C Programming Essentials
Unit 2: Structured Programming

CHAPTER 5: STRUCTURED PROGRAMMING (LOOPS)

DR. ERIC CHOU IEEE SENIOR MEMBER

LECTURE 1

While Loop

Statements

- Expressions, when terminated by a semicolon, become statements.
- Examples

```
X = 5; I++; IsPrime(c); c = 5 * ( f - 32 ) / 9;
```

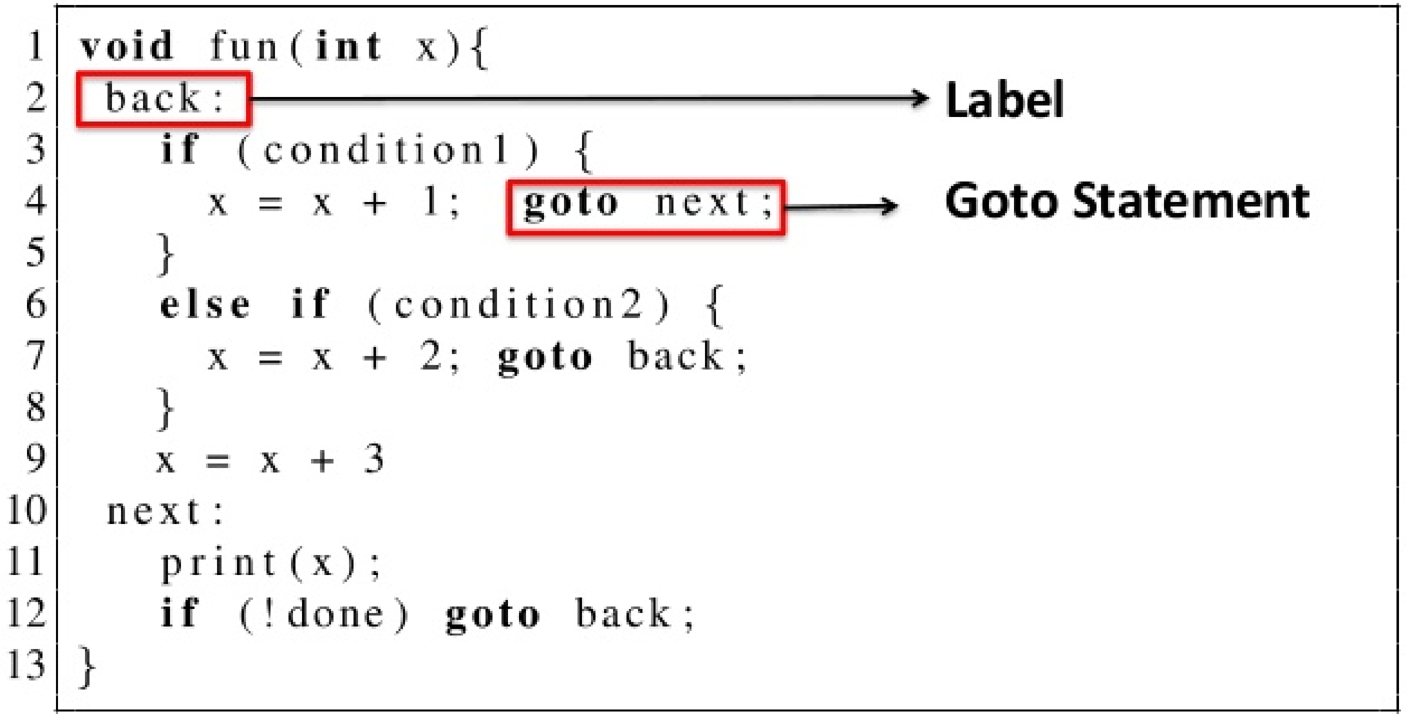
- One can construct compound statements by putting statements and declarations in Braces { and }. Also called blocks.
- Example

```
if ( rad > 0.0 ){
  float area = pi * rad * rad;
  float peri = 2 * pi * rad;
  printf( “Area = %f\n” , area );
  printf( “Peri = %f\n” , peri );
}
else
  printf( “Negative radius\n”);
```

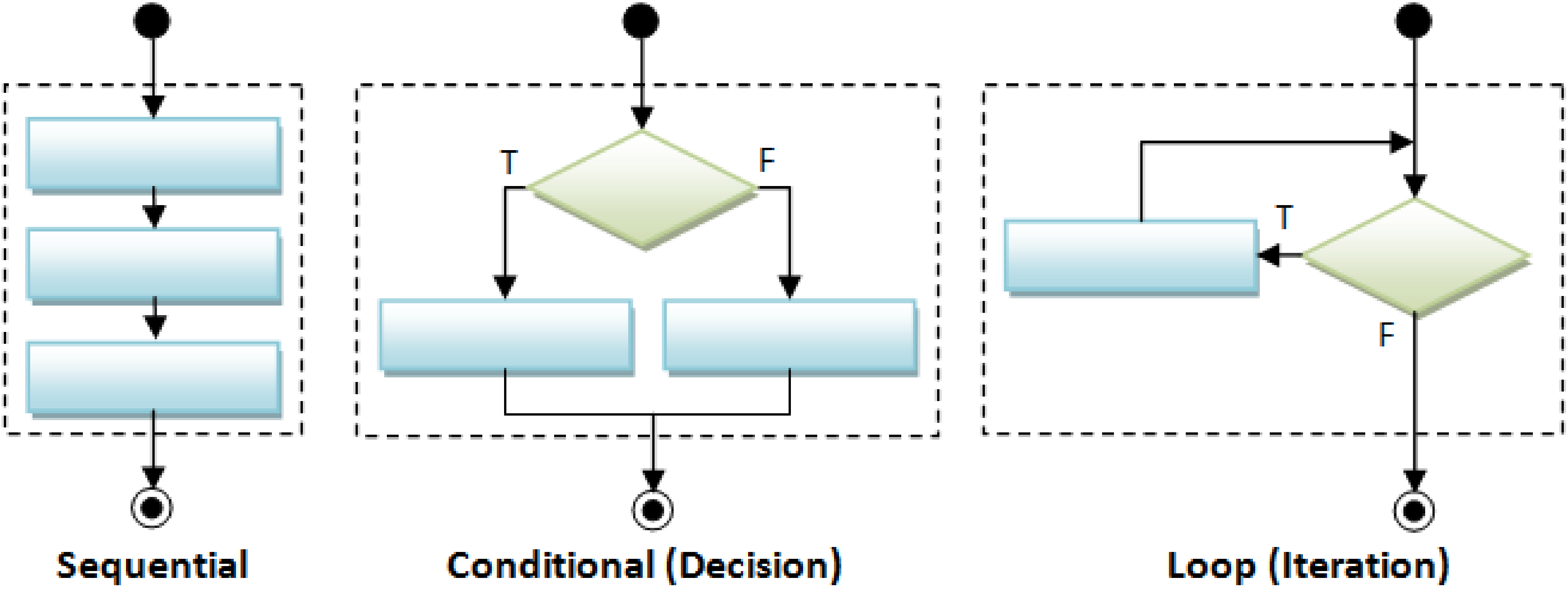
Labeled statements

- One can give a label to any statement. The form is
- Identifier: statement
- There are other labeled statements like case and default, to be used with switch statement.
- In C, Basic. But not in Java or modern languages.
- Example (On the right side)

Example



```
void fun(int x){
 back:
   if (condition1) {
     x = x + 1; goto next;
   }
   else if (condition2) {
     x = x + 2; goto back;
   }
   x = x + 3
 next:
   print(x);
   if (!done) goto back;
}
```

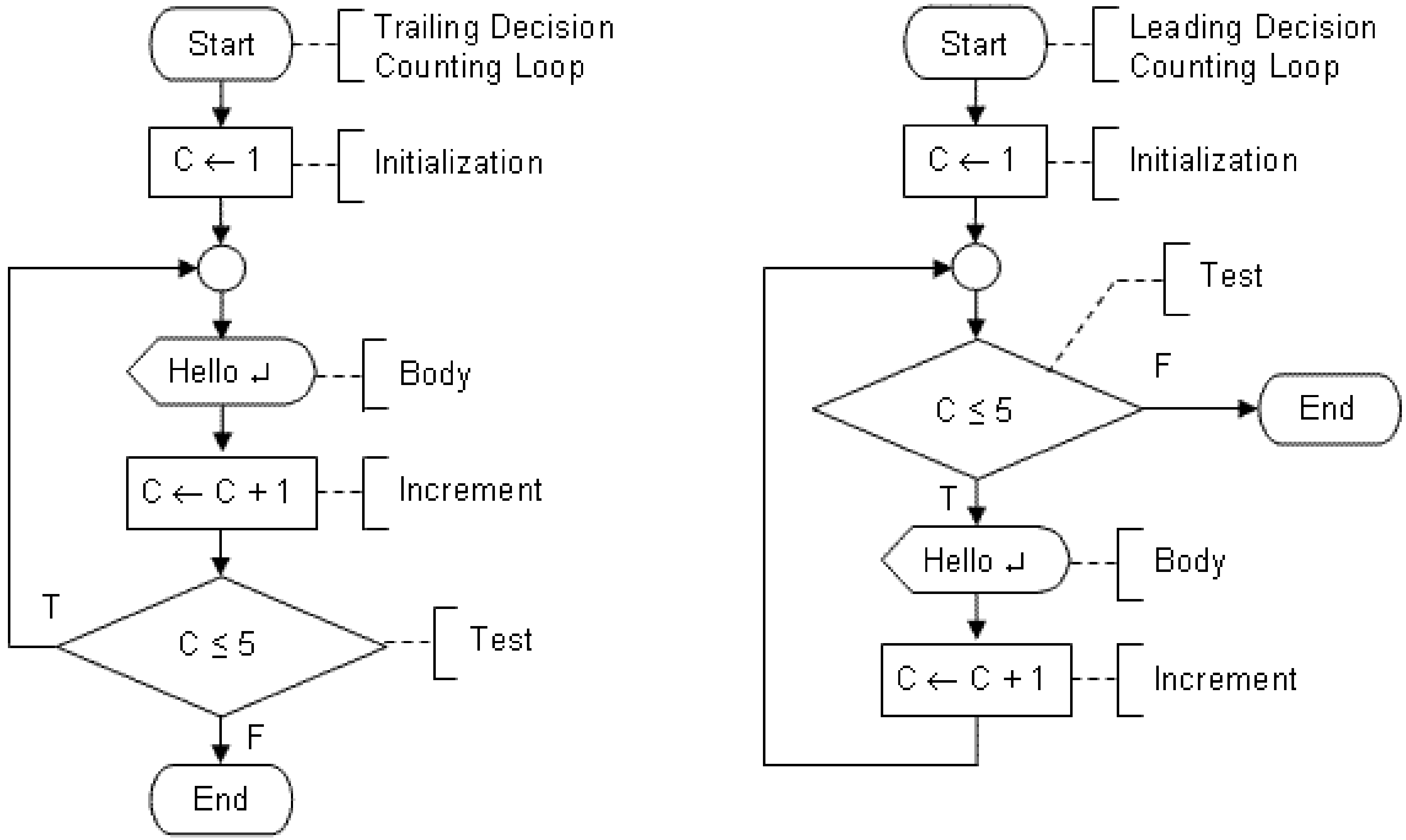


Sequential

Conditional (Decision)

Loop (Iteration)

Structured Programming


Start
Trailing Decision Counting Loop
C ← 1
Initialization
Hello ↵
Body
C ← C + 1
Increment
T
C ≤ 5
Test
F
End
Start
Leading Decision Counting Loop
C ← 1
Initialization
Test
F
C ≤ 5
End
T
Hello ↵
Body
C ← C + 1
Increment

Iterations

- The three types of loops are given below.

  while (*expression*) *statement*

  for (*expression1*$_{opt}$; *expression2*$_{opt}$; *expression3*$_{opt}$) { *statements; }*

  do { *statements }* while(*expression*);

- In while loop the expression (which must be arithmetic) is evaluated and if the value is nonzero, then the statement is executed. The process is continued till the expression becomes zero. The expression is evaluated before the iteration.
- For statement is equivalent to

```
expression1;
while ( expression2 ) {
     statement
   expression3;
}
```

While and do-while

Syntax is

- while(expr) stmt
  - As long as expr is true, keep on executing the stmt in
  - loop
- do stmt while(expr)
  - Same as before, except that the stmt is executed
  - at least once.

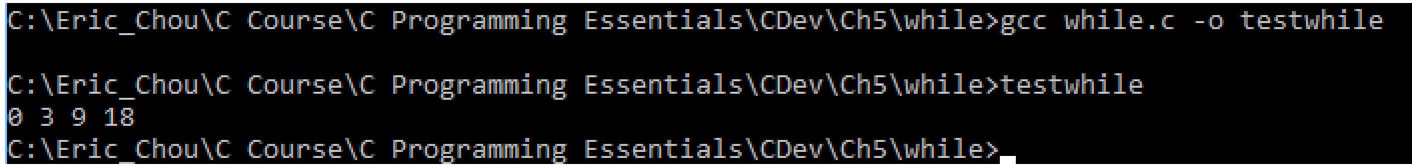
Example:

```
int i=0, x=0 ;
while (i<10) {
      if(i%3==0) {
          x += i;
          printf("%d ", x);
       }
      ++i;
 }
```

What is the output here?

while.c

	Init	+=		+=		+=		+=	
i	0		1 2	3	4 5	6	7 8	9	10
x	0	0		3		9		18	✗
		0		3		9		18	



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\while>gcc while.c -o testwhile

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\while>testwhile
0 3 9 18
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\while>_
```

LECTURE 2

Loops for Repetition

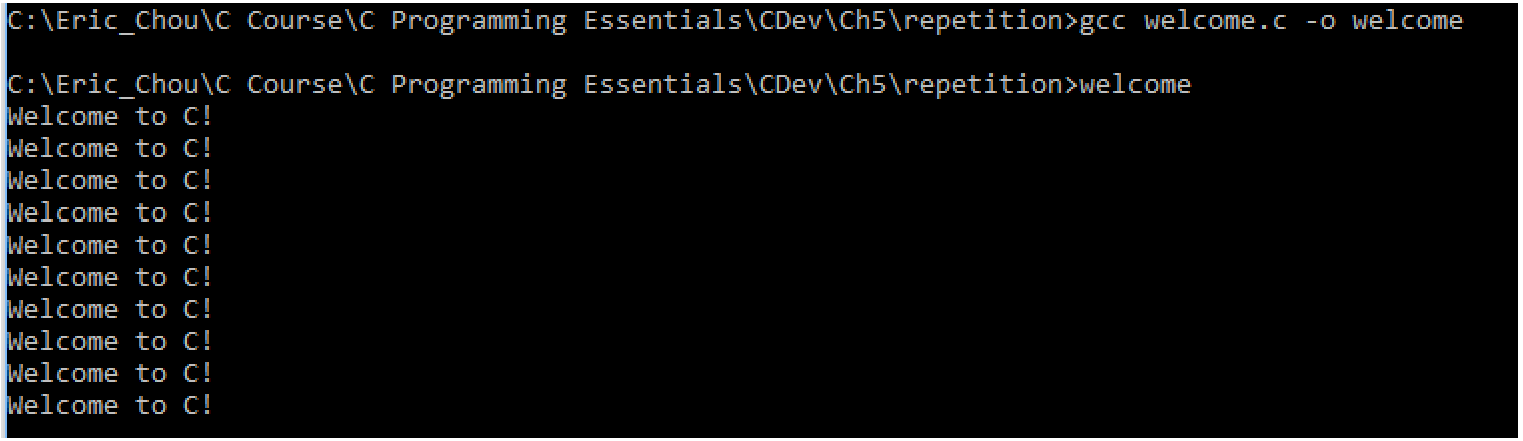
Repetition

Demo Program: Welcome to C! (welcome.c)

Go gcc!!!

Note:

1. Repeat for 10 times
2. This is also called counted loop



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>gcc welcome.c -o welcome

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>welcome
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
Welcome to C!
```

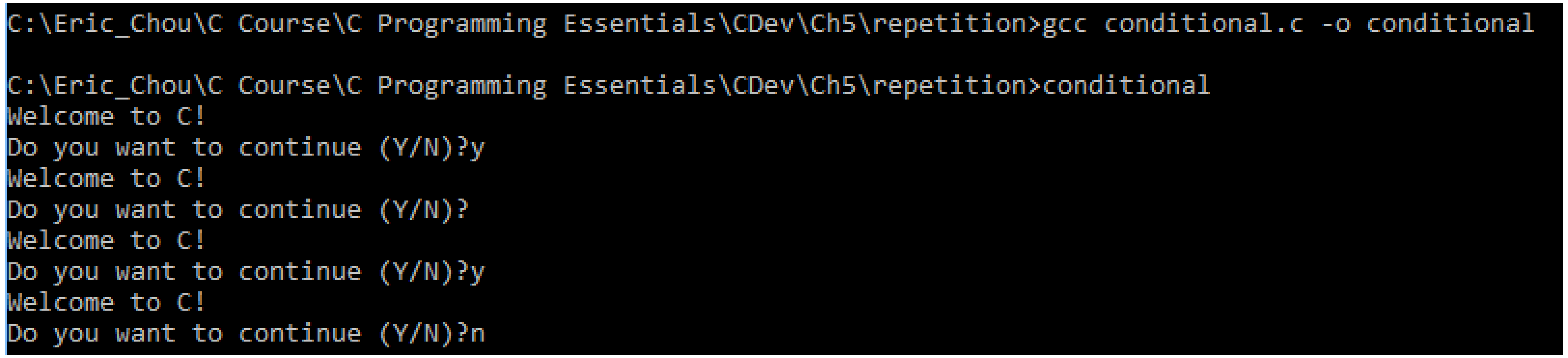
Conditional Repetition

Demo Program: condition.c

Go gcc!!!

Note:

1. getch() will get a keyboard stroke but will not print the key directly.
2. Use the flag continued to determine whether the loop should be terminated or not.
3. This is also called sentinel loop.



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>gcc conditional.c -o conditional

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>conditional
Welcome to C!
Do you want to continue (Y/N)?y
Welcome to C!
Do you want to continue (Y/N)?
Welcome to C!
Do you want to continue (Y/N)?y
Welcome to C!
Do you want to continue (Y/N)?n
```

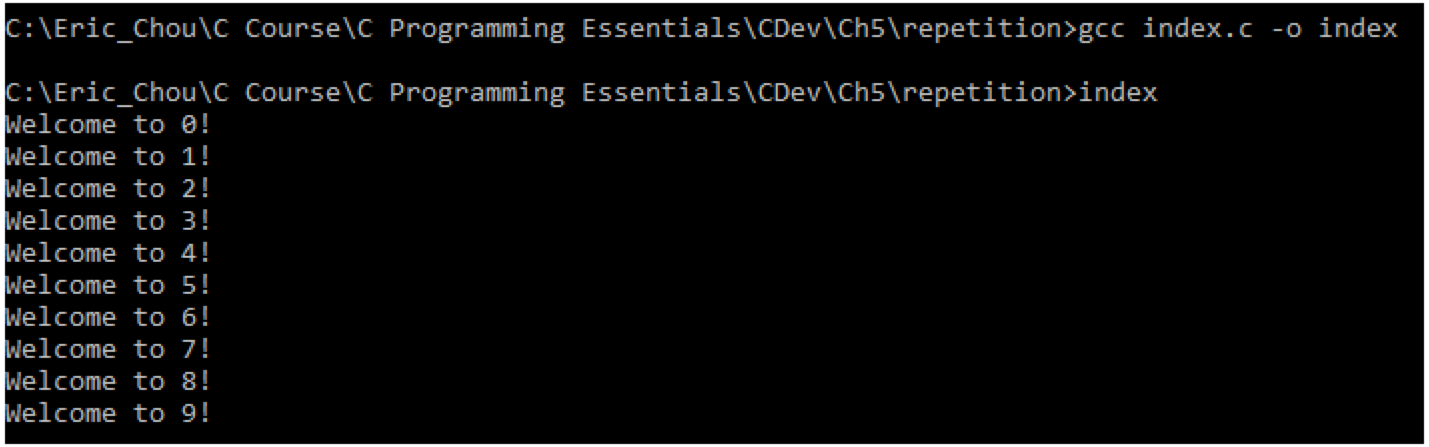
Indexed Repetition

Demo Program: index.c

Go gcc!!!

Note:

1. getch() will get a keyboard stroke but will not print the key directly.
2. Use an index to determine whether the loop should be done or not.
3. This is also called indexed loop.



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>gcc index.c -o index

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>index
Welcome to 0!
Welcome to 1!
Welcome to 2!
Welcome to 3!
Welcome to 4!
Welcome to 5!
Welcome to 6!
Welcome to 7!
Welcome to 8!
Welcome to 9!
```

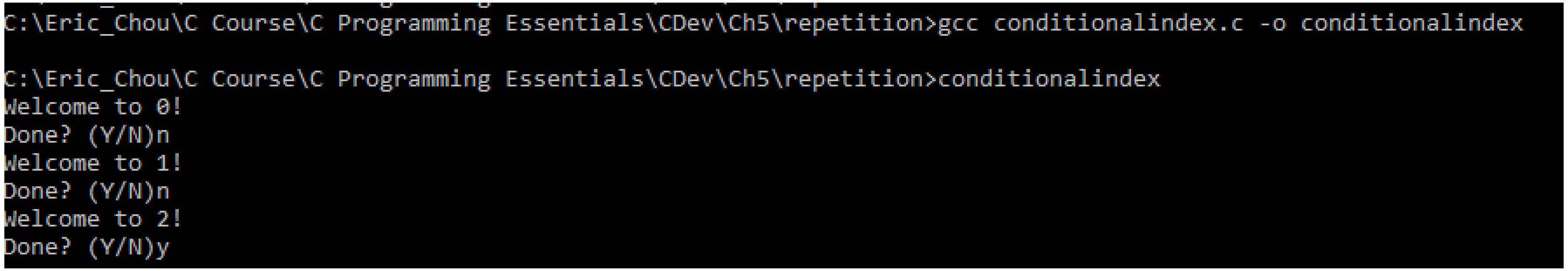
Conditional Indexed Repetition

Demo Program: conditionalindex.c

Go gcc!!!

