C Programming for Engineers

Structured Program

UNIVERSITY AT ALBANY
State University of New York

ICEN 360– Spring 2017

Prof. Dola Saha

Switch Statement

- Used to select one of several alternatives
- useful when the selection is based on the value of
  - a single variable
  - or a simple expression
- values may be of type int or char
  - not double

`switch` Statement

```
switch (controlling expression) {
        label set1
                statements1
                break;
        label set2
                statements2
                break;
                .
                .
                .
        label setn
                statementsn
                break;
```

switch Statement Example

```c
#include <stdio.h>

int main(void)
{
    int grade = 80;
    switch (grade)
    {
        case 90:
            printf("The grade is 90\n");
            break;
        case 80:
            printf("The grade is 80\n");
            break;
        default:
            printf("The grade is unknown\n");
            break;
    }
}
```

Class Assignment

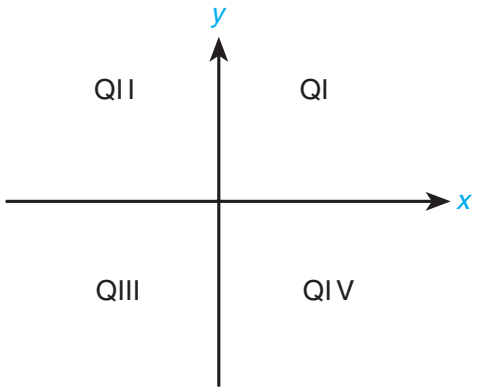
- Write a program that takes letter grade ‘A’, ‘B’, ‘C’, ‘D’ or ‘E’ and prints the range of score as below.

Grading Scale

A: 100-95 points A-: 94-90 points
B+: 89-87 points B: 86-84 points B-: 83-80 points
C+: 79-77 points C: 76-73 points C-: 72-70 points
D+: 69-67 points D: 66-63 points D-: 62-60 points
E: 59 points and below

Class Assignment

- Write a program that takes the *x*–*y* coordinates of a point in the Cartesian plane and prints a message telling either an axis on which the point lies or the quadrant in which it is found.



- Sample lines of output:
  - (-1.0, -2.5) is in quadrant III
  - (0.0, 4.8) is on the *y*-axis

Iteration Statement

- Repeat a set of actions while some condition remains TRUE
- Example:
  - *While there are more students in class raising their hand* ← Condition
    Answer a question and let him/her lower the hand ← Action
  - *While there are more items on my shopping list*
    Purchase next item and cross it off my list
- Usually, action statements modify variables that affect the condition
- CAUTION: Check when the condition becomes FALSE
- Can be single statement or multiple (block)

While Statement in C

```
while (condition)
{
…
}
```

While Statement in C

```
while (condition)
{
…
}
```

- Previously used Equation: ax^3+7
- Code to compute x^3

```
times=1; product =1;
while ( times <= 3 ) {
  product = x * product;
       times = times+1;
} // end while
```

Formulating algorithm – counter controlled iteration

- Consider the following problem statement:
  - *A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.*

- $$class\ average = \frac{\sum grade}{number\ of\ students}$$

Counter controlled iteration

- Uses a variable called a counter to specify the number of times a set of statements should execute.
- Counter-controlled iteration is often called definite iteration because the number of iterations is known *before* the loop begins executing.

Counter controlled iteration - pseudocode

```
1   Set total to zero
2   Set grade counter to one
3
4   While grade counter is less than or equal to ten
5       Input the next grade
6       Add the grade into the total
7       Add one to the grade counter
8
9   Set the class average to the total divided by ten
10  Print the class average
```

Counter controlled iteration – C code

```
1  // Fig. 3.6: fig03_06.c
2  // Class average program with counter-controlled iteration.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     unsigned int counter; // number of grade to be entered next
9     int grade; // grade value
10    int total; // sum of grades entered by user
11    int average; // average of grades
12
13    // initialization phase
14    total = 0; // initialize total
15    counter = 1; // initialize loop counter
16
17    // processing phase
18    while ( counter <= 10 ) { // loop 10 times
19       printf( "%s", "Enter grade: " ); // prompt for input
20       scanf( "%d", &grade ); // read grade from user
21       total = total + grade; // add grade to total
22       counter = counter + 1; // increment counter
23    } // end while
24
```

Counter controlled iteration – C code continued

```
25      // termination phase
26      average = total / 10; // integer division
27
28      printf( "Class average is %d\n", average ); // display result
29   } // end function main
```

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

Initialization phase

- A total is a variable used to accumulate the sum of a series of values.
- A counter is a variable used to count—in this case, to count the number of grades entered.
- An uninitialized variable contains a "garbage" value—the value last stored in the memory location reserved for that variable.

