Chapter 4 functions and program structure

Speaker: Lung-Sheng Chien

OutLine

- Example: grep
  - organization of program structure
  - compilation process
- Precedence and Associativity
- External/Internal objects
- Scope and Lifetime
- Preprocessor (前處理器)

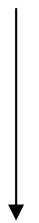
grep – print lines matching a pattern

Ah Love! could you and I with Fate conspire

To grasp this sorry Scheme of Things entire,

Would not we shatter it to bite -- and then

Re-mould it nearer to the Heart's Desire!



Search for the pattern "ould"

Ah Love! c**ould** you and I with Fate conspire

W**ould** not we shatter it to bite -- and then

Re-m**ould** it nearer to the Heart's Desire!

Structure of the program

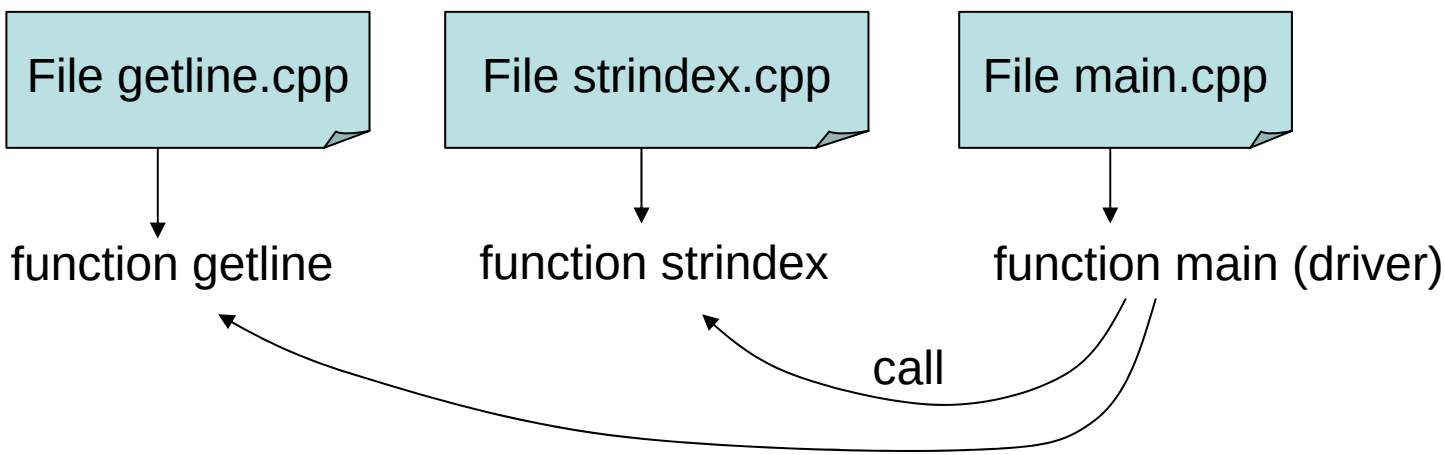
Pseudocode

```
while ( there is another line)
        if ( the line contains the pattern ) then
                print it
        endif
endwhile
```

Function getline

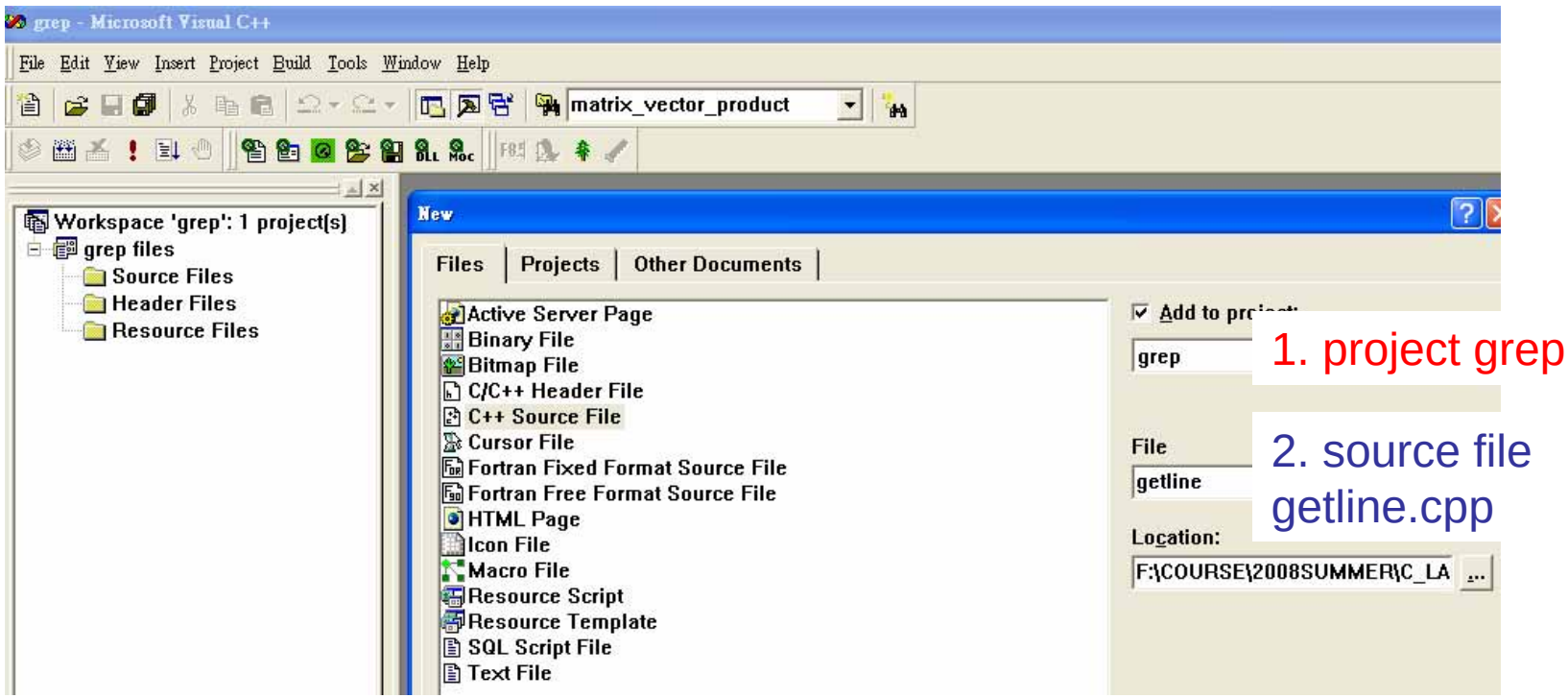
Function strindex

Function printf

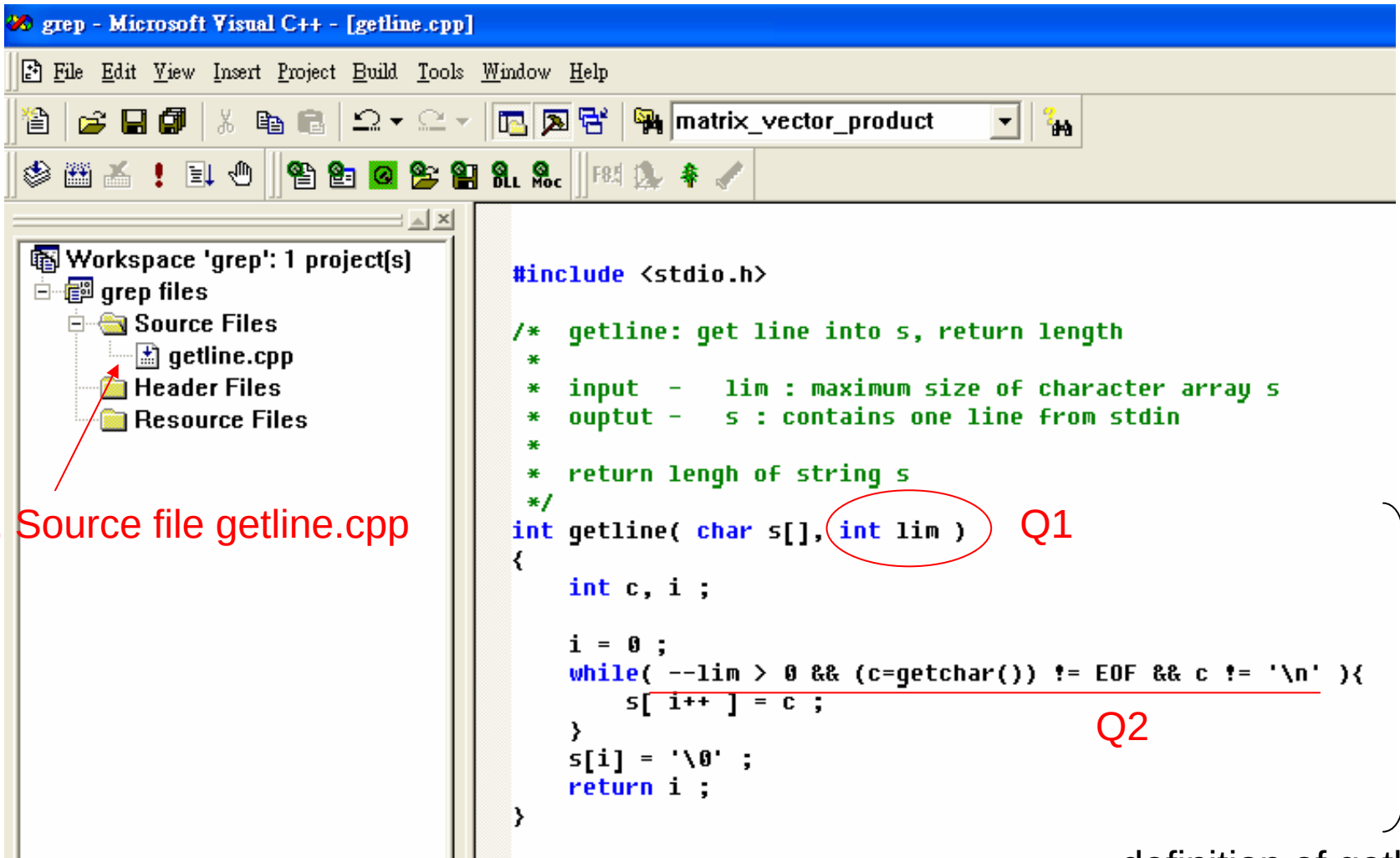


grep project – getline function [1]

Project → Add to Project → New



grep project – getline function [2]



