ICS 103 Lab05: More on Repetition (Loop) Statements

Objectives:

Practice on:

* Sentinel controlled loops
* do-while loops
* Nested loops: independent and dependent

**Sentinel controlled loops**

In a program, a sentinel is a value that marks the end of a series of data values; but is not a data value itself. Sentinels may be used to control conditional loops:

Example1: Write a C program fragment that prompts for and reads student grades in a quiz. It then calculates and displays the average. Use a negative value or a value > 100 as the sentinel.

int count = 0;

double grade, sumOfGrades = 0.0;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

scanf("%lf", &grade);

while(grade >= 0 && grade <= 100){

count++;

sumOfGrades += grade;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

scanf("%lf", &grade);

}

if(count == 0)

printf("Error: No valid grade entered\n");

else

printf("Average = %.2f\n", sumOfGrades / count);

Exampe2: Modify the above C program fragment such that it also displays the maximum and minimum grade.

int count = 0;

double grade, sumOfGrades = 0.0;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

scanf("%lf", &grade);

double max = grade; // Assume the first grade is the max

double min = grade; // Assume the first grade is the min

while(grade >= 0 && grade <= 100){

count++;

sumOfGrades += grade;

if(grade > max)

max = grade;

else if(grade < min)

min = grade;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

scanf("%lf", &grade);

}

if(count == 0)

printf("Error: No valid grade entered\n");

else{

printf("Average = %.2f\n", sumOfGrades / count);

printf("Maximum = %.2f\n", max);

printf("Minimum = %.2f\n", min);

}

* **For sentinel controlled loops.**

For-loops are usually used when the number of repetitions of the loop is known (i.e., for- loops are usually used as counter-controlled loops). It is also possible to use for-loops as sentinel controlled loops.

Example3: Example1 above can be written as:

int count = 0;

double grade, sumOfGrades = 0.0;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

for(scanf("%lf", &grade);grade >= 0 && grade <= 100; scanf("%lf", &grade)){

count++;

sumOfGrades += grade;

printf("Enter grade#%d (-ve value or value > 100 to terminate)\n", count+1);

}

if(count == 0)

printf("Error: No valid grade entered\n");

else

printf("Average = %.2f\n", sumOfGrades / count);

* **do-while statement**

A do-while statement is used to execute a statement or a compound-statement one or more times as long as the do-while condition is true:

do

do-while body

while(condition);

The do-while body can be a simple statement in which case it must be terminated by a semicolon or it may be a compound-statement in which case it MUST NOT be terminated by a semicolon:

|  |  |
| --- | --- |
| Simple statement do-while body | Compound-statement do-while body |
| do  statement;  while(condition); | do  compound\_statement  while(condition); |
|  | |

Note: Because the condition of a do-while loop is at the end; a do-while loop executes one or more times.

Examples:

|  |  |
| --- | --- |
| do-while loop | output |
| int n = 1;  do{  printf("%d ", n);  n += 2;  } while(n <= 10); | 1 3 5 7 9 |
| int x = 25;  do  printf("%d ", x -= 5);  while(x > 0); | 20 15 10 5 0 |

**Example:** A do-while loop can be used to validate input

int n;

do{

printf("Enter an integer number in the [10,100] interval: \n");

scanf("%d", &n);

if(n <10 || n > 100)

printf("Sorry wrong input, try again\n");

}while (n < 10 || n > 100);

printf("Now your input is correct\n");

**Example:** A do-while loop can be used in a menu driven program. The example shown below will continue running as long as the user did not enter the number 6.

int choice;

do{ printf("1-Addition\n");

printf("2-Subtraction\n");

printf("3-Multiplication\n");

printf("4-Division\n");

printf("5-Exit\n");

printf("Enter your choice: \n");

scanf("%d", &choice);

// Here come selection statement to do the different operations(+,-,\*,/)

} while (choice != 5);

**Note:** See and execute the given **MenuDrivenProgram.c**

* **Nested loops**

A loop statement may contain in its body one or more loop statements. There are two types of nested loops: independent and dependent. A nested loop is independent if its number of repetitions does not depend on an outer loop. A nested loop is dependent if its number of repetitions depends on an outer loop.

