Midterm

Speaker: Lung-Sheng Chien

Exercise 1: organization of output

Exercise 1 (Pascal's triangle): In mathematics, Pascal's triangle is a geometric arrangement of the binomial coefficients in a triangle. This construction is related to the binomial coefficients by Pascal's rule, with states if $C_k^n = \dfrac{n!}{k!(n-k)!}$ is the k-th binomial coefficinet in the binomial expansion of $(x+y)^n$, then $C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$ for any $n \geq 0$ and $k = 0, 1, 2, \cdots, n$.

```
parscal's triangle
                                        1
                                    1       1
                                1       2       1
                            1       3       3       1
                        1       4       6       4       1
                    1       5      10      10       5       1
                1       6      15      20      15       6       1
            1       7      21      35      35      21       7       1
        1       8      28      56      70      56      28       8       1
    1       9      36      84     126     126      84      36       9       1
1      10      45     120     210     252     210     120      45      10       1
```

http://en.wikipedia.org/wiki/Pascal's_triangle

http://mathworld.wolfram.com/PascalsTriangle.html

http://www.cecm.sfu.ca/organics/papers/granville/support/pascalform.html

Program requirement [1]

(1) read n and output filename from command

[command] [n] –o [output filename]

```
[imsl@linux pascal_triangle]$ ./a.out  15  -o  output.txt
```

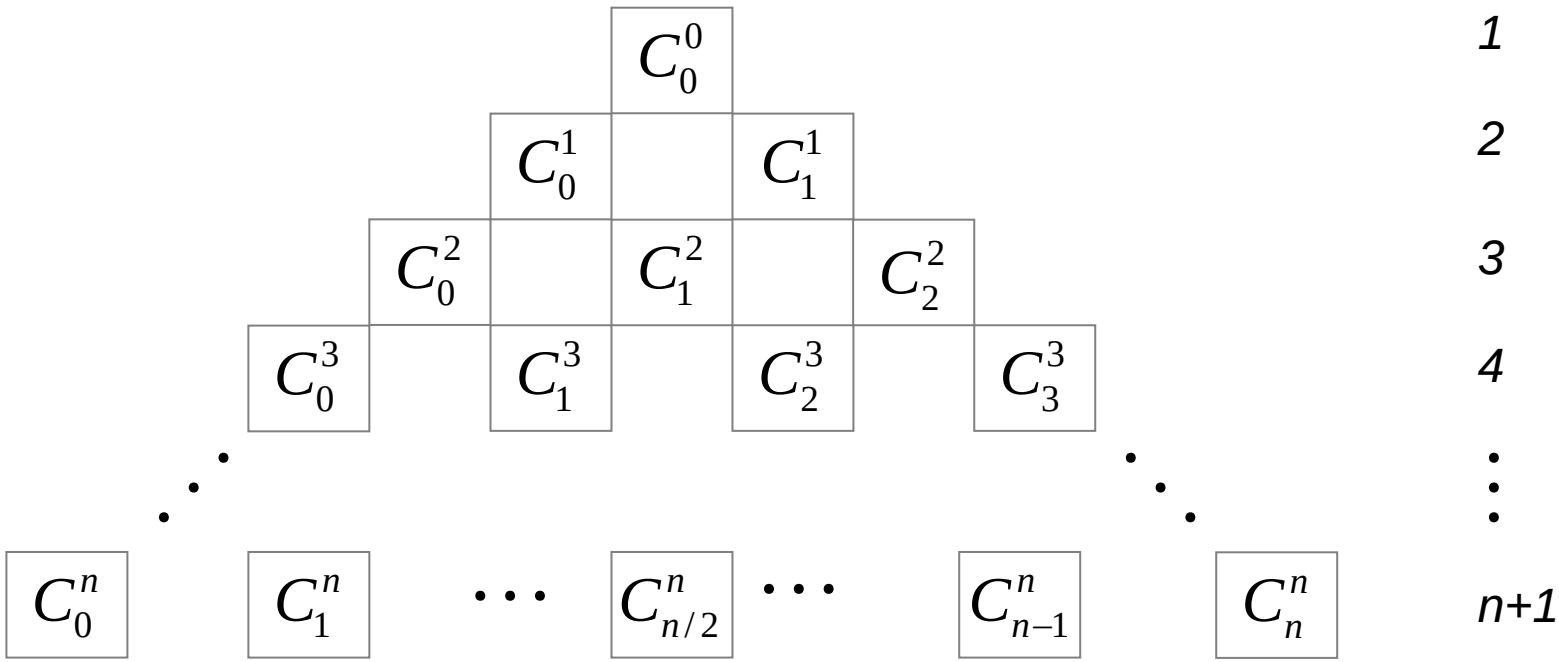
if user does not match the format, then output correct usage,