Note:

1. getch() will get a keyboard stroke but will not print the key directly.
2. Use an index to determine whether the loop should be done or not.
3. You may also finish the loop with the sentinel value.
4. This is also called conditional indexed loop.
5. In conditional.c, we use **positive** continue flag. In this conditionalindex.c, we use **negative** done flag. Both works.



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>gcc conditionalindex.c -o conditionalindex

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\repetition>conditionalindex
Welcome to 0!
Done? (Y/N)n
Welcome to 1!
Done? (Y/N)n
Welcome to 2!
Done? (Y/N)y
```

LECTURE 3

Sum Loops

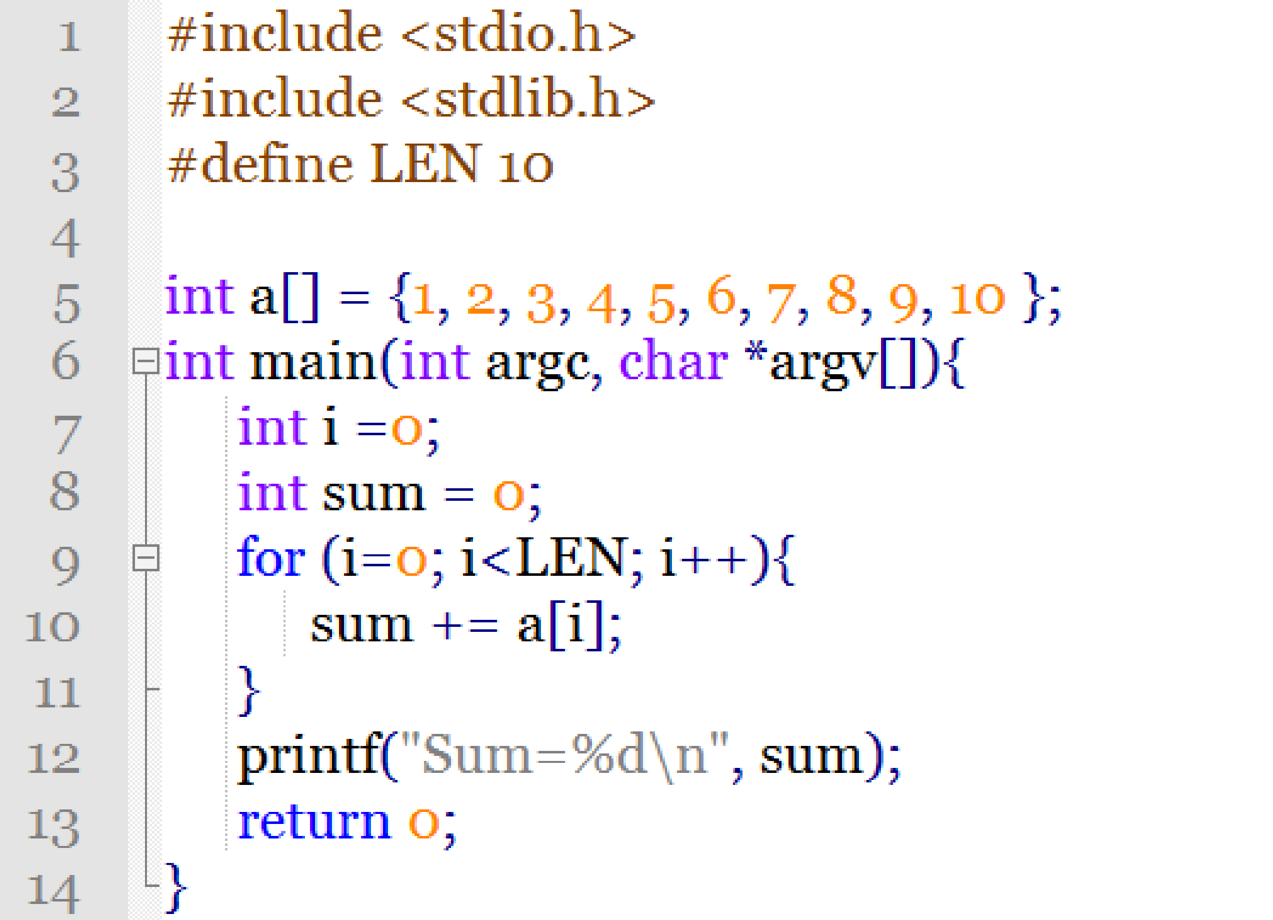
Sum loops

1. Indexed Sum Loops (for-loop)
2. Conditional Sum Loops (while-loop)
3. Sum loops with console inputs
4. Sum loops with random inputs

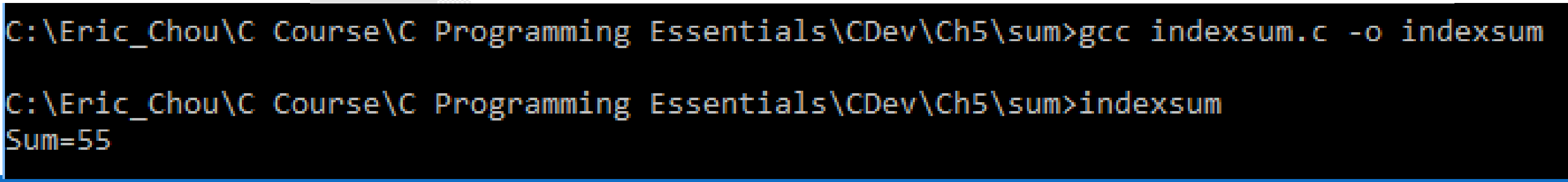
Indexed Sum Loop

Demo Program: indexsum.c

Go gcc!!!



```c
#include <stdio.h>
#include <stdlib.h>
#define LEN 10

int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int main(int argc, char *argv[]){
    int i =0;
    int sum = 0;
    for (i=0; i<LEN; i++){
        sum += a[i];
    }
    printf("Sum=%d\n", sum);
    return 0;
}
```



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>gcc indexsum.c -o indexsum

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>indexsum
Sum=55
```

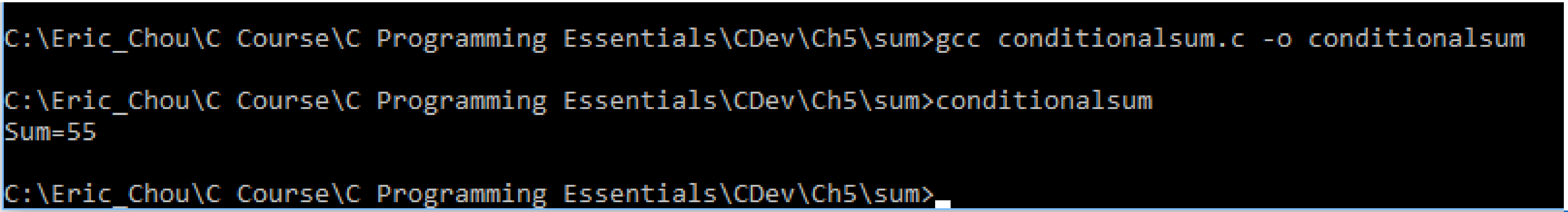
Conditional Sum Loop

Demo Program: conditionalsum.c (includes limits.h)

Go gcc!!!

Note:

No need to know about the length of array in advance.



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>gcc conditionalsum.c -o conditionalsum

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>conditionalsum
Sum=55

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>_
```

Integer Maximum and Minimum used as Sentinels

In C: (Maximum/Minimum Integer and Floating Point)

```
#include <limits.h>
```

then use

```
int imin = INT_MIN; // minimum value
int imax = INT_MAX;
```

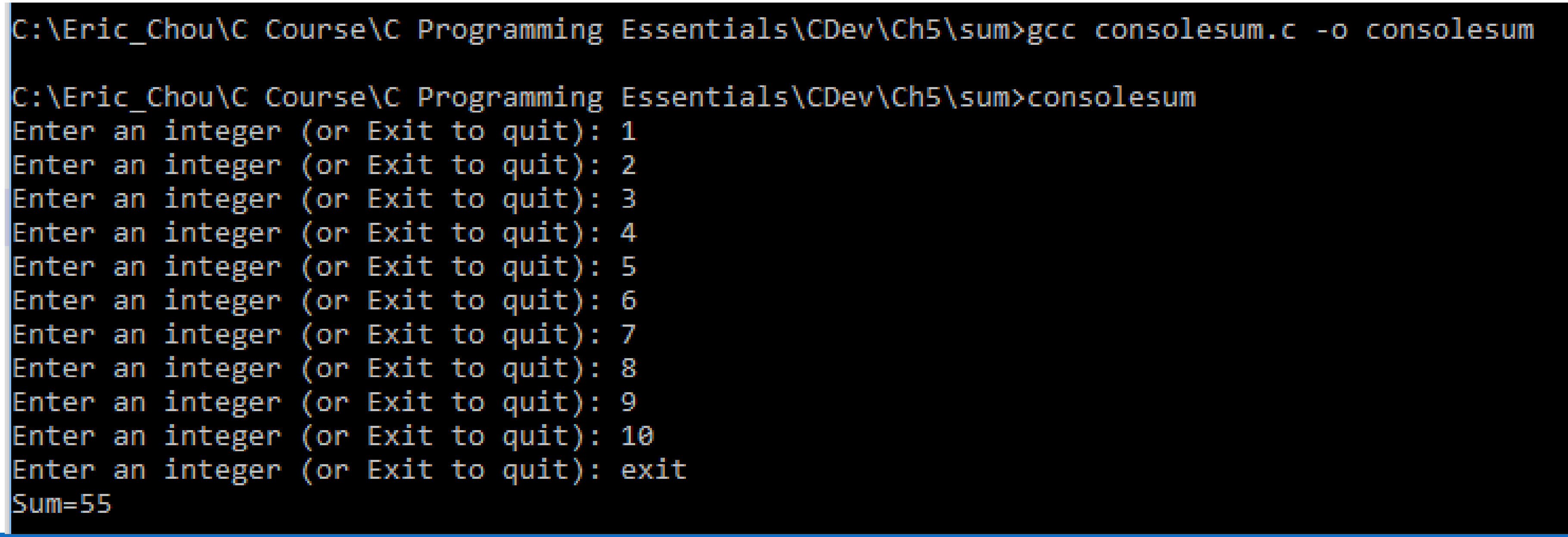
or

```
#include <float.h>
float fmin = FLT_MIN;  // minimum positive value
double dmin = DBL_MIN; // minimum positive value
float fmax = FLT_MAX;
double dmax = DBL_MAX;
```

Sum Loop with Console Inputs

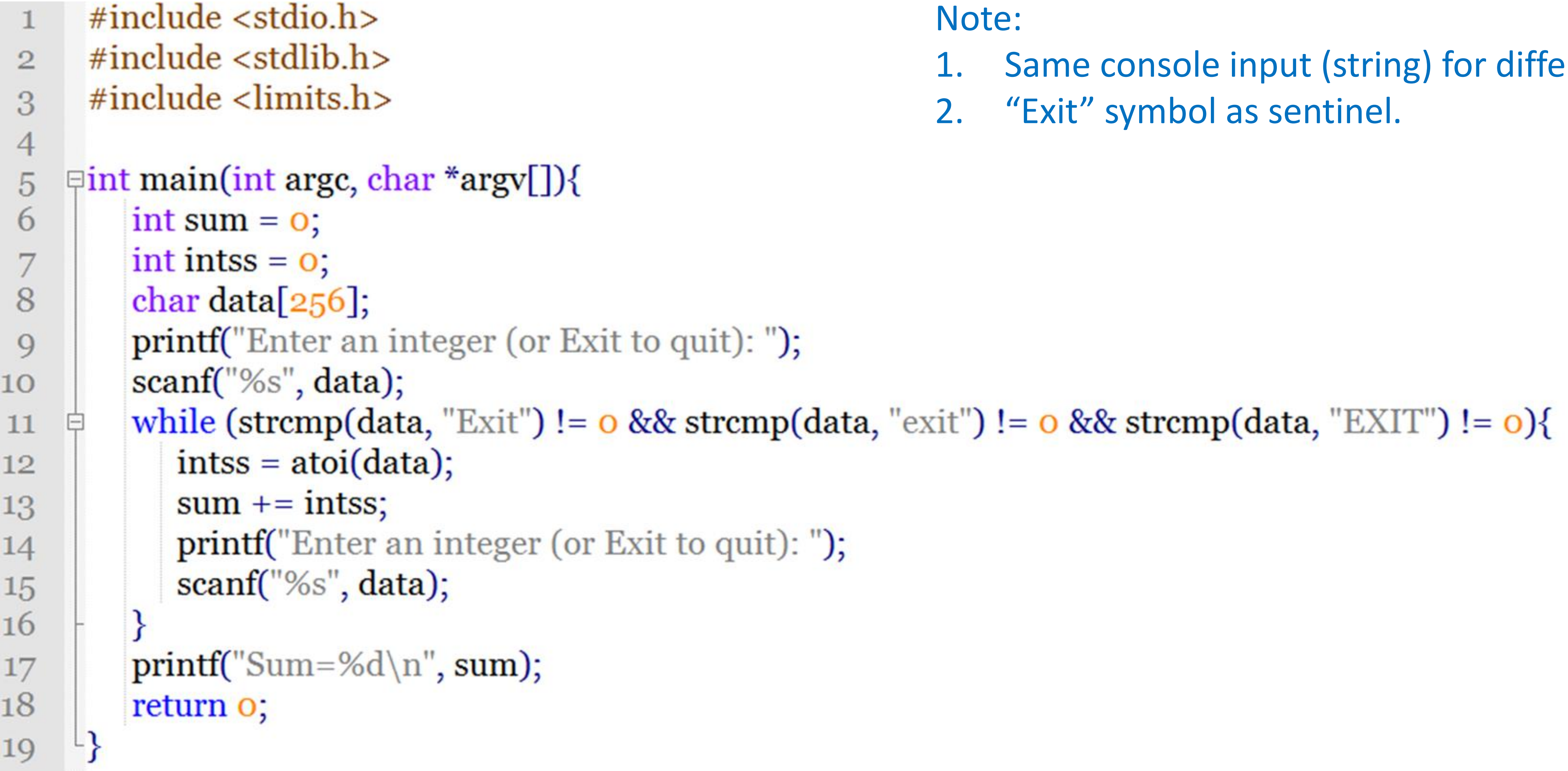
Demo Program: consolesum.c

Go gcc!!!



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>gcc consolesum.c -o consolesum

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>consolesum
Enter an integer (or Exit to quit): 1
Enter an integer (or Exit to quit): 2
Enter an integer (or Exit to quit): 3
Enter an integer (or Exit to quit): 4
Enter an integer (or Exit to quit): 5
Enter an integer (or Exit to quit): 6
Enter an integer (or Exit to quit): 7
Enter an integer (or Exit to quit): 8
Enter an integer (or Exit to quit): 9
Enter an integer (or Exit to quit): 10
Enter an integer (or Exit to quit): exit
Sum=55
```

Sum of Integers from Console: consolesum.c



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

int main(int argc, char *argv[]){
    int sum = 0;
    int intss = 0;
    char data[256];
    printf("Enter an integer (or Exit to quit): ");
    scanf("%s", data);
    while (strcmp(data, "Exit") != 0 && strcmp(data, "exit") != 0 && strcmp(data, "EXIT") != 0){
        intss = atoi(data);
        sum += intss;
        printf("Enter an integer (or Exit to quit): ");
        scanf("%s", data);
    }
    printf("Sum=%d\n", sum);
    return 0;
}
```

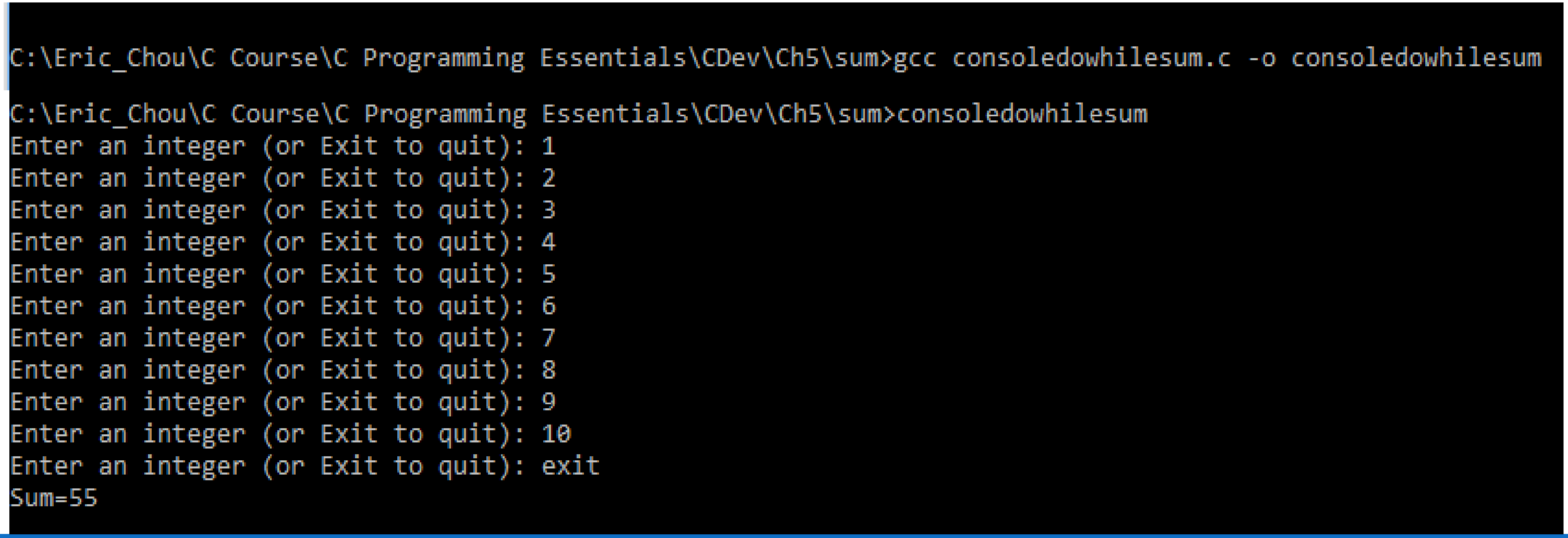
Note:

1. Same console input (string) for different purpose.
2. “Exit” symbol as sentinel.

do-while Sum Loop with Console Inputs

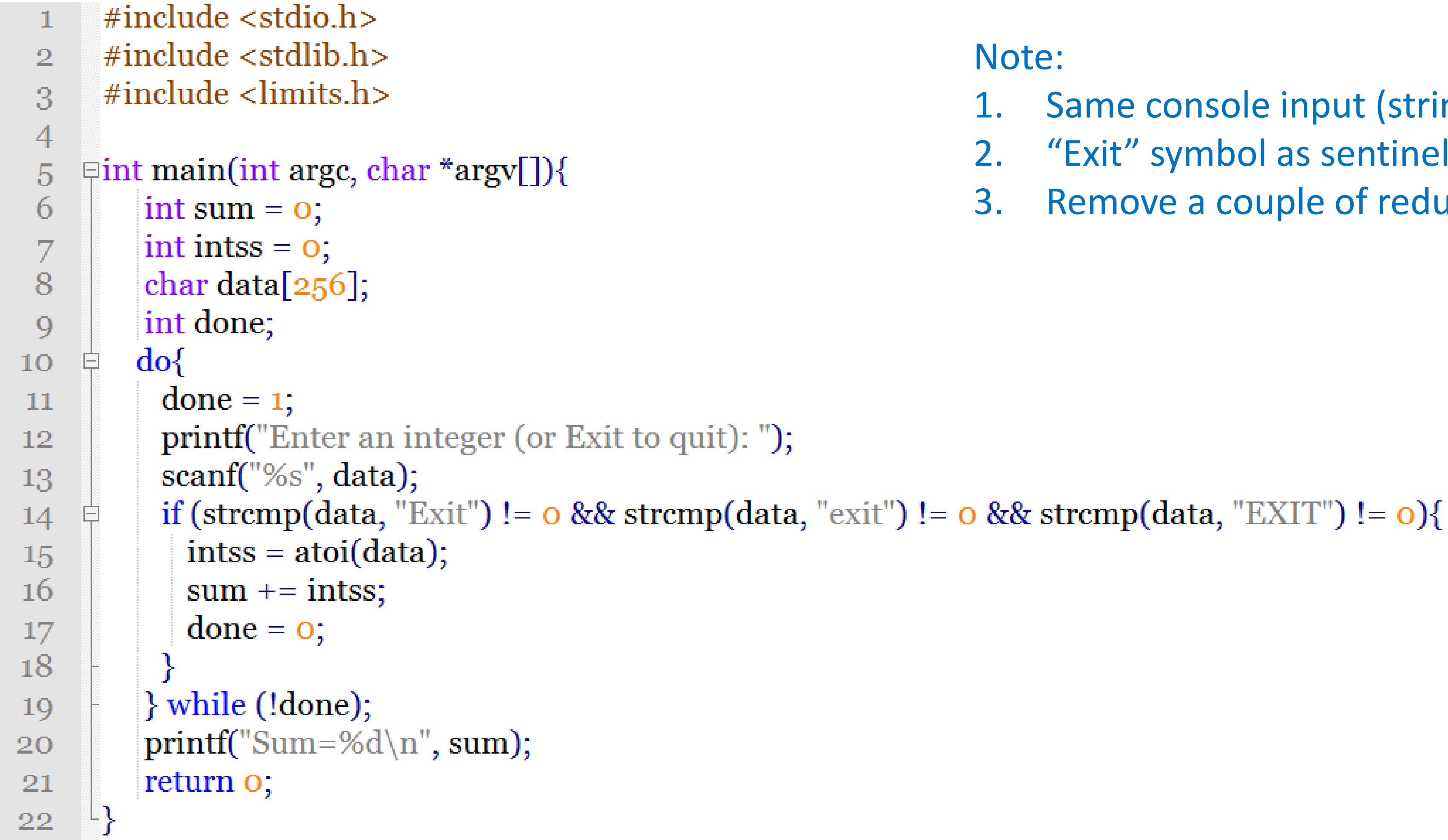
Demo Program: consoledowhilesum.c

Go gcc!!!



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>gcc consoledowhilesum.c -o consoledowhilesum

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>consoledowhilesum
Enter an integer (or Exit to quit): 1
Enter an integer (or Exit to quit): 2
Enter an integer (or Exit to quit): 3
Enter an integer (or Exit to quit): 4
Enter an integer (or Exit to quit): 5
Enter an integer (or Exit to quit): 6
Enter an integer (or Exit to quit): 7
Enter an integer (or Exit to quit): 8
Enter an integer (or Exit to quit): 9
Enter an integer (or Exit to quit): 10
Enter an integer (or Exit to quit): exit
Sum=55
```

