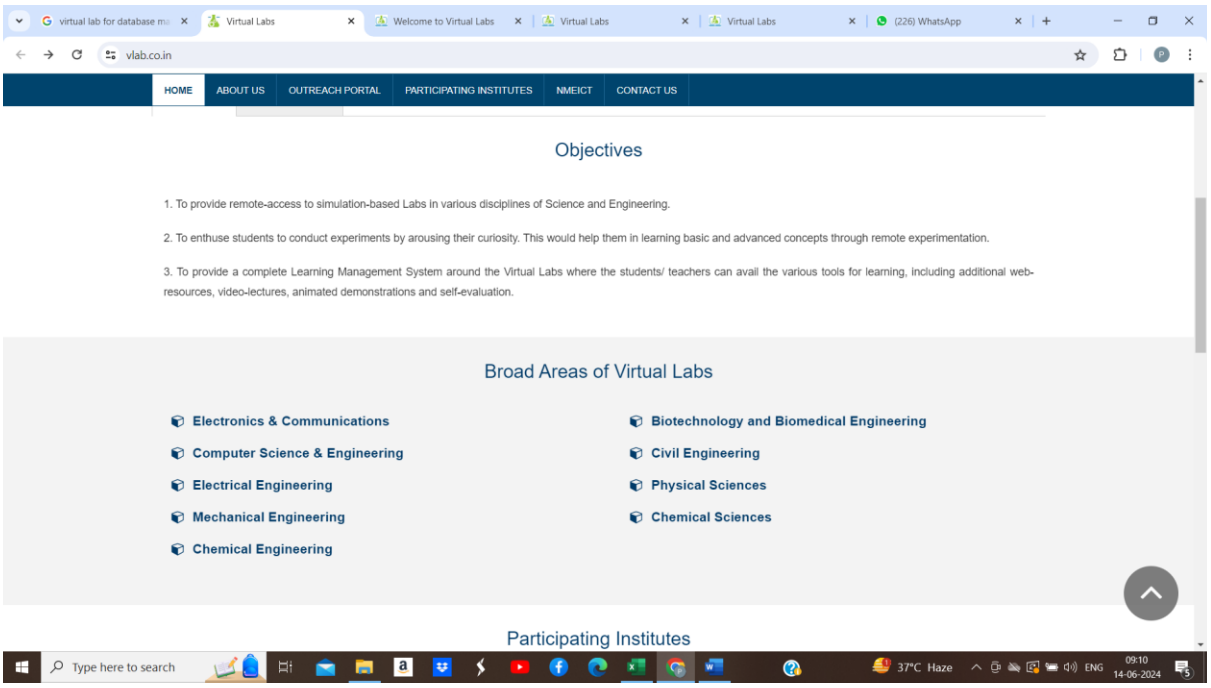
Teaching Aids

Faculty use charts, models and posters etc, to explain some topics in their subject which motivates the students to take interest in the subjects. Some of the charts made were 'Control Structures in C/C++', 'Class Hierarchy in Java' etc. Posters are used to illustrate difficult concepts and recent advances in the respective subjects and used in the classroom teaching.

Working Models/ Charts/ Monograms etc:

S. No	**Chart Description**	**Lab Name**	**Room No.**
1	Logic Gates	System Design LAB	D-208
2	8085 Instruction Set	System Design LAB	D-208
3	8086 Instruction Set	System Design LAB	D-208
4	Full-Subtractor	System Design LAB	D-208
5	D-Flip Flop	System Design LAB	D-208
6	2 x 4 Decoder	System Design LAB	D-208
7	Full-Adder	System Design LAB	D-208
8	4-Bit parallel Adder	System Design LAB	D-208
9	4-to-1 Multiplexer	System Design LAB	D-208
10	4-Bit-Register with parallel load	System Design LAB	D-208
11	Introduction to "C" programming		First Year Lab
12	Model Chart on Instruction Prefetching for Server Workloads		Server Lab



