2022-08-30 LECTURE 05 – STRUCTURED PROGRAM DEVELOPMENT – II

- contents
  - repetition control structure
    - the while iteration statement
    - countered controlled iteration
      - take a number and print all number up to that number
      - top-down stepwise refinement
        - take two numbers and print all numbers between them
      - take marks of 5 students in a subject and print their average
    - sentinel controlled iteration
      - take marks of students in a subject till user enters 0 or a negative number and displays their class average
  - nested control statements
    - selection in repetition
      - count pass and fail status of students entered by the user [1=pass, 2=fail] until the user enters 0, and display percentage of passed and failed students at console
      - repeat if user does not valid option [0=exit, 1=pass, 2=fail]
  - concepts discussed in the lecture
  - exercises
- repetition control structure
  - recall that in repetition control structure a set of statements are executed over and over again till a particular condition is true (1), when the condition becomes false (0) then the program control skips the repetition block and move on next instruction in the program
  - there are two types of repetition control structures
    - counter controlled repetition
      - number of repetitions of a set of statements which should be performed are known
      - repetition is dependent on a value of a variable usually called **counter** which keeps counting how many repetitions have been performed
    - sentinel controlled repetition
      - number of repetitions of a set of statements which should be performed are unknown
      - repetition is dependent on a value of a variable usually called **sentinel value** which checks on every repetitions whether to continue of terminate repetitions
      - usually the values are continuously entered by the user and when user enters a sentinel value then repetition stops
  - counter controlled repetition
    - syntax of a counter controlled repetition is as follows
      define a **counter** variable
      initialize the **counter** variable
      while (write a **condition** on **counter** variable) {

> ***set-of-statements*** *which should be executed if the condition in parentheses is true*
>
> **increment in counter variable**
>
> }

- **example (using while counter controlled repetition):** a program that prompts user to enter a number and displays numbers starting form 0 till that number at console.

// L05-C01

```
1.  int main (void) {
2.     int a;
3.     printf("enter a number\t");
4.     scanf("%d", &a);
5.     int counter = 0;
6.     while ( counter <= a) {
7.        printf("%d\t", counter);
8.        counter = counter + 1;
9.     }
10. }
```

- **input:** enter a number ***5***
- **output:** 0 1 2 3 4 5

- **Line 5:** define and initialize a counter variable named **counter** with value 0.
- **Line 6**: defines a condition on the **counter** variable which would be true (1) if the value in **counter** variable would be less than or equal to the value in variable a (value entered by the user in this program)
- **Line 7:** displays the value of counter at console as required
- **Line 8: is** an increment statement which increases the value of counter variable by 1 on each repetition

- program refinement
  - **example (using while counter controlled repetition):** a program that prompts user to enter two numbers and displays all numbers between them at console.
  - **top-down stepwise refinement** technique is used to develop this program as follows
  - **a top task for this program is**
    - top task is a single statement that conveys overall program functionality
      - ***print all numbers between two user inputted numbers***
    - **top** is usually a complete requirement of the program but it is rarely, directly implementable in a programming language, hence we need further refinements in the **top** to convert it into an algorithm
    - refinement means to divide **top** into smaller tasks written in an order in which these tasks should be executed
    - **first refinement** could be as follows

-
    - define variables
    - ask user to enter number
    - input two numbers
    - print all members between two numbers
- here only sequential control structure is used above steps should be executed in the above mentioned order
- note that each refinement is an algorithm itself, only level of details is varied
- **second refinement** could be as follows
    - define variables could be refined as
        - define first-number, second-number and a counter
    - ask user to enter two number could be refined as
        - print “enter two number” at console
    - input two numbers could be refined as
        - take input of first-number
        - take input of second-number
        - set value of the counter variable equal to one more than first-number
    - print all number between two numbers could be refined as
        - while the value of counter is less than second-number
        - print value of the counter
        - increment one number into the value of counter variable
    - a complete second refinement would be
        - define first-number, second-number and a counter
        - print “enter two number” at console
        - take input of first-number
        - take input of second-number
        - set value of the counter variable equal to one more than first-number
        - while the value of counter is less than the second-number
            - print value of the counter
            - increment one number into the value of counter variable
- in **third refinement** (if required), datatypes of the variables could be defined
- a C language implementation of the above algorithm/pseudo-code could be as follows