Sum of Integers from Console: consoledowhilesum.c



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

int main(int argc, char *argv[]){
    int sum = 0;
    int intss = 0;
    char data[256];
    int done;
    do{
        done = 1;
        printf("Enter an integer (or Exit to quit): ");
        scanf("%s", data);
        if (strcmp(data, "Exit") != 0 && strcmp(data, "exit") != 0 && strcmp(data, "EXIT") != 0){
            intss = atoi(data);
            sum += intss;
            done = 0;
        }
    } while (!done);
    printf("Sum=%d\n", sum);
    return 0;
}
```

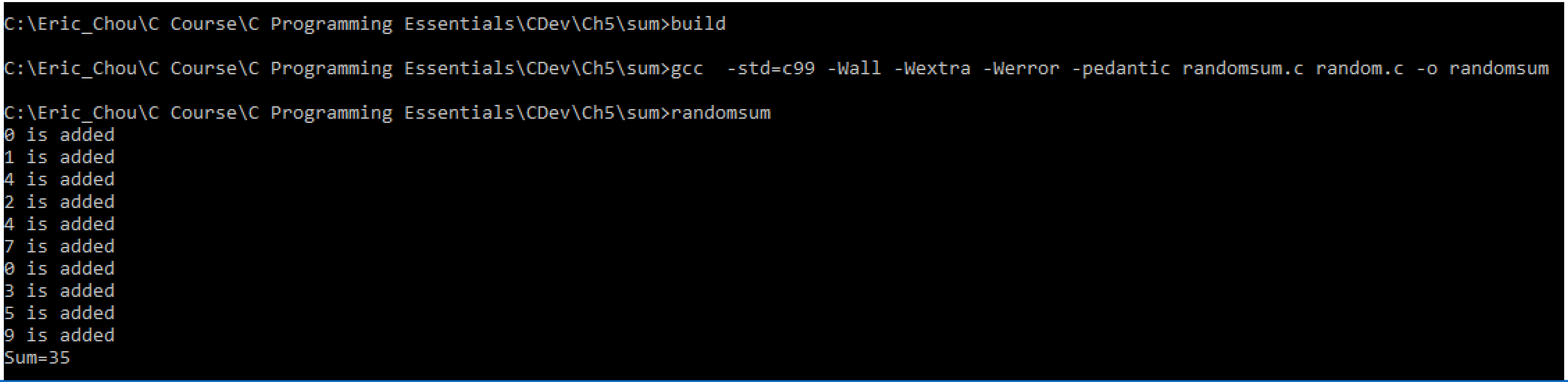
Note:

1. Same console input (string) for different purpose.
2. “Exit” symbol as sentinel.
3. Remove a couple of redundant lines.

Demo Program:

randomsum.c + random.c

Go gcc!!!

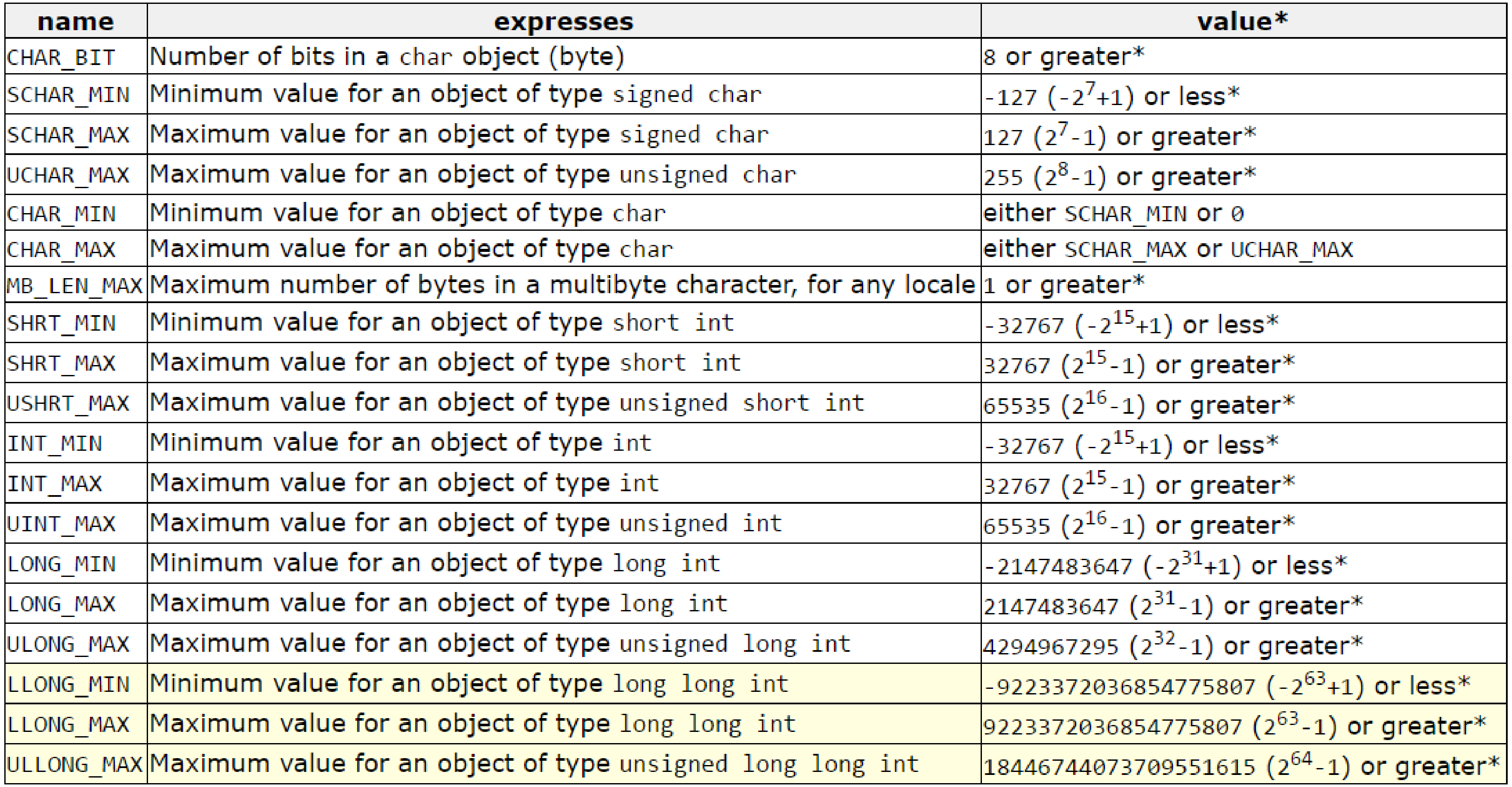


```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>build

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>gcc  -std=c99 -Wall -Wextra -Werror -pedantic randomsum.c random.c -o randomsum

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\sum>randomsum
0 is added
1 is added
4 is added
2 is added
4 is added
7 is added
0 is added
3 is added
5 is added
9 is added
Sum=35
```

LECTURE 4

Basic Statistics

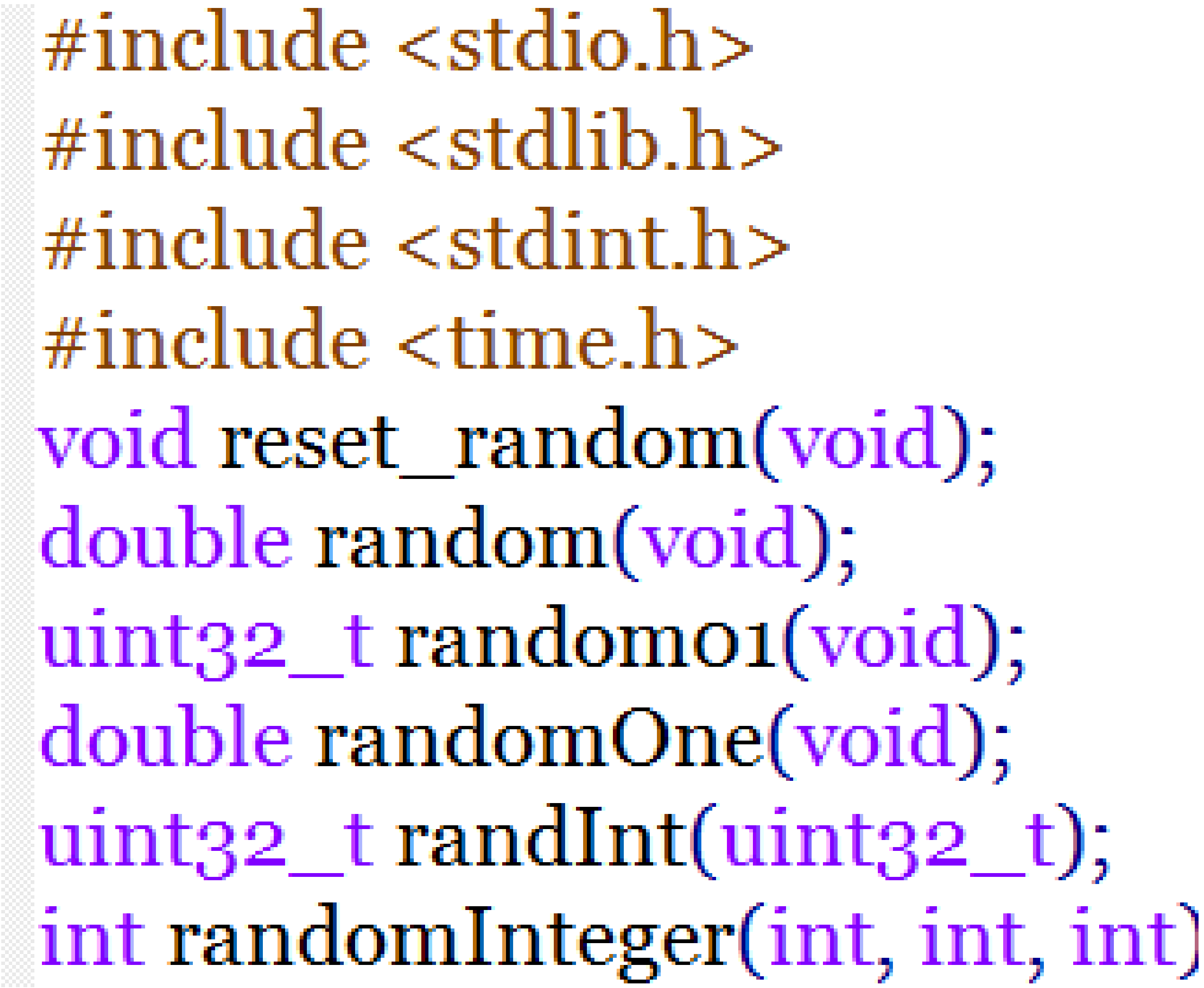
#include <limits.h>



name	expresses	value*
CHAR_BIT	Number of bits in a char object (byte)	8 or greater*
SCHAR_MIN	Minimum value for an object of type signed char	-127 ($-2^{7}+1$) or less*
SCHAR_MAX	Maximum value for an object of type signed char	127 ($2^{7}-1$) or greater*
UCHAR_MAX	Maximum value for an object of type unsigned char	255 ($2^{8}-1$) or greater*
CHAR_MIN	Minimum value for an object of type char	either SCHAR_MIN or 0
CHAR_MAX	Maximum value for an object of type char	either SCHAR_MAX or UCHAR_MAX
MB_LEN_MAX	Maximum number of bytes in a multibyte character, for any locale	1 or greater*
SHRT_MIN	Minimum value for an object of type short int	-32767 ($-2^{15}+1$) or less*
SHRT_MAX	Maximum value for an object of type short int	32767 ($2^{15}-1$) or greater*
USHRT_MAX	Maximum value for an object of type unsigned short int	65535 ($2^{16}-1$) or greater*
INT_MIN	Minimum value for an object of type int	-32767 ($-2^{15}+1$) or less*
INT_MAX	Maximum value for an object of type int	32767 ($2^{15}-1$) or greater*
UINT_MAX	Maximum value for an object of type unsigned int	65535 ($2^{16}-1$) or greater*
LONG_MIN	Minimum value for an object of type long int	-2147483647 ($-2^{31}+1$) or less*
LONG_MAX	Maximum value for an object of type long int	2147483647 ($2^{31}-1$) or greater*
ULONG_MAX	Maximum value for an object of type unsigned long int	4294967295 ($2^{32}-1$) or greater*
LLONG_MIN	Minimum value for an object of type long long int	-9223372036854775807 ($-2^{63}+1$) or less*
LLONG_MAX	Maximum value for an object of type long long int	9223372036854775807 ($2^{63}-1$) or greater*
ULLONG_MAX	Maximum value for an object of type unsigned long long int	18446744073709551615 ($2^{64}-1$) or greater*

random.c and random.h

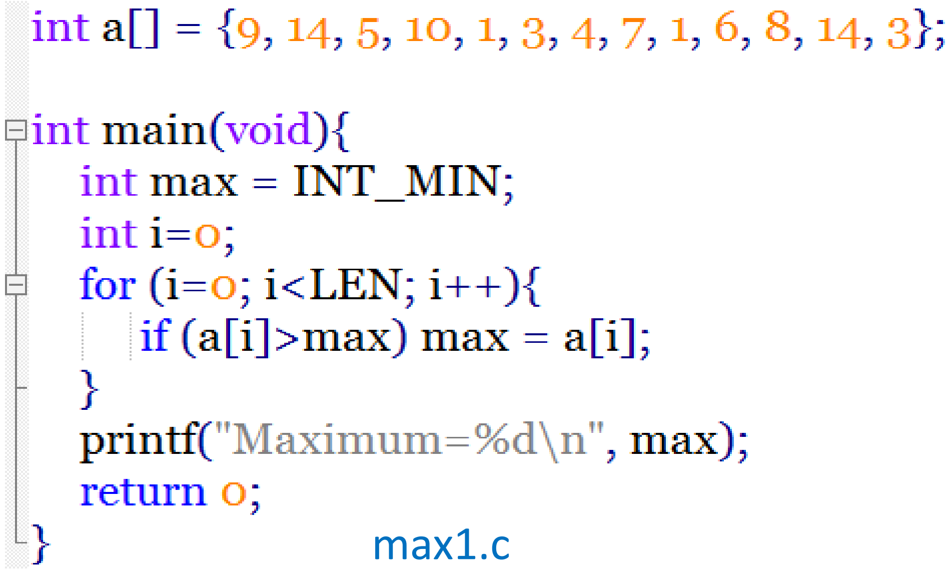
Random module for statistical experiments.