Example:

|  |  |
| --- | --- |
| Nested loops | output |
| // Example of independent nested //loop  int m, n;  for(m = 5; m >= 1; m--){  printf("m is now %d\n", m);  for(n = 1; n <= 4; n++)  printf("n = %d ", n);  printf("\n");  } | m is now 5  n = 1 n = 2 n = 3 n= 4  m is now 4  n = 1 n = 2 n = 3 n= 4  m is now 3  n = 1 n = 2 n = 3 n= 4  m is now 2  n = 1 n = 2 n = 3 n= 4  m is now 1  n = 1 n = 2 n = 3 n= 4 |
| //Example of dependent nested loop  int k,m;  for(k=1;k<=9;k++){  for(m = 1; m <= k; m++){  printf("%d",k);  }  printf("\n");  } |  |

**Example**: Write a C program that prompts for and reads three quiz grades for each student in a class of four students. The program then computes and displays the average for each student. Your program must be easily modifiable to handle any number of students and quizzes.

#include <stdio.h>

#include <stdlib.h>

#define NUMSTUDENTS 4

#define NUMQUIZES 3

int main(void){

double grade, studentTotal, studentAverage;

int m, n;

for(m = 1; m <= NUMSTUDENTS; m++){

studentTotal = 0.0;

for(n = 1; n <= NUMQUIZES; n++){

printf("Enter QuizGrade#%d for student#%d\n", n, m);

scanf("%lf", &grade);

studentTotal += grade;

}

studentAverage = studentTotal / NUMQUIZES;

printf("The average for student#%d is %.2f\n", m, studentAverage);

}

return 0;

}

**Homework Exercise**: Modify the above program such that it also computes and displays the class average.

**Laboratory Tasks**

**Task 1** Write a C program that displays the following options to the user:

1-Addition

2-Multiplication

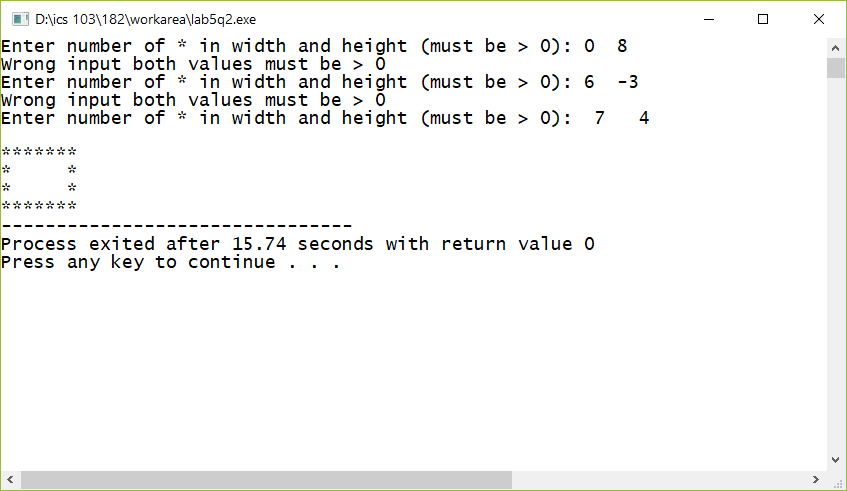
3-Exit

The program keeps doing additions and multiplications until the user chooses 3. Then it will exit the loop and stop.

Sample run. The highlighted values are entered by the user.

|  |
| --- |
| 1-Addition  2-Multiplication  3-Exit  Enter your choice: 4  Wrong choice, Enter 1, 2, or 3  1-Addition  2-Multiplication  3-Exit  Enter your choice: 2  Enter 2 int values to multiply: 5 8  5 \* 8 = 40  1-Addition  2-Multiplication  3-Exit  Enter your choice: 1  Enter 2 int values to add: 24 16  24 + 16 = 40  1-Addition  2-Multiplication  3-Exit  Enter your choice: 6  Wrong choice, Enter 1, 2, or 3  1-Addition  2-Multiplication  3-Exit  Enter your choice: 3  Thanks for using the program |

**Task 2:** Write a C program that reads the width and height of a rectangle (both have to be >0). Then it will display the boundary of this rectangle using the \* symbol as shown in the sample run.



