        matrix::inserter<compressed2D<double> > ins(A);
        ins[0][0] << 2.0;
        ins[1][2] << 0.5;
        ins[2][1] << 3.0;
    } // ins is destroyed here

    std::cout << "A is \n" << A; // we can access A now

    return 0;
}
```

15.10 eh_mtl_element_matrix.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: element_matrix.cpp

#include <iostream>
#include <vector>
#include <boost/numeric/mtl/mtl.hpp>

using namespace mtl;

template <typename Matrix>
void fill(Matrix& m)
{
    // Matrices are not initialized by default
    m = 0.0;

    // Type of m's elements
    typedef typename Collection<Matrix>::value_type value_type;

    // Create inserter for matrix m
    // Existing values are not overwritten but inserted
    matrix::inserter<Matrix, update_plus<value_type> > ins(m, 3);

    // Define element matrix (array)
    double m1[2][2] = {{1.0, -.4}, {-0.5, 2.0}};

    // Corresponding indices of the elements
    std::vector<int> v1(2);
    v1[0] = 1; v1[1] = 3;

    // Insert element matrix
    ins << element_array(m1, v1);

    // Insert same array with different indices
    v1[0] = 0; v1[1] = 2;
    ins << element_array(m1, v1);

    // Use element matrix type with dynamic size
    dense2D<double> m2(2, 3);
    m2[0][0] = 1; m2[0][1] = 0.2; m2[0][2] = 0.1;
    m2[1][0] = 2; m2[1][1] = 1.2; m2[1][2] = 1.1;

    // Vector for column indices
    dense_vector<int> v2(3);
    // Indices can be out of order
    v2[0] = 4; v2[1] = 1; v2[2] = 3;
}
```

```

        // Use element_matrix and separate vectors for row and column indices
        ins << element_matrix(m2, v1, v2);
    }

    int main(int, char**)
    {
        // Matrices of different types
        compressed2D<double>          A(5, 5);
        dense2D<double>               B(5, 5);
        morton_dense<float, morton_mask> C(5, 5);

        // Fill the matrices generically
        fill(A); fill(B); fill(C);
        std::cout << "A is \n" << with_format(A, 4, 3)
                   << "\nB is \n" << with_format(B, 4, 3)
                   << "\nC is \n" << with_format(C, 4, 3);

        return 0;
    }

```

15.11 ei_mtl_insert_class_expensive.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// Filename: insert_class_expensive.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

using namespace mtl;

class world_matrix
{
public:
    world_matrix(unsigned nrows, unsigned ncols) : A(nrows, ncols) {}

    void add_entry(unsigned row, unsigned col, double value)
    {
        // Extremely expensive -> must not be done
        matrix::inserter<compressed2D<double>, update_plus<double> > ins(A);
        ins[row][col] << value;
    }

    friend inline std::ostream& operator<< (std::ostream& os, const world_matrix& w)
    { return os << w.A; }

private:
    compressed2D<double> A;
};

int main(int, char**)
{
    world_matrix A(3, 3);

    A.add_entry(0, 0, 2.0);
    A.add_entry(1, 2, 0.5);
    A.add_entry(2, 1, 3.0);

    std::cout << "A is \n" << A; // we can access A now

    return 0;
}
```

15.12 ej_mtl_insert_scope.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// Filename: insert_scope.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

using namespace mtl;

class world_matrix
{
    typedef matrix::inserter<compressed2D<double>, update_plus<double> > inserter_type;
public:
    world_matrix(unsigned nrows, unsigned ncols) : A(nrows, ncols), ins(0) {}

    void add_entry(unsigned row, unsigned col, double value)
    { (*ins)[row][col] << value; }

    void start_insertion() // must be called before first insertion
    { if (!ins) ins= new inserter_type(A); }

    void finish_insertion() // must be called before first usage
    {
        if (ins) {
            delete ins;
            ins= 0;
        }
    }

    friend inline std::ostream& operator<< (std::ostream& os, const world_matrix& w)
    { return os << w.A; }

private:
    compressed2D<double> A;
    inserter_type* ins;
};

int main(int, char**)
{
    world_matrix A(3, 3);

    A.start_insertion();
    A.add_entry(0, 0, 2.0);
    A.add_entry(1, 2, 0.5);
    A.add_entry(2, 1, 3.0);
}
```

```
A.finish_insertion();  
  
std::cout << "A is \n" << A;  // we can access A now  
  
return 0;  
}
```


15.13 ek_mtl_array_initialization.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: array_initialization.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    double array[][4] = {{3, 7.2, 0, 6},
                          {2, 4.444, 5, 3},
                          {1, 7, 9, 2}};

    dense2D<double> A(array);

    std::cout << "A = \n" << A << "\n";

    return 0;
}
```

15.14 el_mtl_vector_expr.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: vector_expr.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    typedef std::complex<double>    cdouble;
    dense_vector<cdouble>           u(10), v(10);
    dense_vector<double>            w(10), x(10, 4.0);

    for (unsigned i= 0; i < size(v); i++)
        v[i]= cdouble(i+1, 10-i), w[i]= 2 * i + 2;

    u= v + w + x;
    std::cout << "u is " << u << "\n";

    u-= 3 * w;
    std::cout << "u is " << u << "\n";

    u*= 6 ;
    std::cout << "u is " << u << "\n";

    u/= 2;
    std::cout << "u is " << u << "\n";

    u+= dot(v, w) * w + 4.0 * v + 2 * w;
    std::cout << "u is " << u << "\n";

    std::cout << "i * w is " << cdouble(0,1) * w << "\n";

    return 0;
}
```