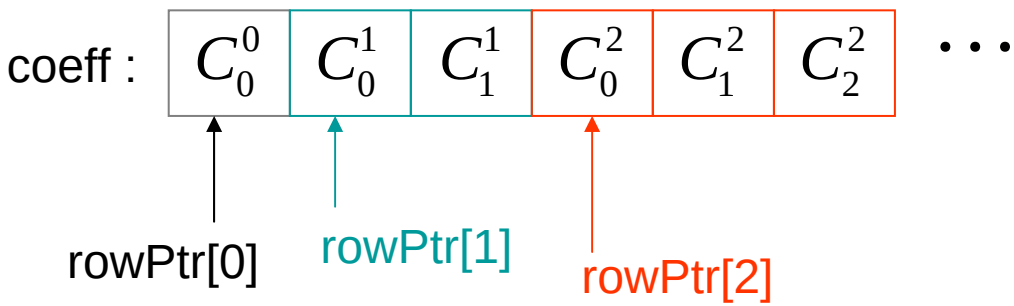
```
[imsl@linux pascal_triangle]$ ./a.out
usage: [command] [n] -o [output filename]
```

```c
#ifdef _WIN32
// for VC, no parameter follows command
    n = 10 ;
    strcpy( filename, DEFAULT_OUT_FILE ) ;
#else
    if ( 4 != argc ){
        print_usage() ;
    }
    argv++ ; // remove "[command]"
    if ( !isInteger( argv[0] ) ){
        print_usage() ;
    }
    n = atoi( argv[0] ) ;  // read "[n]"
    argv++ ;
    if ( 0 != strcmp( argv[0], "-o") ){
        print_usage() ;
    }
    argv++ ;
    strcpy( filename, argv[0] ) ;
#endif
```

Program requirement [2]

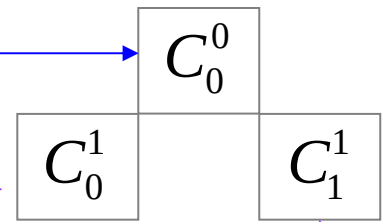
Total number of elements for each *n*

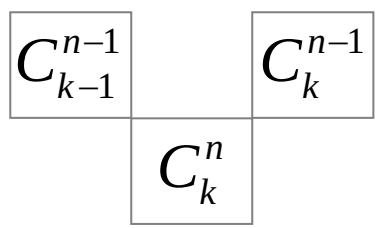




Program requirement [2] conti.

```
// level k has k+1 coeffients, total is 1+2+3+ ... + (n+1)
    total_num_coeff = (n+1)*(n+2) >> 1 ;

    coeff  = (int*) malloc( sizeof(int)* total_num_coeff ) ;
    assert( coeff) ;

// level 0, 1,2, ..., n
    rowPtr = (int **) malloc( sizeof(int*)*(n+1) ) ;
    assert( rowPtr ) ;

    rowPtr[0] = coeff ;
    coeff[0]  = 1 ; // C(0,0)

    if ( 1 <= n ){
        rowPtr[1] = rowPtr[0] + 1 ; // n = 0 has one element
        coeff[1]  = 1 ; // C(1,0)
        coeff[2]  = 1 ; // C(1,1)
    }

    if ( 2 <= n ){
        rowPtr[2] = rowPtr[1] + 2 ; // n = 1 has two elements
    }
    for( i = 2 ; i <=n ; i++ ){
        row_i = rowPtr[i] ;
        row_i_minus1 = rowPtr[i-1] ;
        row_i[0] = 1 ; // C(i,0)
        row_i[i] = 1 ; // C(i,i)
        for (j=1 ; j < i ; j++){
// C(i,j) = C(i-1,j-1) + C(i-1, j)
            row_i[j] = row_i_minus1[j-1] + row_i_minus1[j] ;
        }
        rowPtr[i+1] = rowPtr[i] + i+1 ; // n = i has i+1 elements
    }
```





Program requirement [2] conti.