// L05-C02

```
int main (void) {
   int a, b;
   printf(“enter two numbers\t”);
   scanf(“%d”, &a);
   scanf(“%d”, &b);
6.      int counter = a + 1;
7.      while ( counter < b) {
8.          printf("%d\t", counter);
9.          counter = counter + 1;
10.     }
11. }
```

- **input:** enter two numbers ***5*** ***9***
- **output:** 6 7 8

- **Line 6:** define and initialize a counter variable named **counter** with value one greater than the value in variable **a** (first value entered by the user)
- **Line 7**: defines a condition on the **counter** variable which would be true (1) if the value in **counter** variable would be less than to the value in variable **a** (second value entered by the user)
- **Line 8:** displays the value of counter at console as required
- **Line 9: is** an increment statement which increases the value of counter variable by 1 on each repetition

- **example (using while counter controlled repetition):** a program that prompts user to enter marks of 5 students in a subject, takes marks of 5 students and print their average at console.

// L05-C03

```
1.  int main (void) {
2.      int a, sum=0;
3.      float average;
4.      int counter = 0;
5.      while ( counter < 5) {
6.          printf("enter marks of a student:\t");
7.          scanf("%d", &a);
8.          sum = sum + a;
9.          counter = counter + 1;
10.     }
11.     average = (float) sum / counter;
12.     printf("average marks of the class are: \t%.2f", average);
13. }
```

- **output/input:** enter marks of a student: ***65***
- **output/input:** enter marks of a student: ***92***
- **output/input:** enter marks of a student: ***73***
- **output/input:** enter marks of a student: ***67***
- **output/input:** enter marks of a student: ***55***
- **output:** average marks of the class are: 70.04

- **Line 3:** a float type variable named **average** is defined to average of the class
- **Line 4:** define and initialize a counter variable named **counter** with value 0

    - **Line 5**: defines a condition on the **counter** variable which would be true (1) if the value in **counter** variable would be less than to the value 5
    - **Line 7:** displays the value of counter at console as required
    - **Line 8:** sums the values entered by the user one by one in each repetition
    - **Line 9:** is an increment statement which increases the value of counter variable by 1 on each repetition
    - **Line 11:** to avoid integer division and convert this division of two integer numbers into a float **explicit type casting** of expression **sum / counter** is used with an expression (float) in front it
    - **Line 12:** average marks of the class are displayed using **format specifier %.2f** which will display the value of floating point variable average till to decimal places
- sentinel controlled iteration
    - syntax of a sentinel controlled repetition is as follows

      define a **sentinel** variable

      initialize the **sentinel** variable

      while (write a **condition** on **sentinel** variable) {

      > ***set-of-statements*** *which should be executed if the condition in parentheses is true*
      >
      > **take input of a sentinel variable or set sentinel variable to specific value using some condition**

      }
    - **example (using while sentinel controlled repetition):** a program that prompts user to enter marks of students in a subject till user enters a **0** or a **negative number,** then display average of students marks at console.

// L05-C04

```
1.  int main (void) {
2.     int a, sum=0;
3.     float average;
4.     int counter = 0;
5.     int sentinel = 1;
6.     while ( sentinel != 0 ) {
7.        printf("enter marks of a student:\t");
8.        scanf("%d", &a);
9.        if ( a > 0 ) {
10.       sum = sum + a;
11.       counter = counter + 1;
12.       }
13.       else {
14.       sentinel = 0;
15.       }
16.    }
17.    average = (float) sum / counter;
18.    printf("average marks of the class are: \t%.2f", average);
19. }
```

- **output/input:** enter marks of a student: ***65***
- **output/input:** enter marks of a student: ***92***
- **output/input:** enter marks of a student: ***73***
- **output/input:** enter marks of a student: ***67***
- **output/input:** enter marks of a student: ***0***
- **output:** average marks of the class are: 74.25