Objectives

1. To provide remote-access to simulation-based Labs in various disciplines of Science and Engineering.

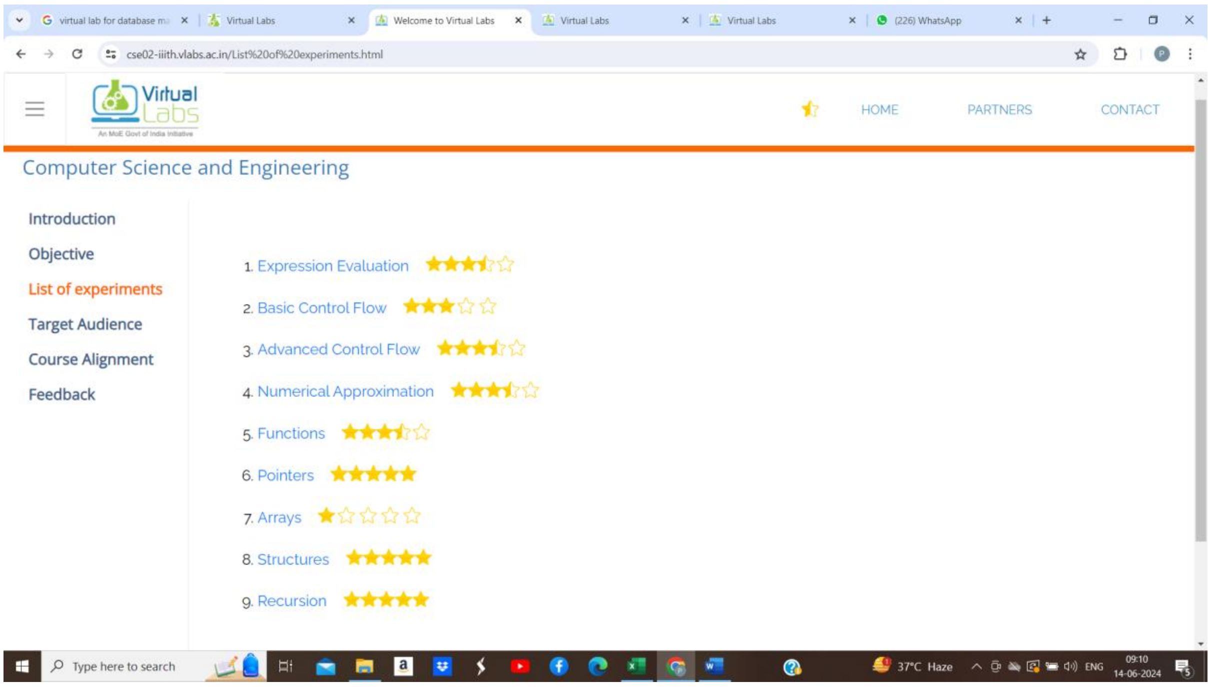
2. To enthuse students to conduct experiments by arousing their curiosity. This would help them in learning basic and advanced concepts through remote experimentation.

3. To provide a complete Learning Management System around the Virtual Labs where the students/ teachers can avail the various tools for learning, including additional web-resources, video-lectures, animated demonstrations and self-evaluation.

Broad Areas of Virtual Labs

- Electronics & Communications
- Computer Science & Engineering
- Electrical Engineering
- Mechanical Engineering
- Chemical Engineering
- Biotechnology and Biomedical Engineering
- Civil Engineering
- Physical Sciences
- Chemical Sciences

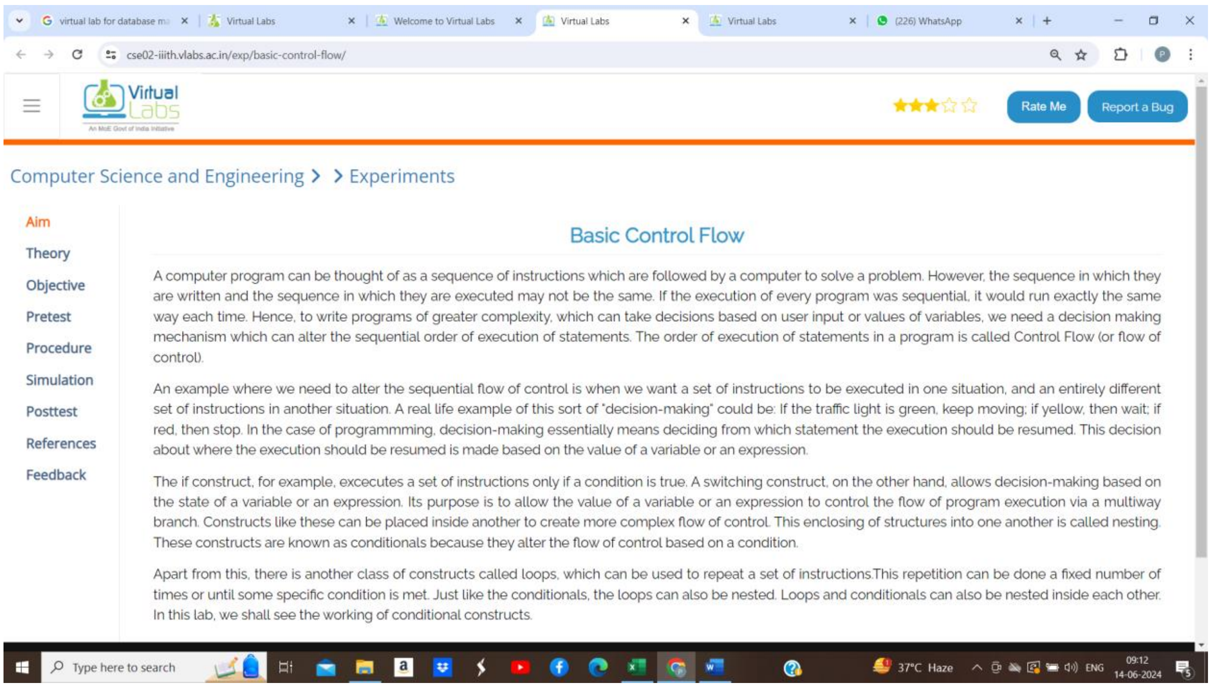
Participating Institutes


virtual lab for database ma
Virtual Labs
Welcome to Virtual Labs
Virtual Labs
Virtual Labs
(226) WhatsApp
cse02-iiith.vlabs.ac.in/List%20of%20experiments.html
Virtual Labs
An MoE Govt of India Initiative
HOME
PARTNERS
CONTACT