```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>
void reset_random(void);
double random(void);
uint32_t random01(void);
double randomOne(void);
uint32_t randInt(uint32_t);
int randomInteger(int, int, int)
```

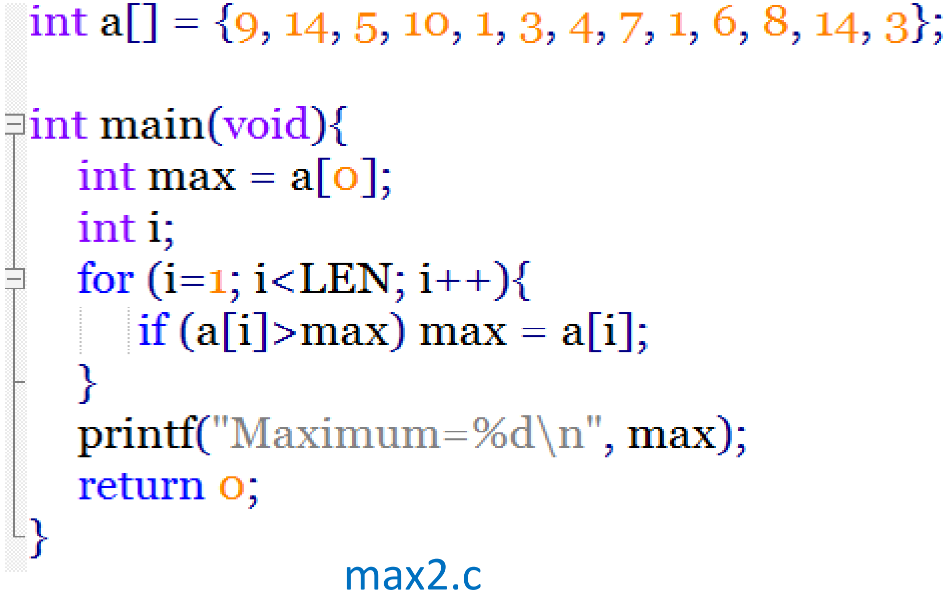
Finding maximum among a group of numbers

1. First maximum occurrence using precondition variable. (max1.c)
2. First maximum occurrence using pre-selected variable. (max2.c)
3. Last maximum occurrence using precondition variable. (max3.c)
4. Last maximum occurrence using pre-selected variable. (max4.c)
5. First maximum occurrence's index using precondition variable. (max5.c)
6. First maximum occurrence's index using pre-selected variable. (max6.c)
7. Last maximum occurrence's index using precondition variable. (max7.c)
8. Last maximum occurrence's index using pre-selected variable. (max8.c)



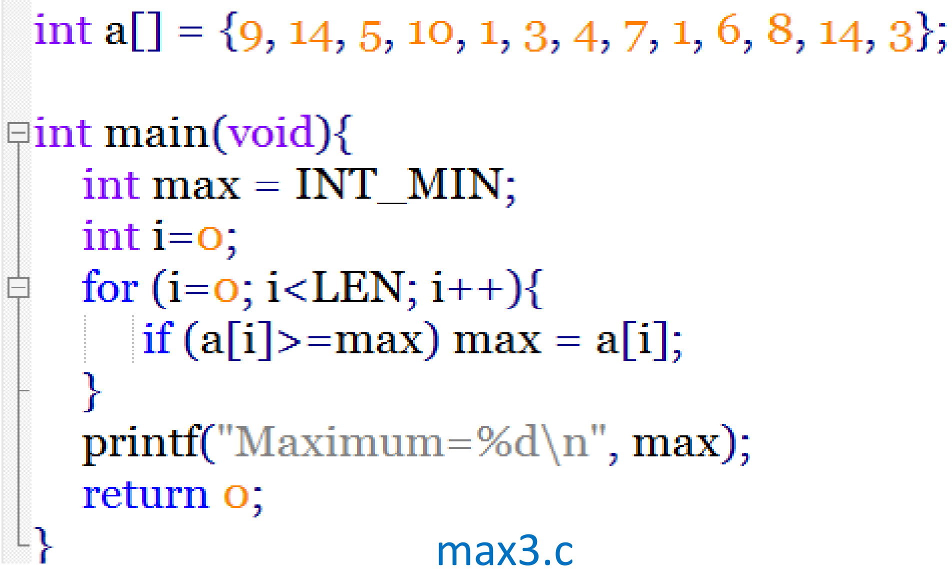
```
int a[] = {9, 14, 5, 10, 1, 3, 4, 7, 1, 6, 8, 14, 3};

int main(void){
    int max = INT_MIN;
    int i=0;
    for (i=0; i<LEN; i++){
        if (a[i]>max) max = a[i];
    }
    printf("Maximum=%d\n", max);
    return 0;
}
```

max1.c



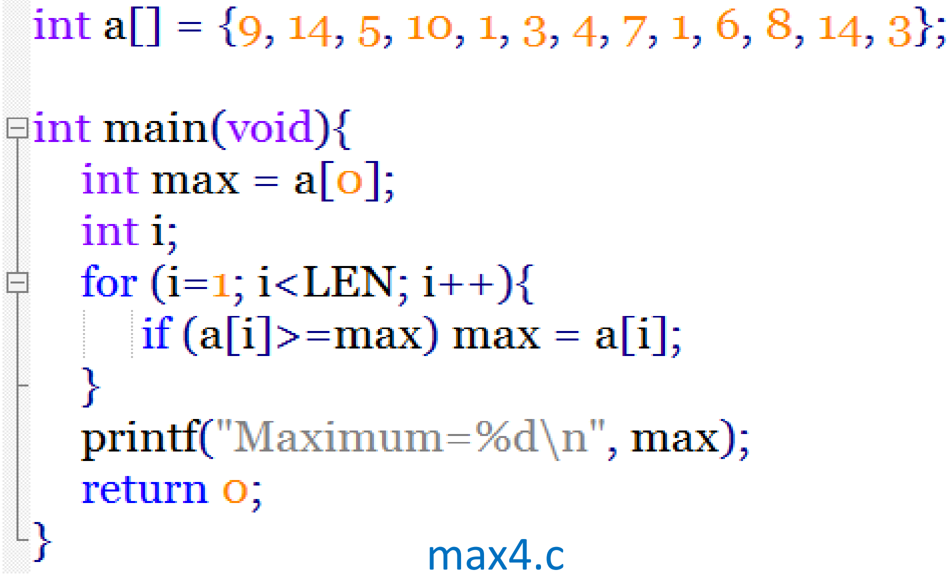
```
int a[] = {9, 14, 5, 10, 1, 3, 4, 7, 1, 6, 8, 14, 3};

int main(void){
    int max = a[0];
    int i;
    for (i=1; i<LEN; i++){
        if (a[i]>max) max = a[i];
    }
    printf("Maximum=%d\n", max);
    return 0;
}
```

max2.c



```
int a[] = {9, 14, 5, 10, 1, 3, 4, 7, 1, 6, 8, 14, 3};

int main(void){
    int max = INT_MIN;
    int i=0;
    for (i=0; i<LEN; i++){
        if (a[i]>=max) max = a[i];
    }
    printf("Maximum=%d\n", max);
    return 0;
}
```

max3.c



```
int a[] = {9, 14, 5, 10, 1, 3, 4, 7, 1, 6, 8, 14, 3};

int main(void){
    int max = a[0];
    int i;
    for (i=1; i<LEN; i++){
        if (a[i]>=max) max = a[i];
    }
    printf("Maximum=%d\n", max);
    return 0;
}
```

max4.c

Discussion About the Algorithm of Finding Minimum

- Same algorithm as finding maximum except that INT_MAX need to be the initial value for the min variable. > need to be replaced by <, while >= need to be replaced by <=.

Demo Program:

Stats package (max, min, avg, sum)

Go gcc!!!

Simple Stats Package

Maximum:

max1.c
max2.c
max3.c
max4.c
max5.c
max6.c
max7.c
max7.c

Minimum:

min1.c
min2.c
min3.c
min4.c
min5.c
min6.c
min7.c
min7.c

Sum:

indexsum.c
conditionalsum.c
consolesum.c
consoledowhilesum.c
randomsum.c

Average:

indexavg.c
conditionalavg.c
consoleavg.c
consoledowhileavg.c
randomavg.c

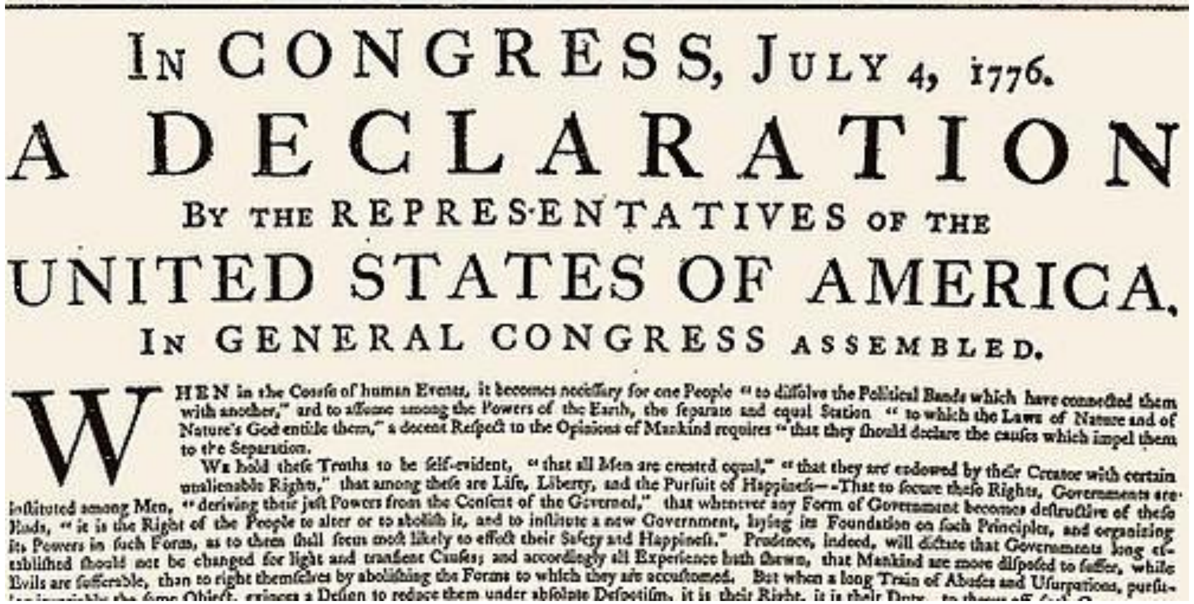
LECTURE 5

Letter Count of A File

Lab Project: Count the letters in a file.

Letter count of a Text File (declare.txt)

Go gcc!!!



IN CONGRESS, JULY 4, 1776.

A DECLARATION

BY THE REPRESENTATIVES OF THE

UNITED STATES OF AMERICA,

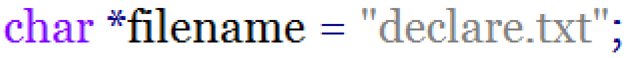
IN GENERAL CONGRESS ASSEMBLED.

WHEN in the Courſe of human Events, it becomes neceſſary for one People "to diſſolve the Political Bands which have connected them with another," and to aſſume among the Powers of the Earth, the ſeparate and equal Station "to which the Laws of Nature and of Nature's God entitle them," a decent Reſpect to the Opinions of Mankind requires "that they ſhould declare the cauſes which impel them to the Separation.

WE hold theſe Truths to be ſelf-evident, "that all Men are created equal," "that they are endowed by their Creator with certain unalienable Rights," that among theſe are Life, Liberty, and the Purſuit of Happineſs—That to ſecure theſe Rights, Governments are inſtituted among Men, "deriving their juſt Powers from the Conſent of the Governed," that whenever any Form of Government becomes deſtructive of theſe Ends, "it is the Right of the People to alter or to aboliſh it, and to inſtitute a new Government, laying its Foundation on ſuch Principles, and organizing its Powers in ſuch Form, as to them ſhall ſeem moſt likely to effect their Safety and Happineſs." Prudence, indeed, will dictate that Governments long eſtabliſhed ſhould not be changed for light and tranſient Cauſes; and accordingly all Experience hath ſhewn, that Mankind are more diſpoſed to ſuffer, while Evils are ſufferable, than to right themſelves by aboliſhing the Forms to which they are accuſtomed. But when a long Train of Abuſes and Uſurpations, purſuing invariably the ſame Object, evinces a Deſign to reduce them under abſolute Deſpotiſm, it is their Right, it is their Duty,

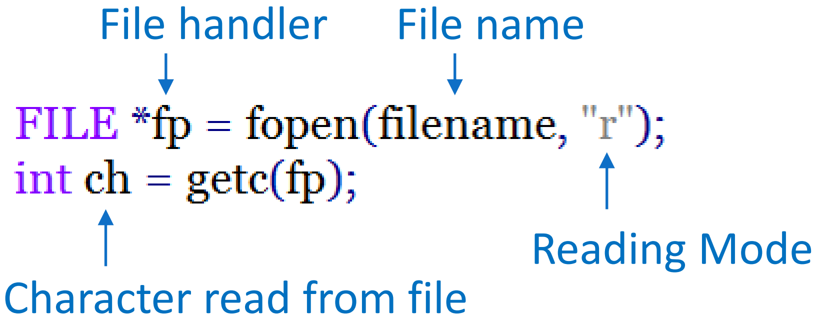
Notes

1. A string of characters is represented by a point of character array.



```
char *filename = "declare.txt";
```

2. Open a text file and read characters from the file.



File handler

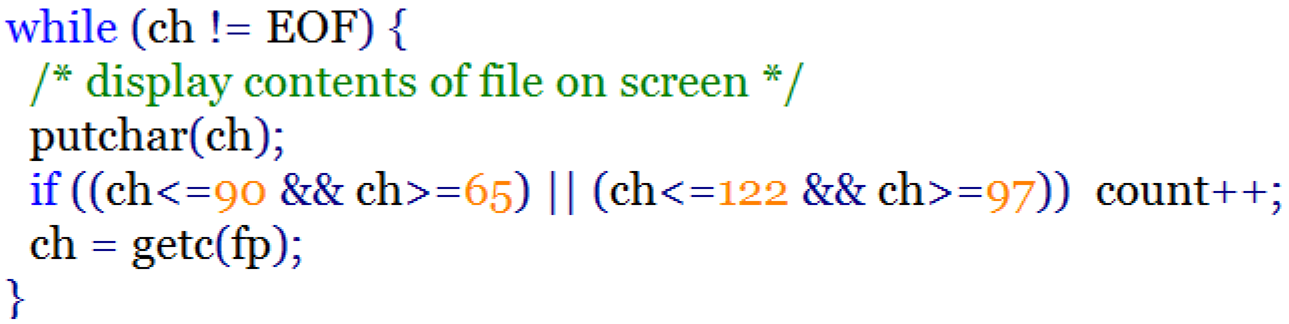
File name

```
FILE *fp = fopen(filename, "r");
int ch = getc(fp);
```

Reading Mode

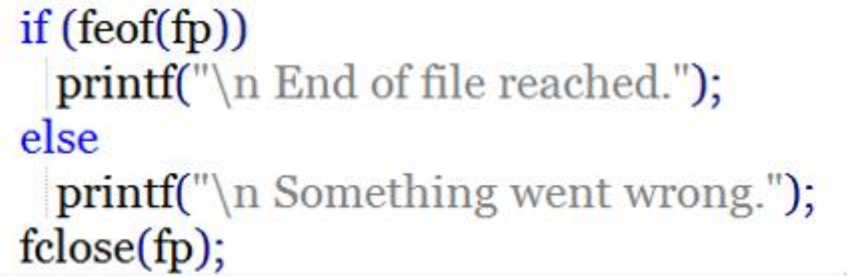
Character read from file

3. End of File mark: EOF

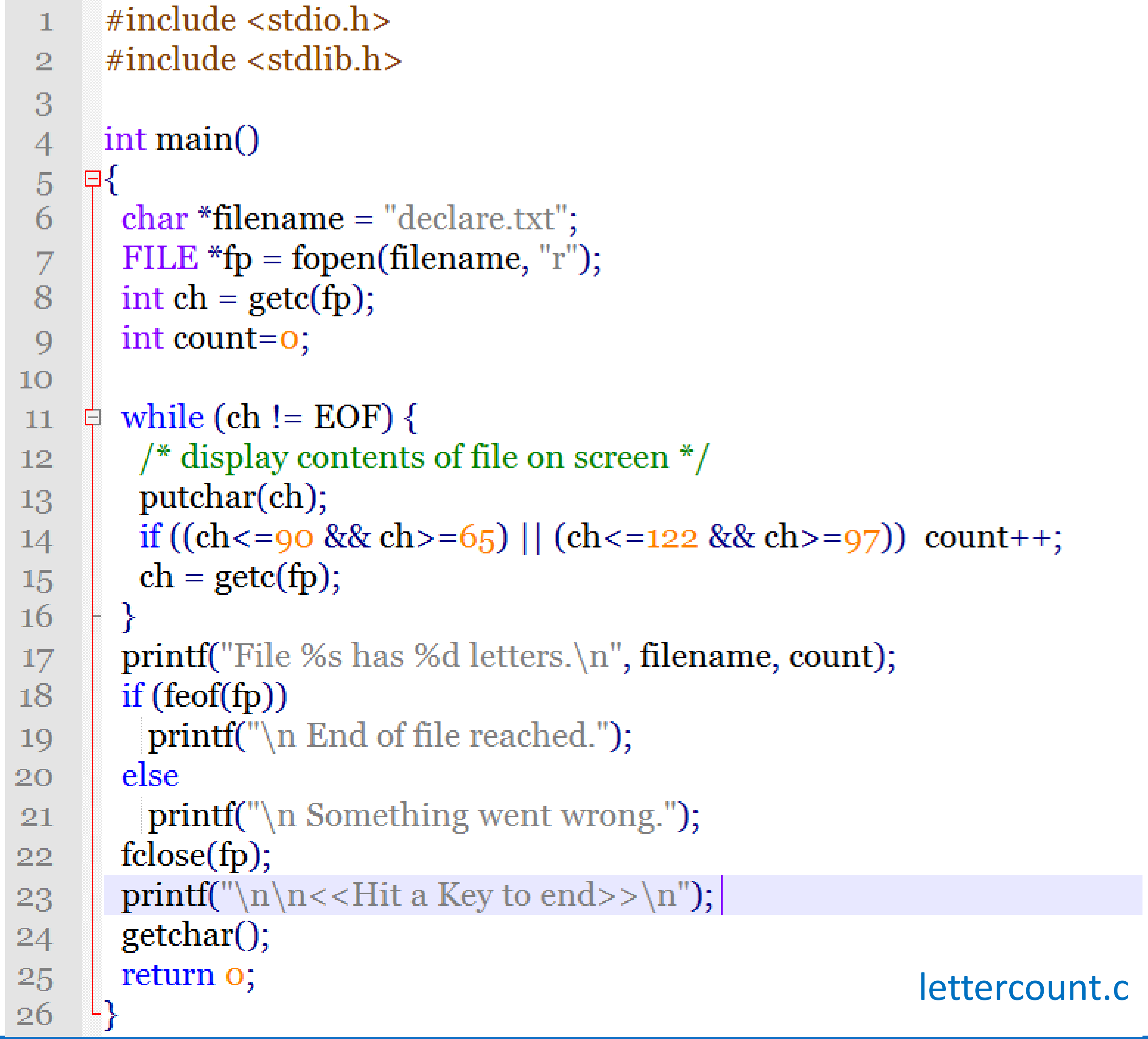


```
while (ch != EOF) {
  /* display contents of file on screen */
  putchar(ch);
  if ((ch<=90 && ch>=65) || (ch<=122 && ch>=97))  count++;
  ch = getc(fp);
}
```

4. End of File check function: feof(fp) check the file, pointed by fp, ended or not.



```
if (feof(fp))
   printf("\n End of file reached.");
else
   printf("\n Something went wrong.");
fclose(fp);
```



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

int main()
{
  char *filename = "declare.txt";
  FILE *fp = fopen(filename, "r");
  int ch = getc(fp);
  int count=0;

  while (ch != EOF) {
    /* display contents of file on screen */
    putchar(ch);
    if ((ch<=90 && ch>=65) || (ch<=122 && ch>=97))  count++;
    ch = getc(fp);
  }
  printf("File %s has %d letters.\n", filename, count);
  if (feof(fp))
     printf("\n End of file reached.");
  else
     printf("\n Something went wrong.");
  fclose(fp);
  printf("\n\n<<Hit a Key to end>>\n");
  getchar();
  return 0;
}
```

lettercount.c

LECTURE 6

For-Loop

`for` statement

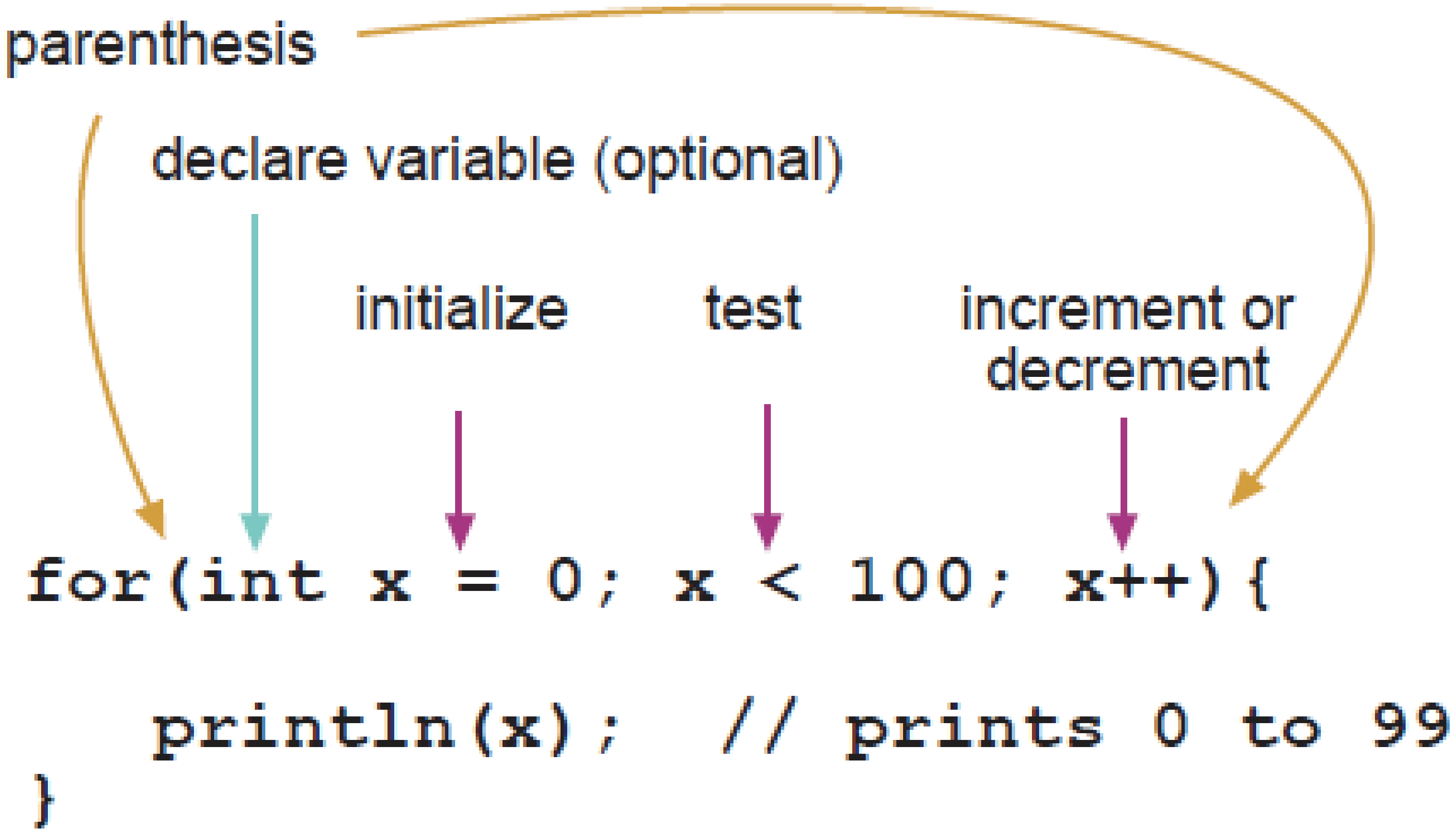
Syntax is

```
for(expr1; expr2; expr3) {
    stmts;
}
```

- Initialization: expr1 is used to initialize some parameters
- Continue Condition: expr2 represents a condition that must be true for the loop to continue
- Update Actions: expr3 is used to modify the values of some parameters.