- **Line 3:** a float type variable named **average** is defined to average of the class
- **Line 4**: define and initialize a counter variable named **counter** with value 0
- **Line 5**: define and initialize a sentinel variable named **sentinel** with value 1
- **Line 6**: defines a condition on the **sentinel** variable which would be true (1) if the value in **sentinel** variable is not equal to 0
- **Line 7&8:** prompts user to enter student makers, takes input and store it in variable **a**
- **Line 9-15:** (line 9) checks whether the value entered by the user a is greater than 0, if it is true then (line 10&11) add this value in sum and increase the value of counter variable, otherwise in else block (line 12-15) set the value of sentinel variable equal to zero (0) which will terminates this repetition structure before next repetition
- **Line 17:** to avoid integer division and convert this divisi on of two integer numbers into a float **explicit type casting** of expression **sum / counter** is used with an expression (float) in front it
- **Line 18:** average marks of the class are displayed using format specifier **%.2f** which will display the value of floating point variable average till to decimal places

- nested control statements
  - recall that a nested control structure means to embed one control structure into another control structure
    - for example see program **L05-04** as discussed above where if-else control structure is embedded into a while control structure
    - **example (using while nested control statements):** a program that prompts user to enter students status (1:pass and 2:fail) in a subject till user enters a **0** or a **negative number,** then display the percentage passed and failed students in the subject at console.

// L05-C05

```
1.  int main (void) {
2.     int a;
3.     float average;
4.     int counter = 0, pass=0, fail=0;
5.     int sentinel = 1;
6.
7.     while ( sentinel != 0 ) {
8.        printf("enter the result of a student (1=pass, 2=fail, 0=exit):\t");
9.        scanf("%d", &a);
10.         if ( a == 1 ) {
11.         pass = pass + 1;
12.         counter = counter + 1;
13.         }
14.         else if ( a == 2 ) {
15.         fail = fail + 1;
16.         counter = counter + 1;
17.         }
18.         else {
19.         sentinel = 0;
20.         }
21.     }
22.
23.     if (counter > 0) {
24.         average = (float) pass / counter;
25.         printf("percentage of passed students is: \t%.2f", average);
26.         average = (float) fail / counter;
27.         printf("percentage of failed students is: \t%.2f", average);
28.     }
29.     else {
30.         printf("no data is entered");
31.     }
32. }
```

- **output/input:** enter the result of a student (1=pass, 2=fail, 0=exit): ***1***
- **output/input:** enter the result of a student (1=pass, 2=fail, 0=exit): ***1***
- **output/input:** enter the result of a student (1=pass, 2=fail, 0=exit): ***2***
- **output/input:** enter the result of a student (1=pass, 2=fail, 0=exit): ***1***
- **output/input:** enter the result of a student (1=pass, 2=fail, 0=exit): ***0***
- **output:** percentage of passed students is: 75.00
- **output:** percentage of failed students is: 25.00

- **Line 3:** a float type variable named **average** is defined to average of the class
- **Line 4**: define and initialize a counter variable named **counter** with value 0, **pass** and **fail** variable to store number of passed and failed students with value 0
- **Line 5**: define and initialize a sentinel variable named **sentinel** with value 1
- **Line 7**: defines a condition on the **sentinel** variable which would be true (1) if the value in **sentinel** variable is not equal to 0
- **Line 8&9:** prompts user to enter result of a student, takes input and store it in variable **a**
- **Line 10-20:** (line 10) checks whether the value entered by the user in variable **a** is equal to 1, if it is true then (line 11&12) add one in the variable pass and increase the value of counter variable, otherwise in else if block (line 14) checks whether the value entered by the user in variable **a** is equal to 2, if it is true then (line 15&16) add on in the variable fail and increase the value of counter variable,

otherwise else block (line 18-20) sets the value of sentinel variable equal to zero (0) which will terminates this while structure before next repetition

- **Line 23:** after coming out of the while, program checks if the value of counter is greater than 0, if it is true than the program calculate average of passed and failed students and display these averages at the console
- **Line 24&26:** to avoid integer division and convert this division of two integer numbers into a float **explicit type casting** of expression **pass / counter** is used with an expression (float) in front it
- **Line 25&27:** average of pass and failed students of the class are displayed using format specifier **%.2f** which will display the value of floating point variable average till to decimal places
- **Line 29:** if the user has not entered any valid **pass** or **fail** status of the student then the counter variable will have value 0, hence the else block of the if statement would be executed it will display "no data is entered" at console

o concepts discussed in the lecture
  - counter, sentinel value, top-down stepwise refinement, first refinement, second refinement, third refinement, explicit type casting, format specifier %.2f