Computer Science and Engineering

- Introduction
- Objective
- List of experiments
- Target Audience
- Course Alignment
- Feedback

1. Expression Evaluation ★★★⯪☆
2. Basic Control Flow ★★★☆☆
3. Advanced Control Flow ★★★⯪☆
4. Numerical Approximation ★★★⯪☆
5. Functions ★★★⯪☆
6. Pointers ★★★★★
7. Arrays ★☆☆☆☆
8. Structures ★★★★★
9. Recursion ★★★★★



Computer Science and Engineering > > Experiments

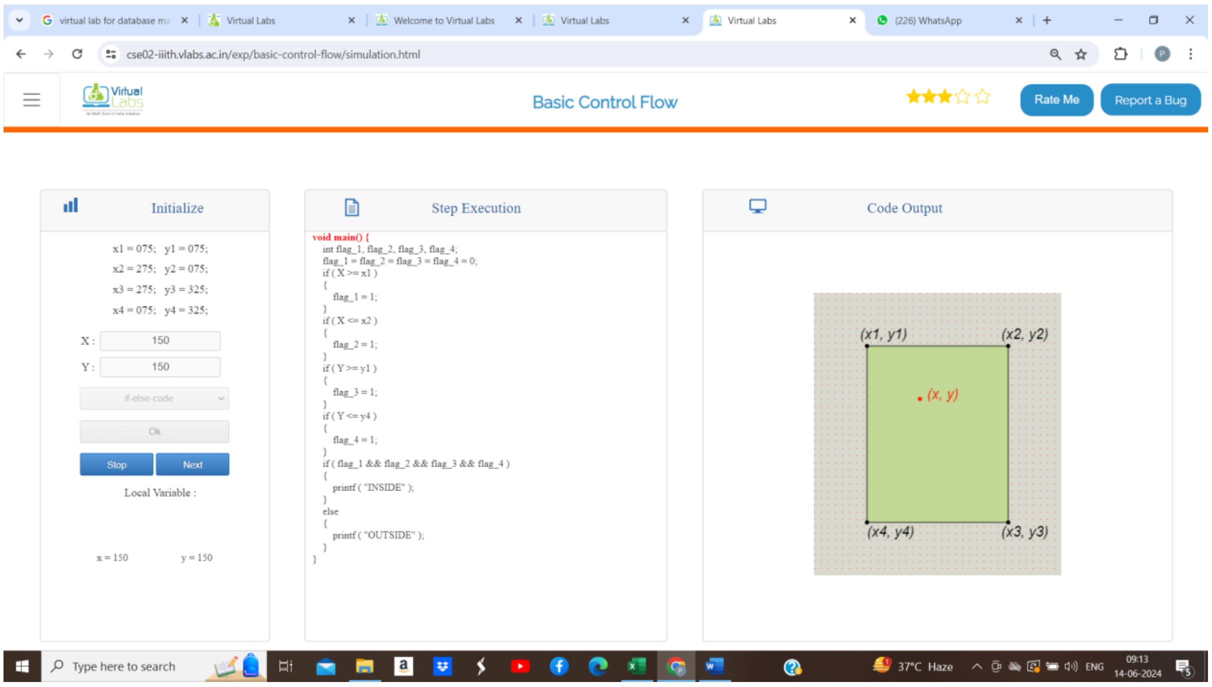
Basic Control Flow

A computer program can be thought of as a sequence of instructions which are followed by a computer to solve a problem. However, the sequence in which they are written and the sequence in which they are executed may not be the same. If the execution of every program was sequential, it would run exactly the same way each time. Hence, to write programs of greater complexity, which can take decisions based on user input or values of variables, we need a decision making mechanism which can alter the sequential order of execution of statements. The order of execution of statements in a program is called Control Flow (or flow of control).

An example where we need to alter the sequential flow of control is when we want a set of instructions to be executed in one situation, and an entirely different set of instructions in another situation. A real life example of this sort of "decision-making" could be: If the traffic light is green, keep moving; if yellow, then wait; if red, then stop. In the case of programmming, decision-making essentially means deciding from which statement the execution should be resumed. This decision about where the execution should be resumed is made based on the value of a variable or an expression.

The if construct, for example, excecutes a set of instructions only if a condition is true. A switching construct, on the other hand, allows decision-making based on the state of a variable or an expression. Its purpose is to allow the value of a variable or an expression to control the flow of program execution via a multiway branch. Constructs like these can be placed inside another to create more complex flow of control. This enclosing of structures into one another is called nesting. These constructs are known as conditionals because they alter the flow of control based on a condition.

Apart from this, there is another class of constructs called loops, which can be used to repeat a set of instructions.This repetition can be done a fixed number of times or until some specific condition is met. Just like the conditionals, the loops can also be nested. Loops and conditionals can also be nested inside each other. In this lab, we shall see the working of conditional constructs.

