SINHGAD COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT CODE: 310248

LAB MANUAL
Laboratory Practice-I

Semester – I, Academic Year: 2022-23

310248: Laboratory Practice-I

Teaching Scheme:
Practical: 04 Hours/Week

Examination Scheme and Marks:
Term work: 25 Marks
Practical: 25 Marks

List of Laboratory Assignments

Sr.No	Group A	Page No
1	Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.	4
2	Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro- processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.	8
3	Write a program to create Dynamic Link Library for any mathematical operation and write an application program to test it. (Java Native Interface / Use VB or VC++).	10
	Group B	
4	Write a program to solve Classical Problems of Synchronization using Mutex and Semaphore.	24
5	Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).	26
6	Write a program to simulate Memory placement strategies – best fit, first fit, next fit and worst fit.	28
7	Write a program to simulate Page replacement algorithm.	
	Part II : Elective I	
	Internet of Things and Embedded Systems	
1	Understanding the connectivity of Raspberry-Pi / Adriano with IR sensor. Write an application to detect obstacle and notify user using LEDs.	34
2	Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.	36
3	Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.	38
4	Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.	41
	Human Computer Interface	
1	Design a paper prototype for selected Graphical User Interface.	47
2	Implement GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.	48
3	Design a User Interface in Python.	51
4	To redesign existing Graphical User Interface with screen complexity.	52

Instructions:

Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a reference and hands-on resource. It should include prologue (about University/program/ institute/ department/foreword/ preface), curriculum of the course, conduction and Assessment guidelines, topics under consideration, concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality and
Guidelines for Practical Examination
Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student’s understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus. For the elective subjects students should form group of 3-4 students. The faculty coordinator will take care that all the assignment should be assigned to class and minimum two assignments are compulsory for each group.
Programming tools recommended: - Human computer Interface-GUI in python
Internet of Things and Embedded System- Raspberry Pi/Arduino Programming; Arduino IDE/Python Interfacing. Other IoT devices
Software project management-MS project/Gantt Project/Primavera

Virtual Laboratory:

- http://cse18- iiith.vlabs.ac.in/Introduction.html?domain=Computer%20Scie nce
- http://vlabs.iitb.ac.in/vlabs-dev/labs/cglab/index.php

Course Objectives:

- To learn system programming tools.
- To learn modern operating system

Course Outcome:

On completion of this course, learners will be able to

- CO1: Implement different system software's like assembler, macro processor, DLL, etc.
- CO2: Implement concept of synchronization and concurrency.
- CO3: Implement scheduling policies and memory management concepts of operating system

Assignment No.: 01

Problem Statement: Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

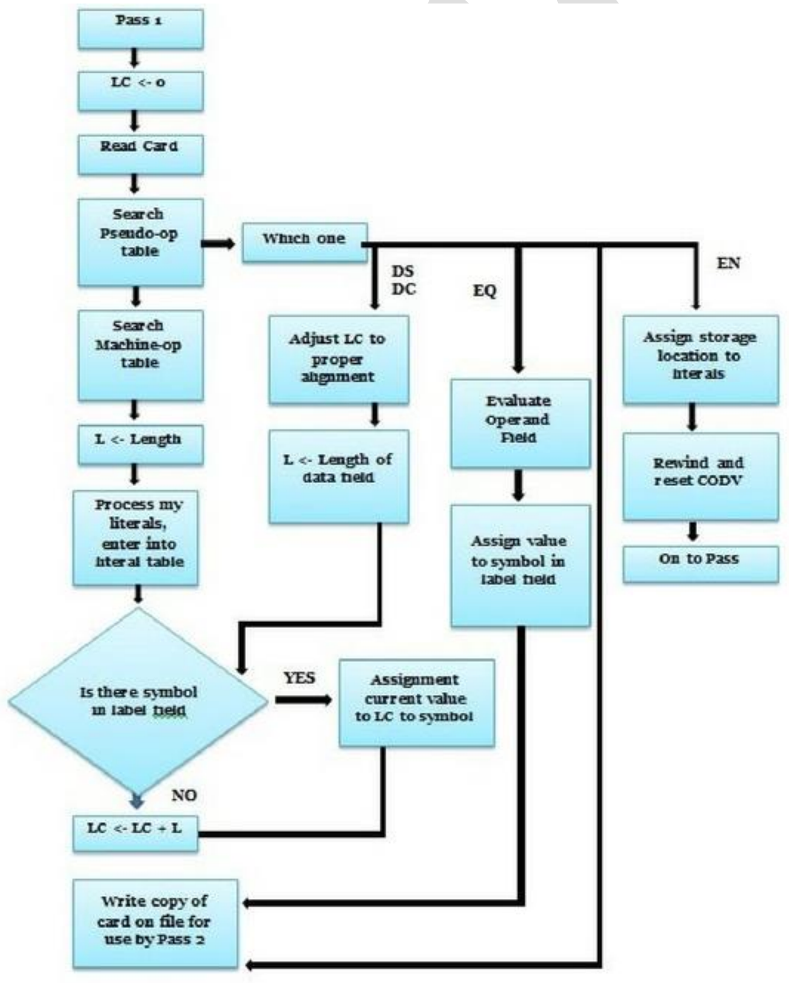
Objectives:

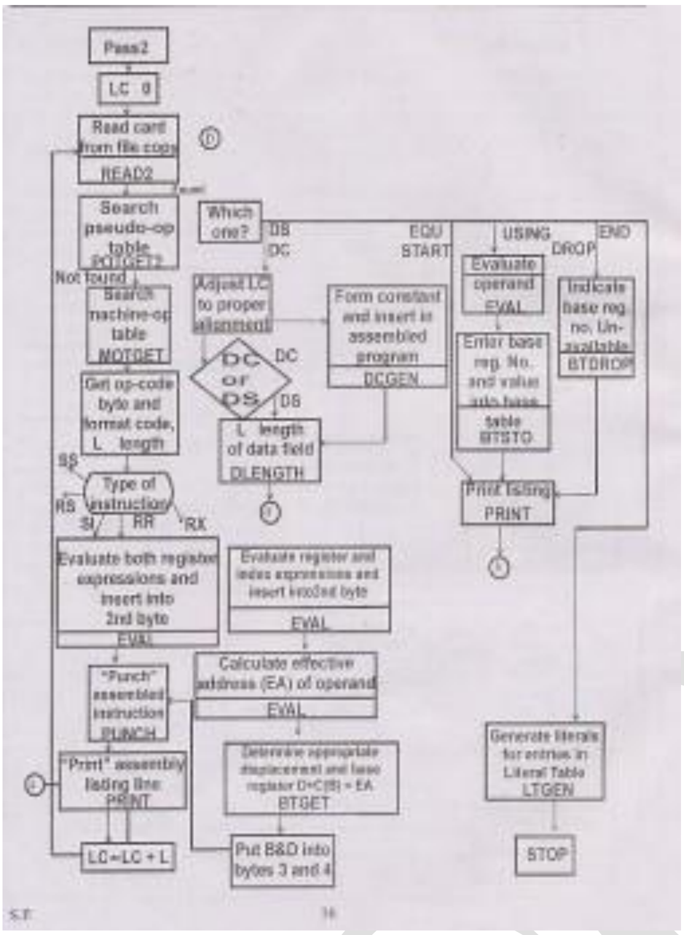
1. To study the design and implementation of pass I and pass II of two pass assembler.
2. To study the categorized instruction set of assembler.
3. To study the data structure used in assembler implementation.

Theory:

1. Explain various Data and Instruction formats of assembly language programming.
2. Explain the design of Pass- I and Pass 2 of assembler with the help of flowchart and example.
3. Discuss various Data structure used in Pass-I and Pass-2 along with its format and significance of each field.

Algorithm/Flowchart:


Pass 1
LC <- 0
Read Card
Search Pseudo-op table
Which one
DS DC
EQ
EN
Adjust LC to proper alignment
L <- Length of data field
Evaluate Operand Field
Assign value to symbol in label field
Assign storage location to literals
Rewind and reset CODV
On to Pass
Search Machine-op table
L <- Length
Process my literals, enter into literal table
Is there symbol in label field
YES
Assignment current value to LC to symbol
NO
LC <- LC + L
Write copy of card on file for use by Pass 2


Pass2
LC 0
Read card from file copy
READ2
Search pseudo-op table
POTGET2
Not found
Which one?
DS
DC
EQU
START
USING
DROP
END
Search machine-op table
MOTGET
Adjust LC to proper alignment
Form constant and insert in assembled program
DCGEN
Evaluate operand
EVAL
Indicate base reg. no. Un-available
BTDROP
Enter base reg. No. and value into base table
BTSTO
DC or DS
Get op-code byte and format code, L length
L length of data field
DLENGTH
Type of instruction
SS
RS
SI
RR
RX
Print listing
PRINT
Evaluate both register expressions and insert into 2nd byte
EVAL
Evaluate register and index expressions and insert into 2nd byte
EVAL
"Punch" assembled instruction
PUNCH
Calculate effective address (EA) of operand
EVAL
Generate literals for entries in Literal Table
LTGEN
"Print" assembly listing line
PRINT
Determine appropriate displacement and base register D+C(B) = EA
BTGET
LC=LC + L
Put B&D into bytes 3 and 4
STOP

Input:

Source code of Assembly Language

```
          SAMPLE START 100
                    USING *, 15
                    L 1, FOUR
                    A 1, =F'3'
                    ST 1, RESULT
                    SR 1, 2
                    LTORG L 2,
                    FIVE
                    A 2, =F'5'
                    A 2, =F'7'
          FIVE DC F'5' FOUR
          DC F'4' RESULT DS
          1F
END
```

Output:

```
100 SAMPLE START 100
100 USING *, 15
100 L 1, FOUR
104 A 1, =F'3'
108 ST 1, RESULT
```

```
112 SR 1, 2
114 LTORG
124 L 2, FIVE
128 A 2, =F'5'
132 A 2, =F'7'
136 FIVE DC F'5'
140 FOUR DC F'4'
144 RESULT DS 1F
152 5
156 7
160 END
```