reserve 3 blanks

reserve 2 blanks

reserve 1 blanks

			C_0^0			
		C_0^1		C_1^1		
	C_0^2		C_1^2		C_2^2	
C_0^3		C_1^3		C_2^3		C_3^3

size

$$10^{k-1} \le \max(coeff) < 10^k$$

$$size = k + 1$$

$$k = \#\text{ of digits of } \max(coeff)$$

$$Example: \log_{10}(252) = 2.4$$

```
max_coeff = 1 ;
row_i = rowPtr[n] ;
for (j=1 ; j < n ; j++){
    max_coeff = MAX(max_coeff, row_i[j] ) ;
}
size = (int)ceil( log10( max_coeff ) ) + 1 ;
```

```
                    1       5       10      10      5       1
                1       6       15      20      15      6       1
            1       7       21      35      35      21      7       1
        1       8       28      56      70      56      28      8       1
    1       9       36      84      126     126     84      36      9       1
1       10      45      120     210     252     210     120     45      10      1
```

Program requirement [2] conti.

```
    fout = fopen( filename, "w") ;
    assert(fout) ;

    fprintf(fout, "\n  parscal's triangle\n") ;
    sprintf( fmt, "%c%dd", '%', size) ;
    for( i = 0 ; i <= n ; i++ ){
        print_space( fout, size, n-i ) ; // reserve (n-i) blanks
        for ( j = 0 ; j <= i ; j++){
            row_i = rowPtr[i] ;
            fprintf( fout, fmt, row_i[j] ) ;
            print_space( fout, size, 1 ) ; // reserve 1 blanks between adjacent elements
                                            // of the same row
        }
        fprintf( fout, "\n") ;
    }

    fclose( fout ) ;
```

reserve 3 blanks

reserve 2 blanks

reserve 1 blanks

			C_0^0			
		C_0^1		C_1^1		
	C_0^2		C_1^2		C_2^2	
C_0^3		C_1^3		C_2^3		C_3^3

Exercise 2: profiling of sorting algorithm

```
// step 1: allocate integer array "intArray" with n elements
//  assignement monotone decreasing integer to "intArray"
    intArray = (int*) malloc( sizeof(int)*n ) ;
    assert( intArray ) ;

    for(i = 0 ; n > i ; i++){
        intArray[i] = n - i ;
    }
// step 2: execute quick sort
    start_time = time( NULL ) ;
    qsort( (void*) intArray, (size_t) n, sizeof(int),
        (int (*)(const void*, const void*))  &int_comp  ) ;
    end_time = time( NULL ) ;

    printf("n = %d, qsort needs %8.4f (s)\n", n, difftime( end_time, start_time) ) ;


    for(i = 0 ; n > i ; i++){
        intArray[i] = n - i ;
    }
// step 3: execute bubble sort
    start_time = time( NULL ) ;
    bubble_sort( (void*) intArray, (size_t) n, sizeof(int),
        (int (*)(const void*, const void*))  &int_comp  ) ;
    end_time = time( NULL ) ;

    printf("n = %d, bubble sort needs %8.4f (s)\n", n, difftime( end_time, start_time) ) ;
```

time_t *time* (*time_t* *tp)

time returns the current calendar time or -1 if the time is not available. If tp is not NULL, the return value is also assigned to *tp

double **difftime** (*time_t* time2, *time_t* time1)

difftime returns time2 – time1 expressed in seconds.

Timing report

Intel(R) Pentium(R) 4 CPU 3.00GHz, Cache 1MB, 2GB memory

Compiler: icpc 10.0

n	*Bubble sort*	*Quick sort*
10000	1s	0
20000	4s	0
40000	17s	0
80000	68s	0
160000	275s	0
10000000	???	2s
20000000	???	6s
40000000	???	11s
80000000	???	24s
160000000	???	32s
320000000	???	67s

$$bubble\ \ sort : O\left(n^2\right)$$

$$quick\ \ sort : O\left(n \log n\right)$$

Exercise 3: find filename o directory [1]

```
F:\course\2008summer\c_lang\example\midterm\system>dir
 磁碟區 F 中的磁碟沒有標籤。
 磁碟區序號:  C065-A3C9

 F:\course\2008summer\c_lang\example\midterm\system 的目錄

2008/07/30  上午 11:36    <DIR>          .
2008/07/30  上午 11:36    <DIR>          ..
2008/07/13  下午 06:41    <DIR>          Debug
2008/07/30  上午 11:25               564 getline.cpp
2008/07/30  上午 11:36             1,361 main.cpp
2008/07/13  下午 06:41               779 output.txt
2008/07/13  下午 05:54             4,718 system.dsp
2008/07/08  上午 10:55               535 system.dsw
2008/07/19  上午 10:48            41,984 system.ncb
2008/07/30  上午 11:36            49,664 system.opt
2008/07/13  下午 06:41             1,310 system.plg
               8 個檔案         100,915 位元組
               3 個目錄  11,560,648,704 位元組可用
```