- It is equiv to

```
expr1;
while (expr2) {
      stmt
      expr3;
}
```



```
for(int x = 0; x < 100; x++){

   println(x);  // prints 0 to 99
}
```

for statement

This piece of code has equivalent for statement as follows:

```
expr1a;
expr1b;
while (expr2) {
        stmt
        expr3a;
        expr3b;
}
```

```
for ( expr1a, expr1b; expr2; expr3a, expr3b){
   stmt;
}
```

Note that in the for statement expr1, expr2, expr3 need not necessarily be present. If expr2 is not there, then the loop will go forever.

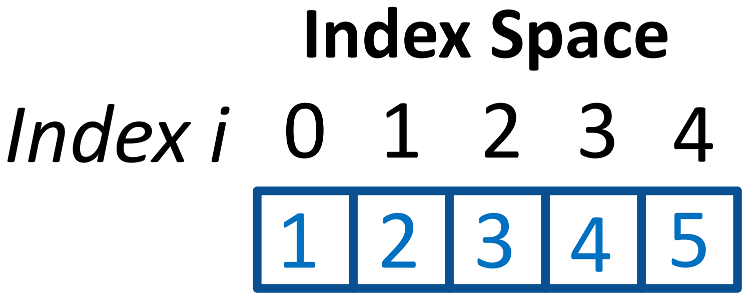
Counted Repetition

Print a message for 100 times.

```
for (int i=0; i<100; i++){
    printf(“Welcome to C !\n”);
}
```

Access of Linear Storage

```
int  a[] = {1, 2, 3, 4, 5};
for (int i=0; i<5; i++){
    sum += a[i];
}
```



Index Space

Index i	0	1	2	3	4
	1	2	3	4	5

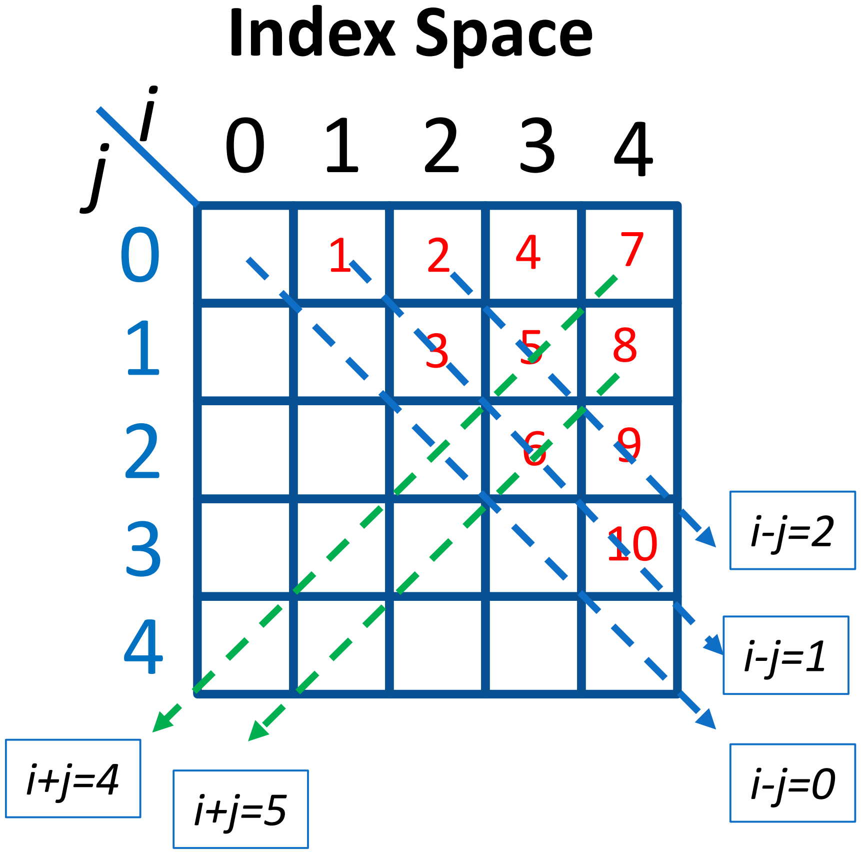
Index Space
for statement: some examples

```
int i, j, x;
for(i=0, x=0; i<5; ++i)
    for(j=0; j<i; ++j) {
        x += (i+j-1);
        printf(“%d ”, x);
    }
```

x’s value when program ends:

0 + 1 + 2 + 2 + 3 +4 + 3 + 4 + 5 +6 = 30

0 1 3 5 8 12 15 19 24 30



Index Space

j \ i	0	1	2	3	4
0		1	2	4	7
1			3	5	8
2				6	9
3					10
4					

LECTURE 7

Do-While Loop

do-while Loop

- Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop in C programming checks its condition at the bottom of the loop.
- A do...while loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

do-while Loop

Syntax

The syntax of a **do...while** loop in C programming language is −

```
do {
   statement(s);
} while( condition );
```

Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop executes once before the condition is tested.

If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop executes again. This process repeats until the given condition becomes false.

do-while loop

- In the do-while loop, the statement is executed and then the expression is evaluated. If the expression is nonzero the process is repeated. Thus the condition is evaluated at the end of each iteration.
- Example

```
x1 = 1;
  do {
     x0 = x1;
     x1 = x0 – f(x0) / fp(x0);
} while ( fabs(x1 – x0) > MIN_STEP ) ;
```

- We needed to evaluate x1 before we could apply the convergence criterion.

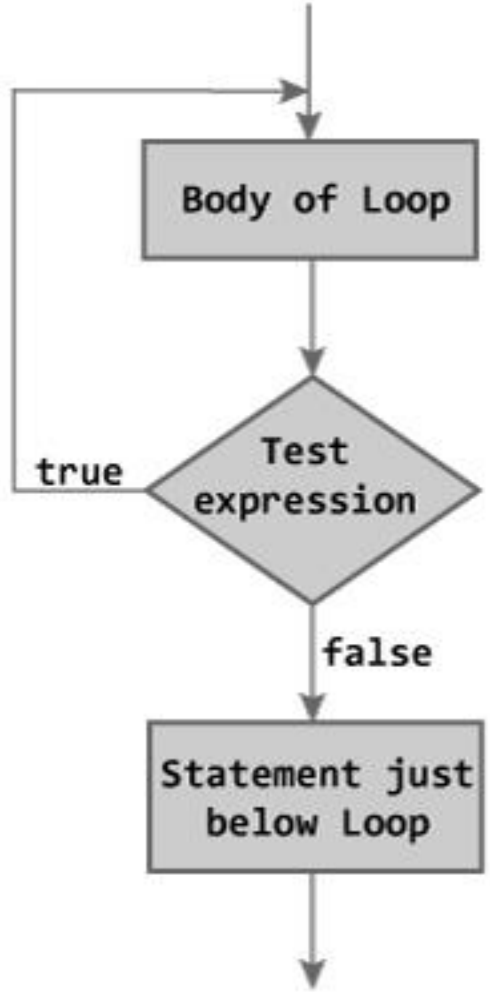
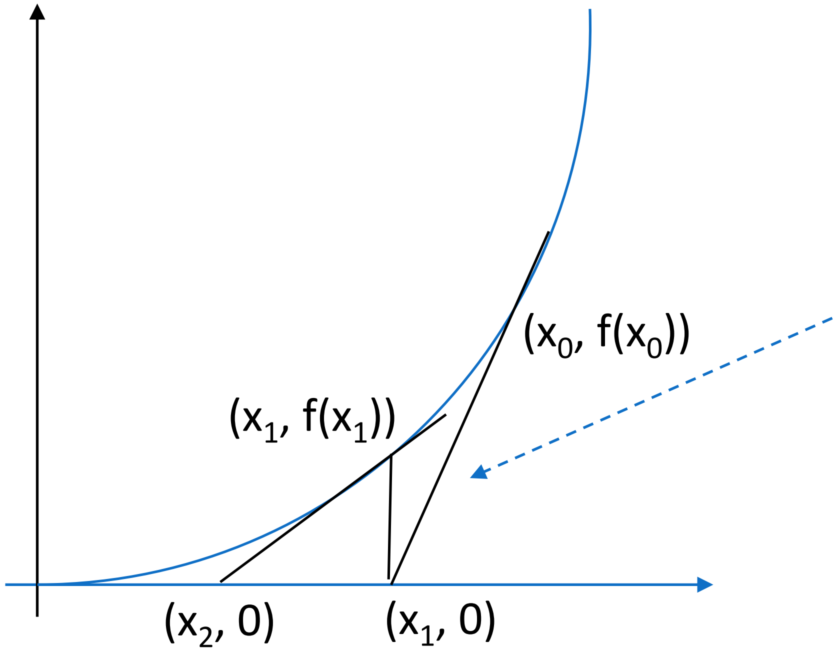


Figure: Flowchart of do...while Loop

Convergence for Solving Numerical Equation (Finding Vertex Point)



$f(x) = x^2$

$f'(x) = 2x$

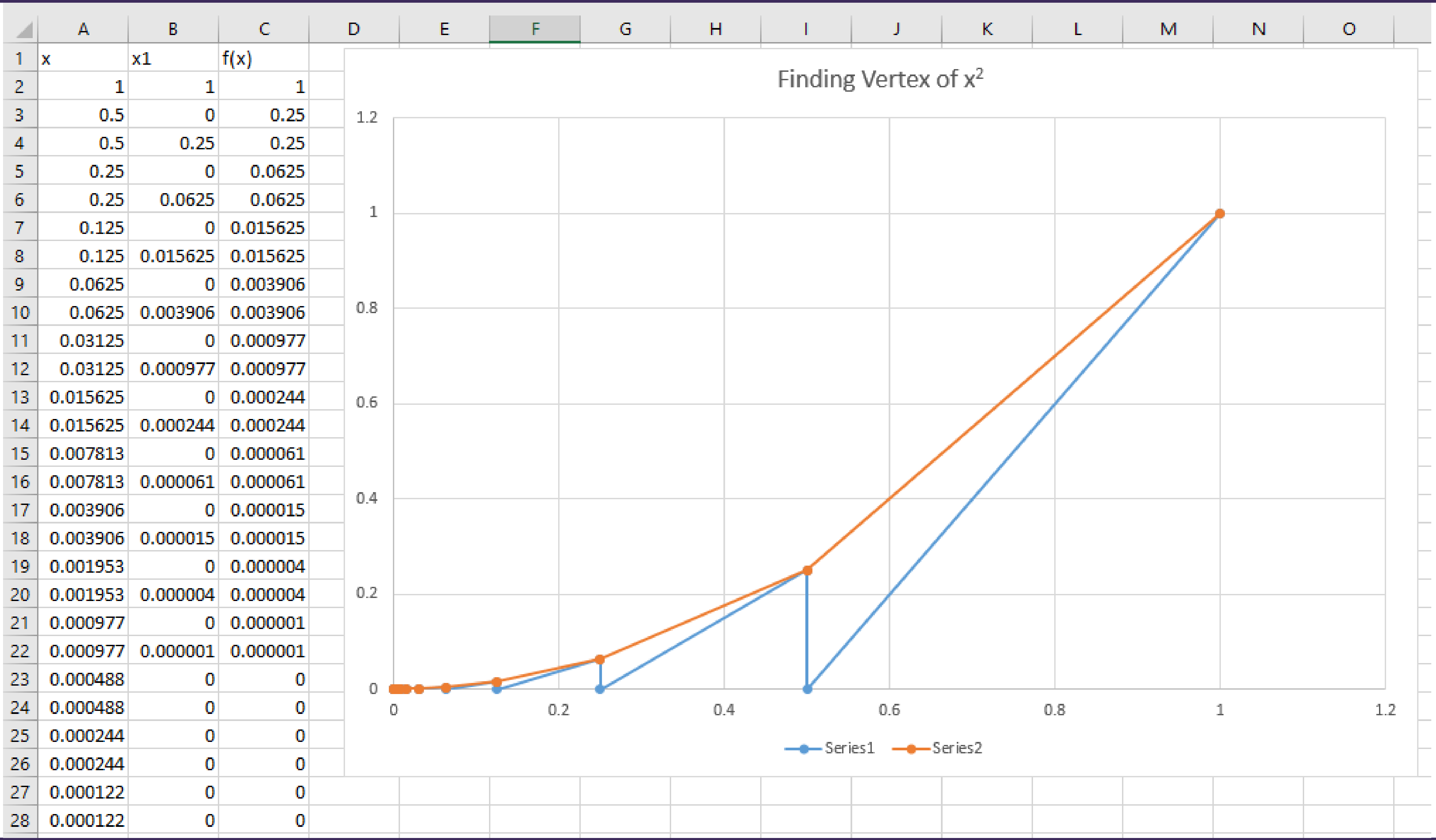
Tangent Line equation:

$y - y_0 = f'(x_0)\,(x - x_0)$

When y = 0, (intersection of y axis)

$x_1 = x_0 - y_0 / f'(x_0)$

When the difference between x0 and x1 is smaller than step size, we consider it converged

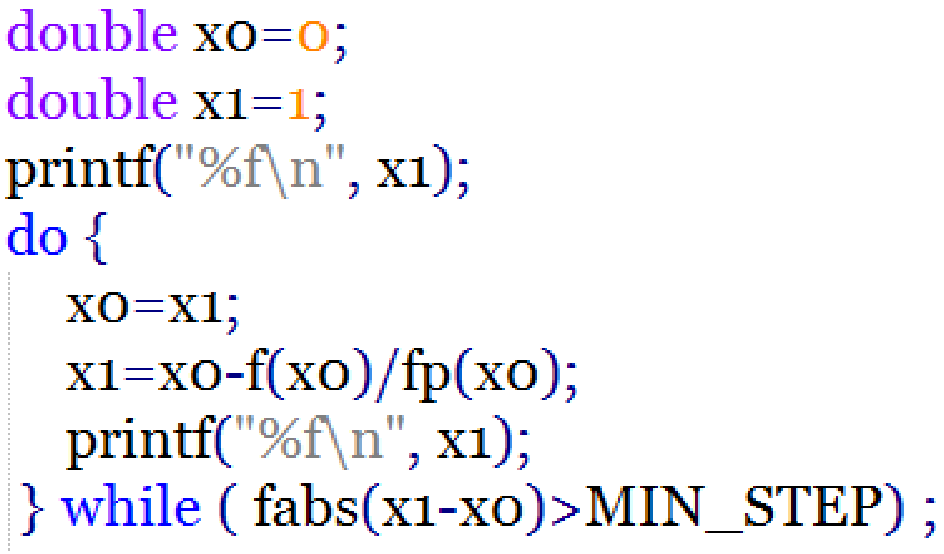


	A	B	C
1	x	x1	f(x)
2	1	1	1
3	0.5	0	0.25
4	0.5	0.25	0.25
5	0.25	0	0.0625
6	0.25	0.0625	0.0625
7	0.125	0	0.015625
8	0.125	0.015625	0.015625
9	0.0625	0	0.003906
10	0.0625	0.003906	0.003906
11	0.03125	0	0.000977
12	0.03125	0.000977	0.000977
13	0.015625	0	0.000244
14	0.015625	0.000244	0.000244
15	0.007813	0	0.000061
16	0.007813	0.000061	0.000061
17	0.003906	0	0.000015
18	0.003906	0.000015	0.000015
19	0.001953	0	0.000004
20	0.001953	0.000004	0.000004
21	0.000977	0	0.000001
22	0.000977	0.000001	0.000001
23	0.000488	0	0
24	0.000488	0	0
25	0.000244	0	0
26	0.000244	0	0
27	0.000122	0	0
28	0.000122	0	0

Demo Program:

convergence.c

Go gcc!!!



```
double x0=0;
double x1=1;
printf("%f\n", x1);
do {
   x0=x1;
   x1=x0-f(x0)/fp(x0);
   printf("%f\n", x1);
} while ( fabs(x1-x0)>MIN_STEP) ;
```

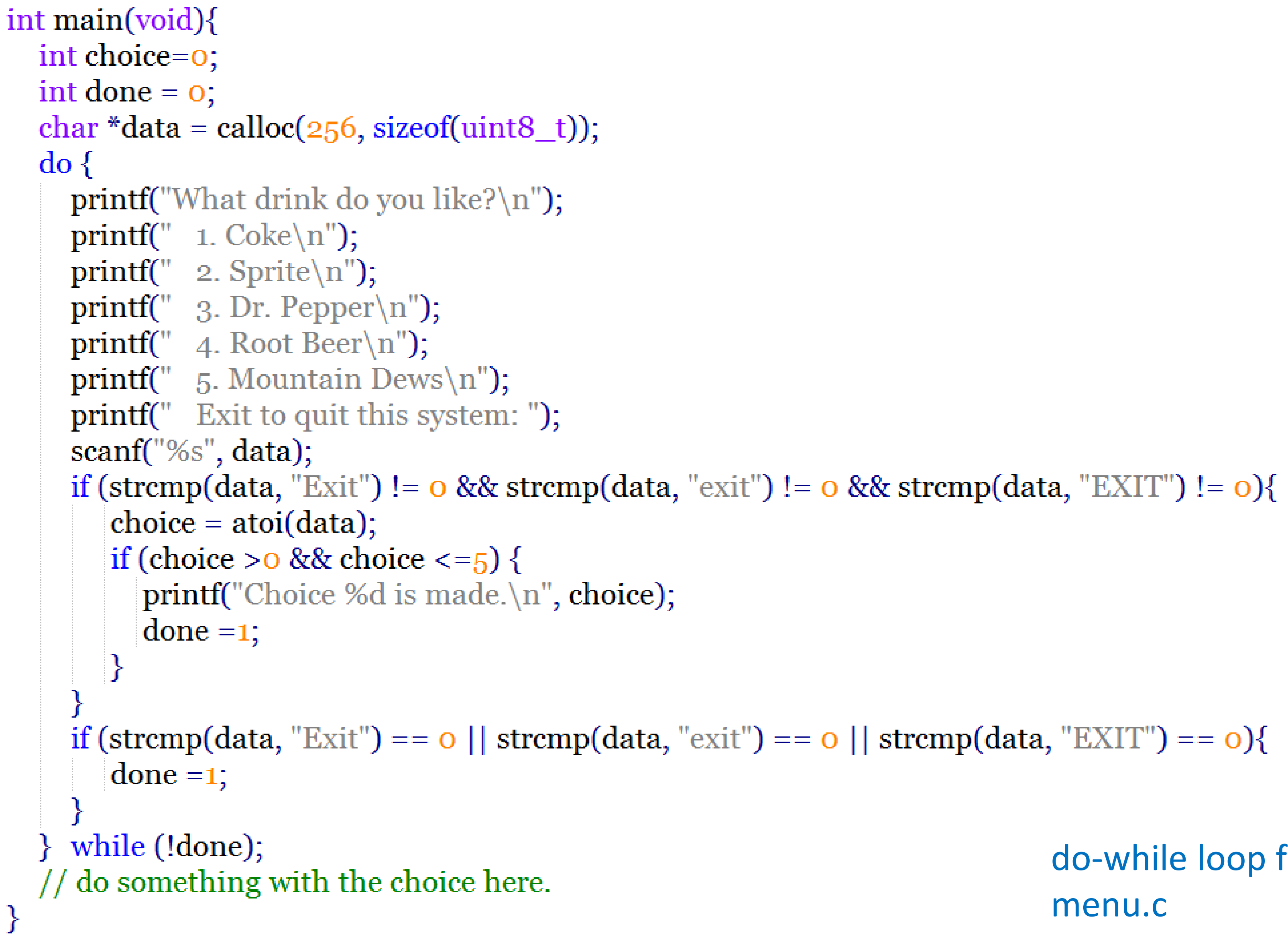
Build: gcc convergence.c –o convergence

Run file: convergence > data.txt   REM redirection of output

The data.txt file is used to make the plot in Excel.