Machine Opcode Table (MOT)

Mnemonic	Hex / Binary Code	Length (Bytes)	Format
L	5A	4	RX
A	1B	4	RX
ST	50	4	RX
SR	18	2	RR

Pseudo Opcode Table (POT)

Pseudo op	Address / Name of Procedure to implement pseudo operation
START	PSTART
USING	PUSING
DC	PDC
DS	PDS
LTORG	PLTORG
END	PEND

Symbol Table (ST)

Sr. No	Symbol name	Address	Value	Length	Relocation
1	SAMPLE	100	--	160	R
2	FIVE	136	5	4	R
3	FOUR	140	4	4	R

4	RESULT	144	__	4	R

Literal Table (LT)

Sr. No	Literal	Address	Length
1	3	120	4
2	5	152	4
3	7	156	4

Base Table (BT)

Register no Availability Value/ Contents1 N --
: : :
: : :
: : :
15 Y 100

Object Code

```
LC OPCODE OPERAND
------------------------------------------------------------------------------
100 5A 1,40(0,15)
104 1B 1,20(0,15)
108 50 1,44(0,15)
112 18 1,2
124 5A 2,36(0,15)
128 1B 2,52(0,15)
132 1B 2,56(0,15)
```

Software Requirement:
1. Fedora
2. Eclipse
3. JDK
4.

Hardware Requirement:
Not specific

Conclusion:
Input assembly language program is processed by applying Pass-I and Pass-II algorithm of assemblerand intermediate data structures, Symbol Table, Literal Table, MOT, POT, BT, etc. are generated.

Assignment No.: 02

Problem Statement: Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro- processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

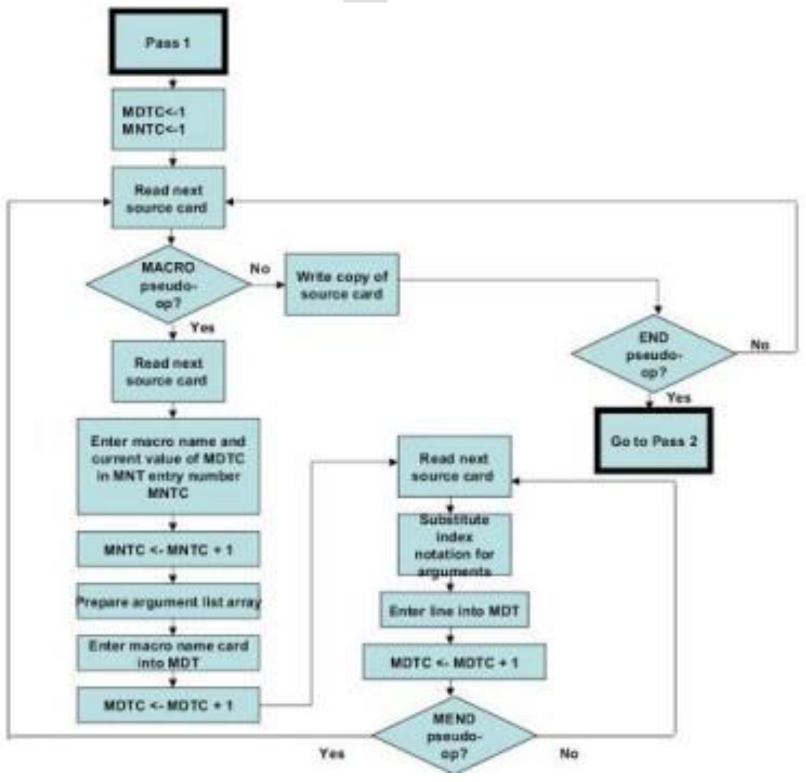
Objectives:

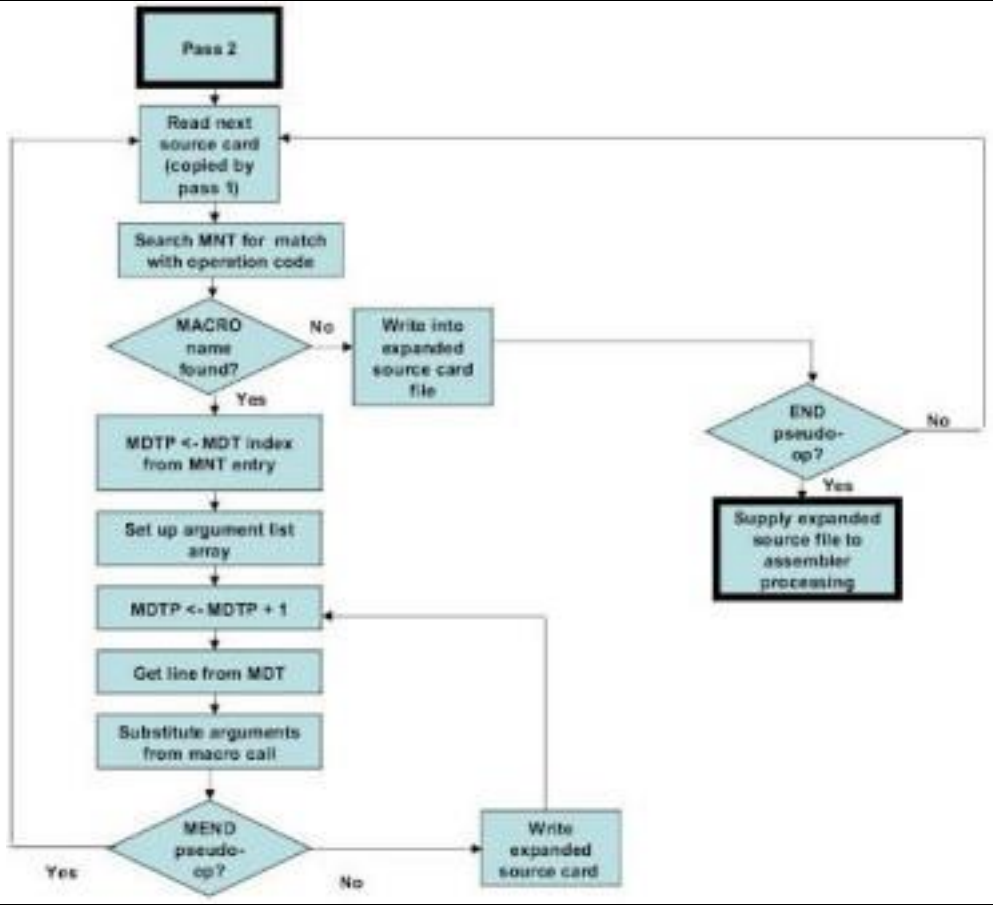
1. To identify and design different data structure used in macro-processor implementation.
2. To apply knowledge in implementation of two pass microprocessor.

Theory:

1. What is macro processor?
2. Differentiate Macro and Function?
3. Explain the design of Pass- I and Pass II of macro-processor with the help of flow chart?
4. Explain the design of Data structure used in Pass-I and Pass-II?
5. Explain the data structures used in Pass-I and Pass-II?

Algorithm/Flowchart:




Pass 2
Read next source card (copied by pass 1)
Search MNT for match with operation code
MACRO name found?
No
Write into expanded source card file
Yes
MDTP <- MDT index from MNT entry
Set up argument list array
MDTP <- MDTP + 1
Get line from MDT
Substitute arguments from macro call
MEND pseudo-op?
Yes
No
Write expanded source card
END pseudo-op?
No
Yes
Supply expanded source file to assembler processing

Input:

Output:

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:
Not Specific

Conclusion: We have successfully completed implementation of Pass-I and Pass-II of two pass macro processor.

Assignment No.: 03

Problem Statement:
Write a program to create a Dynamic Link Library for any mathematical operations (arithmetic, trigonometric and string operation) and write an application program to test it. (Java Native Interface/Use VB/VC++)

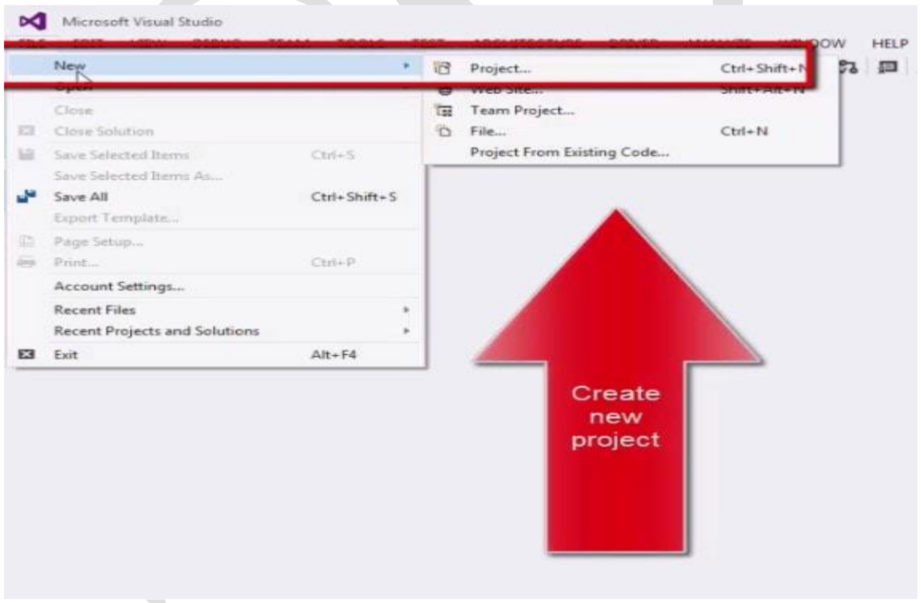
Objectives:
1. To study and understand concept of DLL.
2. To understand VC++.
3. To implement DLL using VC++.