Filename is 5-th token

Filename is 9-th token

```
[imsl@linux system]$ ls -al
total 124
drwxr-xr-x    3 imsl     imsl         4096 Jul 13 21:20 .
drwxr-xr-x    5 imsl     imsl         4096 Jul 13 18:44 ..
-rwxrwxr-x    1 imsl     imsl        28325 Jul 13 21:20 a.out
drwxr-xr-x    2 imsl     imsl         4096 Jul 13 18:41 Debug
-rw-r--r--    1 imsl     imsl          523 Jul 13 18:27 getline.cpp
-rw-r--r--    1 imsl     imsl         1329 Jul 13 18:41 main.cpp
-rw-r--r--    1 imsl     imsl          786 Jul 30 11:14 output.txt
-rw-r--r--    1 imsl     imsl         4718 Jul 13 17:54 system.dsp
-rw-r--r--    1 imsl     imsl          535 Jul  8 10:55 system.dsw
-rw-rw-r--    1 imsl     imsl            0 Jul 13 21:20 system.ncb
-rw-r--r--    1 imsl     imsl        49664 Jul 13 18:41 system.opt
-rw-r--r--    1 imsl     imsl         1310 Jul 13 18:41 system.plg
```

system("ls –al > output.txt")

open file *output.txt*

for each ***line*** in file output.txt

    read each ***token*** of the line and report 9-th token.

endfor

close file *output.txt*

Exercise 3: find filename o directory [2]

```
#ifdef _WIN32
    #define     FILENAME_INDEX      5
#else
    #define     FILENAME_INDEX      9
#endif
#ifdef _WIN32
    sprintf( command, "dir > %s", DEFAULT_OUT_FILE ) ;
#else
    sprintf( command, "ls -al > %s", DEFAULT_OUT_FILE) ;
#endif
    system( command );

    fp = fopen( DEFAULT_OUT_FILE, "r") ;
    assert( fp ) ;

    while( 0 < getline( fp, buffer, MAX_BUFFER_SIZE) ){
        len_buffer = strlen(  buffer ) ;
        buffer[ len_buffer - 1] = '\0' ; // remove newline character

        p =  buffer ;
        token_number = 0 ;
        while (1) {

            len_token = find_token( p, token ) ;

            if ( 0 == len_token ) break ; // no token in this line

            token_number ++ ;

            if ( FILENAME_INDEX == token_number ) {
                printf("filename/directory= %s\n", token ) ;
            }
            p += len_token ;
        }// for each token
    }// for each line
    fclose( fp ) ;
    return 0 ;
}
```

Token is a substring enclosed by space character

Exercise 3: find filename o directory [3]

```c
// return number of character processed
int find_token( char *s, char *token )
{
    char *front = s ;
// remove space
    while( isspace(*s) ){
        s++ ;
    }

    while( !isspace(*s) ){
        if ( '\0' == *s ) break ;
        *token++ = *s++ ;
    }
    *token = '\0' ;
    return s - front ;
}
```

```
-rwxrwxr-x    1 imsl     imsl        28325 Jul 13 21:20 a.out
```

-rwxrwxr-x

1

imsl

imsl

28325

Jul

13

21:20

a.out

Exercise 4: sorting on linked list [1]

Input file *data.txt*

```
auto     0
double   0
int      0
struct   0
break    0
else     0
long     0
switch   0
case     0
enum     0
register   0
typedef    0
char     0
extern     0
return   0
union    0
const    0
float    0
short    0
unsigned     0
continue     0
for      0
signed     0
void     0
default    0
goto     0
sizeof   0
volatile     0
do       0
if       0
static     0
```

Linked List-based

```c
typedef struct keyListEle {
    char word[16] ;          // keyword of C-language
    int  count ;             // number of keyword in a file
    struct keyListEle  *next ; // next entry in the chain
} keyListEleType ;
```

Array-based

```c
typedef struct key {
    char *word ; // keyword of C-language
    int  count ; // number of keyword in a file
} keyType ;

keyType  keytab[] = {
    {"auto"    ,0}, {"double",0}, {"int"     ,0}, {"struct"  ,0},
    {"break"   ,0}, {"else"  ,0}, {"long"    ,0}, {"switch"  ,0},
    {"case"    ,0}, {"enum"  ,0}, {"register",0}, {"typedef" ,0},
    {"char"    ,0}, {"extern",0}, {"return"  ,0}, {"union"   ,0},
    {"const"   ,0}, {"float" ,0}, {"short"   ,0}, {"unsigned",0},
    {"continue",0}, {"for"   ,0}, {"signed"  ,0}, {"void"    ,0},
    {"default" ,0}, {"goto"  ,0}, {"sizeof"  ,0}, {"volatile",0},
    {"do"      ,0}, {"if"    ,0}, {"static"  ,0}, {"while"   ,0}
} ;
```