1. Source file getline.cpp

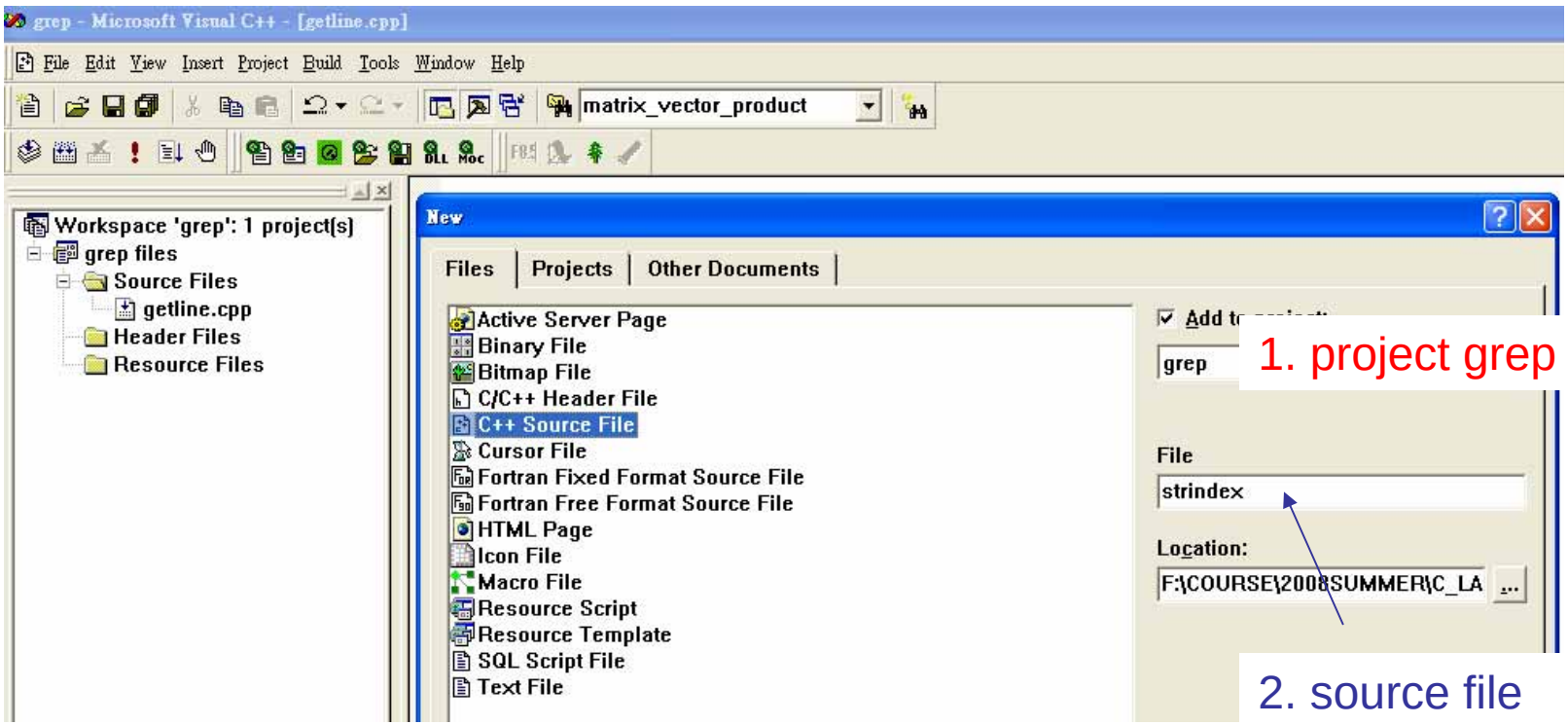
definition of getline

Question 1: why we need input parameter lim?

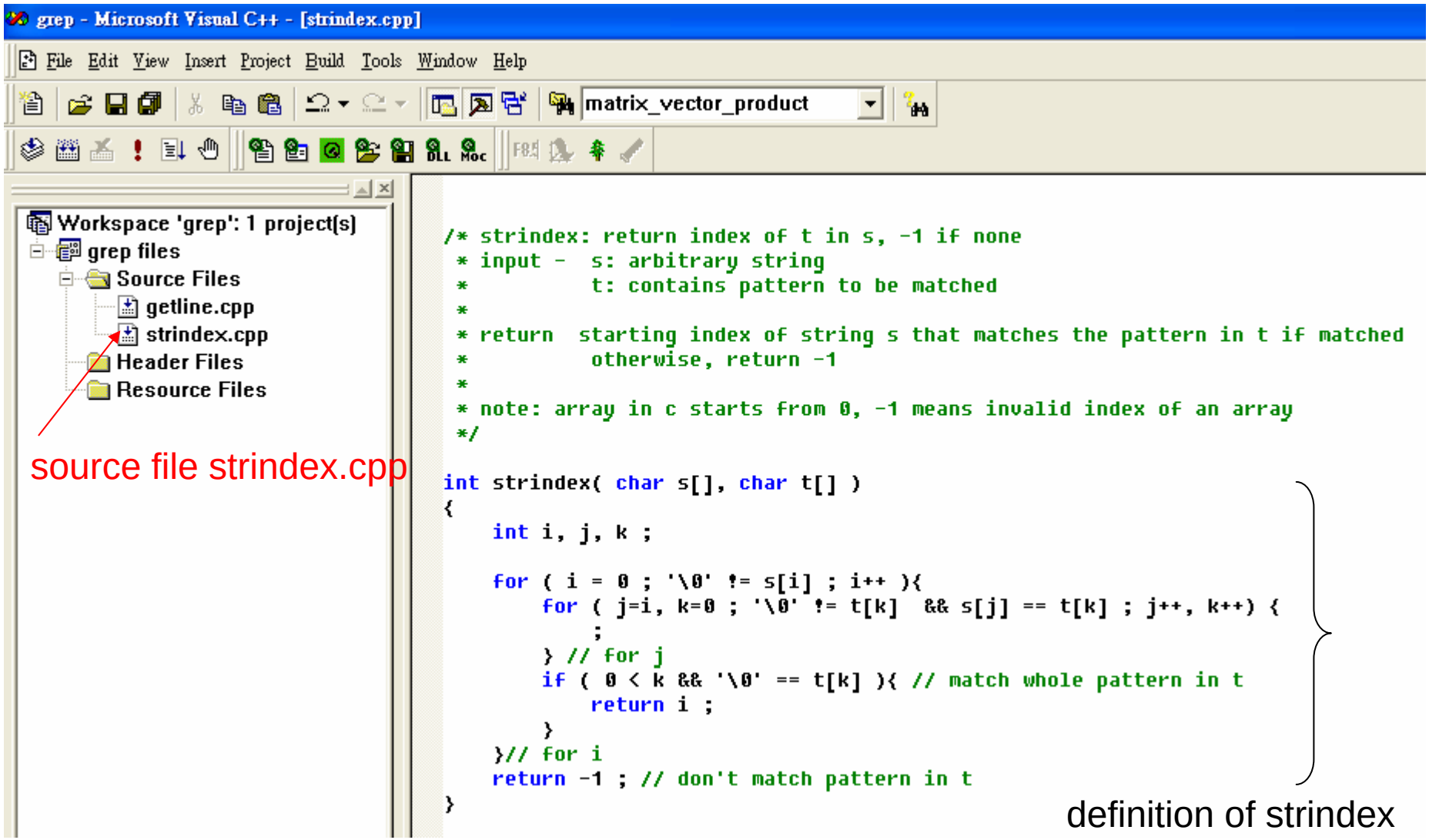
Question 2: what is execution order in predicate of while-loop?

grep project – strindex function [1]

Project → Add to Project → New



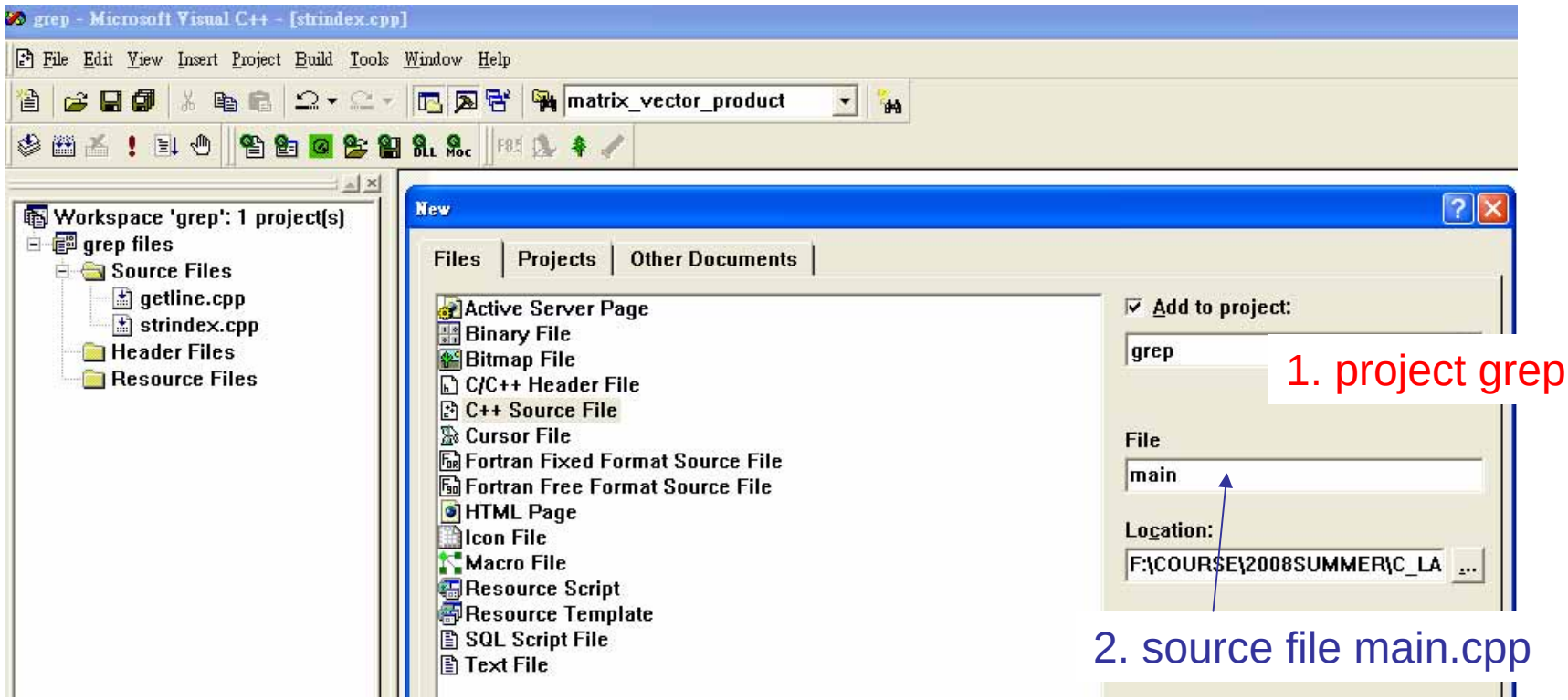
grep project – strindex function [2]