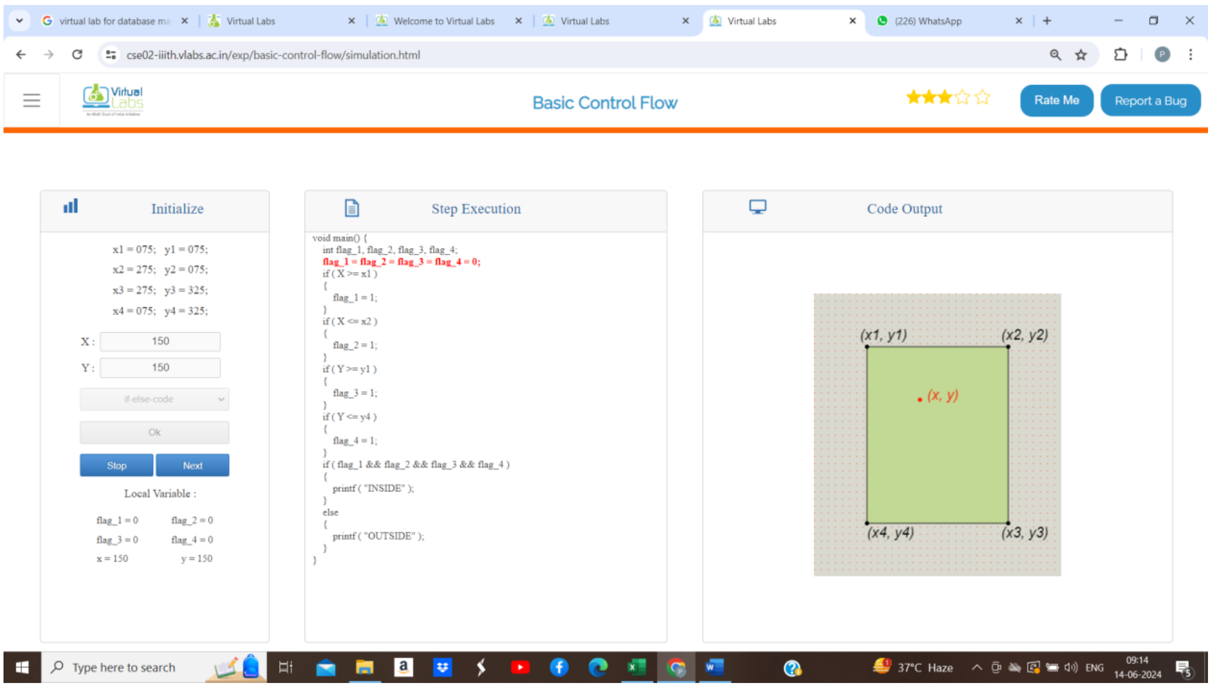


```
x1 = 075;   y1 = 075;
x2 = 275;   y2 = 075;
x3 = 275;   y3 = 325;
x4 = 075;   y4 = 325;

X : 150
Y : 150
if-else-code
Ok
Stop   Next
Local Variable :

x = 150        y = 150
```

```
void main() {
  int flag_1, flag_2, flag_3, flag_4;
  flag_1 = flag_2 = flag_3 = flag_4 = 0;
  if ( X >= x1 )
  {
    flag_1 = 1;
  }
  if ( X <= x2 )
  {
    flag_2 = 1;
  }
  if ( Y >= y1 )
  {
    flag_3 = 1;
  }
  if ( Y <= y4 )
  {
    flag_4 = 1;
  }
  if ( flag_1 && flag_2 && flag_3 && flag_4 )
  {
    printf ( "INSIDE" );
  }
  else
  {
    printf ( "OUTSIDE" );
  }
}
```



cse02-iiith.vlabs.ac.in/exp/basic-control-flow/simulation.html

Basic Control Flow

Initialize

x1 = 075; y1 = 075;
x2 = 275; y2 = 075;
x3 = 275; y3 = 325;
x4 = 075; y4 = 325;

