

Lecture Plan(Objectives):

Input Validation Code:

We will revisit the input validation code and make some modifications to it so that we can validate the type of input too.

Understanding Headers:

A header file is a file with extension **.h** which contains C function declarations and macro definitions to be shared between several source files. There are two types of header files: the files that the programmer writes and the files that comes with your compiler.

You can use a header file in your program by including it with the keyword **#include**, like as we include **<stdio.h>** header file, which comes along with compiler. Each standard library has a corresponding header file containing the function prototypes for all the functions in that library and definitions of various data types and constants needed by those functions.

Learning Random Number Generation:

Consider the following statement: `int value = rand ();`

The `rand` function generates an integer between 0 and `RAND_MAX` (a symbolic constant defined in the **<stdlib.h>** header). The C standard states that `RAND_MAX`'s value must be at least 32,767, which is the maximum value for a two-byte (i.e., 16-bit) integer.

Understanding Scope Rules:

The scope of an identifier is the part of the program in which the identifier can be referenced. For example, a local variable in a block can be referenced only following its definition in that block or in blocks nested within that block. The four identifier scopes are function scope, file scope, block scope and function-prototype scope.

Storage Classes in C:

C provides the storage-class specifiers **auto**, **register**, **extern** and **static**. A storage class determines an identifier's storage duration, scope and linkage. Storage Classes are used to describe the features of a variable/function. These features basically include the scope, visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program.

Function-Call Stack and Stack Frames:

Function call stack is an important mechanism about function execution for programmers to understand. Stack (sometimes referred to as the program execution **stack**). This data structure working “behind the scenes” supports the function call/return mechanism. The function call stack also supports creating, maintaining and destroying each called function’s local variables.

Headers:

Each standard library has a corresponding header containing the function prototypes for all the functions in that library and definitions of various data types and constants needed by those functions. The following table alphabetically lists several standard library headers that may be included in programs. The C standard includes additional headers.

Headers in C:

<ctype.h>	Contains function prototypes for functions that test characters for certain properties, and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa.
<float.h>	Contains the floating-point size limits of the system.
<stdbool.h>	Allows you to use bool as a Boolean data type.
<math.h>	Contains function prototypes for math library functions.
<stdio.h>	Contains function prototypes for the standard input/output library functions and information used by them.
<stdlib.h>	Contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers and other utility functions
<string.h>	Contains function prototypes for string-processing functions.

<time.h>	Contains function prototypes and types for manipulating the time and date.
-----------------------	--

Custom Header:

You can create custom headers. A programmer-defined header can be included by using the `#include` preprocessor directive. For example, if the prototype for our square function was located in the header `square.h`, we'd include that header in our program by using the following directive at the top of the program:

```
#include "square.h"
```

While including custom headers, defined headers are enclosed in quotes (") rather than angle brackets (<>).

Learning Random Number Generation:

In order to generate the expected output, the program must need the proper input. Usually, the inputs are provided by the user but sometimes the program has to pick the input by itself.

In the same way, sometimes we need to have the application generate any random number which could be processed further to get the supposed output. Though it looks random to the user the programming language offers us the mechanism to define the range of the random number. Let's focus on the inbuilt function that is provided by C in order to generate a random number.

Generating Random Integers:

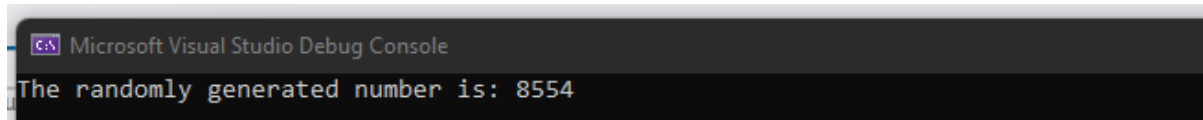
The `rand ()` function is used to generate a random number. Every time it is called, it gives a random number. If the developers add some logic with it, they can generate the random number within a defined range and if the range is not defined explicitly, it will return a totally random integer value.

Code:

The `rand ()` function in C could be used in order to generate the random number and the generated number is totally deleting seed. A seed is a value that is used by `rand` function to generate the random value. If the seed value is kept on changing, the number generated will new every time the program is

compiled else it will return the same value every time that was generated when the program was executed first.

Output:



```
Microsoft Visual Studio Debug Console
The randomly generated number is: 8554
```

In this program, we have used time.h header file which is used to get the system time for generating the random number. As the time changes every time, the value of seed will change every time the program will be executed, gives different random number every time the program is executed.

Rolling a Six-Sided Die:

To demonstrate rand, let's develop a program to simulate 10 rolls of a six-sided die and print each roll's value.

Code:

The rand function's prototype is in <stdlib.h>. In line 6, we use the remainder operator (%) in conjunction with rand as follows `rand () % 6` to produce integers in the range 0 to 5. This is called scaling. The number 6 is called the scaling factor. We then shift the range of numbers produced by adding 1 to our previous result. The output confirms that the results are in the range 1 to 6, the order in which these random values are chosen might vary by compiler.

Output:



```
Microsoft Visual Studio Debug Console
6 6 5 5 6 5 1 1 5 3
```

IN LAB TASKS

Task 1:

Write statements that assign random integers to the variable n in the following ranges:

- a) $1 \leq n \leq 2$
- b) $1 \leq n \leq 100$
- c) $0 \leq n \leq 9$
- d) $1000 \leq n \leq 1112$
- e) $-1 \leq n \leq 1$
- f) $-3 \leq n \leq 11$

C statements:

Range	C Statement
a	
b	
c	
d	
e	
f	

Task 02:

Guess the number Game Write a C program which:

- Plays the game of “guess the number” as follows: Your program chooses the number to be guessed by selecting an integer. at random in the range 1 to 1000.
- It will ask for the guess a number between 1 and 100. If number has been guessed then output number of passes.
- If the guess is incorrect, your program should loop until the player guesses the number. Your program should keep telling the player “Too high” or “Too low” to help the player “zero in” on the correct answer.

```
Microsoft Visual Studio Debug Console
Guess My Number Game
Enter a guess between 1 and 100 : 45
Too low!
Enter a guess between 1 and 100 : 70
Too low!
Enter a guess between 1 and 100 : 90
Too high!
Enter a guess between 1 and 100 : 80
Too high!
Enter a guess between 1 and 100 : 75
Too low!
Enter a guess between 1 and 100 : 78
Too high!
Enter a guess between 1 and 100 : 77
Correct! You got it in 7 guesses!
```

Task 03: Remove Fraction [Estimated 40 minutes / 30 marks]

Create a C program that: takes a floating value as input and implement following functions on it.

- int truncate (float num)
- int round (float num)
- int ceiling (float num)
- int floor (float num)

Using these functions display the value after truncation, rounding, ceiling and flooring.

Write a main function to input a float value then pass that value to these function and print results.