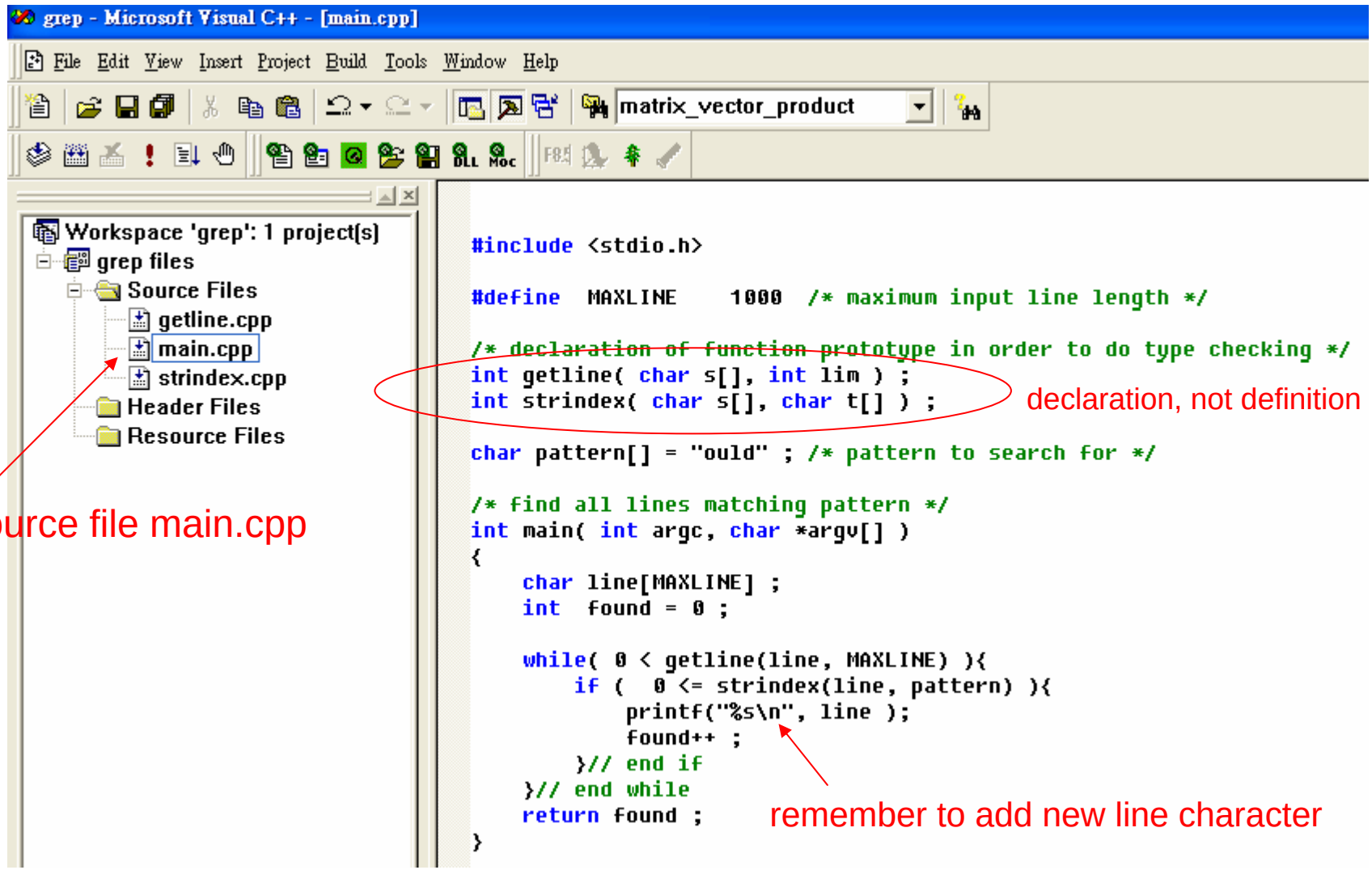
Question: what is procedure of function strindex ?

grep project – main function [1]

Project → Add to Project → New



grep project – main function [2]



Question: who does function **main** return **found** to ?

grep project – in Linux machine

Upload project **grep** by sftp first

```
[imsl@linux grep]$
[imsl@linux grep]$ ls
article.txt  getline.cpp  grep.dsw  grep.opt  main.cpp
Debug        grep.dsp     grep.ncb  grep.plg  strindex.cpp
[imsl@linux grep]$ icpc main.cpp strindex.cpp getline.cpp
[imsl@linux grep]$ ls
a.out        Debug        grep.dsp  grep.ncb  grep.plg  strindex.cpp
article.txt  getline.cpp  grep.dsw  grep.opt  main.cpp
[imsl@linux grep]$ ./a.out < article.txt
Ah Love! could you and I with Fate conspire
Would not we shatter it to bite -- and then
Re-mould it nearer to the Heart's Desire!e
[imsl@linux grep]$ echo $?
3
[imsl@linux grep]$
```

1. compile 3 source files

2. feed article.txt into a.out

3. 上一個指令的傳回值

article.txt

Ah Love! could you and I with Fate conspire

To grasp this sorry Scheme of Things entire,

Would not we shatter it to bite -- and then

Re-mould it nearer to the Heart's Desire!

“grep” is embedded in Linux machine

```
[imsl@linux grep]$
[imsl@linux grep]$ which grep
/bin/grep
[imsl@linux grep]$
[imsl@linux grep]$ man grep
```

Search manual of grep

```
GREP(1)                                                    GREP(1)

NAME
       grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
       grep [options] PATTERN [FILE...]
       grep [options] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
       Grep  searches the named input FILEs (or standard input if no files are
       named, or the file name - is given) for lines containing a match to the
       given PATTERN.  By default, grep prints the matching lines.

       In addition, two variant programs egrep and fgrep are available.  Egrep
       is the same as grep -E.  Fgrep is the same as grep -F.
```

1. pattern

2. File

```
[imsl@linux grep]$
[imsl@linux grep]$ grep  ould  article.txt
Ah Love! could you and I with Fate conspire
Would not we shatter it to bite -- and then
Re-mould it nearer to the Heart's Desire!t
[imsl@linux grep]$
```

Function definition

白話文

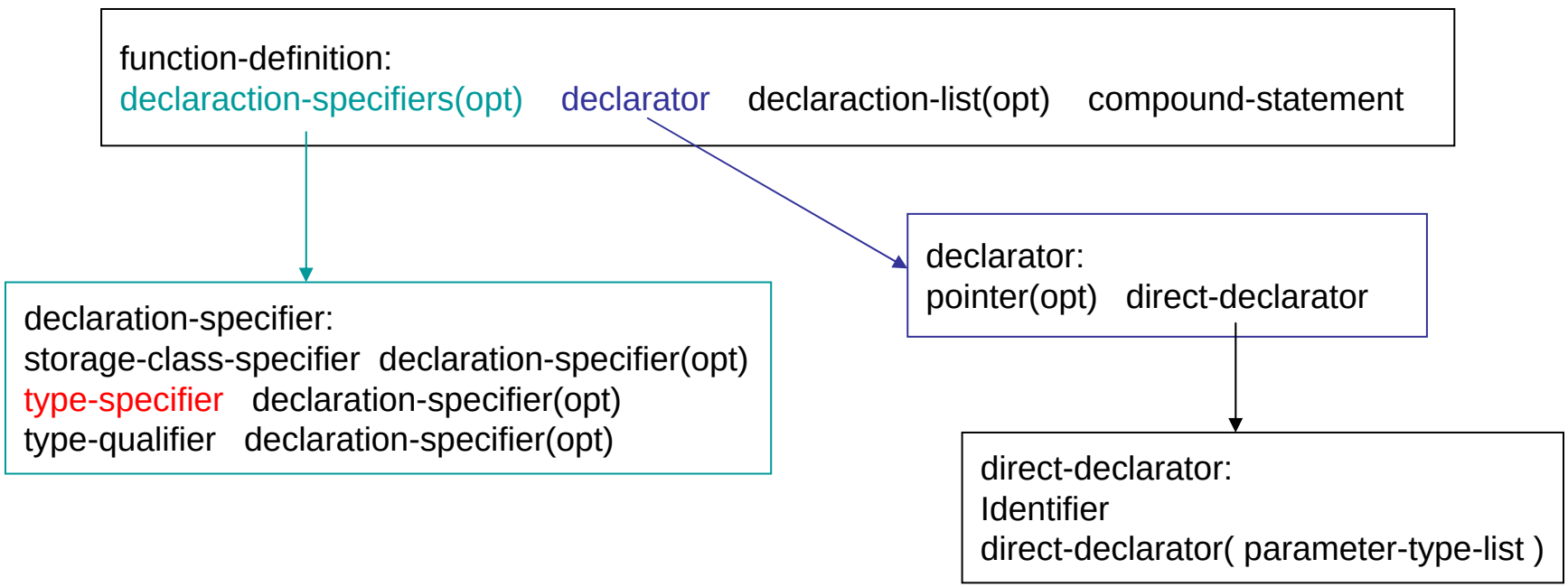
return-type function-name (argument declarations)

{

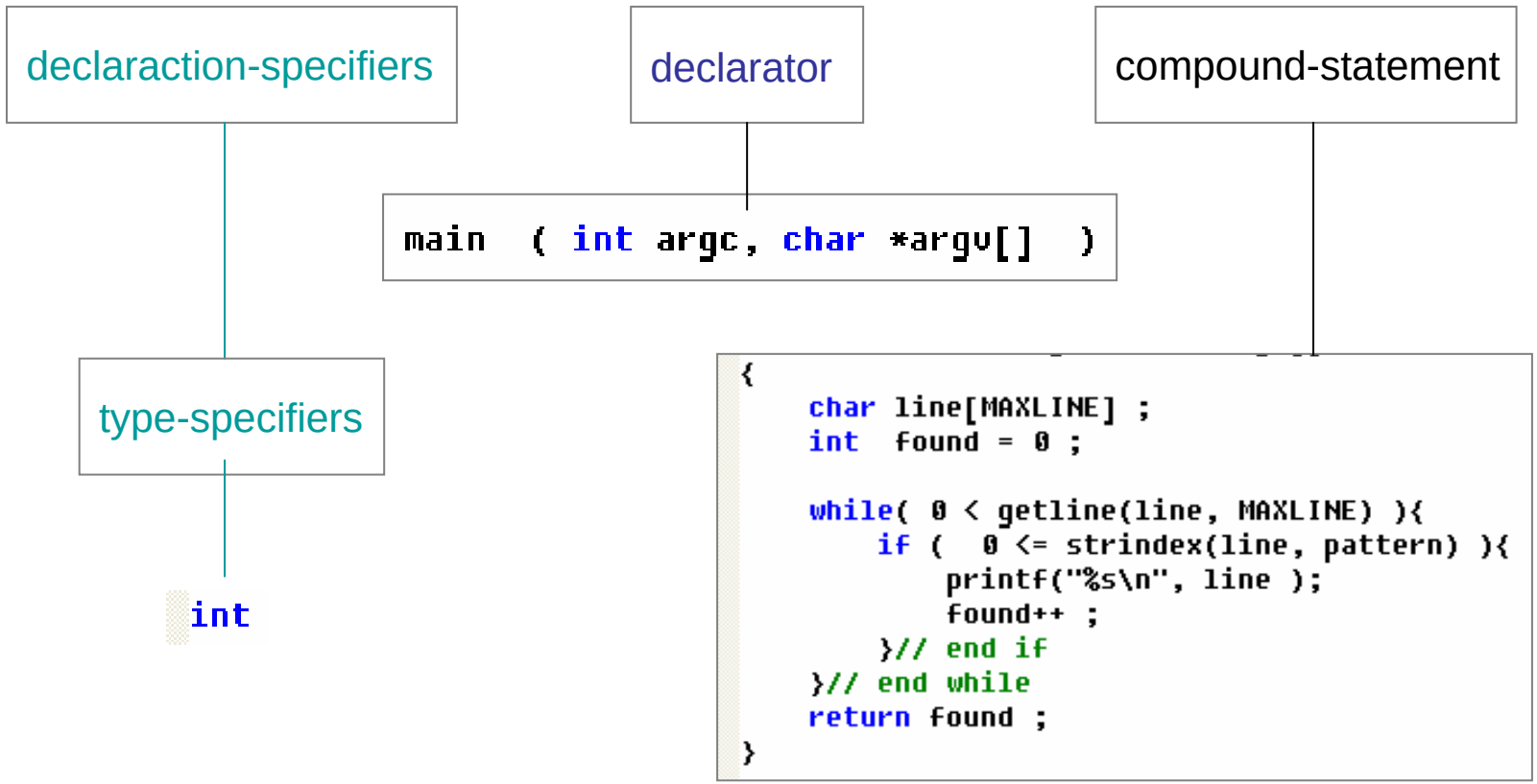
declarations and statements

}

文言文 (grammar, page 234, 235)



definition of main function [1]



definition of main function [2]

declarator:
Pointer(opt) direct-declarator

direct-declarator:
direct-declarator (parameter-type-list)