Theory:
1. What is DLL? Significance of DLL. Advantages/ Disadvantages of DLL?
2. Explain Import and Export functions used in DLL?

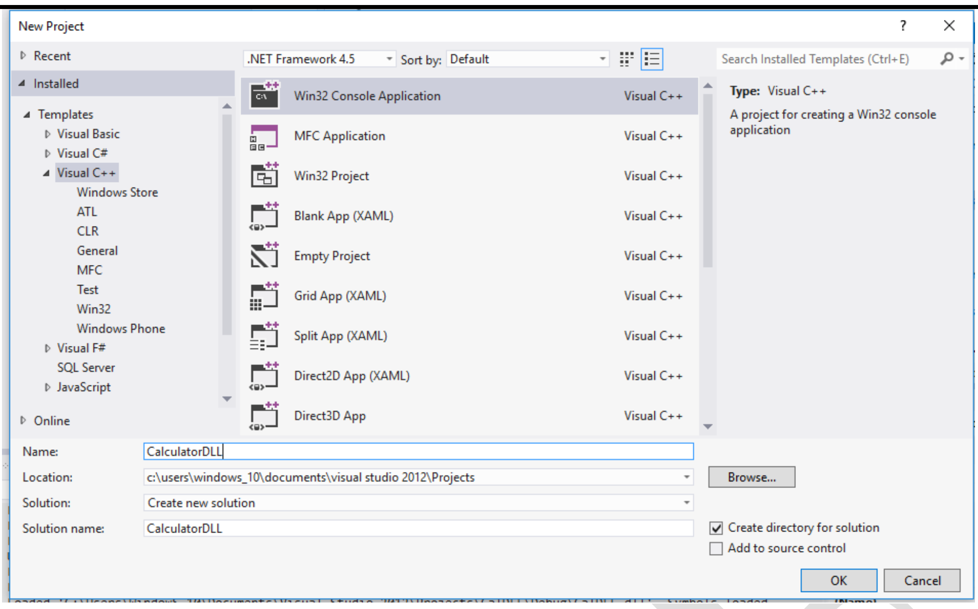
Algorithm/Flow Chart:

Steps to create DLL in C++

- Open the visual studio and click on the menu bar to create a new project. See the below Image.



- After selecting the new project, a new dialog box will be open, here select the project type Win32 and give the name to the DLL project.


New Project

Recent
Installed
Templates
Visual Basic
Visual C#
Visual C++
Windows Store
ATL
CLR
General
MFC
Test
Win32
Windows Phone
Visual F#
SQL Server
JavaScript
Online

.NET Framework 4.5 Sort by: Default

Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Blank App (XAML)	Visual C++
Empty Project	Visual C++
Grid App (XAML)	Visual C++
Split App (XAML)	Visual C++
Direct2D App (XAML)	Visual C++
Direct3D App	Visual C++

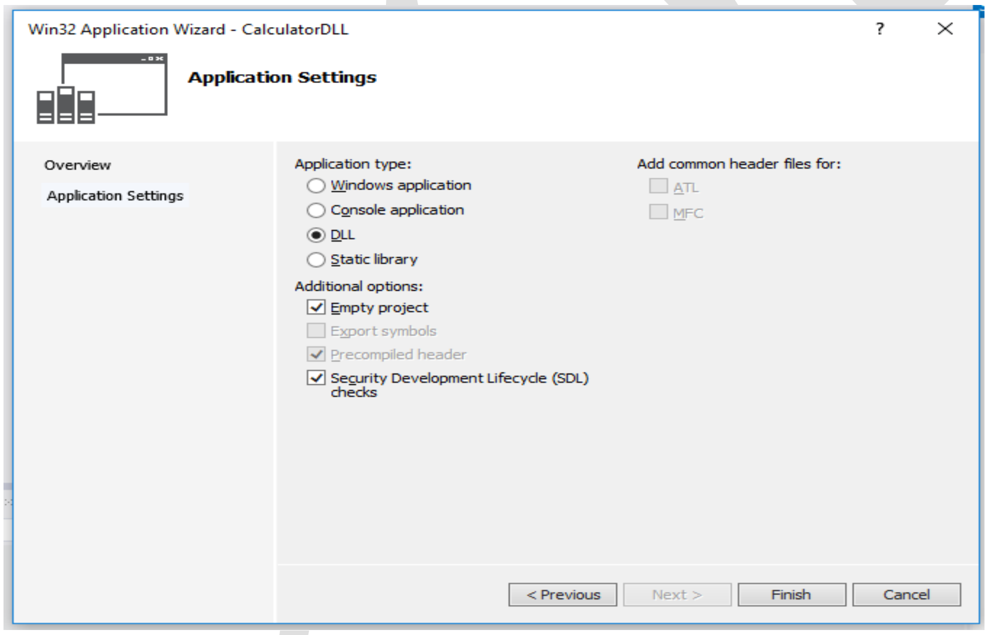
Search Installed Templates (Ctrl+E)

Type: Visual C++
A project for creating a Win32 console application

Name: CalculatorDLL
Location: c:\users\windows_10\documents\visual studio 2012\Projects
Solution: Create new solution
Solution name: CalculatorDLL

[✓] Create directory for solution
[ ] Add to source control

- On the Overview page of the Win32 Application Wizard dialog box, choose the Next button. After clicking the next button a new window will open. It is Application setting window here we will select the type of the application and click on the finish button to create the DLL project.


Win32 Application Wizard - CalculatorDLL

Application Settings

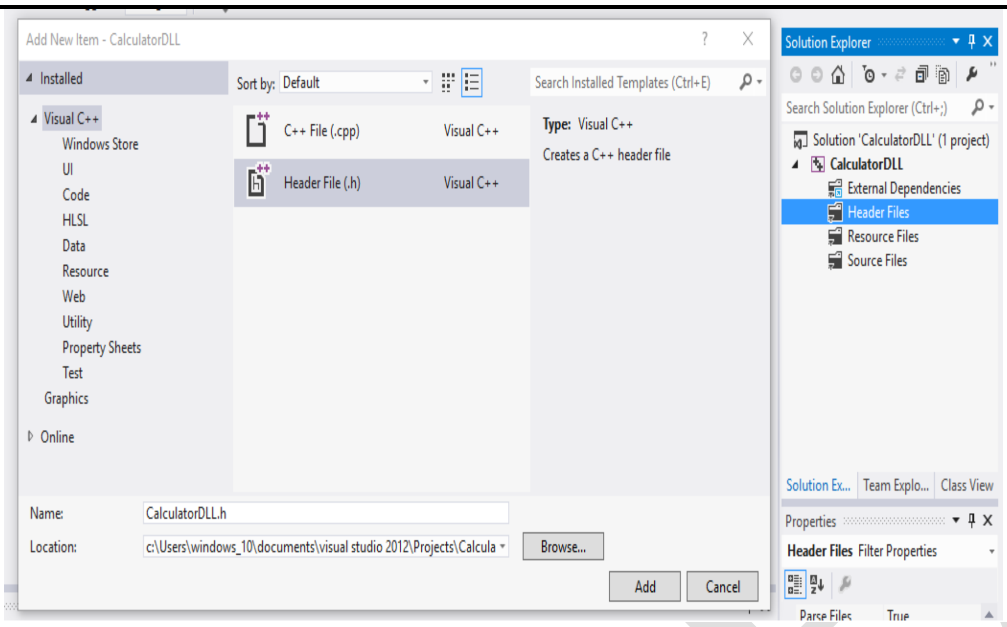
Overview
Application Settings

Application type:
( ) Windows application
( ) Console application
(●) DLL
( ) Static library

Additional options:
[✓] Empty project
[ ] Export symbols
[✓] Precompiled header
[✓] Security Development Lifecycle (SDL) checks

Add common header files for:
[ ] ATL
[ ] MFC

- After creating the DLL project you have to add the header files and source file as per your requirements. Here I am adding only one header file.


Add New Item - CalculatorDLL
Installed
Visual C++
Windows Store
UI
Code
HLSL
Data
Resource
Web
Utility
Property Sheets
Test
Graphics
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
C++ File (.cpp) Visual C++
Header File (.h) Visual C++
Type: Visual C++
Creates a C++ header file
Browse...
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'CalculatorDLL' (1 project)
CalculatorDLL
External Dependencies
Header Files
Resource Files
Source Files
Solution Ex... Team Explo... Class View
Properties
Header Files Filter Properties
Parse Files True

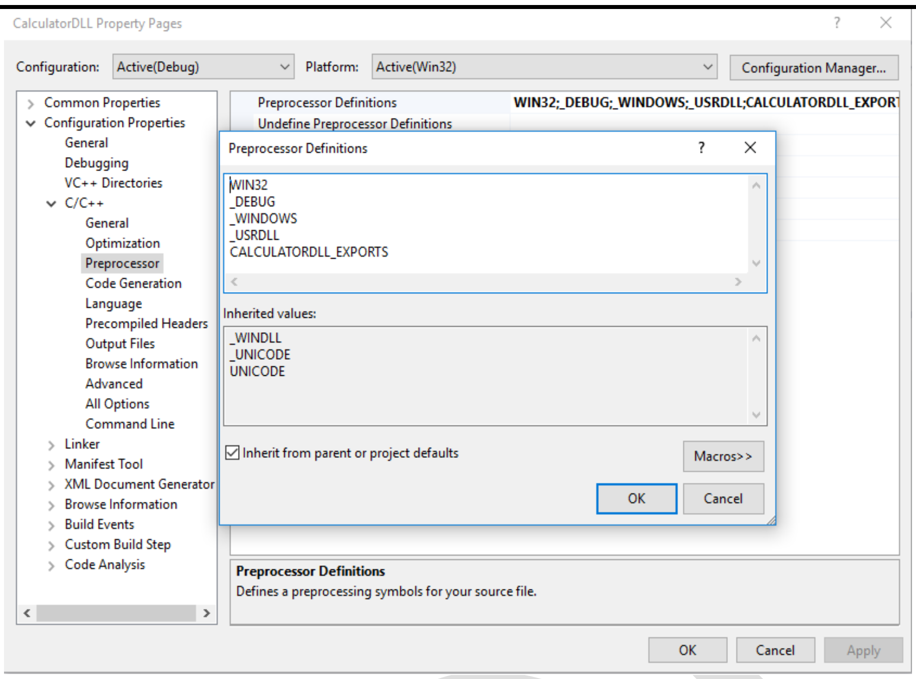
Name: CalculatorDLL.h

Location: c:\Users\windows_10\documents\visual studio 2012\Projects\Calcula

- When you have created the header file then write the desired content as per the requirements. Here I am creating a library that performs some basic arithmetic operation like addition.

```
#ifndef _CALCULATORDLL_h_
#define _CALCULATORDLL_h_
#ifdef CALCULATORDLL_EXPORTS
#define CALCULATORDLL_API __declspec(dllexport)
#else
#define CALCULATORDLL_API __declspec(dllimport)
#endif
CALCULATORDLL_API int Addition(int x,int y);
#endif
```

Note: When you have created a DLL project then automatically PROJECTNAME_EXPORTS is defined in preprocessor symbols of the DLL project. In this example, CALCULATIONDLL_EXPORTS is defined when your CALCULATIONDLL DLL project is built.