X : 150
Y : 150

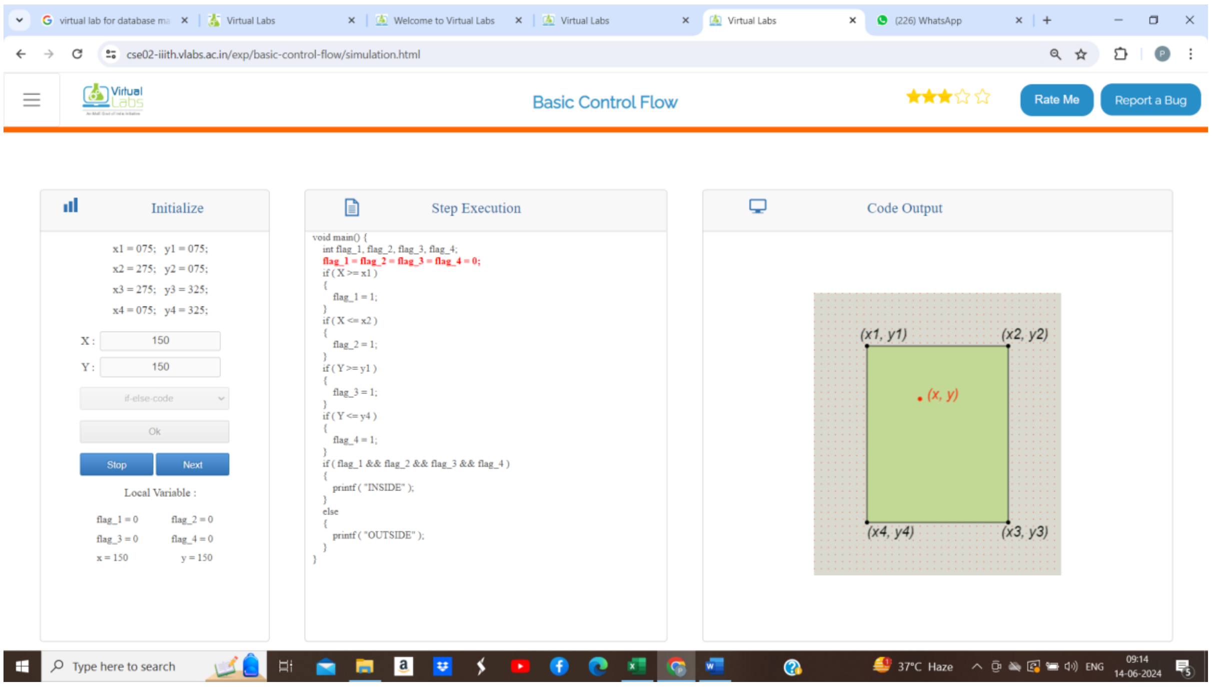
Local Variable :

flag_1 = 0   flag_2 = 0
flag_3 = 0   flag_4 = 0
x = 150   y = 150

Step Execution

```
void main() {
  int flag_1, flag_2, flag_3, flag_4;
  flag_1 = flag_2 = flag_3 = flag_4 = 0;
  if ( X >= x1 )
  {
    flag_1 = 1;
  }
  if ( X <= x2 )
  {
    flag_2 = 1;
  }
  if ( Y >= y1 )
  {
    flag_3 = 1;
  }
  if ( Y <= y4 )
  {
    flag_4 = 1;
  }
  if ( flag_1 && flag_2 && flag_3 && flag_4 )
  {
    printf ( "INSIDE" );
  }
  else
  {
    printf ( "OUTSIDE" );
  }
}
```

Code Output



cse02-iiith.vlabs.ac.in/exp/basic-control-flow/simulation.html

Basic Control Flow

Initialize

x1 = 075; y1 = 075;
x2 = 275; y2 = 075;
x3 = 275; y3 = 325;
x4 = 075; y4 = 325;

X : 150
Y : 150

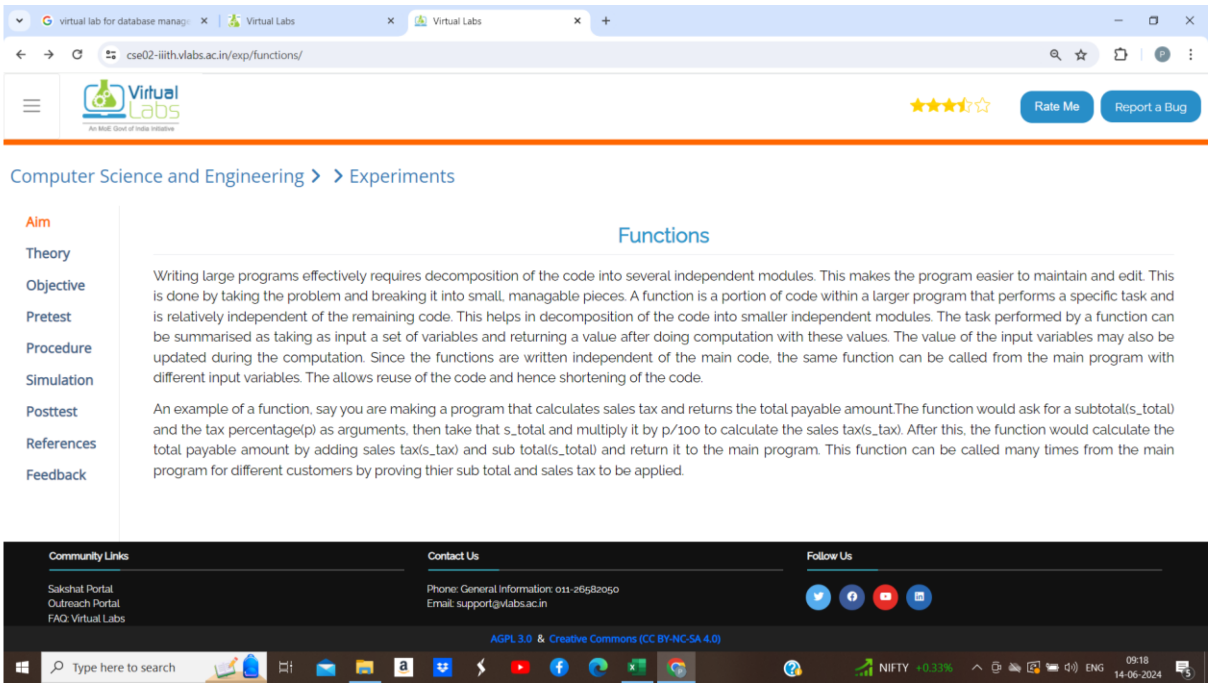
Local Variable :