Identifier

main

```
int argc, char *argv[]
```

compound-statement:
{ declaration-list statement-list }

```
char line[MAXLINE] ;
int  found = 0 ;
```

```
while( 0 < getline(line, MAXLINE) ){
    if (  0 <= strindex(line, pattern) ){
        printf("%s\n", line );
        found++ ;
    }// end if
}// end while
return found ;
```

Why we need declaration of getline and strindex in main.cpp

```
#include <stdio.h>

#define  MAXLINE    1000  /* maximum input line length */

/* declaration of function prototype in order to do type checking */
int getline( char s[], int lim ) ;
int strindex( char s[], char t[] ) ;

char pattern[] = "ould" ; /* pattern to search for */

/* find all lines matching pattern */
int main( int argc, char *argv[]  )
{
    char line[MAXLINE] ;
    int  found = 0 ;

    while( 0 < getline(line, MAXLINE) ){
        if (  0 <= strindex(line, pattern) ){
            printf("%s\n", line );
            found++ ;
        }// end if
    }// end while
    return found ;
}
```

Compiler 是逐個檔案作編譯, 當他讀 main.cpp 且看見

getline(line, MAXLINE) 時需作 type checking, 可是 getline 的 defintion 是在 getline.cpp 內, 並不在 main.cpp, 所以我們需在 call getline 之前宣告 getline 的原型,

注意：compiler 只讀檔一次, 所以必需在compiler作 type checking 之前就告訴他函數的原型

編譯順序 -- [1] main.cpp

秀出編譯過程

```
[imsl@linux grep]$
[imsl@linux grep]$
[imsl@linux grep]$ g++ -v  main.cpp getline.cpp strindex.cpp
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --en
able-shared --enable-threads=posix --disable-checking --with-system-zlib --enable-__cxa_atexit --h
ost=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cclplus -v -D__GNUC__=3 -D__GNUC_MINOR__=2 -D__GNUC_PATC
HLEVEL__=2 -D__GXX_ABI_VERSION=102 -D__ELF__ -Dunix -D__gnu_linux__ -Dlinux -D__ELF__ -D__unix__ -
D__gnu_linux__ -D__linux__ -D__unix -D__linux -Asystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -
D_GNU_SOURCE -Acpu=i386 -Amachine=i386 -Di386 -D__i386 -D__i386__ -D__tune_i386__ main.cpp -D__GNU
G__=3 -D__DEPRECATED -D__EXCEPTIONS -quiet -dumpbase main.cpp -version -o /tmp/cc7Rx0Gb.s
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cpplib) (i386 Linux/ELF)
GNU C++ version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (i386-redhat-linux)
        compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
igmoring nonexistent directory "/usr/i386-redhat-linux/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/include/c++/3.2.2
 /usr/include/c++/3.2.2/i386-redhat-linux
 /usr/include/c++/3.2.2/backward
 /usr/local/include
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include
 /usr/include
End of search list.
 as -V -Qy -o /tmp/ccEg6xsf.o /tmp/cc7Rx0Gb.s
```

c++ 編譯器

先編譯 main.cpp

編譯爲組合語言

GNU assembler

翻譯組語成 object code, **ccEg6xsf.o** 檔

編譯順序 -- [2] getline.cpp

c++ 編譯器

```
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cclplus -v -D__GNUC__=3 -D__GNUC_MINOR__=2 -D__GNUC_PATC
HLEVEL__=2 -D__GXX_ABI_VERSION=102 -D__ELF__ -Dunix -D__gnu_linux__ -Dlinux -D__ELF__ -D__unix__ -
D__gnu_linux__ -D__linux__ -D__unix -D__linux -Asystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -
D_GNU_SOURCE -Acpu=i386 -Amachine=i386 -Di386 -D__i386 -D__i386__ -D__tune_i386__ getline.cpp -D__
GNUG__=3 -D__DEPRECATED -D__EXCEPTIONS -quiet -dumpbase getline.cpp -version -o /tmp/cc7Rx0Gb.s
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cpplib) (i386 Linux/ELF)
GNU C++ version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (i386-redhat-linux)
        compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
ignoring nonexistent directory "/usr/i386-redhat-linux/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/include/c++/3.2.2
 /usr/include/c++/3.2.2/i386-redhat-linux
 /usr/include/c++/3.2.2/backward
 /usr/local/include
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include
 /usr/include
End of search list.
 as -V -Qy -o /tmp/ccnNjsxn.o /tmp/cc7Rx0Gb.s
GNU assembler version 2.13.90.0.18 (i386-redhat-linux) using BFD version 2.13.90.0.18 20030206
```

編譯爲組合語言

再編譯 getline.cpp

翻譯組語成 object code, **ccnNjsxn.o** 檔

編譯順序 -- [3] strindex.cpp

c++ 編譯器

```
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cc1plus -v -D__GNUC__=3 -D__GNUC_MINOR__=2 -D__GNUC_PATC
HLEVEL__=2 -D__GXX_ABI_VERSION=102 -D__ELF__ -Dunix -D__gnu_linux__ -Dlinux -D__ELF__ -D__unix__ -
D__gnu_linux__ -D__linux__ -D__unix -D__linux -Asystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -
D_GNU_SOURCE -Acpu=i386 -Amachine=i386 -Di386 -D__i386 -D__i386__ -D__tune_i386__ strindex.cpp -D_
_GNUG__=3 -D__DEPRECATED -D__EXCEPTIONS -quiet -dumpbase strindex.cpp -version -o /tmp/cc7Rx0Gb.s
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cpplib) (i386 Linux/ELF)
GNU C++ version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (i386-redhat-linux)
        compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
ignoring nonexistent directory "/usr/i386-redhat-linux/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/include/c++/3.2.2
 /usr/include/c++/3.2.2/i386-redhat-linux
 /usr/include/c++/3.2.2/backward
 /usr/local/include
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include
 /usr/include
End of search list.
 as -V -Qy -o /tmp/ccQJUtuu.o /tmp/cc7Rx0Gb.s
GNU assembler version 2.13.90.0.18 (i386-redhat-linux) using BFD version 2.13.90.0.18 20030206
```

編譯爲組合語言

最後編譯 strindex.cpp

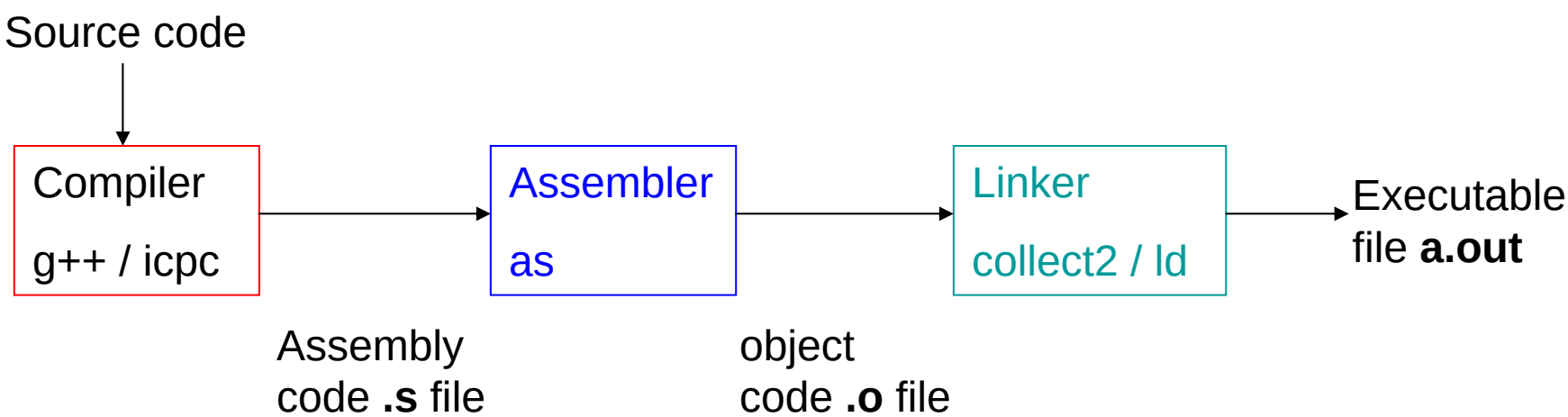
翻譯組語成 object code, **ccQJUtuu.o** 檔

連接程序 – linking process

Collect2 被作爲ld (GNU linker 連接器)

```
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/collect2 --eh-frame-hdr -m elf_i386 -dynamic-li
nker /lib/ld-linux.so.2 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../crt1.o /usr/lib
/gcc-lib/i386-redhat-linux/3.2.2/../../../crti.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2
/crtbegin.o -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2 -L/usr/lib/gcc-lib/i386-redhat-lin
ux/3.2.2/../../.. /tmp/ccuM0qvw.o /tmp/cc867imh.o /tmp/cc0Gl5x0.o -lstdc++ -lm -lgcc_s -l
gcc -lc -lgcc_s -lgcc /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/crtend.o /usr/lib/gcc-lib/
i386-redhat-linux/3.2.2/../../../crtn.o
[imsl@linux grep]$
```

main.cpp getline.cpp strindex.cpp



OutLine

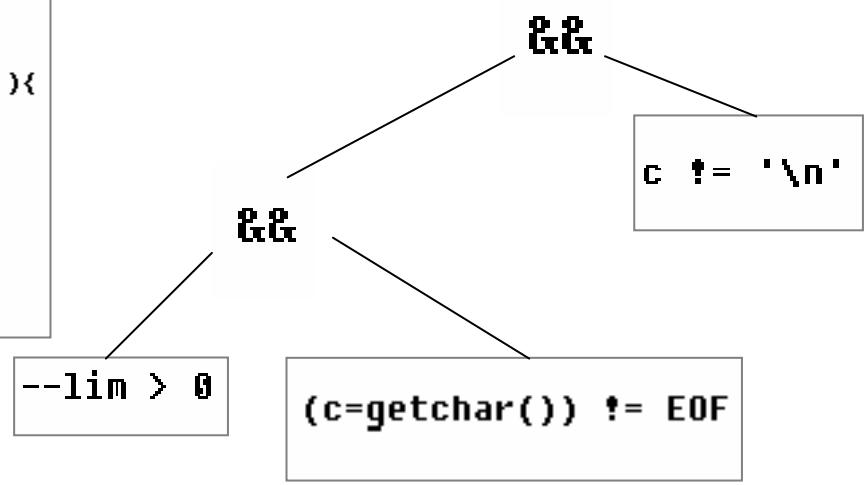
- Example: grep
- Precedence and Associativity
- External/Internal objects
- Scope and Lifetime
- Preprocessor (前處理器)

Precedence and Associativity of operators [1]

```c
#include <stdio.h>

/*  getline: get line into s, return length
 *
 *  input  -   lim : maximum size of character array s
 *  ouptut -   s : contains one line from stdin
 *
 *  return lengh of string s
 */
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while( --lim > 0 && (c=getchar()) != EOF && c != '\n' ){

        s[ i++ ] = c ;

    }
    s[i] = '\0' ;
    return i ;
}
```

Q2

Execution order



Question 2: what is execution order in predicate of while-loop?