LECTURE 8

Menu Selection



```c
int main(void){
    int choice=0;
    int done = 0;
    char *data = calloc(256, sizeof(uint8_t));
    do {
        printf("What drink do you like?\n");
        printf("   1. Coke\n");
        printf("   2. Sprite\n");
        printf("   3. Dr. Pepper\n");
        printf("   4. Root Beer\n");
        printf("   5. Mountain Dews\n");
        printf("   Exit to quit this system: ");
        scanf("%s", data);
        if (strcmp(data, "Exit") != 0 && strcmp(data, "exit") != 0 && strcmp(data, "EXIT") != 0){
            choice = atoi(data);
            if (choice >0 && choice <=5) {
                printf("Choice %d is made.\n", choice);
                done =1;
            }
        }
        if (strcmp(data, "Exit") == 0 || strcmp(data, "exit") == 0 || strcmp(data, "EXIT") == 0){
            done =1;
        }
    }  while (!done);
    // do something with the choice here.
}
```

do-while loop for menu design.
menu.c

Do-while Loop for Menu Design

menu.c

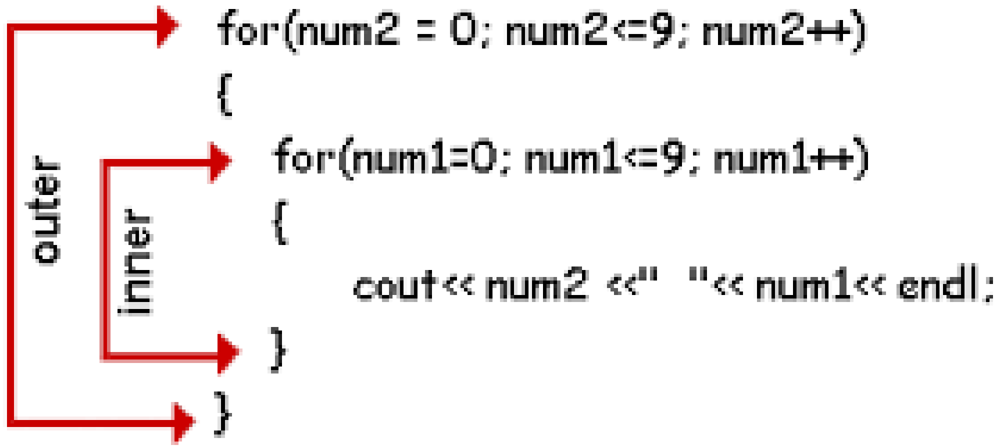
Go gcc!!!

LECTURE 9

Nested Loop (Multiplication table)

Nested Loop

- A loop inside another loop is called a nested loop. The depth of nested loop depends on the complexity of a problem.
- We can have any number of nested loops as required.
- Consider a nested loop where the outer loop runs *n* times and consists of another loop inside it.
- The inner loop runs *m* times. Then, the total number of times the inner loop runs during the program execution is *n*m*.



```
for(num2 = 0; num2<=9; num2++)
{
   for(num1=0; num1<=9; num1++)
   {
        cout<< num2 <<"  "<< num1<< endl;
   }
}
```

Nested Loop

Each layer of loop has its own functionality:

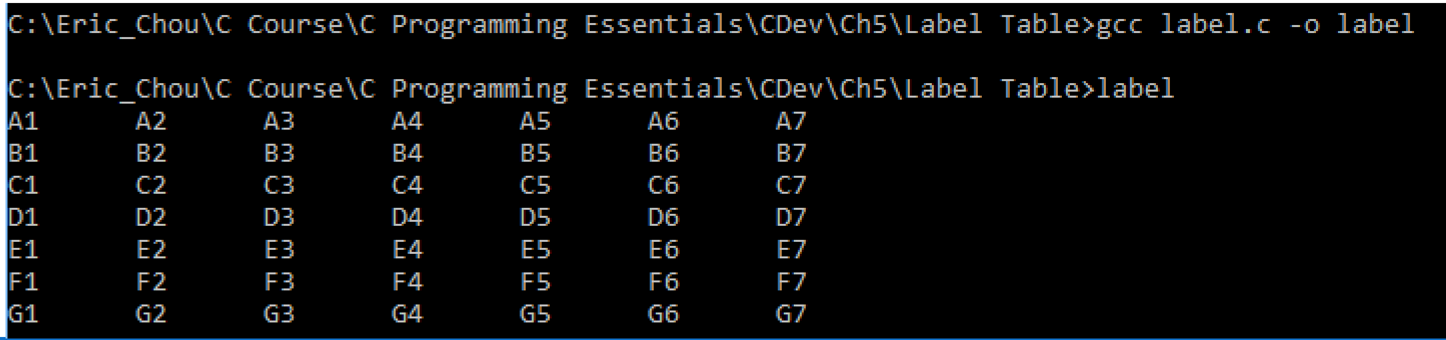
- Repetition-loop performs a certain task many times.
- Index-loop creates index space.
- Token-loop access new tokens.
- Sentinel-Controlled loop waits for a condition to finish repetition.
- While loop for state machine.

Nested Loop (1): Index Space

Demo Program: label.c

```c
#include <stdio.h>
int main()
{
  int alpha, code;
  for(alpha='A'; alpha<='G'; alpha=alpha+1){
    for(code=1; code<=7; code=code+1){
      printf("%c%d\t",alpha,code);
    }
    putchar('\n');   /* end a line of text */
  }
  return(0);
}
```

Go gcc!!!

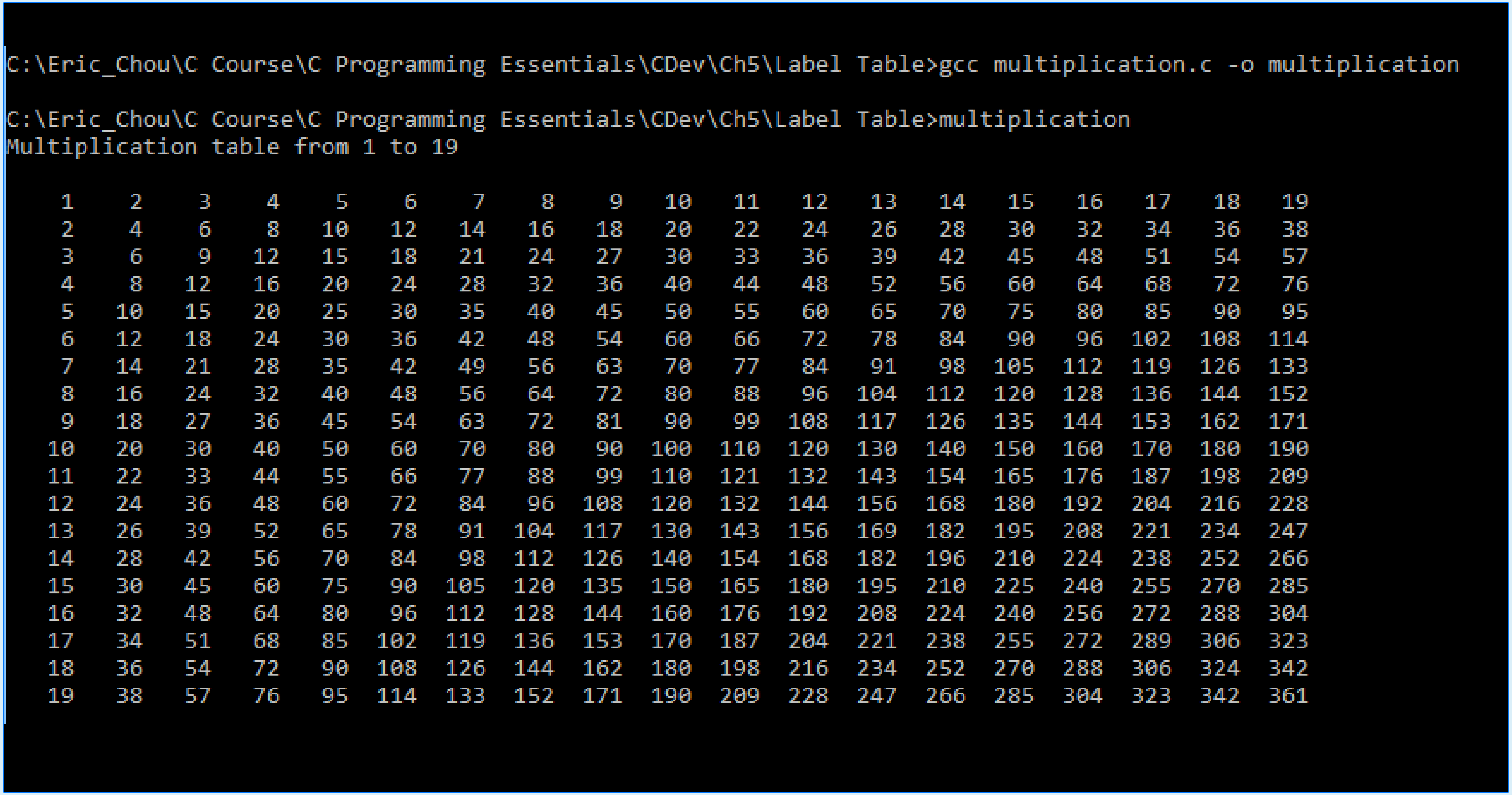


```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\Label Table>gcc label.c -o label

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\Label Table>label
A1      A2      A3      A4      A5      A6      A7
B1      B2      B3      B4      B5      B6      B7
C1      C2      C3      C4      C5      C6      C7
D1      D2      D3      D4      D5      D6      D7
E1      E2      E3      E4      E5      E6      E7
F1      F2      F3      F4      F5      F6      F7
G1      G2      G3      G4      G5      G6      G7
```

Multiplication Table (Multiple Index Loop)

Demo Program: multiplication.c (in Label Table package)

Go gcc!!!



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\Label Table>gcc multiplication.c -o multiplication

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\Label Table>multiplication
Multiplication table from 1 to 19

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19
    2    4    6    8   10   12   14   16   18   20   22   24   26   28   30   32   34   36   38
    3    6    9   12   15   18   21   24   27   30   33   36   39   42   45   48   51   54   57
    4    8   12   16   20   24   28   32   36   40   44   48   52   56   60   64   68   72   76
    5   10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95
    6   12   18   24   30   36   42   48   54   60   66   72   78   84   90   96  102  108  114
    7   14   21   28   35   42   49   56   63   70   77   84   91   98  105  112  119  126  133
    8   16   24   32   40   48   56   64   72   80   88   96  104  112  120  128  136  144  152
    9   18   27   36   45   54   63   72   81   90   99  108  117  126  135  144  153  162  171
   10   20   30   40   50   60   70   80   90  100  110  120  130  140  150  160  170  180  190
   11   22   33   44   55   66   77   88   99  110  121  132  143  154  165  176  187  198  209
   12   24   36   48   60   72   84   96  108  120  132  144  156  168  180  192  204  216  228
   13   26   39   52   65   78   91  104  117  130  143  156  169  182  195  208  221  234  247
   14   28   42   56   70   84   98  112  126  140  154  168  182  196  210  224  238  252  266
   15   30   45   60   75   90  105  120  135  150  165  180  195  210  225  240  255  270  285
   16   32   48   64   80   96  112  128  144  160  176  192  208  224  240  256  272  288  304
   17   34   51   68   85  102  119  136  153  170  187  204  221  238  255  272  289  306  323
   18   36   54   72   90  108  126  144  162  180  198  216  234  252  270  288  306  324  342
   19   38   57   76   95  114  133  152  171  190  209  228  247  266  285  304  323  342  361
```

LECTURE 9

Nested Token-Loop and index-loop

Lab Project: Word Count of a Text file

Goal:

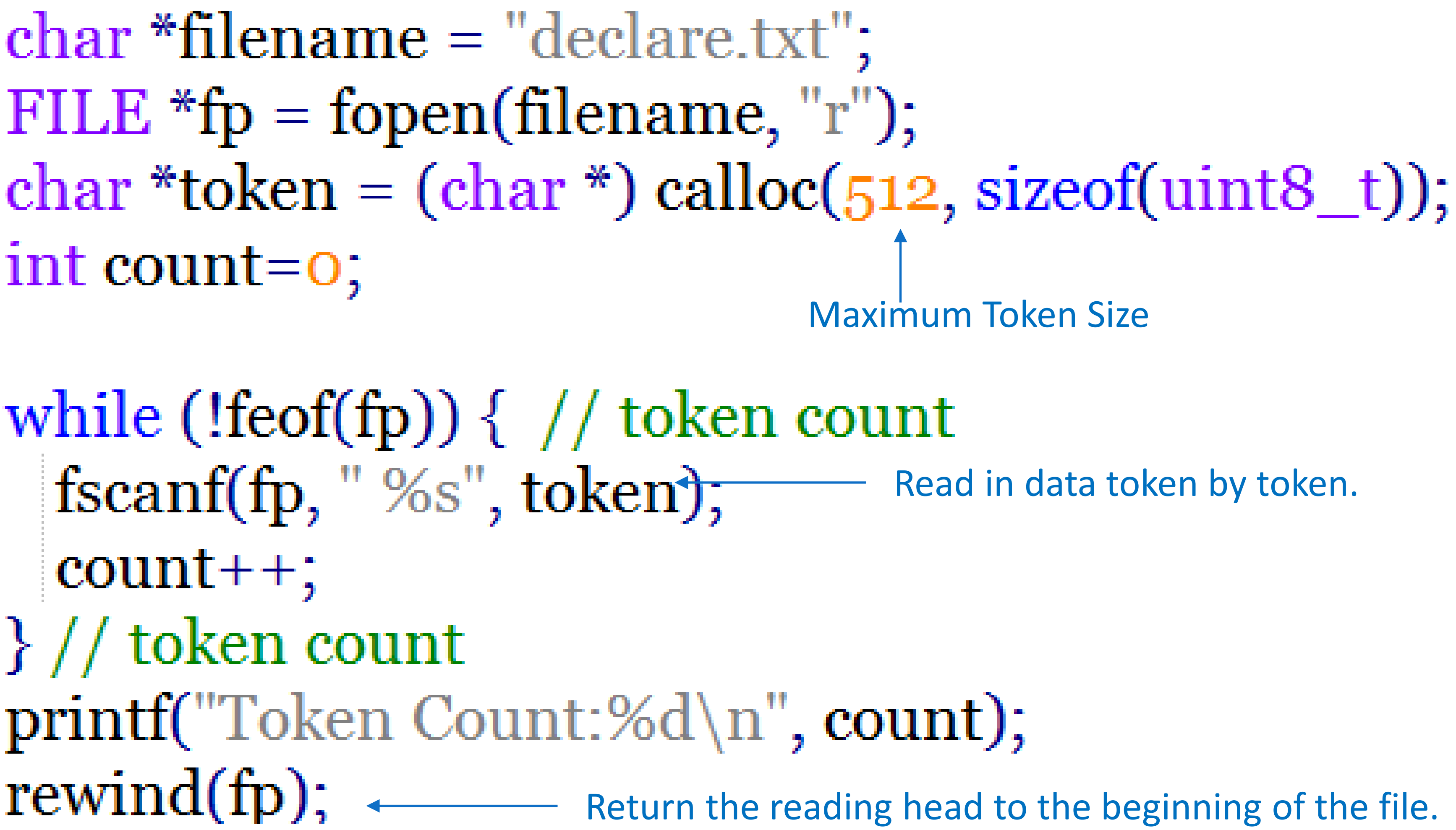
Calculate the total identical words used in a text file. All of the words should only be counted once.

Techniques:

- Nested Loop.
- Outer Loop: traversal through the text file. Read in all tokens as a string.
- Pre-processing on each token. Trim non-letter characters.
- Pre-processing on converting all of the tokens to lowercase.
- Inner Loop: check if the incoming token already been counted. Otherwise, add it into the counted token array.

Step 1:

- Decide how many tokens are in the file.
  1. Open the text file.
  2. Read in strings one by one.
  3. Advance the counter for number of token read.
  4. Allocate an array for the tokens being identified as distinct.
  5. Rewind the text file.



```
char *filename = "declare.txt";
FILE *fp = fopen(filename, "r");
char *token = (char *) calloc(512, sizeof(uint8_t));
int count=0;

while (!feof(fp)) {  // token count
    fscanf(fp, " %s", token);
    count++;
} // token count
printf("Token Count:%d\n", count);
rewind(fp);
```

Maximum Token Size

Read in data token by token.

Return the reading head to the beginning of the file.

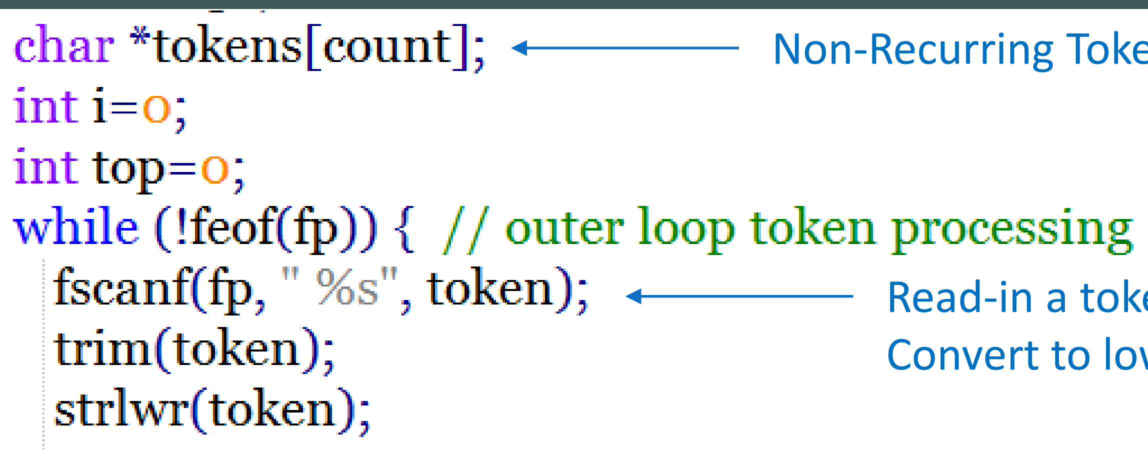
Step2: Outer Loop – Read-in Tokens

Outer loop:

- Read in a token
- Remove non-letter characters
- Convert to lowercase.

Inner Loop:

- Check if the token already been included in the token list.
- If not, add it into the list.



```
char *tokens[count];
int i=0;
int top=0;
while (!feof(fp)) {  // outer loop token processing
  fscanf(fp, " %s", token);
  trim(token);
  strlwr(token);
```

Non-Recurring Tokens

Read-in a token. Rip off white spaces and punctuations. Convert to lowercase.

//Put Inner Loop Here

Check if the token already exists

If not add it in

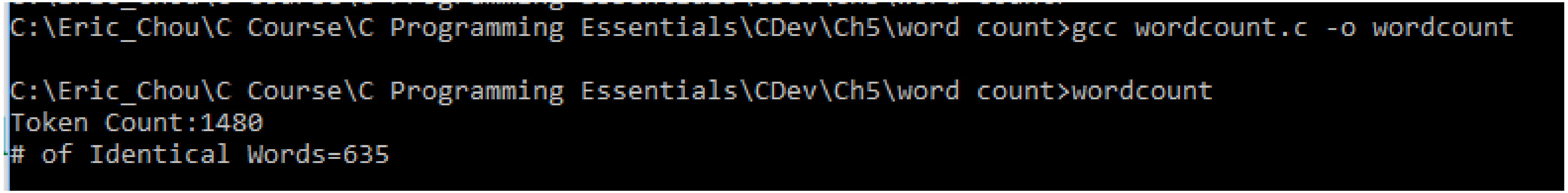
```
} // outer loop token processing
```

Note:

1. Conditional compilation by #ifdef DEBUG directives.

```
#ifdef DEBUG
for (i=0; i<top; i++) printf("%s\n", tokens[i]);
#endif
```

2. Results:



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\word count>gcc wordcount.c -o wordcount

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\word count>wordcount
Token Count:1480
# of Identical Words=635
```

3. Application of this nested loop:
    - Finding identical tokens (case-sensitive or case-insensitive)
    - Finding the non-recurring elements from a text file.

Demo Program:

wordcount.c

Go gcc!!!

LECTURE 10

Lottery Game

Lab Project: 4 digit number lottery
(Lottery.java, Sample Answer: LotteryAnswer.java)

Write a program to allow player to play three different lottery games. All of these three games are based on guess a lottery number of 4 digits (such as 1234, leading 0s are allowed, 0000 is OK).

In each play, computer generates a lottery number and player guesses another 4 digit number.

Lab Project: 4 digit number lottery

(Lottery.java, Sample Answer: LotteryAnswer.java)

Game 1: if all 4 numbers matched, player win $50 prize.

Game 2: If player's guess number match first two digits, we call it front pair matched, win $5.