flag_1 = 0   flag_2 = 0
flag_3 = 0   flag_4 = 0
x = 150   y = 150

Step Execution

void main() {
  int flag_1, flag_2, flag_3, flag_4;
  flag_1 = flag_2 = flag_3 = flag_4 = 0;
  if (X >= x1)
  {
    flag_1 = 1;
  }
  if (X <= x2)
  {
    flag_2 = 1;
  }
  if (Y >= y1)
  {
    flag_3 = 1;
  }
  if (Y <= y4)
  {
    flag_4 = 1;
  }
  if (flag_1 && flag_2 && flag_3 && flag_4)
  {
    printf ("INSIDE");
  }
  else
  {
    printf ("OUTSIDE");
  }
}

Code Output


cse02-iiith.vlabs.ac.in/exp/functions/

Computer Science and Engineering > > Experiments

Aim
Theory
Objective
Pretest
Procedure
Simulation
Posttest
References
Feedback

Functions

Writing large programs effectively requires decomposition of the code into several independent modules. This makes the program easier to maintain and edit. This is done by taking the problem and breaking it into small, managable pieces. A function is a portion of code within a larger program that performs a specific task and is relatively independent of the remaining code. This helps in decomposition of the code into smaller independent modules. The task performed by a function can be summarised as taking as input a set of variables and returning a value after doing computation with these values. The value of the input variables may also be updated during the computation. Since the functions are written independent of the main code, the same function can be called from the main program with different input variables. The allows reuse of the code and hence shortening of the code.

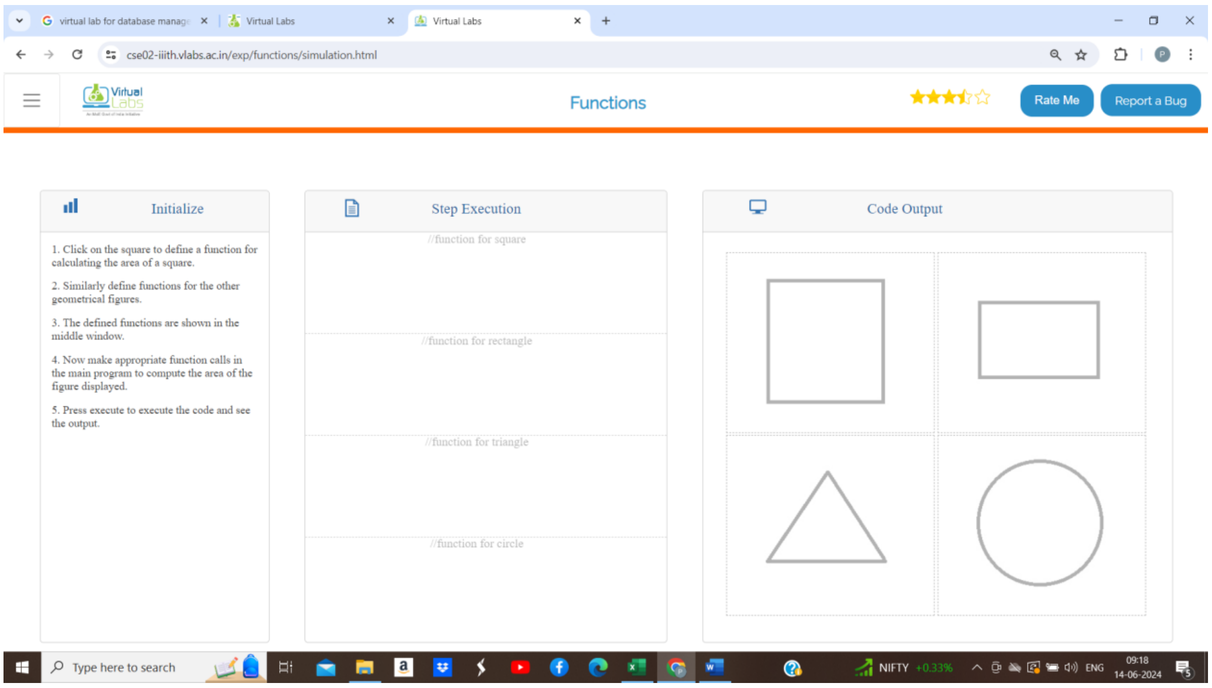
An example of a function, say you are making a program that calculates sales tax and returns the total payable amount.The function would ask for a subtotal(s_total) and the tax percentage(p) as arguments, then take that s_total and multiply it by p/100 to calculate the sales tax(s_tax). After this, the function would calculate the total payable amount by adding sales tax(s_tax) and sub total(s_total) and return it to the main program. This function can be called many times from the main program for different customers by proving thier sub total and sales tax to be applied.