Precedence and Associativity of operators, page 53 [2]

Precedence and Associativity of C Operators

Symbol1	Type of Operation	Associativity
[ ] ( ) . –> postfix ++ and postfix ––	Expression	Left to right
prefix ++ and prefix –– **sizeof & * + – ~ !**	Unary	Right to left
typecasts	Unary	Right to left
* / %	Multiplicative	Left to right
+ –	Additive	Left to right
<< >>	Bitwise shift	Left to right
< > <= >=	Relational	Left to right
== !=	Equality	Left to right
&	Bitwise-AND	Left to right
^	Bitwise-exclusive-OR	Left to right
|	Bitwise-inclusive-OR	Left to right
&&	Logical-AND	Left to right
||	Logical-OR	Left to right
? :	Conditional-expression	Right to left
= *= /= %= += –= <<= >>=&= ^= |=	Simple and compound assignment2	Right to left
,	Sequential evaluation	Left to right

arithmetic, 先乘除後加減

relational

logical

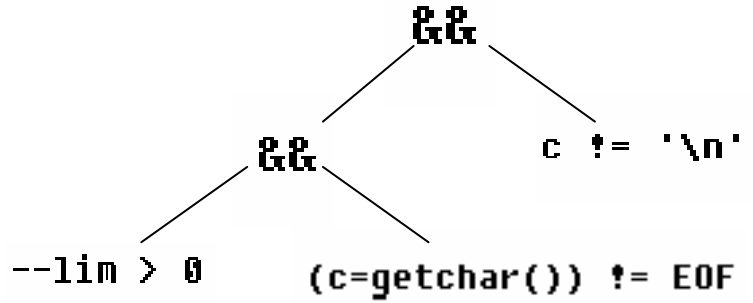
assignment

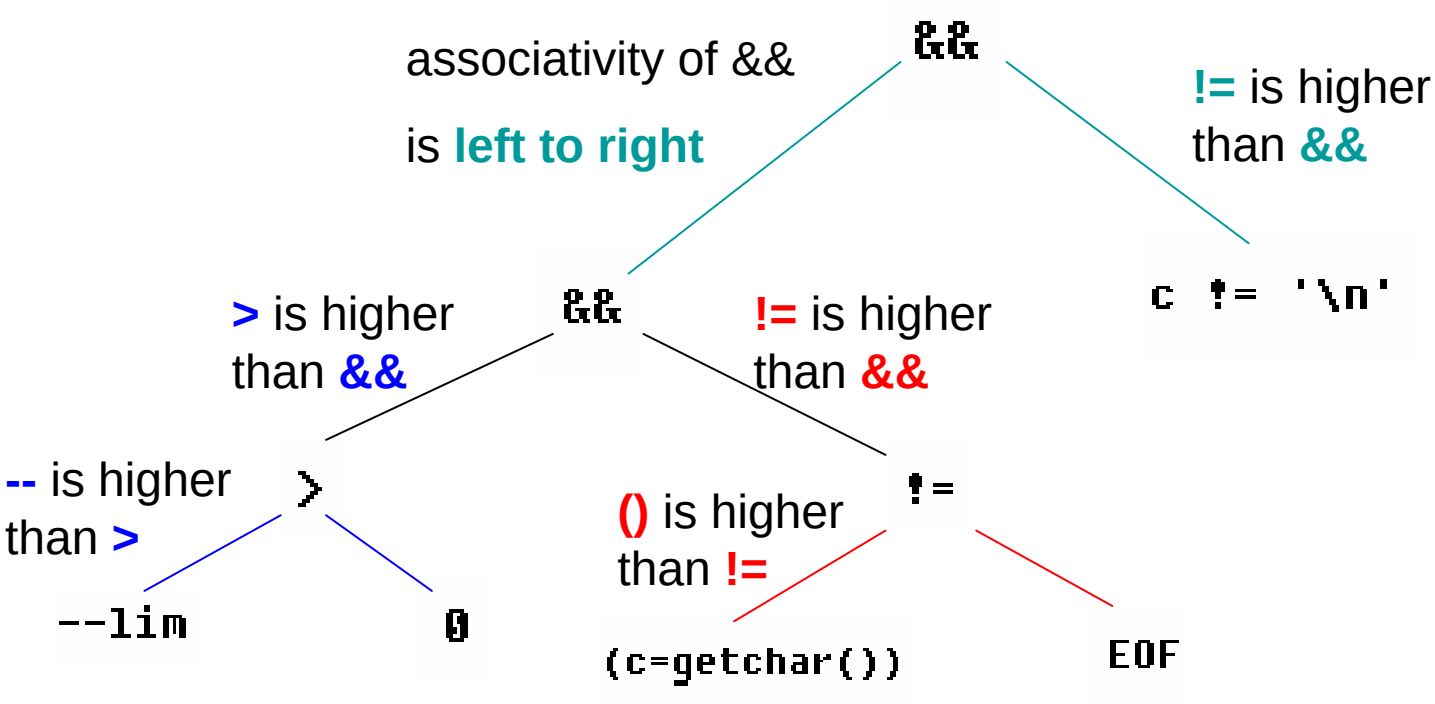
Precedence : () > arithmetic > relational > logical > assignment

Parsing tree of `--lim > 0 && (c=getchar()) != EOF && c != '\n'`

Execution order

1. bottom-up
2. Left subtree → right subtree





Three equivalent coding style

```
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(1){
        lim = lim - 1 ;
        if ( 0 < lim ){
            c = getchar() ;
            if ( EOF != c ){
                if ( '\n' != c ){
                    s[ i++ ] = c ;
                }else {
                    break ;
                }// endif ( '\n' != c )
            }else{
                break ;
            } // endif ( EOF != c )
        }else{
            break ;
        }// endif ( 0 < lim )
    }// forever
    s[i] = '\0' ;
    return i ;
}
```

c is well-defined

```
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while( --lim > 0 && (c=getchar()) != EOF && c != '\n' ){

        s[ i++ ] = c ;

    }
    s[i] = '\0' ;
    return i ;
}
```

```
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(  ( --lim > 0 ) &&
            ( (c=getchar()) != EOF  ) &&
            ( c != '\n' )
         ){

        s[ i++ ] = c ;

    }
    s[i] = '\0' ;
    return i ;
}
```

用括號表示優先次序

Question: what's procedure of function strindex ?

Given pattern string t

for each position i of string s

```
for ( i = 0 ; '\0' != s[i] ; i++ ){
```

check if substring $s\left[i : i + strlen(t)\right] = t$

```
for ( j=i, k=0 ; '\0' != t[k]  && s[j] == t[k] ; j++, k++) {
    ;
} // for j
```

endfor

if found, then return starting index of matched string in s

```
if ( 0 < k && '\0' == t[k] ){ // match whole pattern in t
    return i ;
}
```

otherwise, return -1

```
return -1 ; // don't match pattern in t
```

OutLine

- Example: grep
- Precedence and Associativity
- External/Internal objects
- Scope and Lifetime
- Preprocessor (前處理器)

External/internal objects [1]

- External objects
  - external variable (global variable): defined outside any function
  - functions
- Internal objects
  - variables inside function without declaring static
  - parameters of a function

Grammar in page 234

translation-unit:
    external-declaration
    translation-unit external-declaration

external-declaration:
    function-definition
    declaration

declaration:
declaration-sepcifiers init-declarator-list(opt)

External/internal objects [2]

file main.cpp

External variable's definition

External object: function **main**

```
#include <stdio.h>

#define  MAXLINE    1000  /* maximum input line length */

/* declaration of function prototype in order to do type checking */
int getline( char s[], int lim ) ;
int strindex( char s[], char t[] ) ;

char pattern[] = "ould" ; /* pattern to search for */

/* find all lines matching pattern */
int main( int argc, char *argv[] )
{

   char line[MAXLINE] ;
   int  found = 0 ;

   while( 0 < getline(line, MAXLINE) ){
       if (  0 <= strindex(line, pattern) ){
           printf("%s\n", line );
           found++ ;
       }// end if
   }// end while
   return found ;
}
```

Local variables

External object:

function **strindex**

```
int strindex( char s[], char t[] )
{
    int i, j, k ;

    for ( i = 0 ; '\0' != s[i] ; i++ ){

        for ( j=i, k=0 ; '\0' != t[k]  && s[j] == t[k] ; j++, k++) {
            ;
        } // for j

        if ( 0 < k && '\0' == t[k] ){ // match whole pattern in t
            return i ;
        }
    }// for i

    return -1 ; // don't match pattern in t
}
```

Local variables

file strindex.cpp

file getline.cpp

```
#include <stdio.h>

/*  getline: get line into s, return length
 *
 *  input  -   lim : maximum size of character array s
 *  ouptut -   s : contains one line from stdin
 *
 *  return lengh of string s
 */

int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(  --lim > 0  &&  (c=getchar()) != EOF && c != '\n' ){
        s[ i++ ] = c ;
    }
    s[i] = '\0' ;
    return i ;
}
```