CalculatorDLL Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Preprocessor Definitions WIN32;_DEBUG;_WINDOWS;_USRDLL;CALCULATORDLL_EXPORT

Preprocessor Definitions

```
WIN32
_DEBUG
_WINDOWS
_USRDLL
CALCULATORDLL_EXPORTS
```

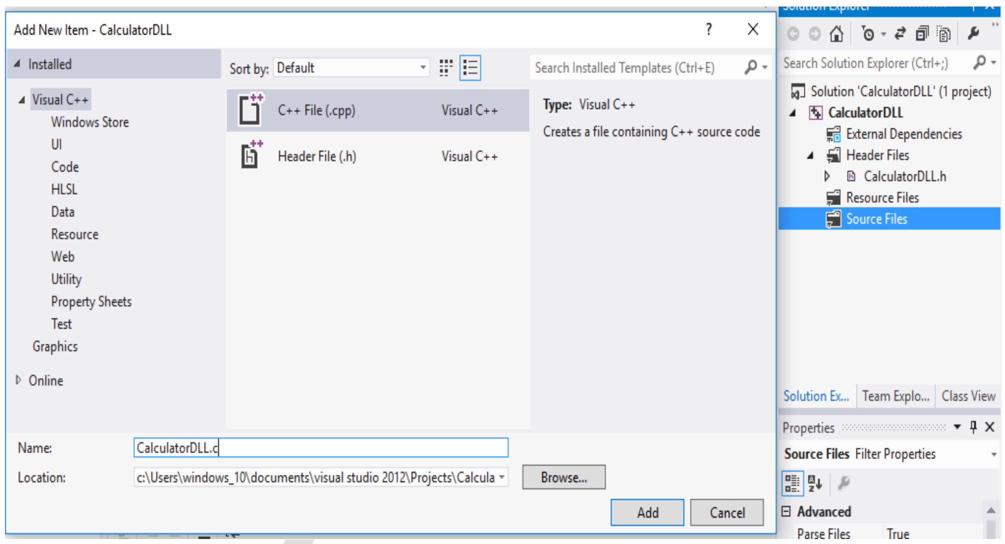
Inherited values:

```
_WINDLL
_UNICODE
UNICODE
```

☑ Inherit from parent or project defaults

Preprocessor Definitions
Defines a preprocessing symbols for your source file.

- Now it’s time to define your class member function in the source file. Here I am defining all member functions in CalculatorDLL.C file.


Add New Item - CalculatorDLL

Sort by: Default

C++ File (.cpp) Visual C++

Header File (.h) Visual C++

Type: Visual C++
Creates a file containing C++ source code

Name: CalculatorDLL.c

Location: c:\Users\windows_10\documents\visual studio 2012\Projects\Calcula

Solution 'CalculatorDLL' (1 project)

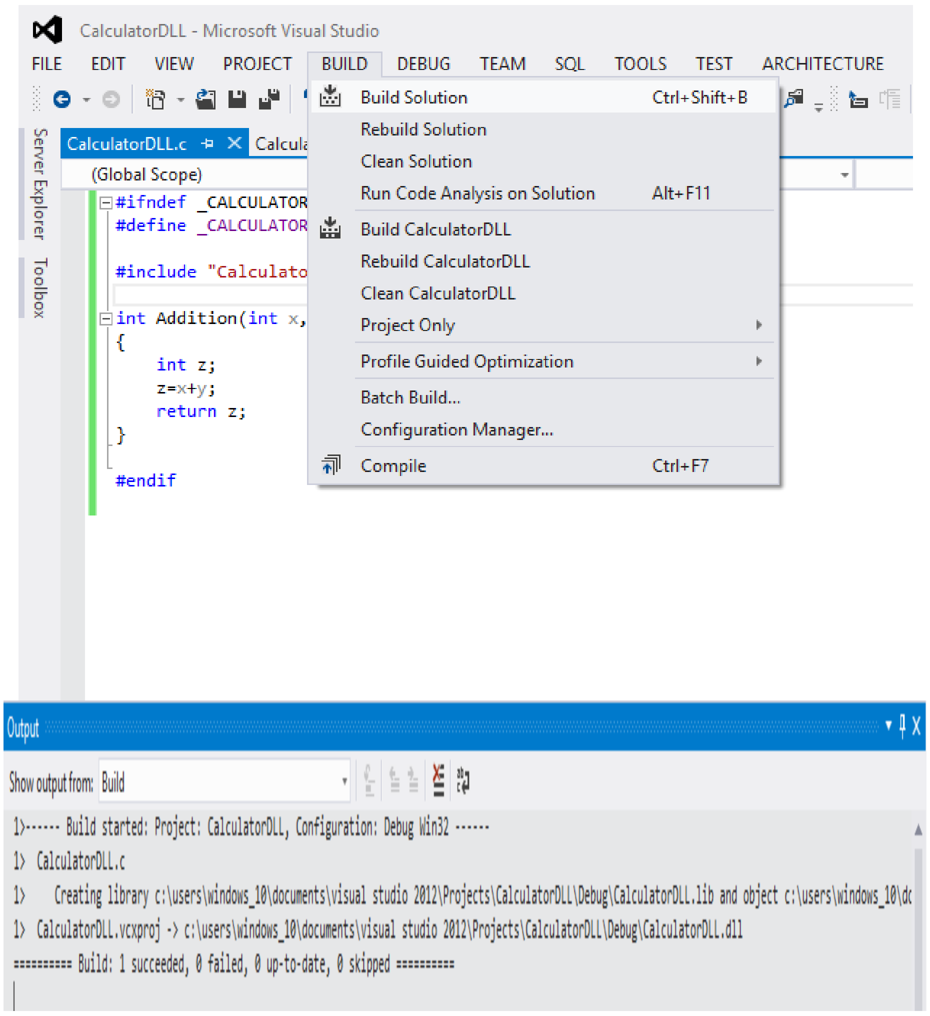
Source Files Filter Properties

Parse Files True

```
#ifndef _CALCULATORDLL_c_
#define _CALCULATORDLL_c_

#include "CalculatorDLL.h"

int Addition(int x,int y)
{
        int z;
        z=x+y;
        return z;
}
```

```
#endif
```

Now source and header files are added to the DLL project, to create the DLL and lib just build the DLL project. If everything is fine and your DLL project compiles perfectly without any error then a DLL and .lib file will be generated.