use function *fscanf* to read keyword and count from *data.txt*

Exercise 4: sorting on linked list [2]

use function *fscanf* to read keyword and count from *data.txt*

```
fp = fopen( DEFAULT_OUT_FILE , "r") ;
assert(fp) ;
while( 1 ){
    numOfInput = fscanf(fp, "%s", word ) ;
    if ( EOF == numOfInput ) { break ; }
    if (  !numOfInput  ){
        printf("Error: first field must be a string\n");
        exit(1) ;
    }
    numOfInput = fscanf(fp, "%d\n", &count )  ;
    if ( !numOfInput ){
        printf("Error: second field must be an integer \n");
        exit(1) ;
    }

    unitEle = (keyListEleType*) malloc( sizeof(keyListEleType) ) ;
    assert( unitEle ) ;
    strcpy( unitEle->word, word ) ;
    unitEle->count = count ;
    unitEle->next = NULL ;
    if ( NULL == keytabList ){
        keytabList = unitEle ;
        elePtr = keytabList ;
    }else{
        elePtr->next = unitEle ;
        elePtr = elePtr->next ;
    }
}// forever
fclose( fp ) ;
```

create linked list

Exercise 4: sorting on linked list - bubble sort [3]

Given un-sorted array $a[0:n]$

$$
\begin{aligned}
&\textit{for}\ \ k = n:-1:1 \\
&\quad \textit{for}\ \ j = 0:1:k-1 \\
&\qquad \text{if } a[j] > a[j+1] \text{ then } swap\big(a[j], a[j+1]\big) \\
&\quad \textit{endfor} \\
&\textit{endfor}
\end{aligned}
$$

```
void  bubble_sort_key( keyListEleType  *keytabList, int n )
{
    int k , j ;
    keyListEleType  *elePtr = NULL ;
    keyListEleType  tmp ;

    for ( k = n-1 ; 0 < k ; k-- ){
        elePtr = keytabList ;
        for ( j = 0 ; j < k ; j++ ){
            assert( elePtr->next );
            if ( strcmp(elePtr->word, elePtr->next->word) > 0 ) {
// swap (elePtr->word, elePtr->next->word)
                strcpy( tmp.word, elePtr->word ) ;
                tmp.count = elePtr->count ;

                strcpy( elePtr->word, elePtr->next->word ) ;
                elePtr->count = elePtr->next->count ;

                strcpy( elePtr->next->word, tmp.word ) ;
                elePtr->next->count = tmp.count ;
            }
            elePtr = elePtr->next ;
        }// for j
    }// for k
}
```