Local variables

External object:

function **getline**

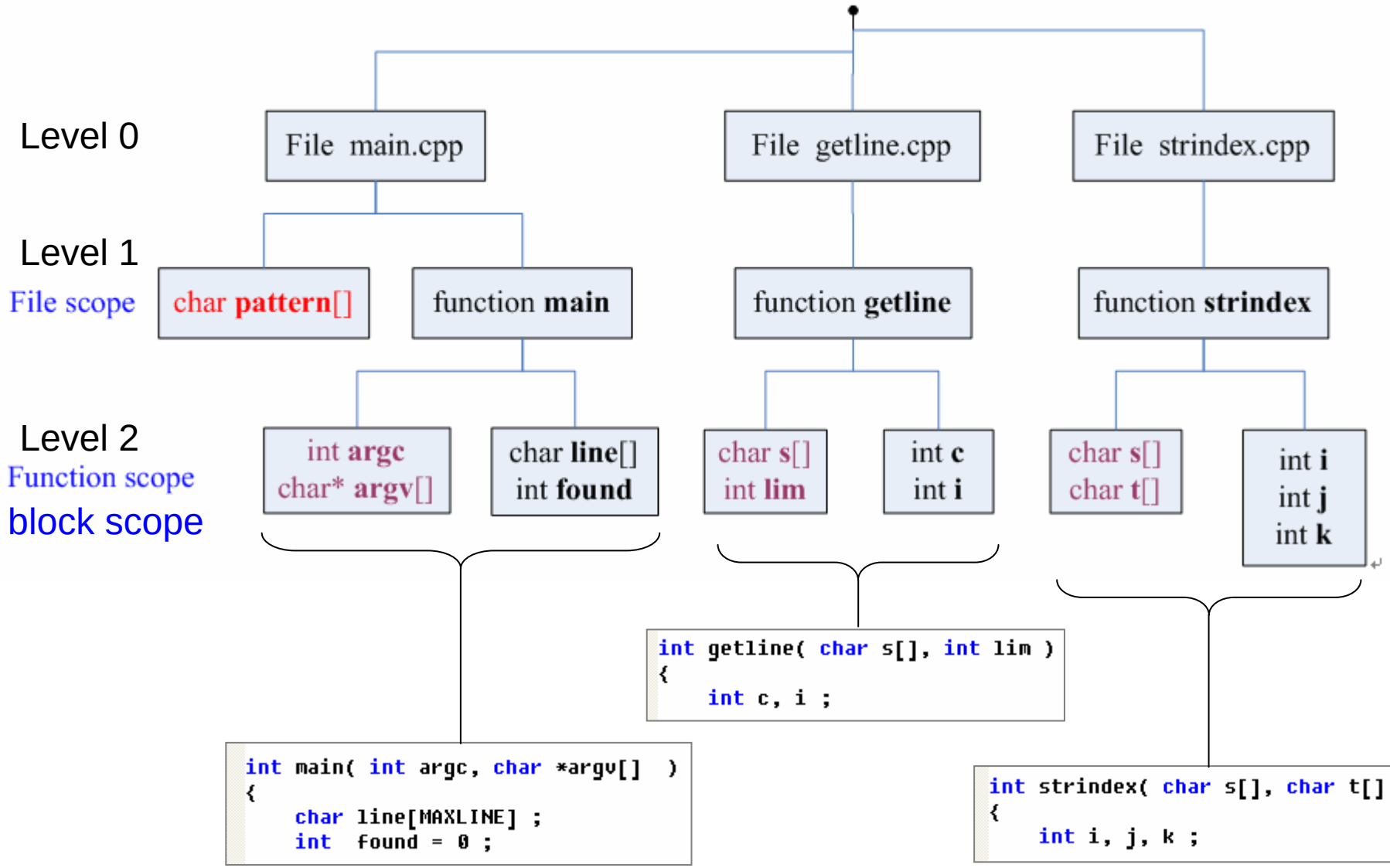
OutLine

- Example: grep
- Precedence and Associativity
- External/Internal objects
- Scope and Lifetime
- Preprocessor (前處理器)

Scope and visibility

- The scope of a name is the part of the program within which the name can be used.
- For automatic variable (automatic storage) declared at the beginning of a function, the scope is the function in which the name is declared.
- Local variables of the same name in different functions are unrelated. The same is true of the parameters of the function, which are local variables also.
- The scope of an external variable or a function lasts from the point at which it is declared to the end of the file being compiled.
- External variable in file1 can be accessed by functions in file2 if it is declared by keyword **extern** in file2.
- static external variable is encapsulated inside a file1, no other functions of file2 can access it.

Scope graph


Level 0
Level 1
File scope
Level 2
Function scope
block scope
File main.cpp
File getline.cpp
File strindex.cpp
char pattern[]
function main
function getline
function strindex
int argc
char* argv[]
char line[]
int found
char s[]
int lim
int c
int i
char s[]
char t[]
int i
int j
int k
int getline(char s[], int lim)
{
int c, i ;
int main(int argc, char *argv[])
{
char line[MAXLINE] ;
int found = 0 ;
int strindex(char s[], char t[])
{
int i, j, k ;

Lifetime (生命周期) [1]

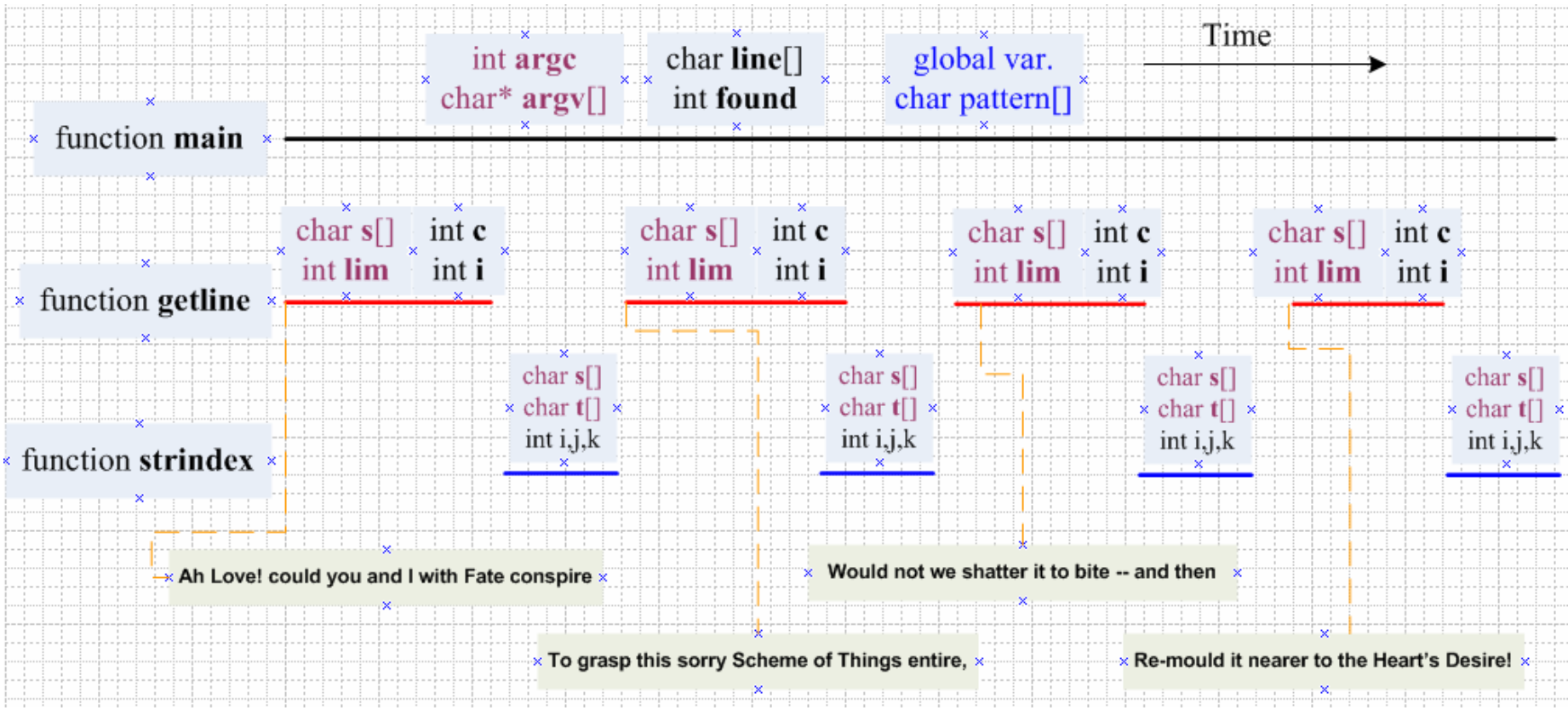
- “Lifetime” is the period during execution of a program in which a variable or function exists. The storage duration of the identifier determines its lifetime.

- Identifiers with static storage duration (global lifetime) are alive during (main) program execution and die when program terminates.
  Example: global variable, static variable

- Identifiers, which are defined inside a function, with automatic storage duration (local lifetime) is allocated new storage each time the function is called and die when function returns.
  Example: local variable, function parameter

- Although an identifier with a *global lifetime* exists throughout the execution of the program, it may not be visible in all parts of the program.

Lifetime (生命周期) [2]



MSDN Library: search “extern”

```
// extern_StorageClassSpecified.c
#include <stdio.h>

void other( void );

int main()
{
    // Reference to i, defined below:
    extern int i;

    // Initial value is zero; a is visible only within main:
    static int a;

    // b is stored in a register, if possible:
    register int b = 0;

    // Default storage class is auto:
    int c = 0;

    // Values printed are 1, 0, 0, 0:
    printf_s( "%d\n%d\n%d\n%d\n", i, a, b, c );
    other();
    return;
}
int i = 1;
```

global variable

```
void other( void )
{
    // Address of global i assigned to pointer variable:
    static int *external_i = &i;

    // i is redefined; global i no longer visible:
    int i = 16;

    // This a is visible only within the other function:
    static int a = 2;

    a += 2;
    // Values printed are 16, 4, and 1:
    printf_s( "%d\n%d\n%d\n", i, a, *external_i );
}
```

extern and static [1]

- A variable declared with the **extern** storage-class specifier is a reference to a variable with the same name defined at the external level in any of the source files of the program.
- The internal **extern** declaration is used to make the external-level variable definition visible within the block.
- Unless otherwise declared at the external level, a variable declared with the **extern** keyword is visible only in the block in which it is declared.

extern and static [2]

file main.cpp

```
#include <stdio.h>
#define  MAXLINE    1000  /* maximum input line length */

/* declaration of function prototype in order to do type checking */
int getline( char s[], int lim ) ;

int strindex( char s[] ) ;

char pattern[] = "ould" ;  definition  earch for */

/* find all lines matching pattern */
int main( int argc, char *argv[]  )
{
    char line[MAXLINE] ;
    int  found = 0 ;

    while( 0 < getline(line, MAXLINE) ){

        if (  0 <= strindex(line) ){

            printf("%s\n", line );
            found++ ;
        }// end if
    }// end while
    return found ;
}
```

file strindex.cpp

```
extern char pattern[] ;   declaration

int strindex( char s[] )
{
    int i, j, k ;
    for ( i = 0 ; '\0' != s[i] ; i++ ){

        for ( j=i, k=0 ; '\0' != pattern[k]  && s[j] == pattern[k] ; j++, k++) {
            ;
        } // for j

        if ( 0 < k && '\0' == pattern[k] ){ // match whole pattern in t
            return i ;
        }
    }// for i

    return -1 ; // don't match pattern in t
}
```

OutLine

- Example: grep
- Precedence and Associativity
- External/Internal objects
- Scope and Lifetime
- Preprocessor (前處理器)

C ProProcessor (cpp) [1]