If player's guess number match middle two digits, we call it middle pair matched, win $5.

If player's guess number match ending two digits, we call it end pair matched, win $5.

Player may win at most 4 matched pairs to win $15.

Game 3: If player's guess number match each of the digits, we call it single digit matched, win $2.

Player may win at most 4 matches to win $8.

Lab Project: 4 digit number lottery

(Lottery.java, Sample Answer: LotteryAnswer.java)

Player can only enter one game but not 2 or 3 games. (He can choose a game of higher risk higher return or another game of lower risk lower return.)

Print out the prize the player won in this current game.

Then, ask if the player wants to play again?

When You are developing, you may create a debug mode to pre-watch the lottery number.

```
final boolean DEBUG_MODE = true; // turn on the debug
mode to show lottery number
```

```
// Somewhre, after the lottery number has been generated.
if (DEBUG_MODE) System.out.println("Lottery: " + lottery);
```

The odds for Game1 to win is only 1/1000. Therefore, it is hard to debug if you just try to guess.

Generation of Lottery Number

Use String, do not use int data type. (integer can not show leading 0s).

```
lottery = "";
lottery += (char) ((Math.random()*10)+'0'); // generate first digit
lottery += (char) ((Math.random()*10)+'0'); // generate second digit
lottery += (char) ((Math.random()*10)+'0'); // generate third digit
lottery += (char) ((Math.random()*10)+'0'); // generate fourth digit
// Up to this point, lottery has been created.
```

Rough Pseudo Code

```
Declare constants DEBUG_MODE;
Declare variables done, play as flags to control games
Declare sum for total won prize;
Declare lottery and guessed number strings.
Declare input stream from System.in
Declare game code
```

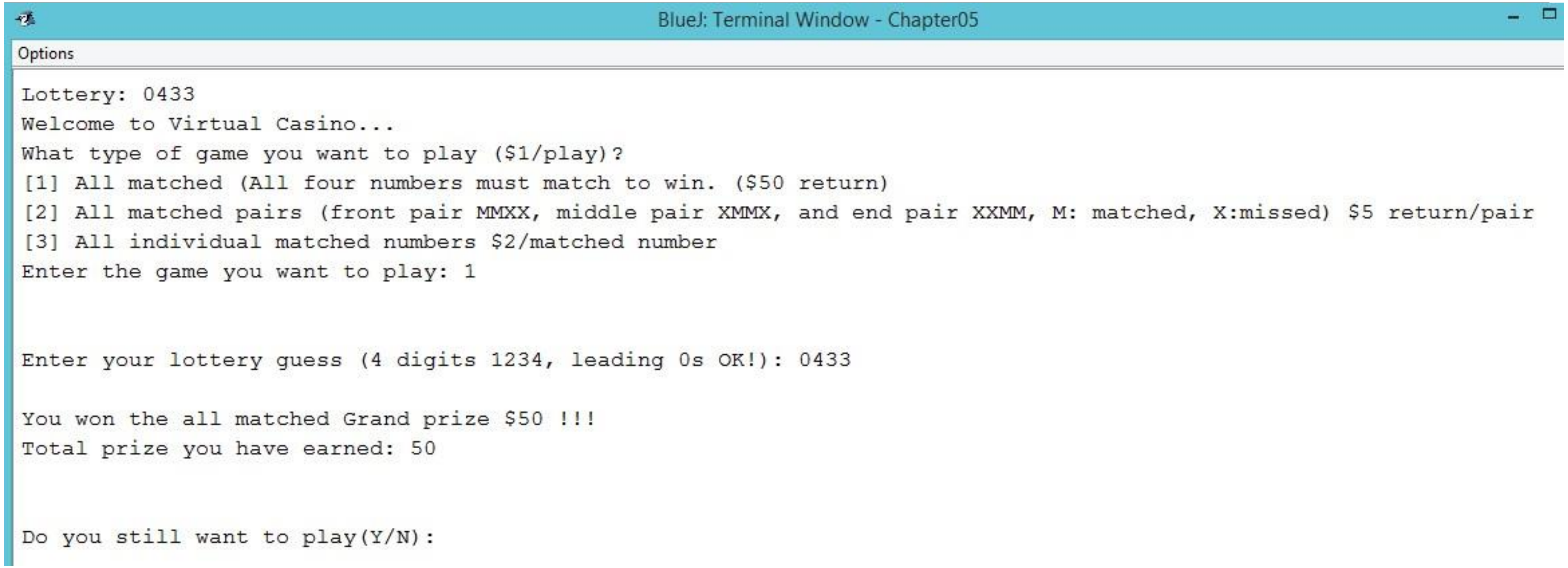
Rough Pseudo Code

```
do { 1. reset variables for the next game play.
     2. generate a lottery number.
        show lottery number if it is in DEBUG_MODE
     3. Show menus for game choice and, then, ask player to enter a game code
     4. Ask for player’s guessing number
     5. if (game == 1) check if he won the grand prize $50
        if (game == 2) check if he won each of the pairs. Sum up the prize won in this game.
        if (game == 3) check if he won each of the digits.  Sum up the prize won in this game.
     6. Show how much money the player won in current game.
     7. ask if the player want to play a new game. set the flags (done, play) properly.
} while (!done)
```

Prize checks

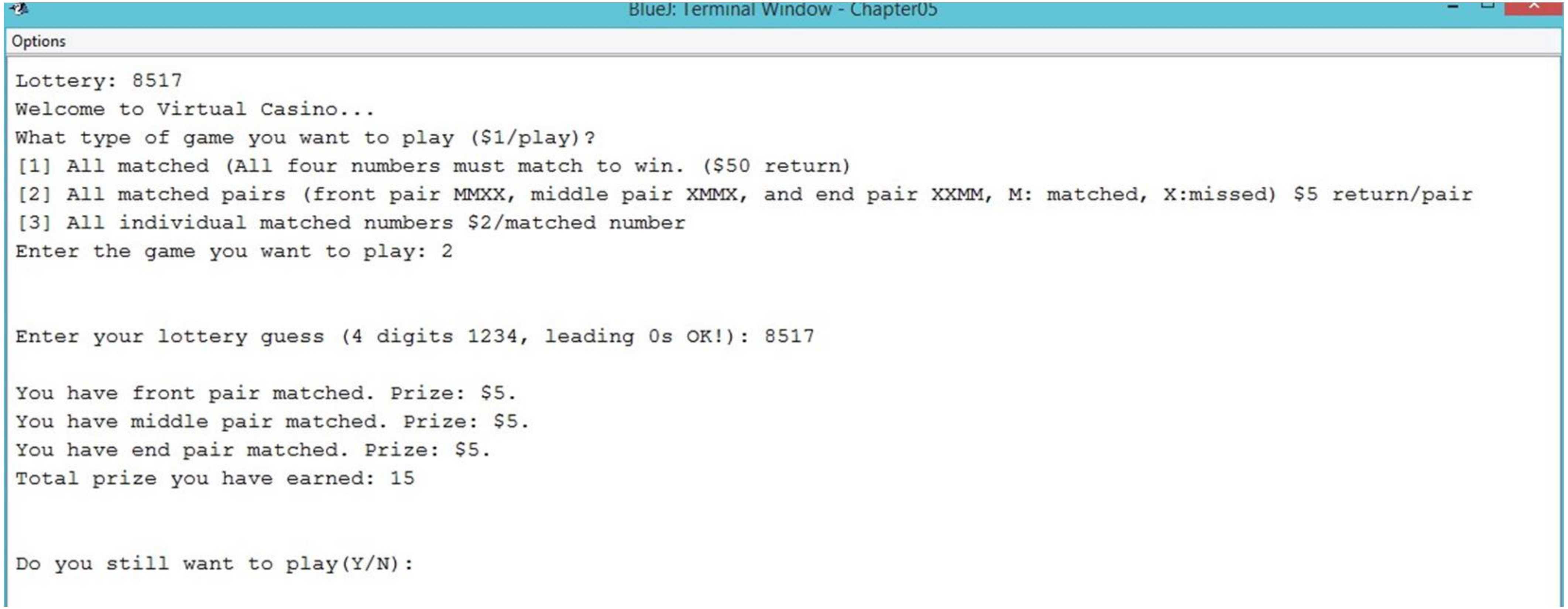
```
// All matched
 if (guess.equals(lottery)) {sum += 50;
      System.out.println("You won the all matched Grand prize $50 !!!");
 }
 // Front pair matched check
if (guess.substring(0,2).equals(lottery.substring(0,2))) {
     sum +=5;
     System.out.println("You have front pair matched. Prize: $5.");
}
// check for one digit
 if (guess.charAt(0)== lottery.charAt(0))
   {sum+=2; System.out.println("You matched the first number. Prize $2."); }
```

Expected Results. (LotteryAnswer.java)



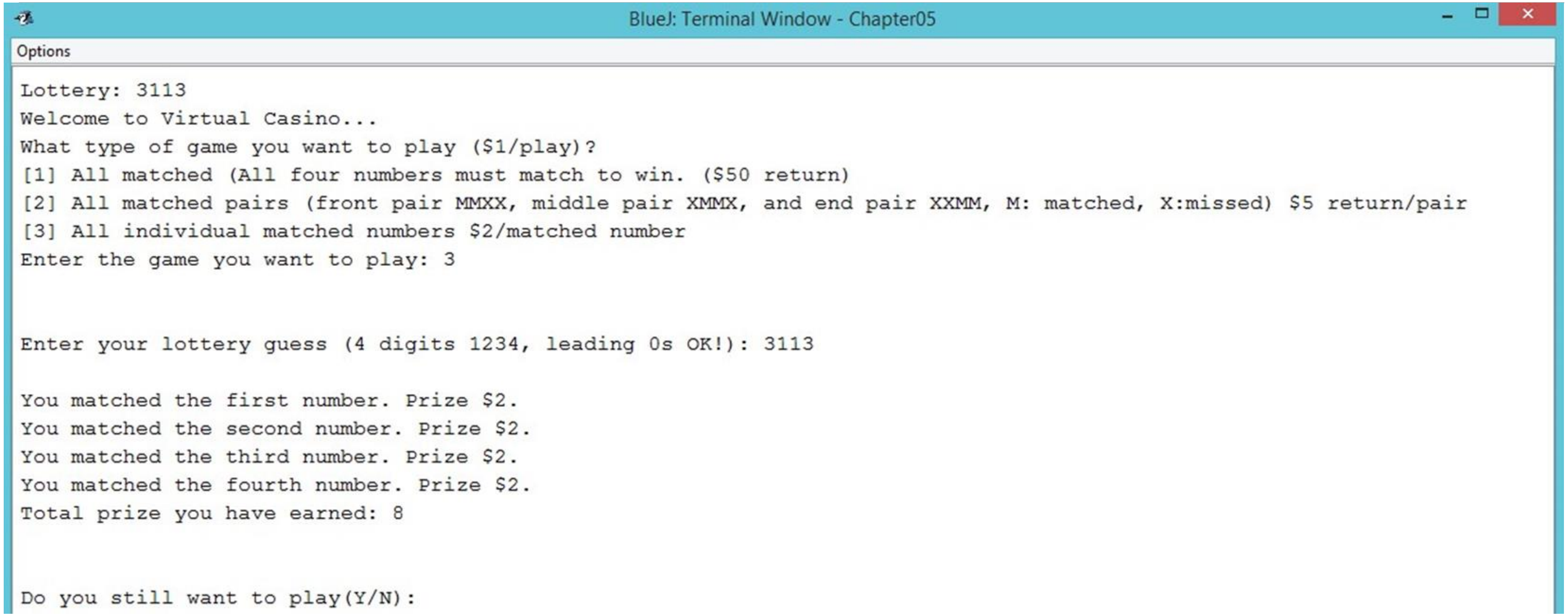
```
Lottery: 0433
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 1


Enter your lottery guess (4 digits 1234, leading 0s OK!): 0433

You won the all matched Grand prize $50 !!!
Total prize you have earned: 50


Do you still want to play(Y/N):
```

Expected Results.



```
Lottery: 8517
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 2


Enter your lottery guess (4 digits 1234, leading 0s OK!): 8517

You have front pair matched. Prize: $5.
You have middle pair matched. Prize: $5.
You have end pair matched. Prize: $5.
Total prize you have earned: 15


Do you still want to play(Y/N):
```

Expected Results.



```
Lottery: 3113
Welcome to Virtual Casino...
What type of game you want to play ($1/play)?
[1] All matched (All four numbers must match to win. ($50 return)
[2] All matched pairs (front pair MMXX, middle pair XMMX, and end pair XXMM, M: matched, X:missed) $5 return/pair
[3] All individual matched numbers $2/matched number
Enter the game you want to play: 3


Enter your lottery guess (4 digits 1234, leading 0s OK!): 3113

You matched the first number. Prize $2.
You matched the second number. Prize $2.
You matched the third number. Prize $2.
You matched the fourth number. Prize $2.
Total prize you have earned: 8


Do you still want to play(Y/N):
```

LECTURE 11

State Machine (while-Loop nested with state machine)

Lab Project
Character Stream and State Machine

Project Goal:

Read in a sequence of characters form from a file.

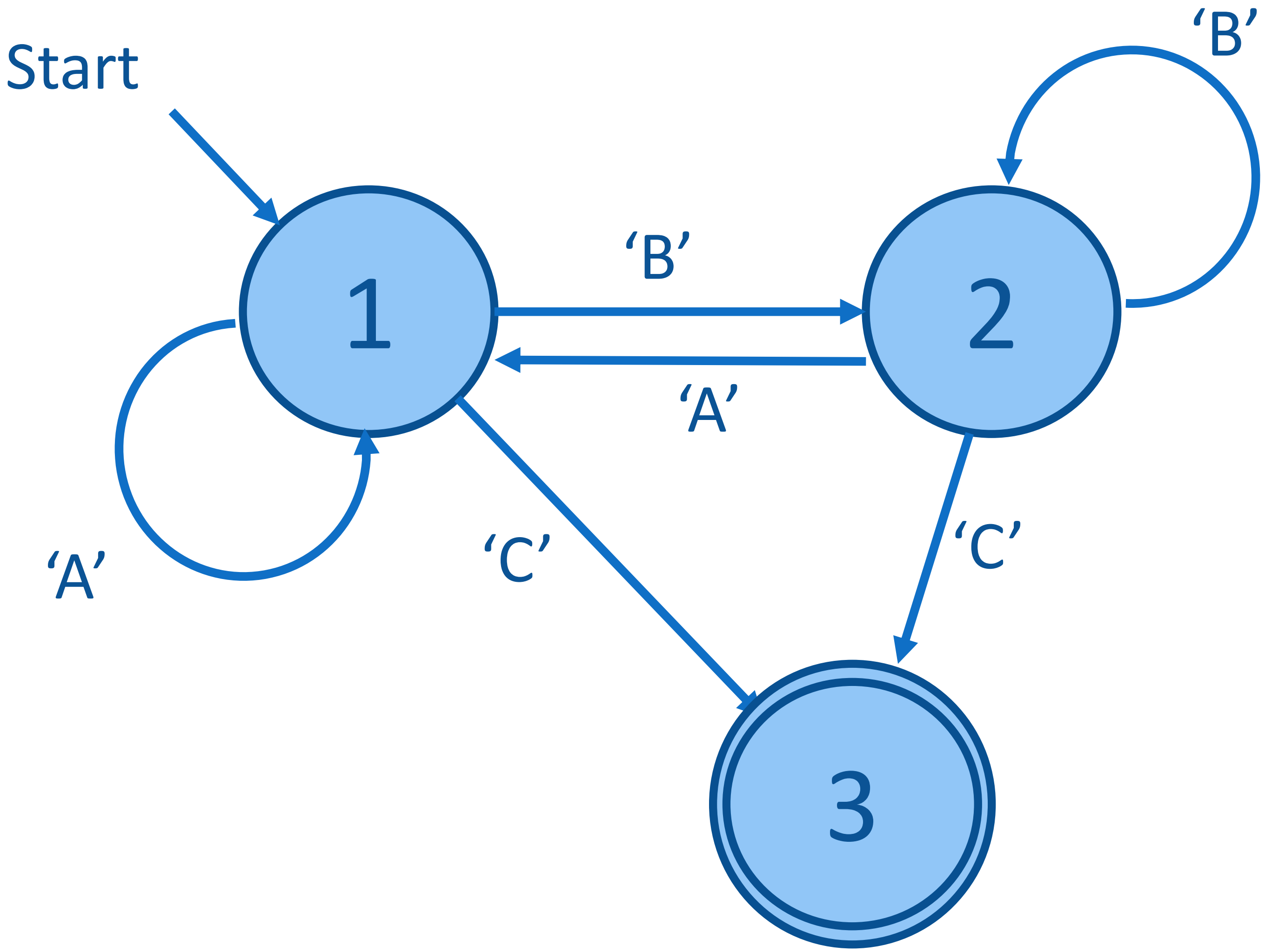
The character sequence needs state machine to handle its input. If the input form does not follow the format, syntactical error will be reported.

If it follows the input format, the input form will be accepted and the student record will be printed.

Techniques:

Token Loop. (Using the outer loop from letter count project.)

State Machine using switch-case statement to manage the state transitions.


Start
1
2
3
'B'
'A'
'B'
'A'
'C'
'C'

Tokenizer and State Machine can be used to Analyze the Input Text Data

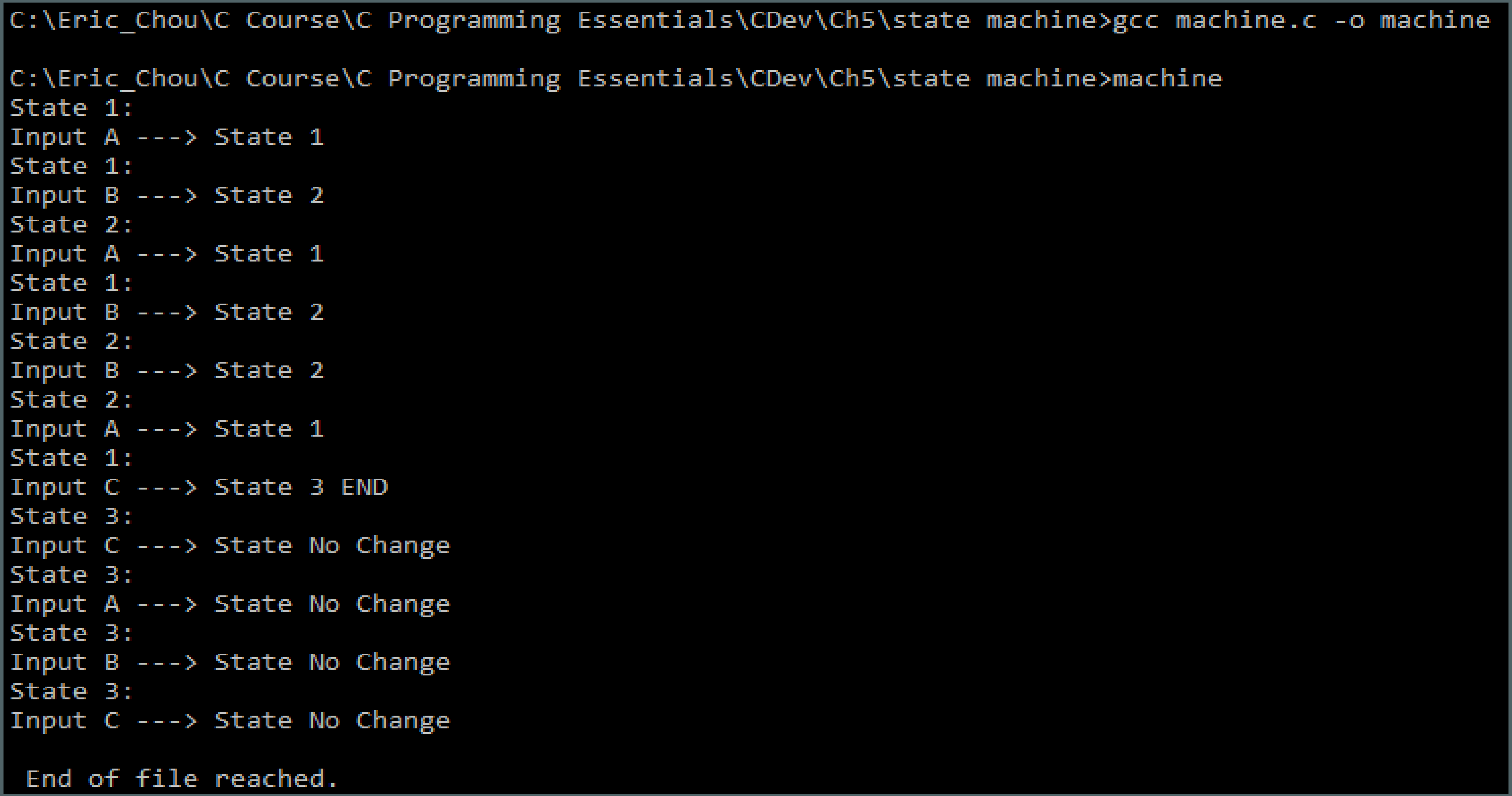
- In this chapter, the letter count program has a letter tokenizer. The word count program has a word(string) tokenizer.
- State machine (nested case-switch) program can be used to perform to analyze the text (letter sequence or token sequence).