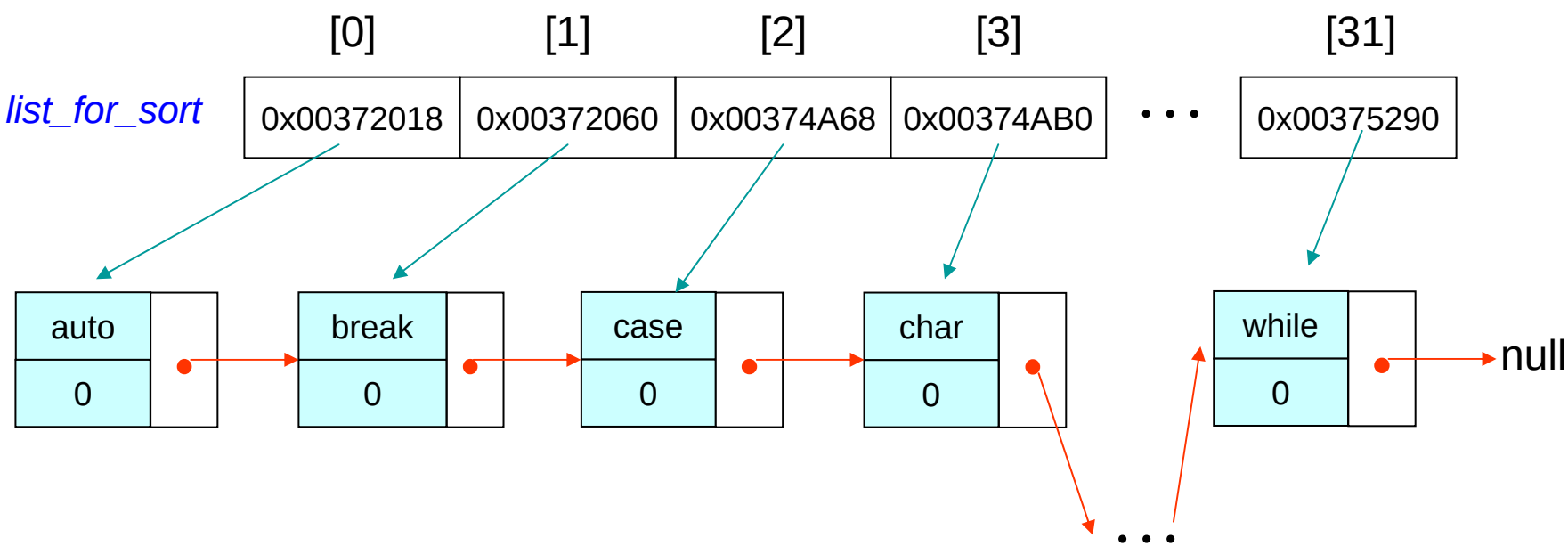
Hard-code *comparison*

Hard-code *swap*

Exercise 4: framework of sorting on linked list [4]

We use pointer array as auxiliary tool to do sorting

```
keyListEleType **list_for_sort ;
int i ;

list_for_sort = (keyListEleType **)malloc(sizeof(keyListEleType *)*list_length ) ;
assert( list_for_sort ) ;
for ( i = 0, elePtr = keytabList ; NULL != elePtr ; elePtr = elePtr->next, i++ ){
    list_for_sort[i] =  elePtr ;
}
```



Exercise 4: framework of sorting on linked list [5]

sort pointer array *keyListEleType *list_for_sort[]*

```
void quickSort( void *v[], int left, int right,
                int (*comp)(void*, void*) )
{
    int i, last ;

    if ( left >= right ){  /* do nothing f array contains */
        return ;          /* fewer than two elements */
    }
    swap(v, left, (left+right)/2 ) ; /* move partition elem */
    last = left ;                    /* to v[0] */
    for(i = left+1 ; i <= right ; i++){ /* partition */
        if ( (*comp)(v[i], v[left]) < 0 ){
            swap(v, ++last, i ) ;
        }
    }
    swap(v, left, last) ;   /* restore partition elem */
    quickSort( v, left, last-1 , comp ) ;
    quickSort( v, last+1, right, comp ) ;

}
```

quick sort in page 120 of textbook

User-defined *swap*

```
void swap( void *v[], int i, int j )
{
    keyListEleType  tmp ;

    keyListEleType *s = (keyListEleType *) v[i] ;
    keyListEleType *t = (keyListEleType *) v[j] ;

// tmp <-- *s
    strcpy( tmp.word, s->word ) ;
    tmp.count = s->count ;
// *s <-- *t
    strcpy( s->word, t->word ) ;
    s->count = t->count ;
// *t <-- tmp
    strcpy( t->word, tmp.word ) ;
    t->count = tmp.count ;
}
```

Exercise 4: framework of sorting on linked list [6]

Question: Can you explain why parameter of function *list_cmp_v2* is (*keyListEleType **), not (*keyListEleType ***) ?

```
int   list_cmp_v2( keyListEleType  *s, keyListEleType  *t )
{
    return strcmp( s->word, t->word ) ;
}


    list_for_sort = (keyListEleType **)malloc(sizeof(keyListEleType *)*list_length ) ;
    assert( list_for_sort ) ;
// copy reference of element into pointer array "list_for_sort"
    for ( i = 0, elePtr = keytabList ; NULL != elePtr ; elePtr = elePtr->next, i++ ){
        list_for_sort[i] =  elePtr ;
    }

    quickSort( (void **)list_for_sort, 0, list_length-1,
               (int (*)(void*, void*))list_cmp_v2 ) ;

// show sorted result
    for ( elePtr = keytabList ; NULL != elePtr ; elePtr = elePtr->next ){
        printf("%8s \t",  elePtr->word ) ;

    }
```

Exercise 4: framework of sorting on linked list [7]

Question: if we use *qsort* directly, it does not work, why?

```
int   list_cmp( keyListEleType  **s, keyListEleType  **t )
{
    return strcmp( (*s)->word, (*t)->word ) ;
}


list_for_sort = (keyListEleType **)malloc(sizeof(keyListEleType *)*list_length ) ;
assert( list_for_sort ) ;
for ( i = 0, elePtr = keytabList ; NULL != elePtr ; elePtr = elePtr->next, i++ ){
    list_for_sort[i] =  elePtr ;
}

qsort( (void*)list_for_sort, (size_t)list_length, (size_t) sizeof(keyListEleType *),
    (int (*)(const void*, const void*) ) list_cmp) ;


 for ( elePtr = keytabList ; NULL != elePtr ; elePtr = elePtr->next ){
     printf("%8s \t",  elePtr->word ) ;

 }
```