- File inclusion
  #include “filename”
  #include <filename>

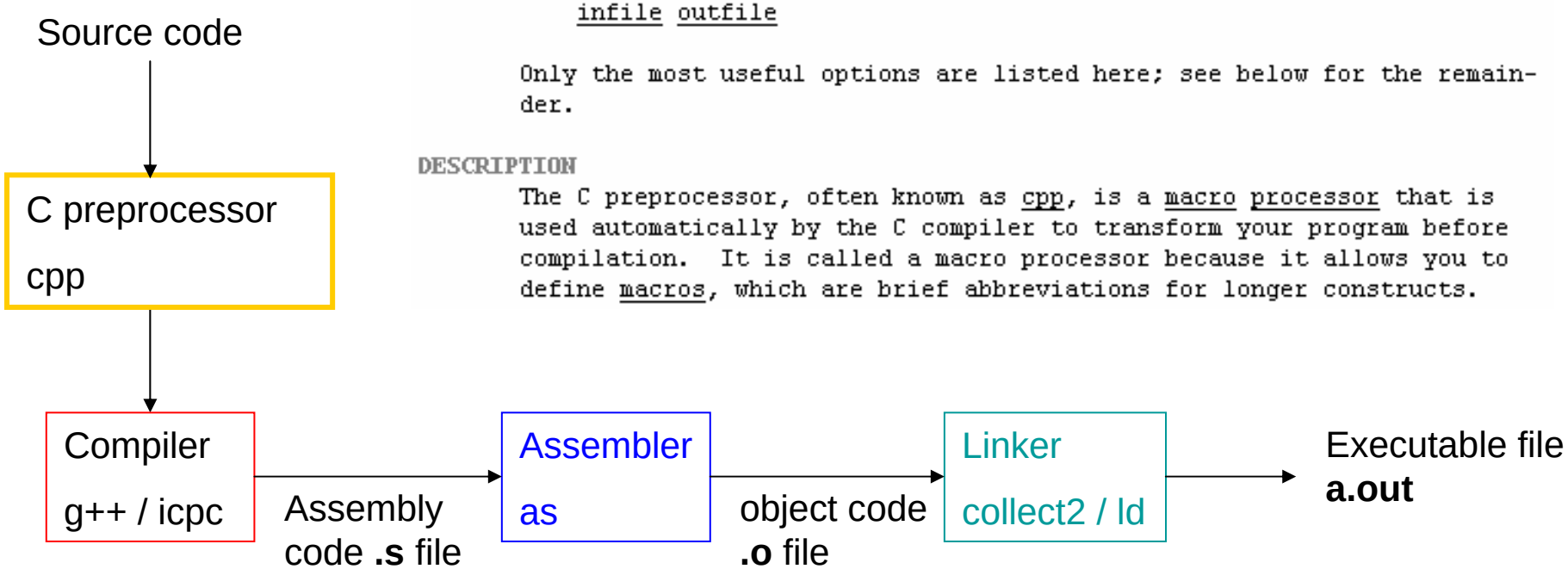
- Macro (巨集) substitution
  #define forever for (; ;) // infinite loop
  #define square(x) (x)*(x)

- Conditional inclusion
  #ifndef _INC_STDIO
  #define _INC_STDIO
  /* content of stdio.h */
  #endif /* _INC_STDIO */

C PrePrecessor (cpp) [2]

```
[imsl@linux grep]$
[imsl@linux grep]$ man cpp
```

```
NAME
       cpp - The C Preprocessor

SYNOPSIS
       cpp [-Dmacro[=defn]...] [-Umacro]
           [-Idir...] [-Wwarn...]
           [-Mâ-MM] [-MG] [-MF filename]
           [-MP] [-MQ target...] [-MT target...]
           [-x language] [-std=standard]
           infile outfile

       Only the most useful options are listed here; see below for the remain-
       der.

DESCRIPTION
       The C preprocessor, often known as cpp, is a macro processor that is
       used automatically by the C compiler to transform your program before
       compilation.  It is called a macro processor because it allows you to
       define macros, which are brief abbreviations for longer constructs.
```



CPP – macro substitution

Question: Why no ***max*** operation in *math.h*

```
#define  MAX( A, B ) ( (A) > (B) ? (A) : (B) )

int main( int argc, char* argv[] )
{
    int x,y,z ;

    x = 3 ;
    y = 5 ;

    z = MAX( x, y ) ;

    return 0 ;
}
```

```
[imsl@linux max]$
[imsl@linux max]$ cpp main.cpp
# 1 "main.cpp"
# 1 "<built-in>"
# 1 "<command line>"
# 1 "main.cpp"



int main( int argc, char* argv[] )
{
        int x,y,z ;

        x = 3 ;
        y = 5 ;

        z = ( (x) > (y) ? (x) : (y) ) ;


        return 0 ;
}
[imsl@linux max]$
```

cpp handles this but the compiler don't see this statement

CPP - #include “filename” [1]

```c
/*
    Program 1: entrace to C-language

    Ref: page 6 in the book, "The C programing Language, Kernighan".

    As you print out the message "Hello World" in the screen, then you
    enter the palace of C-language.
*/

#include <stdio.h>

#define  HELLO_STRING   "hello, world\n"

int main( int argc, char* argv[] )
{
    printf( HELLO_STRING );

    return 0 ;
}
```

cpp 會複製 stdio.h 到此檔案以便 compiler 作 type checking

cpp 將替換 HELLO_STRING

```
[imsl@linux helloWorld]$ ls
a.out  helloWorld.dsp  helloWorld.ncb  helloWorld.plg  output.txt
Debug  helloWorld.dsw  helloWorld.opt  main.cpp
[imsl@linux helloWorld]$
[imsl@linux helloWorld]$ cpp main.cpp  -o main_cpp.txt
[imsl@linux helloWorld]$
[imsl@linux helloWorld]$ cat main_cpp.txt | more
```

輸出至檔案main_cpp.txt

讀檔案 main_cpp.txt 一次一頁

CPP - #include “filename” [2]

```
[imsl@linux helloWorld]$ cat main_cpp.txt | more
# 1 "main.cpp"
# 1 "<built-in>"
# 1 "<command line>"
# 1 "main.cpp"
# 12 "main.cpp"
# 1 "/usr/include/stdio.h" 1 3
# 28 "/usr/include/stdio.h" 3
# 1 "/usr/include/features.h" 1 3
# 291 "/usr/include/features.h" 3
# 1 "/usr/include/sys/cdefs.h" 1 3
# 292 "/usr/include/features.h" 2 3
# 314 "/usr/include/features.h" 3
# 1 "/usr/include/gnu/stubs.h" 1 3
# 315 "/usr/include/features.h" 2 3
# 29 "/usr/include/stdio.h" 2 3

extern "C" {



# 1 "/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include/stdde
# 213 "/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include/std
typedef unsigned int size_t;
# 35 "/usr/include/stdio.h" 2 3

# 1 "/usr/include/bits/types.h" 1 3
# 28 "/usr/include/bits/types.h" 3
# 1 "/usr/include/bits/wordsize.h" 1 3
# 29 "/usr/include/bits/types.h" 2 3


# 1 "/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include/stdde
# 32 "/usr/include/bits/types.h" 2 3
--More--
```

按空白鍵接下一頁

```
[imsl@linux imsl]$ cat /usr/include/stdio.h | more
/* Define ISO C stdio on top of C++ iostreams.
   Copyright (C) 1991,1994-1999,2000,01,02 Free Software Foundation, Inc.
   This file is part of the GNU C Library.

   The GNU C Library is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2.1 of the License, or (at your option) any later version.

   The GNU C Library is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
   Lesser General Public License for more details.

   You should have received a copy of the GNU Lesser General Public
   License along with the GNU C Library; if not, write to the Free
   Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
   02111-1307 USA.  */

/*
 *      ISO C99 Standard: 7.19 Input/output     <stdio.h>
 */

#ifndef _STDIO_H

#if !defined __need_FILE && !defined __need___FILE
# define _STDIO_H       1
# include <features.h>
__BEGIN_DECLS

# define __need_size_t
# define __need_NULL
--More--
```

line 28, include feature.h

Recursively file inclusion, 需滿足因果律

CPP - #include “filename” [3]

```
typedef _G_fpos64_t fpos64_t;
# 138 "/usr/include/stdio.h" 3
# 1 "/usr/include/bits/stdio_lim.h" 1 3
# 139 "/usr/include/stdio.h" 2 3



extern struct _IO_FILE *stdin;
extern struct _IO_FILE *stdout;
extern struct _IO_FILE *stderr;
```

standard input

standard output

```
extern int fprintf (FILE * __stream,
                    const char * __format, ...) ;

extern int printf (const char * __format, ...) ;

extern int sprintf (char * __s,
                    const char * __format, ...) ;


extern int vfprintf (FILE * __s, const char * __format,
                     __gnuc_va_list __arg) ;

extern int vprintf (const char * __format, __gnuc_va_list __arg)
     ;

extern int vsprintf (char * __s, const char * __format,
                     __gnuc_va_list __arg) ;
```

```
extern int fgetc (FILE *__stream) ;
extern int getc (FILE *__stream) ;


extern int getchar (void) ;
```

```
extern void funlockfile (FILE *__stream) ;
# 679 "/usr/include/stdio.h" 3
}
# 13 "main.cpp" 2



int main( int argc, char* argv[] )
{
        printf( "hello, world\n" );

        return 0 ;
}
[imsl@linux helloWorld]$
```

CPP – conditional inclusion [1]

```
#include <stdio.h>

int main( int argc, char* argv[] )
{
    #if defined(_WIN32) || defined(__WIN32__)
            printf("This is Win32 Application\n");
    #else
            printf("This is Linux Application\n");
    #endif

    return 0 ;
}
```

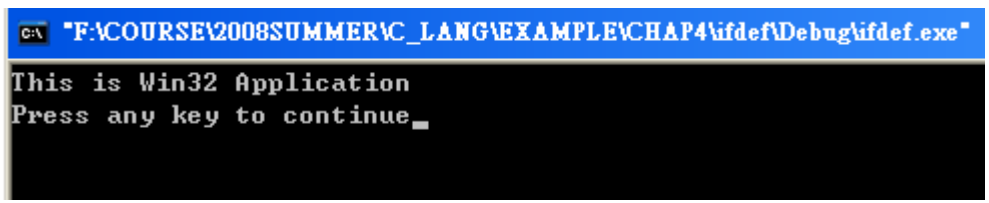
如果巨集 _WIN32 或巨集 __WIN32__ 有定義, 則執行程式碼

printf("This is Win32 Application\n");

否則執行程式碼

printf("This is Linux Application\n");

Result in Visual Studio



Result in Linux

```
[imsl@linux ifdef]$ ls
Debug  ifdef.dsp  ifdef.dsw  ifdef.ncb  ifdef.opt  ifdef.plg  main.cpp
[imsl@linux ifdef]$ icpc main.cpp
[imsl@linux ifdef]$ ls
a.out  Debug  ifdef.dsp  ifdef.dsw  ifdef.ncb  ifdef.opt  ifdef.plg  main.cpp
[imsl@linux ifdef]$ ./a.out
This is Linux Application
[imsl@linux ifdef]$
```

CPP – conditional inclusion [2]