- Tokenizers can be implemented as state machines, but with these important differences:
    - To *succeed* (recognize a token), the tokenizer does not have to reach the end of input; it only has to reach a final state
    - When the tokenizer returns a token, the remainder of the input string is kept for use in getting the remaining tokens
- Tokenizers are almost always implemented as state machines
- We'll do a quick tokenizer to recognize tokens in arithmetic expressions:
    - Integers (digits only)
    - Variables (letters and digits, starting with a letter)
    - Operators, + - * / %
    - Parentheses, ( )
    - Errors (anything not in the above list)

Demo Program:

machine.c

Go gcc!!!



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\state machine>gcc machine.c -o machine

C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\state machine>machine
State 1:
Input A ---> State 1
State 1:
Input B ---> State 2
State 2:
Input A ---> State 1
State 1:
Input B ---> State 2
State 2:
Input B ---> State 2
State 2:
Input A ---> State 1
State 1:
Input C ---> State 3 END
State 3:
Input C ---> State No Change
State 3:
Input A ---> State No Change
State 3:
Input B ---> State No Change
State 3:
Input C ---> State No Change

 End of file reached.
```

LECTURE 12

Break Levels in C language

Break and Continue

- The loops have one expression that decides whether the iterative process should be terminated. It is sometimes convenient to be able to exit from the loop.
- `break` statement provides an early exit from the `for`, `while` and `do` loops.
- It also provides an exit from switch statement.
- `continue` statement causes the jump to the end of the loop body, skipping the statements only once. The loop may continue.

```
while (...) {
    ...
    continue;
    ...
    cont: ;
}
```

```
do {
    ...
    continue;
    ...
    cont: ;
}
```

```
for (...) {
    ...
    continue;
    ...
    cont: ;
}
```

Note:
continue: skip the current iteration

break and continue

- Example

```
for ( i = 0; i < n; i++ ) {
   if ( a[i] < 0 ) continue;
    /*  Process only non-negative elements of the array */
   ...
}
```

- Example

```
for ( i = 2; i <= (int)sqrt(n); i++ ) {
   if ( n % i == 0 ) break;   // break when it is not a prime number
 }
if (!(n%i)) printf(“Prime\n”);
else printf(“Not prime\n”);
```

continue statement

- Used to bypass the remainder of the current pass through a loop.
- Computation proceeds directly to the next pass through the loop.
- Example:

```
for( count=1; x <=100; ++count) {
  scanf(“%f “, &x);
  if (x < 0) {
    printf(“ it’s a negative no\n”)
    continue;
   }
 /*computation for non-negative
  numbers here*/
}
```

Power of break

Syntax is

- `break;`

used to terminate **loop** or exit from a **switch**.

In case of several nested `while`,`do-while`, `for` or `switch` statements, a `break` statement will cause a transfer of control out of the immediate enclosing statement.

break statement: Example

```
int count =0;
while (count <=n) {
  while( c=getchar()!=‘\n’){
      if ( c ==‘@’) break;
       …   …   …
   }
   ++count;
}
```

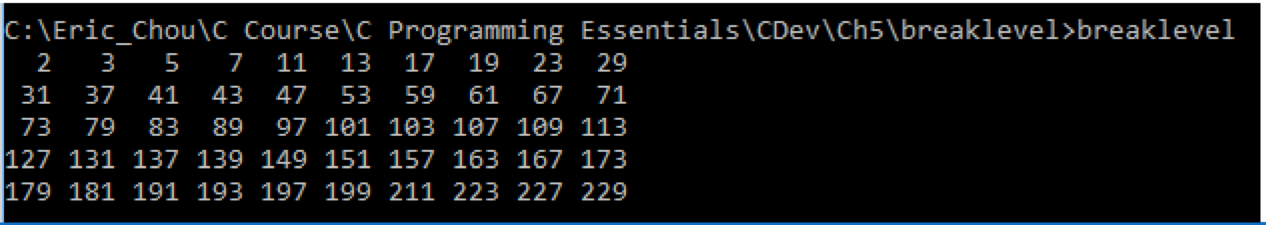
Demo Program:

breaklevel.c (finding the first 50 prime numbers)

Go gcc!!!

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(void){
   int i;
   int count;
   int n=4;  printf("  2   3 ");
   for (count = 2; count< 50; n++){
     for ( i = 2; i <= (int) sqrt(n); i++ ) {
          if ( n % i == 0 ) break;   // break when it is not a prime number
      }
     if (n%i !=0) {
        printf("%3d ", n);
        if (count % 10 == 9) printf("\n");
        count++;
     }
     //n++;
   }
}
```

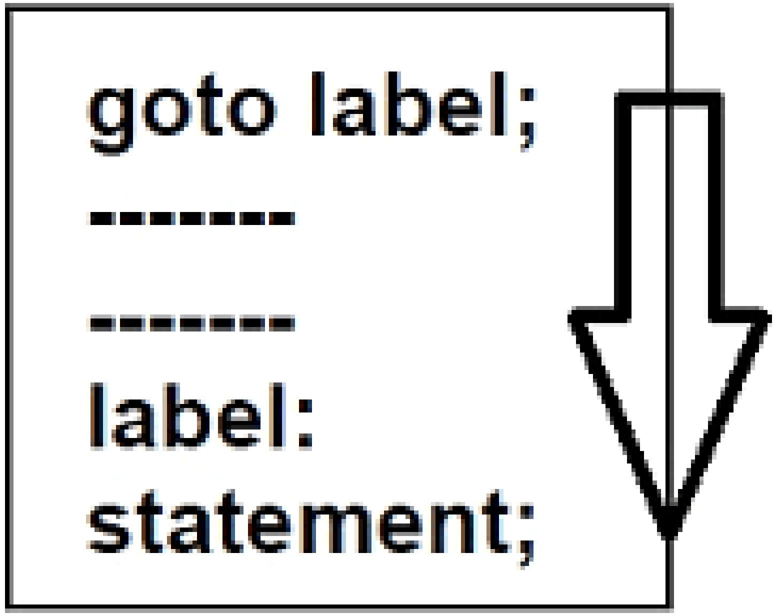
Print prime numbers →



```
C:\Eric_Chou\C Course\C Programming Essentials\CDev\Ch5\breaklevel>breaklevel
  2   3   5   7  11  13  17  19  23  29
 31  37  41  43  47  53  59  61  67  71
 73  79  83  89  97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```

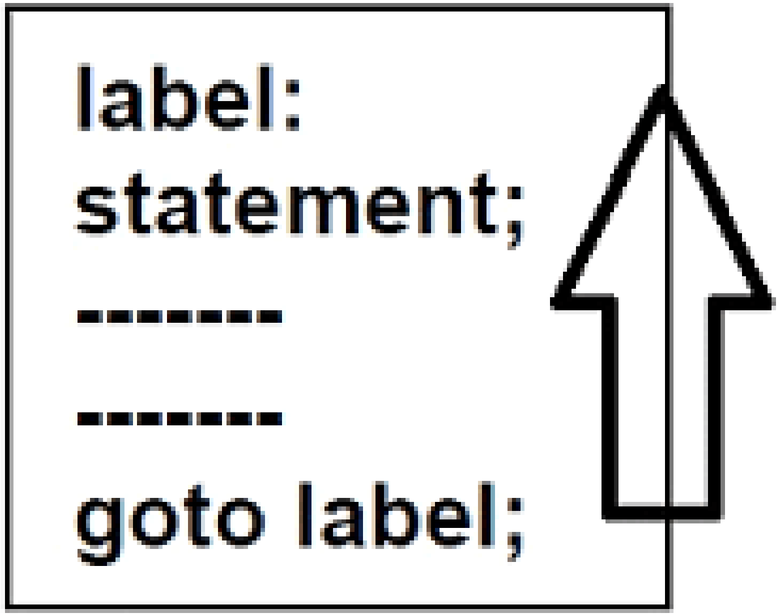
LECTURE 13

Goto and labels



```
goto label;
--------
--------
label:
statement;
```

Forward Jump



```
label:
statement;
--------
--------
goto label;
```

Backward Jump

goto statement

Note that you can tag any statement in C with an identifier.

And then, can use **goto** to directly transfer the program control to that statement .

Example:

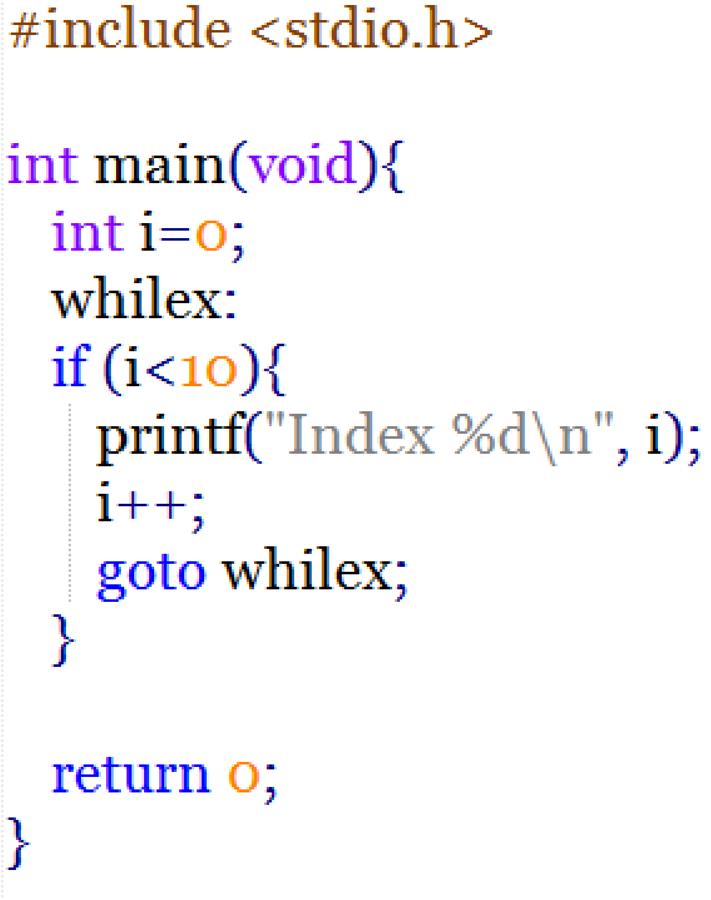
```
 while ( x <= 10) {
        … … …
        if (x<0)  goto chkErr;
        … … …
        scanf(“%f”, &x);
  }
 chkErr: {
         printf(“found a negative value!\n”);
         … … …
  }
```

Note that use of **goto** is discouraged. It encourages logic that skips all over the program . Difficult to track the code. Hard to debug.

Demo Program:

gotox.c (Using label like assembly language)

Go gcc!!!

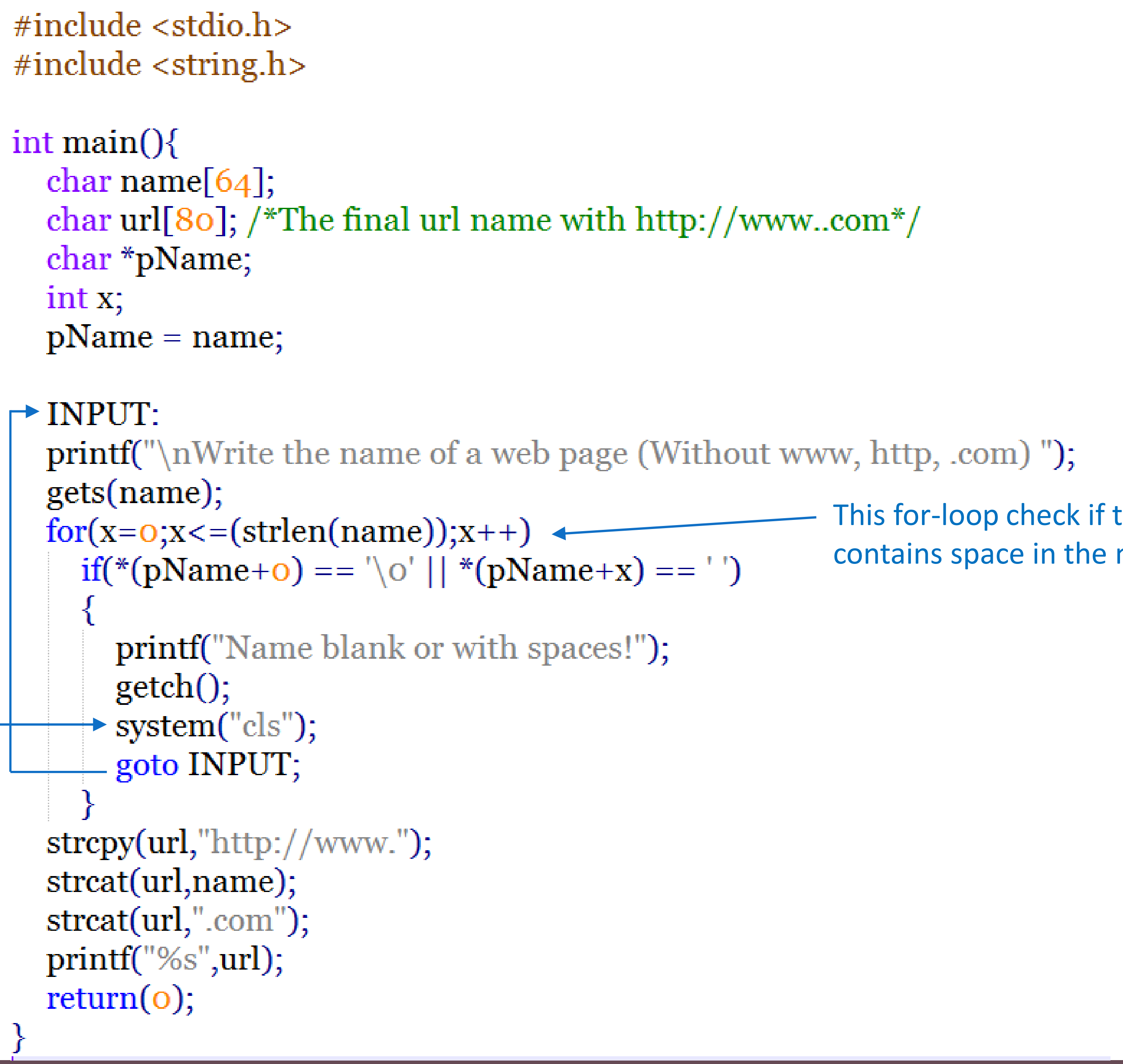


```
#include <stdio.h>

int main(void){
  int i=0;
  whilex:
  if (i<10){
    printf("Index %d\n", i);
    i++;
    goto whilex;
  }

  return 0;
}
```

Using goto to handle errors

Deme Program: url.c

- We try to use goto and labels to handle errors.
- system("cls"); // clear screen.
- Use goto INPUT to repeat the input of domain name if the domain name entered is not correct.
- For all un-structured program, there is always a equivalent structured program which provides the same result.



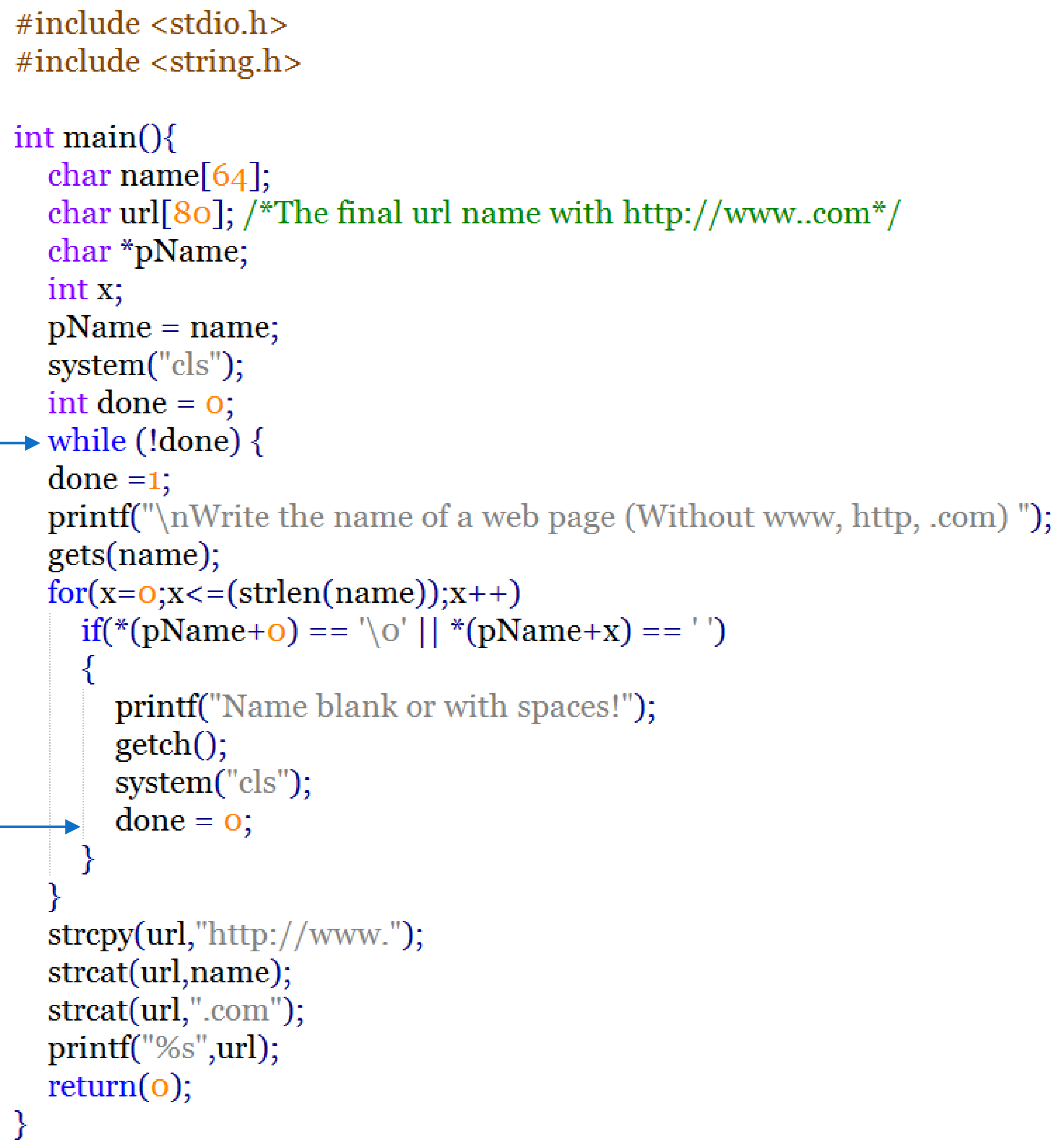
```
#include <stdio.h>
#include <string.h>

int main(){
    char name[64];
    char url[80]; /*The final url name with http://www..com*/
    char *pName;
    int x;
    pName = name;

    INPUT:
    printf("\nWrite the name of a web page (Without www, http, .com) ");
    gets(name);
    for(x=0;x<=(strlen(name));x++)
        if(*(pName+0) == '\0' || *(pName+x) == ' ')
        {
            printf("Name blank or with spaces!");
            getch();
            system("cls");
            goto INPUT;
        }
    strcpy(url,"http://www.");
    strcat(url,name);
    strcat(url,".com");
    printf("%s",url);
    return(0);
}
```

goto gives you a quick way to write problem. But it is hard to debug.

This for-loop check if the string contains space in the middle.

System call for clearance of console screen.

url.c



```c
#include <stdio.h>
#include <string.h>

int main(){
    char name[64];
    char url[80]; /*The final url name with http://www..com*/
    char *pName;
    int x;
    pName = name;
    system("cls");
    int done = 0;
    while (!done) {
    done =1;
    printf("\nWrite the name of a web page (Without www, http, .com) ");
    gets(name);
    for(x=0;x<=(strlen(name));x++)
        if(*(pName+0) == '\0' || *(pName+x) == ' ')
        {
            printf("Name blank or with spaces!");
            getch();
            system("cls");
            done = 0;
        }
    }
    strcpy(url,"http://www.");
    strcat(url,name);
    strcat(url,".com");
    printf("%s",url);
    return(0);
}
```

Sentinel-Controlled
Loop can be used to
replace goto labels.

flag setting to replace goto

urlwhile.c

LECTURE 14

Two Dice Game (TBD)