Formulating algorithm – Sentinel Controlled Iteration

- Consider the following problem:
  - *Develop a class-average program that will process an arbitrary number of grades each time the program is run.*
- In this example, the program must process an *arbitrary number* of grades.

Sentinel Controlled Iteration

- Use a special value called a sentinel value (also called a signal value, a dummy value, or a flag value) to indicate "end of data entry."
- Sentinel-controlled iteration is often called indefinite iteration because the number of iterations isn't known before the loop begins executing.
- Sentinel value must be chosen so that it cannot be confused with an acceptable input value.

Sentinel controlled iteration - pseudocode

```
1   Initialize total to zero
2   Initialize counter to zero
3
4   Input the first grade (possibly the sentinel)
5   While the user has not as yet entered the sentinel
6       Add this grade into the running total
7       Add one to the grade counter
8       Input the next grade (possibly the sentinel)
9
10  If the counter is not equal to zero
11      Set the average to the total divided by the counter
12      Print the average
13  else
14      Print "No grades were entered"
```

Sentinel controlled iteration – C code

```
1  // Fig. 3.8: fig03_08.c
2  // Class-average program with sentinel-controlled iteration.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     unsigned int counter; // number of grades entered
9     int grade; // grade value
10    int total; // sum of grades
11
12    float average; // number with decimal point for average
13
14    // initialization phase
15    total = 0; // initialize total
16    counter = 0; // initialize loop counter
17
18    // processing phase
19    // get first grade from user
20    printf( "%s", "Enter grade, -1 to end: " ); // prompt for input
21    scanf( "%d", &grade ); // read grade from user
22
```

Sentinel controlled iteration – C code

```
23      // loop while sentinel value not yet read from user
24      while ( grade != -1 ) {
25         total = total + grade; // add grade to total
26         counter = counter + 1; // increment counter
27
28         // get next grade from user
29         printf( "%s", "Enter grade, -1 to end: " ); // prompt for input
30         scanf("%d", &grade); // read next grade
31      } // end while
32
33      // termination phase
34      // if user entered at least one grade
35      if ( counter != 0 ) {
36
37         // calculate average of all grades entered
38         average = ( float ) total / counter; // avoid truncation
39
40         // display average with two digits of precision
41         printf( "Class average is %.2f\n", average );
42      } // end if
43      else { // if no grades were entered, output message
44         puts( "No grades were entered" );
45      } // end else
46   } // end function main
```

This would cause an *infinite loop* if -1 is not input as the first grade.

Sentinel controlled iteration - Output

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

```
Enter grade, -1 to end: -1
No grades were entered
```

Nested Control Statement

- One control statement within another
- Consider the following problem statement:
  - *In a class of 10 students, get the result of the student from the user (1=pass, 2=fail) and find the number of students who failed and who passed. If more than 8 students passed, print a statement for bonus to the instructor.*

Nested Control Statement – Pseudocode

```
1   Initialize passes to zero
2   Initialize failures to zero
3   Initialize student to one
4
5   While student counter is less than or equal to ten
6           Input the next exam result
7
8           If the student passed
9               Add one to passes
10          else
11              Add one to failures
12
13          Add one to student counter
14
15  Print the number of passes
16  Print the number of failures
17  If more than eight students passed
18      Print "Bonus to instructor!"
```

Nested Control Statement – C code

```
1  // Fig. 3.10: fig03_10.c
2  // Analysis of examination results.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     // initialize variables in definitions
9     unsigned int passes = 0; // number of passes
10    unsigned int failures = 0; // number of failures
11    unsigned int student = 1; // student counter
12    int result; // one exam result
13
14    // process 10 students using counter-controlled loop
15    while ( student <= 10 ) {
16
17       // prompt user for input and obtain value from user
18       printf( "%s", "Enter result ( 1=pass,2=fail ): " );
19       scanf( "%d", &result );
20
```

Nested Control Statement – C code

```
21          // if result 1, increment passes
22          if ( result == 1 ) {
23             passes = passes + 1;
24          } // end if
25          else { // otherwise, increment failures
26             failures = failures + 1;
27          } // end else
28
29          student = student + 1; // increment student counter
30       } // end while
31
32       // termination phase; display number of passes and failures
33       printf( "Passed %u\n", passes );
34       printf( "Failed %u\n", failures );
35
36       // if more than eight students passed, print "Bonus to instructor!"
37       if ( passes > 8 ) {
38          puts( "Bonus to instructor!" );
39       } // end if
40    } // end function main
```

Nested Control Statement – Output 1

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Passed 6
Failed 4
```

Nested Control Statement – Output 2

```
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 2
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Enter Result (1=pass,2=fail): 1
Passed 9
Failed 1
Bonus to instructor!
```