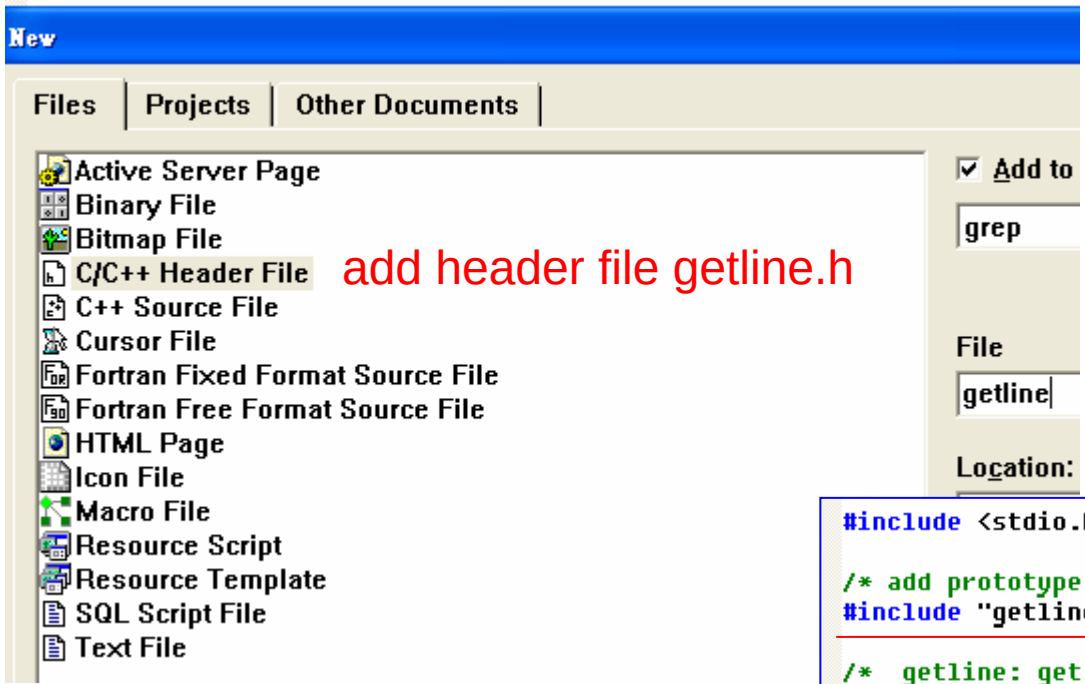
```
[imsl@linux ifdef]$ icpc -D_WIN32=1 main.cpp
[imsl@linux ifdef]$ ./a.out
This is Win32 Application
[imsl@linux ifdef]$
```

Macro **_WIN32** is always defined in Visual Studio

```
[imsl@linux ifdef]$ man gcc
GCC(1)                                                    GNU

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-câ-Sâ-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-pedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] infile...
```

```
[imsl@linux ifdef]$
[imsl@linux ifdef]$ man icpc
      :
      :
      :
  -D<name>[=<value>]
         Define  the  macro  <name>  and  associate it with the specified
         <value>.  Equivalent to a #define preprocessing directive.

         DEFAULT: -D<name> defines the macro <name> with a <value> of  1.
```

CPP – causality (因果律) [1]



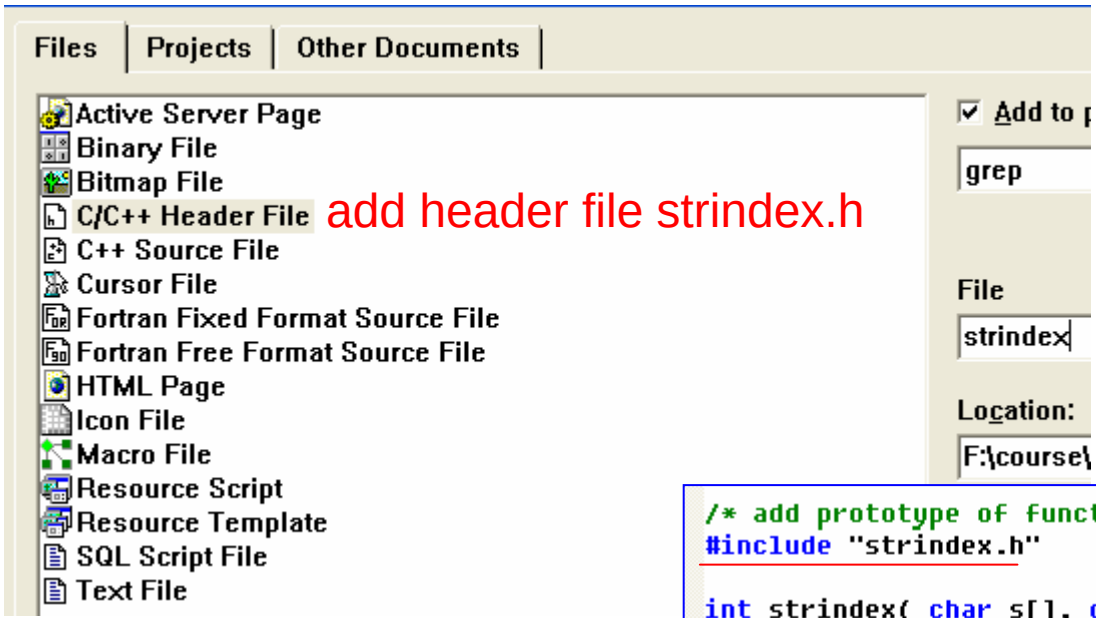
add header file getline.h

content of getline.h

```c
/* test for nested file inclusion */
#include "strindex.h"

/* prototype of function getline */
int getline( char s[], int lim ) ;
```

```c
#include <stdio.h>

/* add prototype of function getline */
#include "getline.h"

/*  getline: get line into s, return length
 *
 *  input  -   lim : maximum size of character array s
 *  ouptut -   s : contains one line from stdin
 *
 *  return lengh of string s
 */
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(  --lim > 0  &&  (c=getchar()) != EOF && c != '\n'  ){
        s[ i++ ] = c ;
    }
    s[i] = '\0' ;
    return i ;
}
```

CPP – causality (因果律) [2]

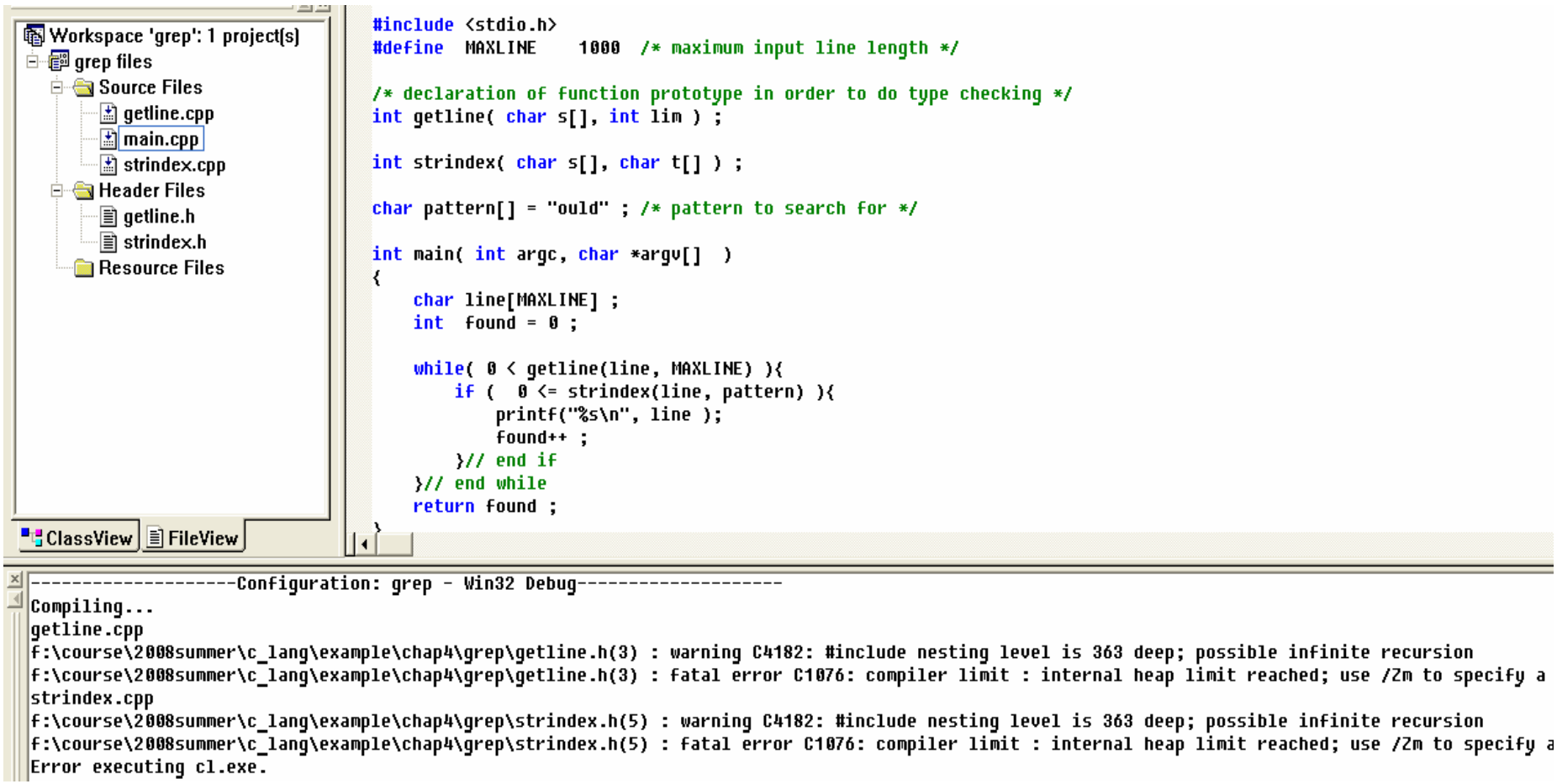


content of strindex.h

```c
/* test for nested file inclusion */
#include "getline.h"

/* prototype of function strindex */
int strindex( char s[], char t[] ) ;
```

```c
/* add prototype of function getline */
#include "strindex.h"

int strindex( char s[], char t[] )
{
    int i, j, k ;

    for ( i = 0 ; '\0' != s[i] ; i++ ){

        for ( j=i, k=0 ; '\0' != t[k]  && s[j] == t[k] ; j++, k++) {
            ;
        } // for j

        if ( 0 < k && '\0' == t[k] ){ // match whole pattern in t
            return i ;
        }
    }// for i

    return -1 ; // don't match pattern in t
}
```

CPP – causality (因果律) [3]



Question 1: why we have infinite recursion?

Question 2: How to solve infinite recursion?

CPP – causality (因果律) [4]

getline.h

```c
#ifndef  GETLINE_H

#define  GETLINE_H

/* test for nested file inclusion */
    #include "strindex.h"

/* prototype of function getline */
    int getline( char s[], int lim ) ;

#endif  // GETLINE_H
```

strindex.h

```c
#ifndef  STRINDEX_H

    #define  STRINDEX_H

/* test for nested file inclusion */
    #include "getline.h"

/* prototype of function strindex */
    int strindex( char s[], char t[] ) ;

#endif  // STRINDEX_H
```

getline.cpp

```c
#include <stdio.h>

/* add prototype of function getline */
#include "getline.h"

/*  getline: get line into s, return length
 *
 *  input  -   lim : maximum size of character array s
 *  ouptut -   s : contains one line from stdin
 *
 *  return lengh of string s
 */
int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(  --lim > 0  &&  (c=getchar()) != EOF && c != '\n'  ){
        s[ i++ ] = c ;
    }
    s[i] = '\0' ;
    return i ;
}
```

strindex.cpp

```c
/* add prototype of function getline */
#include "strindex.h"

int strindex( char s[], char t[] )
{
    int i, j, k ;

    for ( i = 0 ; '\0' != s[i] ; i++ ){

        for ( j=i, k=0 ; '\0' != t[k]  && s[j] == t[k] ; j++, k++) {
            ;
        } // for j

        if ( 0 < k && '\0' == t[k] ){ // match whole pattern in t
            return i ;
        }
    }// for i

    return -1 ; // don't match pattern in t
}
```

CPP – causality (因果律) [5]
preprocess source strindex.cpp