15.15 em_mtl_rich_vector_expr.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

// File: rich_vector_expr.cpp

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    typedef std::complex<double>    cdouble;
    dense_vector<cdouble>           u(10), v(10);
    dense_vector<double>            w(10), x(10, 4.0);

    for (unsigned i= 0; i < size(v); i++)
        v[i]= cdouble(i+1, 10-i), w[i]= 2 * i + 2;

    // Increment w by x
    // and assign the sum of--the incremented--w and v to u
    u= v + (w+= x);
    std::cout << "u is " << u << "\n";

    // w= w * 3; x= 2; v= v + w + x; u= u + v;
    u+= v+= (w*= 3) + (x= 2);
    std::cout << "u is " << u << "w is " << w << "\n";

    return 0;
}
```

15.16 en_mtl_matrix_expressions1.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl;

    const unsigned n= 10;
    compressed2D<double> A(n, n);
    dense2D<int, matrix::parameters<col_major> > B(n, n);
    morton_dense<double, 0x555555f0> C(n, n), D(n, n);

    matrix::laplacian_setup(A, 2, 5);
    matrix::hessian_setup(B, 1);
    matrix::hessian_setup(C, 2.0);
    matrix::hessian_setup(D, 3.0);

    D+= A - 2 * B + C;

    std::cout << "The matrices are: A=\n" << A << "B=\n" << B << "C=\n" << C << "D=\n" << D << "\n";

    return 0;
}
```

15.17 eo_mtl_matrix_expressions2.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl; using namespace mtl::matrix;

    const unsigned n= 20;
    dense2D<double> A(n, n), B(n, n);
    morton_dense<double, doppled_64_row_mask> C(n, n);

    hessian_setup(A, 3.0); hessian_setup(B, 1.0);
    hessian_setup(C, 2.0);

    // Corresponds to A= B * B;
    mult(B, B, A);

    A= B * B;    // use BLAS
    A= B * C;    // use recursion + tiling from MTL4

    A+= B * C;   // Increment A by the product of B and C
    A-= B * C;   // Likewise with decrement

    std::cout << "The matrices are: A=\n" << A << "B=\n" << B << "C=\n" << C;

    return 0;
}
```

15.18 ep_mtl_matrix_expressions3.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <iostream>
#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl; using namespace mtl::matrix;

    const unsigned n= 40;
    dense2D<double> A(n, n), B(n, n);
    morton_dense<double, doppled_64_row_mask> C(n, n), D(n, n);

    hessian_setup(A, 3.0); hessian_setup(B, 1.0);
    hessian_setup(C, 2.0); hessian_setup(D, 11.0);

    A+= B * B + C * B - B * B * C * D;

    std::cout << "The matrices are: A=\n" << A << "B=\n" << B << "C=\n" << C << "D=\n" << D << "\n";

    return 0;
}
```

15.19 eq_mtl_matrix_vector1.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl; using namespace mtl::matrix;

    const unsigned          xd= 2, yd= 5, n= xd * yd;
    dense2D<double>          A(n, n);
    compressed2D<double>     B(n, n);
    hessian_setup(A, 3.0); laplacian_setup(B, xd, yd);

    typedef std::complex<double>  cdouble;
    dense_vector<cdouble>          v(n), w(n);
    for (unsigned i= 0; i < size(v); i++)
        v[i]= cdouble(i+1, n-i), w[i]= cdouble(i+n);

    v+= A * w;
    w= B * v;

    std::cout << "v is " << v << "\n";
    std::cout << "w is " << w << "\n";

    return 0;
}
```

15.20 er_mtl_matrix_vector2.cpp

```
/* program to illustrate the use of mtl
 *
 * example taken from mtl document
 *
 * Rajeev Singh
 * 2013-04-16
 *
 */

#include <boost/numeric/mtl/mtl.hpp>

int main(int, char**)
{
    using namespace mtl; using namespace mtl::matrix;

    const unsigned          xd= 2, yd= 5, n= xd * yd;
    dense2D<double>         A(n, n);
    laplacian_setup(A, xd, yd);
    dense_vector<double>     v(n), w(n, 7.0);

    // Scale A with 4 and multiply the scaled view with w
    v= 4 * A * w;
    std::cout << "v is " << v << "\n";

    // Scale w with 4 and multiply the scaled view with A
    v= A * (4 * w);
    std::cout << "v is " << v << "\n";

    // Scale both with 2 before multiplying
    v= 2 * A * (2 * w);
    std::cout << "v is " << v << "\n";

    // Scale v after the MVP
    v= A * w;
    v*= 4;
    std::cout << "v is " << v << "\n";

    return 0;
}
```