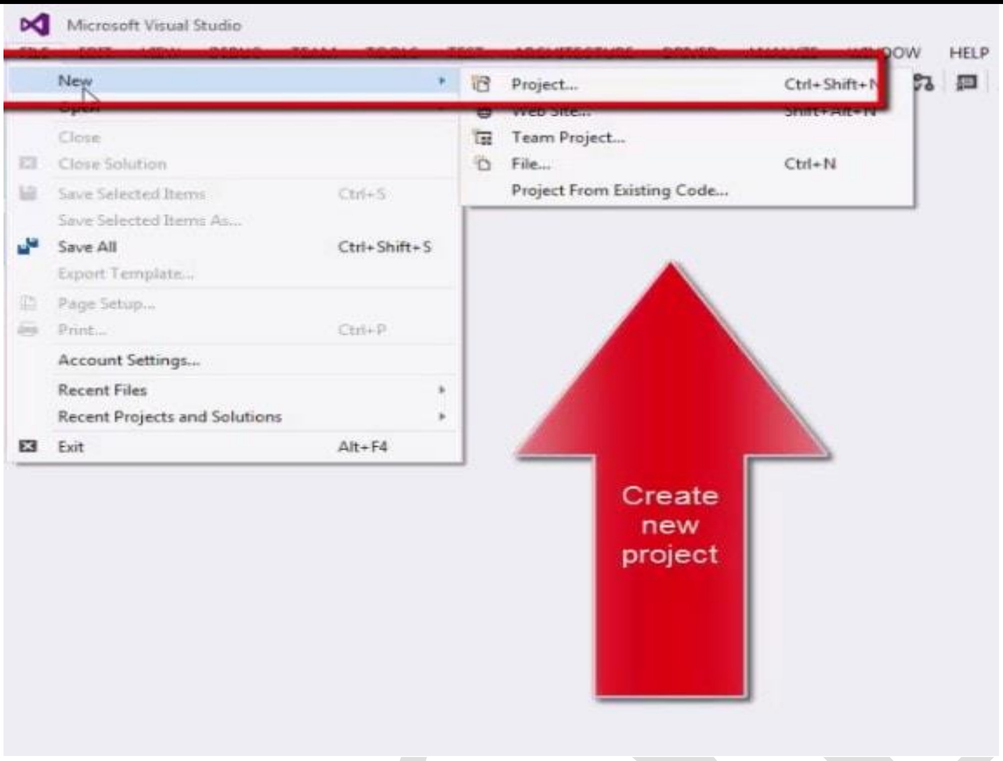
```
#ifndef _CALCULATOR
#define _CALCULATOR

#include "Calculato

int Addition(int x,
{
    int z;
    z=x+y;
    return z;
}

#endif
```

```
1>------ Build started: Project: CalculatorDLL, Configuration: Debug Win32 ------
1>  CalculatorDLL.c
1>     Creating library c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorDLL\Debug\CalculatorDLL.lib and object c:\users\windows_10\dc
1>  CalculatorDLL.vcxproj -> c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorDLL\Debug\CalculatorDLL.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

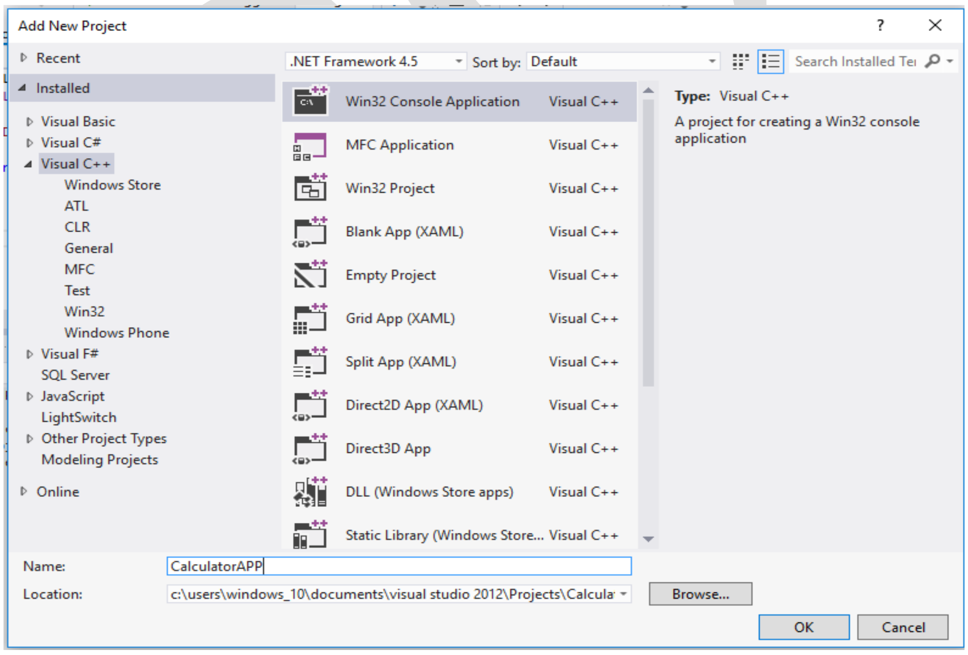
Steps to create a C ++ Application

Here I am creating a c++ application that will use the created DLL.

- Click on the menu bar to create a new c++ Application project that uses the DLL which I have created just now.

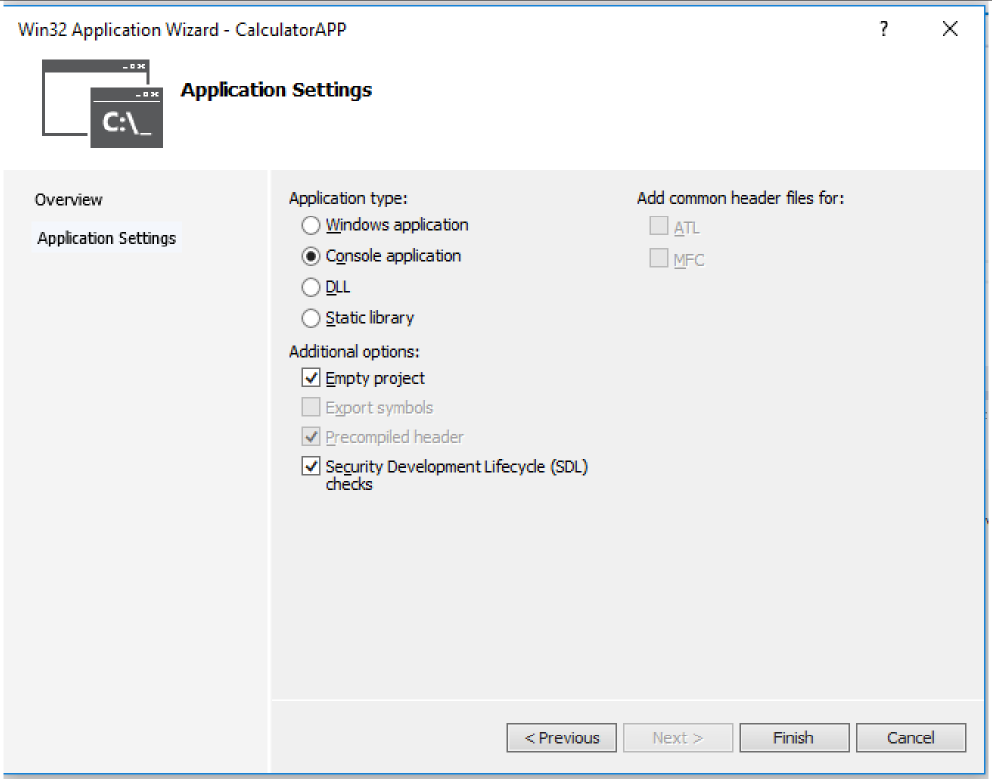


- After selecting the new project a new dialog box will be open, here select the project type Win32 Console Application and give the name to the App project.



Name: CalculatorAPP
Location: c:\users\windows_10\documents\visual studio 2012\Projects\Calcula

- On the Overview page of the Win32 Application Wizard dialog box, choose the Next button. After clicking the next button a new window will open. It is the Application setting window here we will select the type of the application and click on the finish button to create the c++ Console Application project.


Win32 Application Wizard - CalculatorAPP

Application Settings

Application type:
- ○ Windows application
- ● Console application
- ○ DLL
- ○ Static library

Additional options:
- ✓ Empty project
- ☐ Export symbols
- ✓ Precompiled header
- ✓ Security Development Lifecycle (SDL) checks

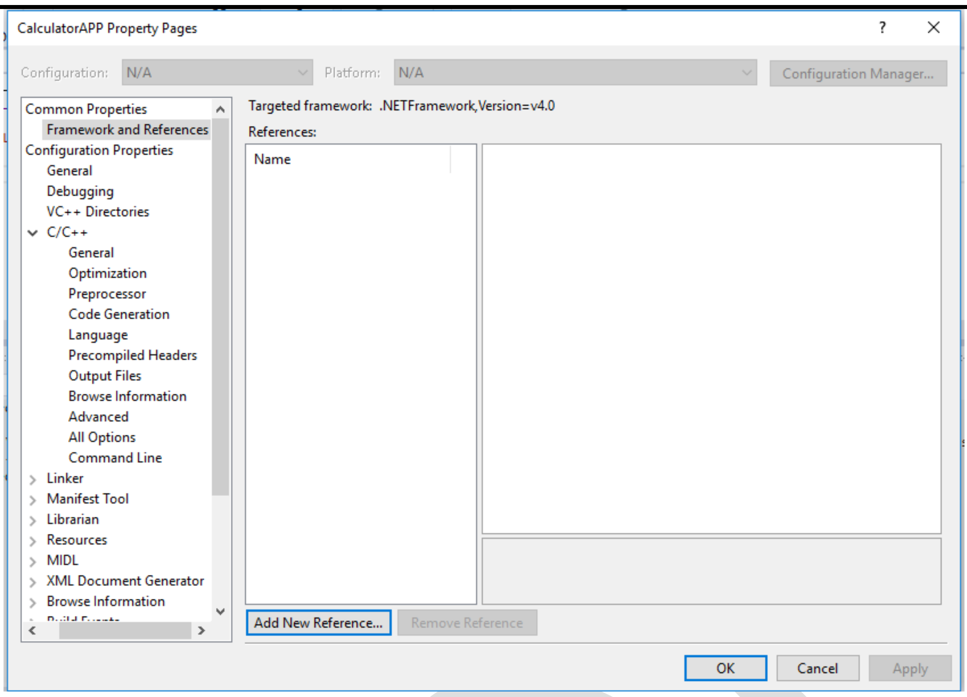
Add common header files for:
- ☐ ATL
- ☐ MFC

Now your C++ application project is ready to use the DLL (Dynamic linking library).

How to Link DLL with c++ Application

Here I am discussing simple steps to link the DLL project with the C++ Application project.

- When we have created the DLL and Application then after that we have to reference the DLL to the Application that makes the enable to Application to use the DLL function as per the requirement. To do this, under the CalculatorAPP project in Solution Explorer, select the References item. On the menu bar, choose Project, Add Reference.


CalculatorAPP Property Pages

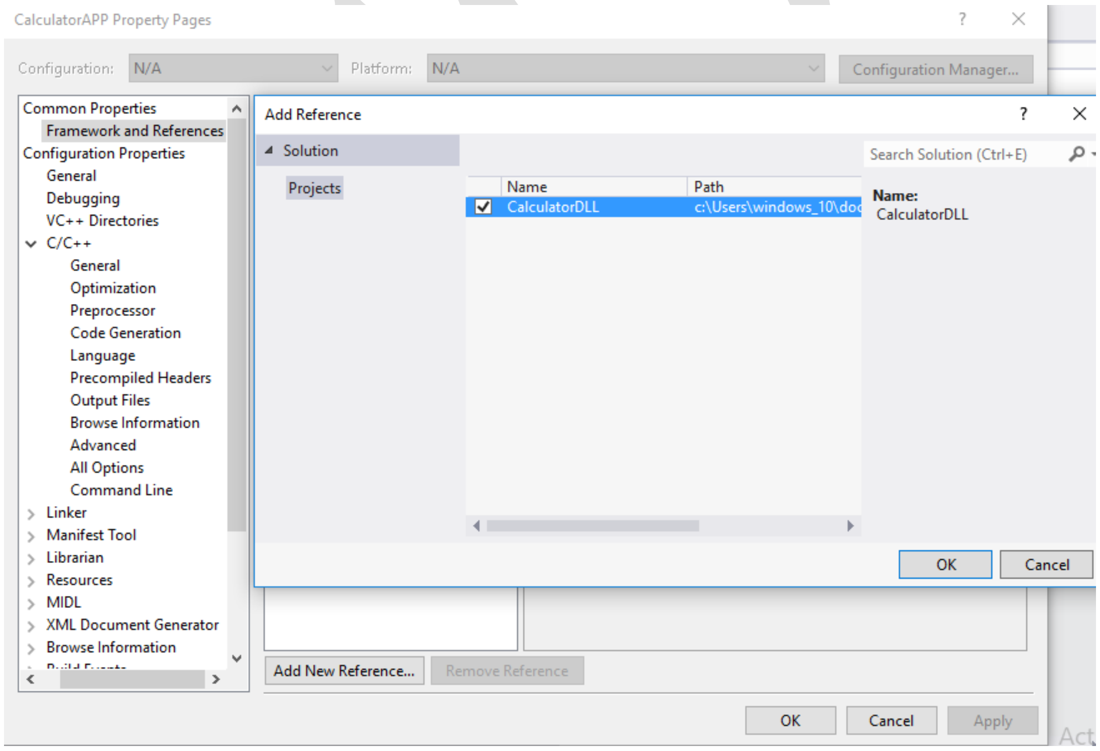
Configuration: N/A Platform: N/A

Targeted framework: .NETFramework,Version=v4.0

References:

- Common Properties
  - Framework and References
- Configuration Properties
  - General
  - Debugging
  - VC++ Directories