Exercise 5: 2-dimensional array (continuous case)

How to construct 2-dimensional array $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

```c
#include <stdio.h>

int main( int argc, char *argv[] )
{
    int A[2][3] = { 1, 2, 3, 4, 5, 6 } ;
    int i, j ;

    for( i =0 ; i < 2; i++){
        for(j=0 ; j < 3 ; j++){
            printf("A[%d][%d] = %d, address = 0x%p\n",
                i,j, A[i][j], &A[i][j]);
        }
    }
    return 0 ;
}
```

continuous array

```
A[0][0] = 1, address = 0x0012FF68
A[0][1] = 2, address = 0x0012FF6C
A[0][2] = 3, address = 0x0012FF70
A[1][0] = 4, address = 0x0012FF74
A[1][1] = 5, address = 0x0012FF78
A[1][2] = 6, address = 0x0012FF7C
Press any key to continue
```

Name	Value
⊟ A	0x0012ff68
⊟ [0]	0x0012ff68
[0]	1
[1]	2
[2]	3
⊟ [1]	0x0012ff74
[0]	4
[1]	5
[2]	6

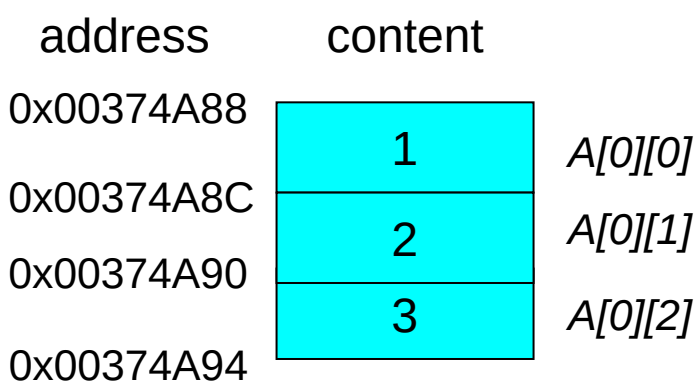
address	content	
0x0012FF68	1	*A[0][0]*
0x0012FF6C	2	*A[0][1]*
0x0012FF70	3	*A[0][2]*
0x0012FF74	4	*A[1][0]*
0x0012FF78	5	*A[1][1]*
0x0012FF7C	6	*A[1][2]*

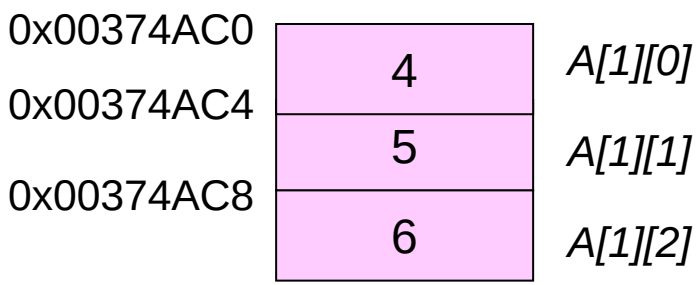
Exercise 5: 2-dimensional array (pointer-array case)

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

int main( int argc, char *argv[] )
{
    int* A[2] ; // A is a pointer array of size 2
    int i, j ;

    for( i =0 ; i < 2; i++){
        A[i] = (int*) malloc(sizeof(int)*3) ;
        assert( A[i] ) ;
    }
    A[0][0] = 1 ; A[0][1] = 2 ; A[0][2] = 3 ;
    A[1][0] = 4 ; A[1][1] = 5 ; A[1][2] = 6 ;

    for( i =0 ; i < 2; i++){
        for(j=0 ; j < 3 ; j++){
            printf("A[%d][%d] = %d, address = 0x%p\n",
                i,j, A[i][j], &A[i][j]);
        }
    }
    return 0 ;
}
```

Non-continuous array

```
A[0][0] = 1, address = 0x00374A88
A[0][1] = 2, address = 0x00374A8C
A[0][2] = 3, address = 0x00374A90
A[1][0] = 4, address = 0x00374AC0
A[1][1] = 5, address = 0x00374AC4
A[1][2] = 6, address = 0x00374AC8
Press any key to continue_
```





Exercise 5: difference between continuous case and pointer-array case [1]