Community Links
Sakshat Portal
Outreach Portal
FAQ: Virtual Labs

Contact Us
Phone: General Information: 011-26582050
Email: support@vlabs.ac.in

Follow Us

AGPL 3.0 & Creative Commons (CC BY-NC-SA 4.0)


cse02-iiith.vlabs.ac.in/exp/functions/simulation.html

Functions

Initialize

1. Click on the square to define a function for calculating the area of a square.

2. Similarly define functions for the other geometrical figures.

3. The defined functions are shown in the middle window.

4. Now make appropriate function calls in the main program to compute the area of the figure displayed.

5. Press execute to execute the code and see the output.

Step Execution

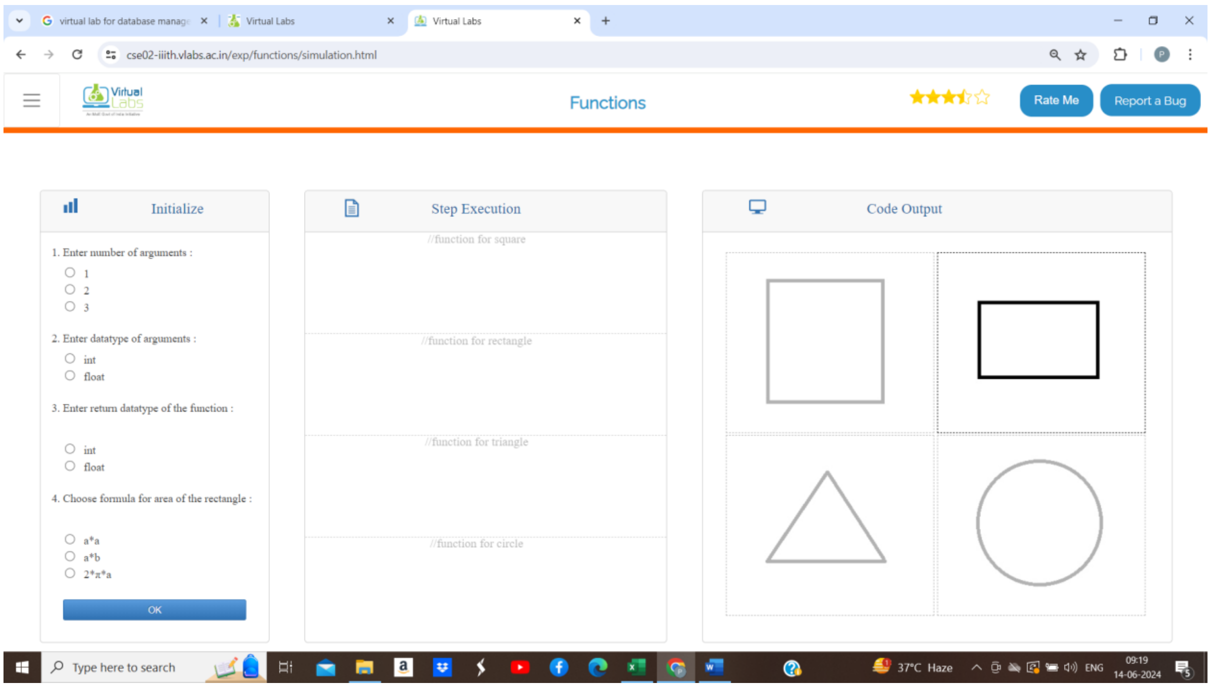
```
//function for square

//function for rectangle

//function for triangle

//function for circle
```

Code Output


cse02-iiith.vlabs.ac.in/exp/functions/simulation.html

Functions

Initialize

1. Enter number of arguments :
   - 1
   - 2
   - 3
2. Enter datatype of arguments :
   - int
   - float
3. Enter return datatype of the function :
   - int
   - float
4. Choose formula for area of the rectangle :
   - a*a
   - a*b
   - 2*π*a

Step Execution

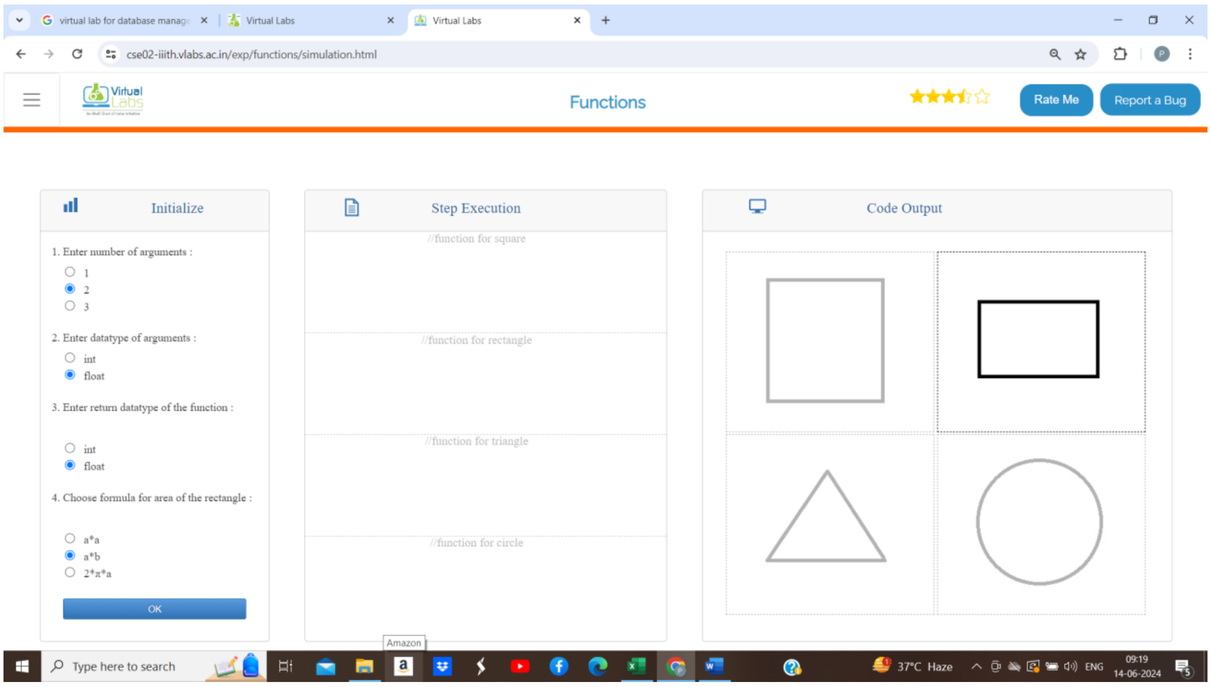
```
//function for square

//function for rectangle

//function for triangle

//function for circle
```