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - All Options
    - Command Line
  - Linker
  - Manifest Tool
  - Librarian
  - Resources
  - MIDL
  - XML Document Generator
  - Browse Information

- When you click on the Add new Reference then a dialog box will be open which has the lists of the library that you can reference. You need to just click on the check button to the required library. Here only one library is showing in the dialog box.

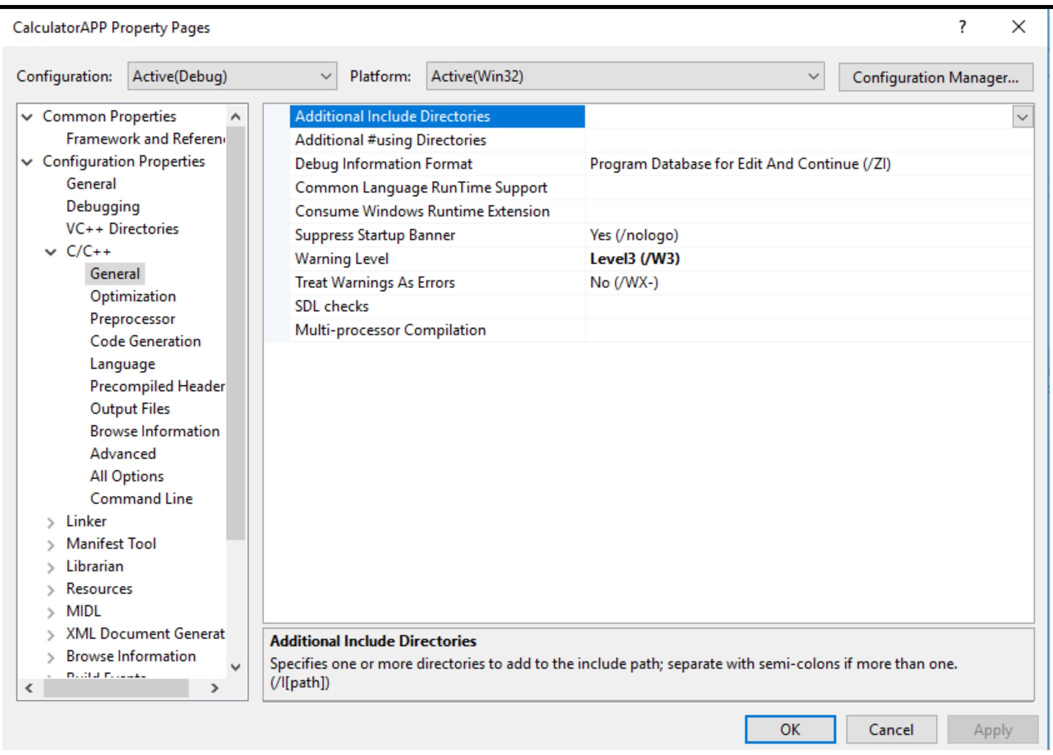

Add Reference

Solution — Projects

Name	Path
✓ CalculatorDLL	c:\Users\windows_10\doc

Name: CalculatorDLL

- Now your created library is linked with the created Application, but before using the DLL in Application you have to add the DLL header file. We just reference the DLL header file to give the path of original DLL header files in Application project included directories path.

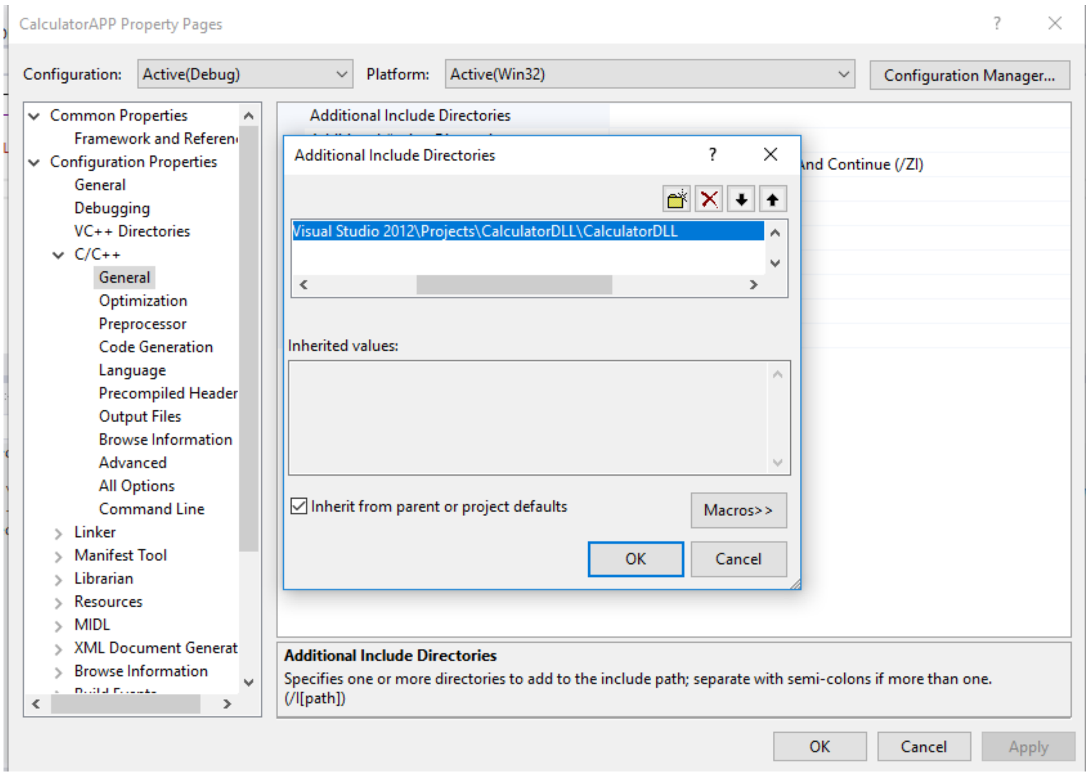


CalculatorAPP Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Additional Include Directories	
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
SDL checks	
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])



CalculatorAPP Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

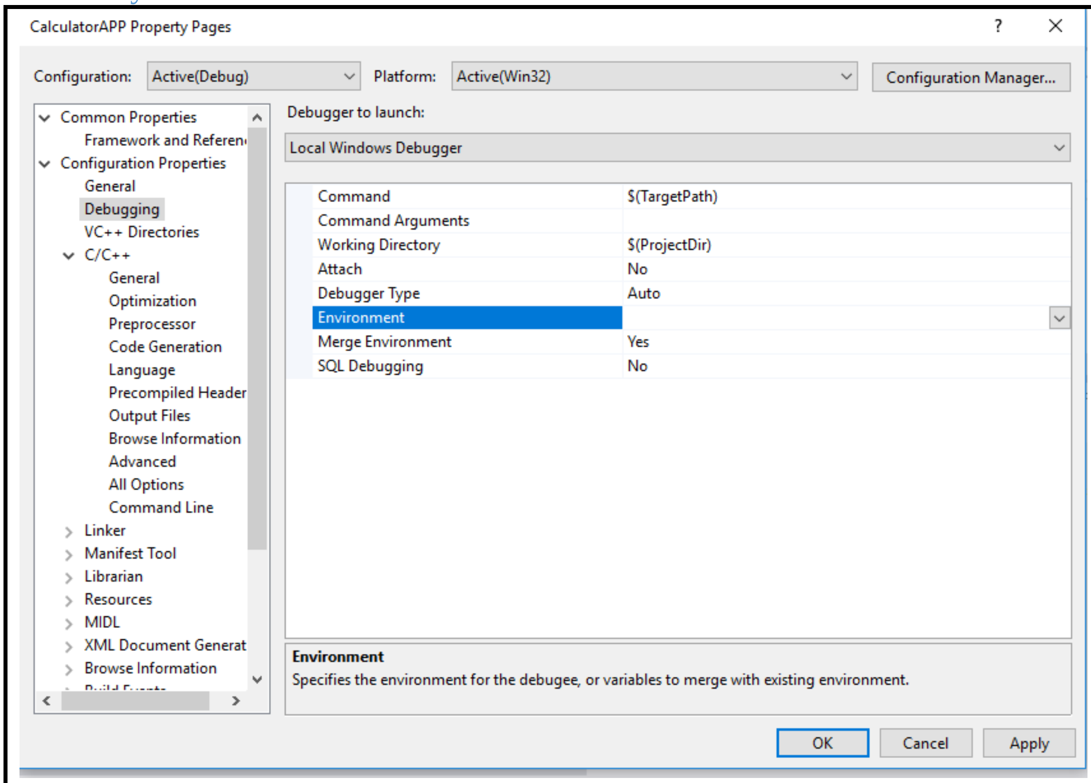
Additional Include Directories

Visual Studio 2012\Projects\CalculatorDLL\CalculatorDLL

Inherited values:

☑ Inherit from parent or project defaults

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])



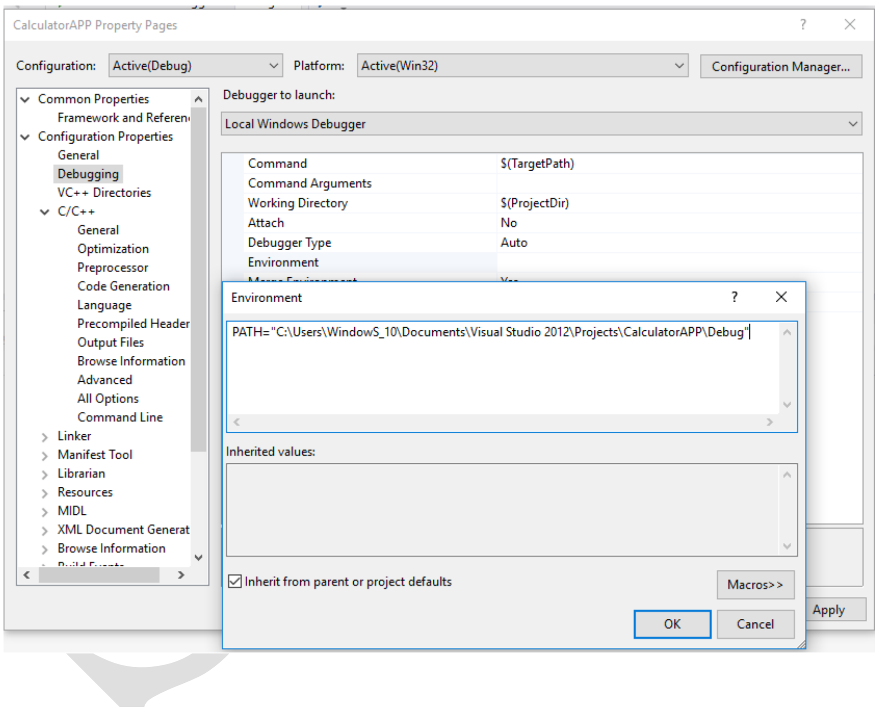
CalculatorAPP Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

Debugger to launch: Local Windows Debugger

Command	$(TargetPath)
Command Arguments	
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No

Environment
Specifies the environment for the debugee, or variables to merge with existing environment.



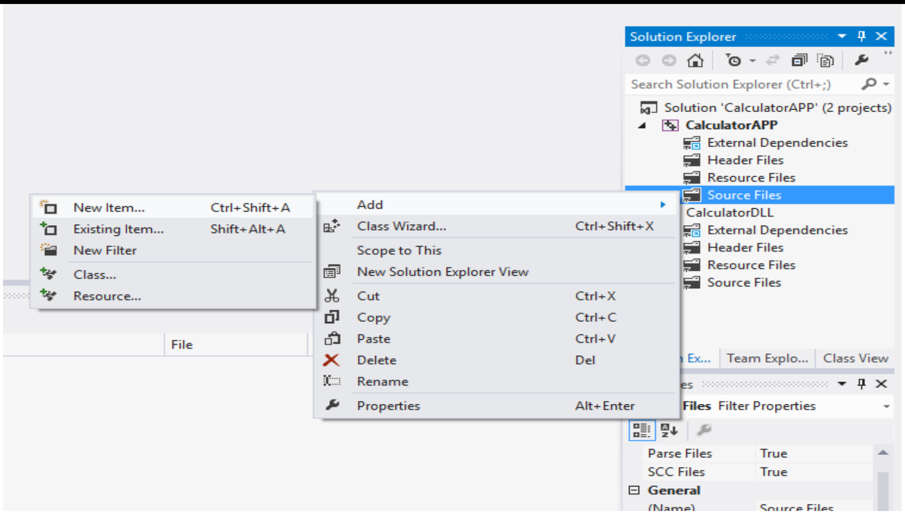
Environment

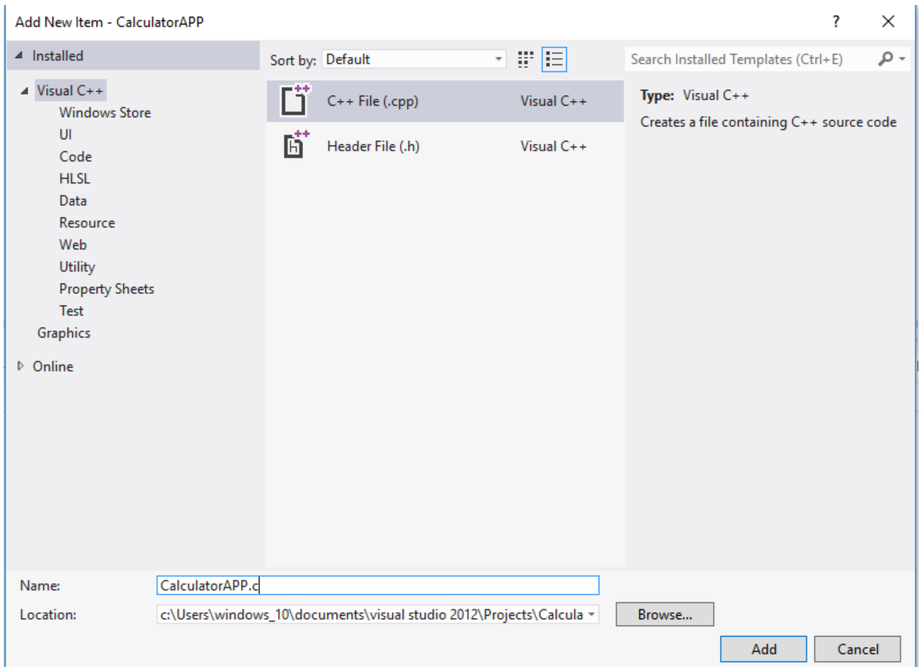
```
PATH="C:\Users\WindowS_10\Documents\Visual Studio 2012\Projects\CalculatorAPP\Debug"
```

Inherited values:

☑ Inherit from parent or project defaults

- Now it’s time to define your class member function in the source file. Here I am calling all member functions in CalculatorAPP.C file.

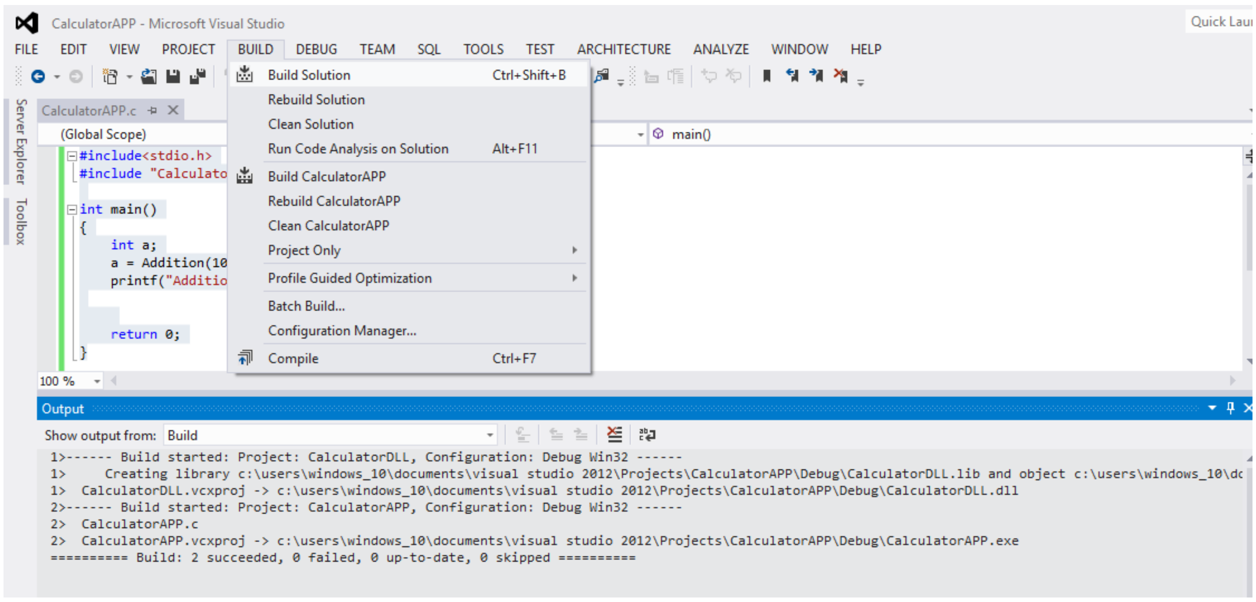



Add New Item - CalculatorAPP
Installed
Visual C++
Windows Store
UI
Code
HLSL
Data
Resource
Web
Utility
Property Sheets
Test
Graphics
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
C++ File (.cpp) Visual C++
Header File (.h) Visual C++
Type: Visual C++
Creates a file containing C++ source code
Browse...
Add
Cancel

Name: CalculatorAPP.c

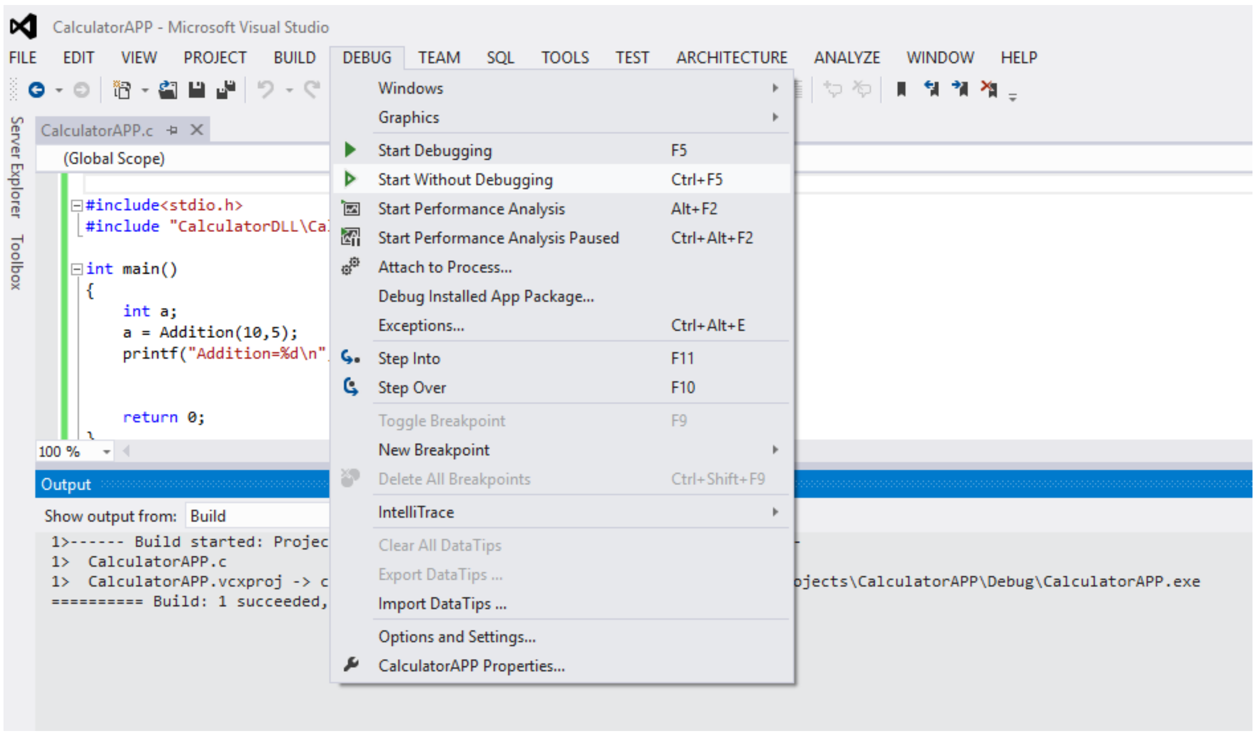
Location: c:\Users\windows_10\documents\visual studio 2012\Projects\Calcula