```
int A[2][3] = { 1, 2, 3, 4, 5, 6 } ;
```

$A[row][col] = A[row \times 3 + col]$ maps 2D array to 1D array

row	*col*	*rowx3+col*
0	0	0
0	1	1
0	2	2
1	0	3
1	1	4
1	2	5

row-major

C-language

$A[row][col] = A[row + col \times 2]$

row	*col*	*row+colx2*
0	0	0
1	0	1
0	1	2
1	1	3
0	2	4
1	2	5

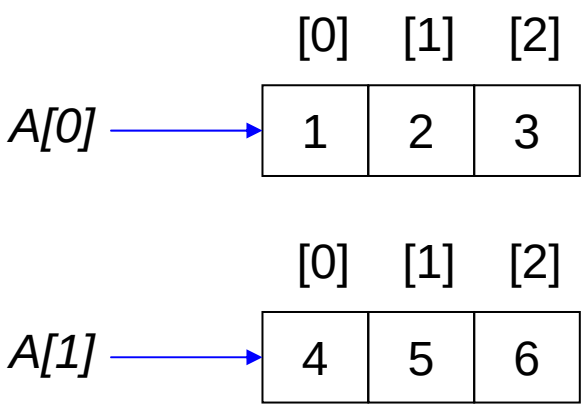
colume-major

Fortran-language

Exercise 5: difference between continuous case and pointer-array case [2]

pointer-array case

```
for( i =0 ; i < 2; i++){
    A[i] = (int*) malloc(sizeof(int)*3) ;
    assert( A[i] ) ;
}
```



Exactly row-major

Question 1: Can you create a 3-dimensional array by using pointer-array?

Question 2: multi-dimensional array starts from 0, can you implement a multi-dimensional array starting anywhere?

$$example: A[-1:5][3:7]$$

Exercise 6: union

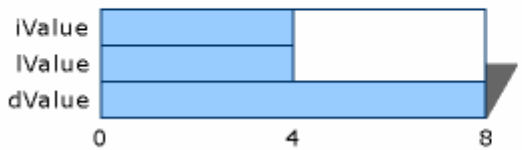
A **union** is a user-defined data or class type that, at any given time, contains only one object from its list of members.

```c
// example of union, from MSDN
#include <stdio.h>

union NumericType
{
    int         iValue;
    long        lValue;
    double      dValue;
};

int main(int argc, char* argv[])
{
    union NumericType Values ;

    Values.iValue = 10 ;

    printf("%d\n", Values.iValue);

    Values.dValue = 3.1416;

    printf("%f\n", Values.dValue);

    return 0 ;
}
```

```
10
3.141600
Press any key to continue
```

The NumericType union is arranged in memory (conceptually) as shown in the following figure.

Storage of Data in NumericType Union




```c
struct {
    char  *name ;
    int   flags ;
    int   utype ;
    union {
        int   ival ;
        float fval ;
        char  *sval ;
    } u ;
} symtab[ NSYM ] ;
```

Question: what is purpose of field *utype* ?

Exercise 7: remove comments in a file

in C-language, comment is delimited by a pair of **/*** and ***/**, in C++, comment starts from **//**, write a program to remove all comments of a given file. You can show result in screen or to another file.

Pseudo-code

```
for each line in a file
    if  line contains "//" not in a string, then
      remove remaining characters after "//".
    if  line contains "/*", then
      find conjugate pair "*/" and remove all characters in between
endfor
```

Question: can above pseudo-code identify following comment?

```
/* getInt: get next // integer from input to *pn */
```

Exercise 8: static variables

- the **static** keyword specifies that the variable has static duration (it is allocated when the program begins and deallocated when the program ends) and initializes it to 0 unless another value is specified.

- A variable declared **static** in a function retains its state between calls to that function.

- In recursive code, a static object or variable is guaranteed to have the same state in different instances of a block of code.

```c
// example of "static", from MSDN
#include <stdio.h>

void showstat( int curr )
{
   static int nStatic;    // Value of nStatic is retained
                          // between each function call
   nStatic += curr;
   printf( "nStatic is %d\n", nStatic ) ;
}

int main(int argc, char *argv[]) {
    for ( int i = 0; i < 5; i++ ){
      showstat( i );
    }
    return 0 ;
}
```

```
nStatic is 0
nStatic is 1
nStatic is 3
nStatic is 6
nStatic is 10
Press any key to continue
```

Question 1: what is *scope* of static variable?

Question 2: Can you access static variable out of the function?