**Task 3:** Write a C **main** function that prompts for and reads a positive integer, it then finds and displays sum of its digits.Your program must recover from input error by displaying an error message and looping as long as the input is zero or negative.

**Note** You need 2 loops; the first for validating the input, and the second for finding the sum of the digits.

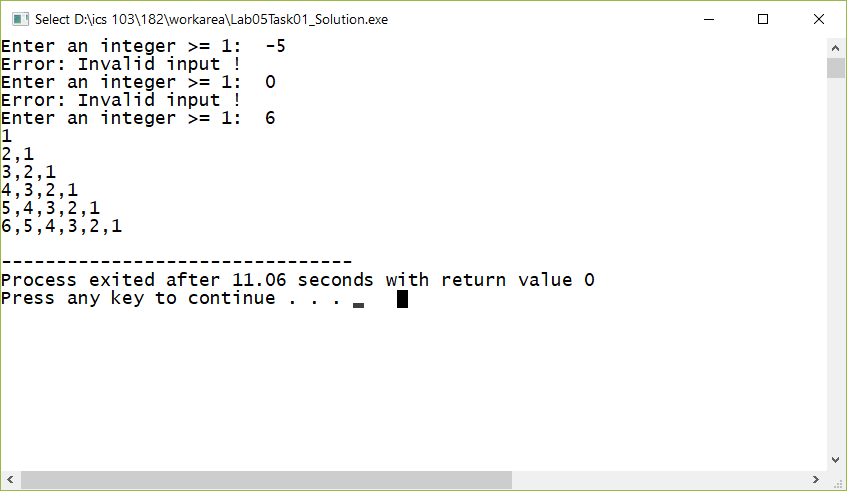
Hint: Use remainder of division by 10 and division by 10.

Sample program runs:

|  |  |
| --- | --- |
|  |  |

**Task 4:**: Write a C program That displays the following sequence shown below based on input n (n >=1) representing the number of rows.

Notice that there is no comma printed after 1 in each row.



**Task 5:** By the Newton-Raphson method**,** the square root of a positive number **n** can be computed by successive approximations x1, x2, x3, x4, . . . :

// or any other suitable initial approximation

. . .

We stop the above iterations when the absolute difference between the square of the ith approximation and **n** is smaller than a given small value **epsilon**:

| xi \* xi - n | < epsilon

Write a complete C program that prompts for and reads the value of positive numbers **n** . Your program must display an appropriate error message and loop if **n** is negative; otherwise it computes the square root of the number using Newton-Raphson method. After that, your program displays both the approximate computed square root, and the square root of **n** returned by invoking the standard square root function.

Note: Use the value **1e-4** for epsilon.

Here also use 2 loops; one for validating the input and another for approximating the square root of n. The condition for the second loop to continue is that | xi \* xi - n | >= epsilon

Sample program runs:

|  |
| --- |
| Enter a positive number: -5.6  Error: -5.600000 is negative  Enter a positive number: -123  Error: -123.000000 is negative  Enter a positive number: 38.87  The square root of 38.870000 by Newton-Raphson is 6.2345809874  The square root of 38.870000 by the standard sqrt function is 6.2345809803 |

**Additional Exercise:**

**Exercise 6:** Write a C program that prompts for and reads an integer **n1** in the range [8, 15], it then prompts for and reads an integer **n2** in the range [3, 6]. It finally generates and displays a multiplication table whose rows start from **n1** to **1**, and whose columns start from **n2** to **1** as shown in the sample program runs below. If the input is not valid, your program must display an appropriate error message and terminate.

|  |
| --- |
|  |
|  |
|  |

**Exercise 7:** Using a **do-while** loop, write a C program to check that the user has typed a character digit. The program must not stop until the user types a character digit.

**Note:**

* Use **" %c"** as the format string for the **scanf** to skip white space.
* The behavior of your program must be similar to the sample program runs given below

Sample runs:

|  |  |
| --- | --- |
|  |  |

**Exercise 8:**Using **nested for loops**, write a **general** C program to print patterns of the form shown in the sample program runs below:

Sample program runs:

|  |  |
| --- | --- |
|  |  |

**Note:** The first line of output contains n A’s where n is the value of the integer read.