```
#include<stdio.h>
#include "CalculatorDLL\CalculatorDLL.h"

int main()
{
        int a;
        a = Addition(10,5);
        printf("Addition=%d\n",a);
        return 0;
}
```



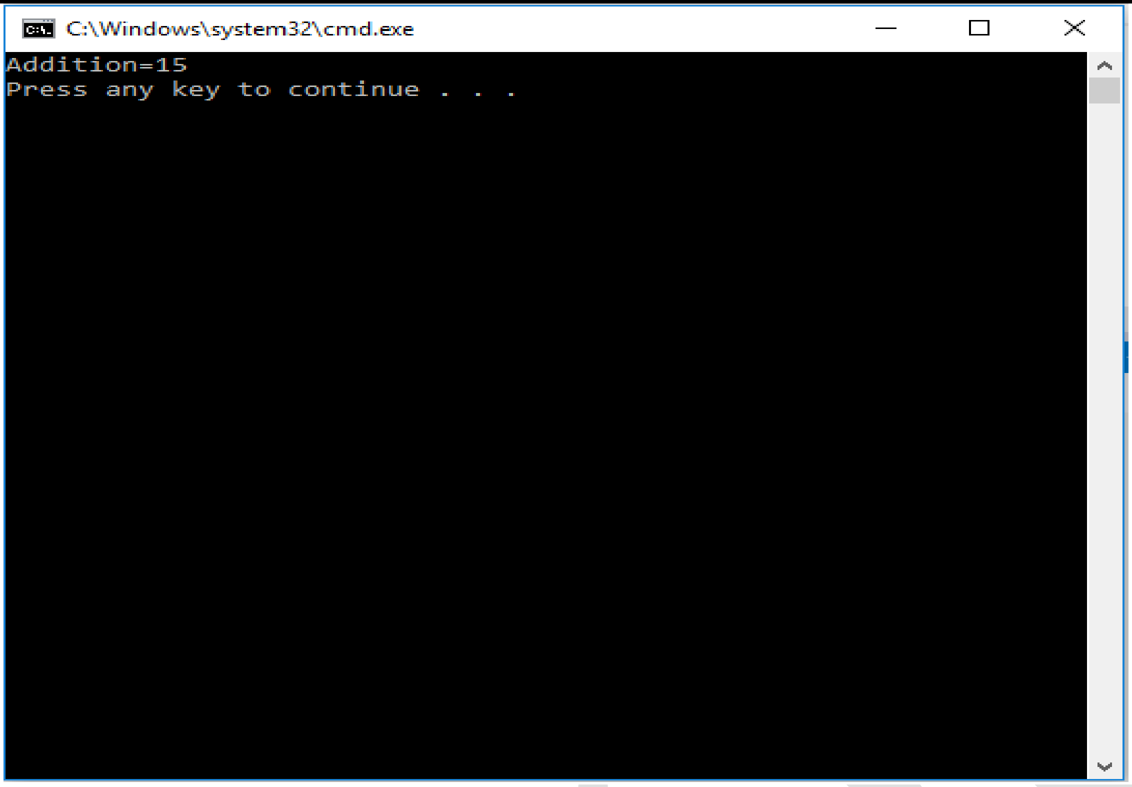
```
#include<stdio.h>
#include "Calculato