Code Output


cse02-iiith.vlabs.ac.in/exp/functions/simulation.html

Functions

Initialize

1. Enter number of arguments :
   - 1
   - [✓ 2]
   - 3
2. Enter datatype of arguments :
   - int
   - [✓ float]
3. Enter return datatype of the function :
   - int
   - [✓ float]
4. Choose formula for area of the rectangle :
   - a*a
   - [✓ a*b]
   - 2*π*a

Step Execution

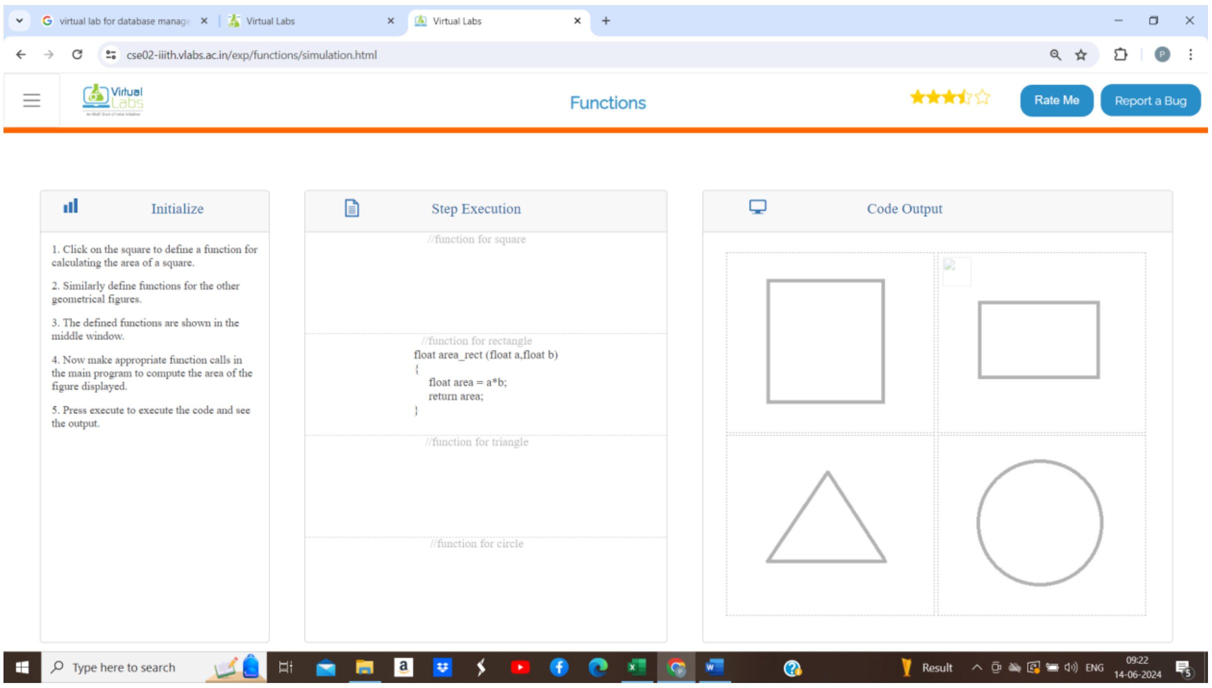
```
//function for square

//function for rectangle

//function for triangle

//function for circle
```

Code Output



1. Click on the square to define a function for calculating the area of a square.

2. Similarly define functions for the other geometrical figures.

3. The defined functions are shown in the middle window.

4. Now make appropriate function calls in the main program to compute the area of the figure displayed.

5. Press execute to execute the code and see the output.

```
//function for square

//function for rectangle
float area_rect (float a,float b)
{
    float area = a*b;
    return area;
}

//function for triangle

//function for circle
```