int main()
{
    int a;
    a = Addition(10
    printf("Additio

    return 0;
}
```

```
1>------ Build started: Project: CalculatorDLL, Configuration: Debug Win32 ------
1>     Creating library c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorDLL.lib and object c:\users\windows_10\dc
1>  CalculatorDLL.vcxproj -> c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorDLL.dll
2>------ Build started: Project: CalculatorAPP, Configuration: Debug Win32 ------
2>  CalculatorAPP.c
2>  CalculatorAPP.vcxproj -> c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorAPP.exe
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```


CalculatorAPP - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Windows
Graphics
Start Debugging F5
Start Without Debugging Ctrl+F5
Start Performance Analysis Alt+F2
Start Performance Analysis Paused Ctrl+Alt+F2
Attach to Process...
Debug Installed App Package...
Exceptions... Ctrl+Alt+E
Step Into F11
Step Over F10
Toggle Breakpoint F9
New Breakpoint
Delete All Breakpoints Ctrl+Shift+F9
IntelliTrace
Clear All DataTips
Export DataTips ...
Import DataTips ...
Options and Settings...
CalculatorAPP Properties...
CalculatorAPP.c
(Global Scope)
Server Explorer
Toolbox
100 %
Output
Show output from: Build

```
#include<stdio.h>
#include "CalculatorDLL\Ca

int main()
{
    int a;
    a = Addition(10,5);
    printf("Addition=%d\n"

    return 0;
}
```

```
1>------ Build started: Projec
1>  CalculatorAPP.c
1>  CalculatorAPP.vcxproj -> c                    ojects\CalculatorAPP\Debug\CalculatorAPP.exe
========== Build: 1 succeeded,
```



```
Addition=15
Press any key to continue . . .
```

Input:
Enter Number1:10
Enter Number2:05

Output:

```
Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:1
Addition:15

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:2
Substration:05

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:3
```

```
Mutiplication:150

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:4
Division:02
```

Software Requirement:

1. Windows.
2. Visual Studio.

Hardware Requirement:
Not Specific

Conclusion:
Successfully implemented DLL and tested it with VC++

Assignment No.: 4

Problem Statement:
Write a program to solve classical problems of synchronization using mutex and semaphore.

Objectives:
1. To understand reader writer synchronization problem
2. To solve reader-writer synchronization problem using mutex and semaphore

Theory:

- There is a data area shared among a number of processor registers.
- The data area could be a file, a block of main memory, or even a bank of processor registers.
- There are a number of processes that only read the data area (readers) and a number that only write to the data area (writers).
- The conditions that must be satisfied are

  Any number of readers may read simultaneously read the file.
  Only one write at a time may write to the file.
  If a writer is writing to the file, no reader may read it.

Semaphore:

Definition: Semaphores are system variables used for synchronization of process **Two types of Semaphore:**

Counting semaphore – integer value can range over an unrestricted domain **Binary semaphore –**

- Integer value can range only between 0 and 1; can be simpler to implement
- Also known as mutex locks

Semaphore functions:

Package: import java.util.concurrent.Semaphore;

1) To initialize a semaphore:

Semaphore Sem1 = new Semaphore(1);

2) To wait on a semaphore:

```
/* Wait (S)
 while S<=0no-
 op;
 S - -;
*/ Sem1.acquire();
```

3) To signal on a semaphore:

```
/* Signal(S)S
 ++;
*/ mutex.release();
```

Algorithm/Flowchart:

__**Algorithm for Reader Writer:**__

1. import java.util.concurrent.Semaphore;
2. Create a class RW
3. Declare semaphores – mutex and wrt
4. Declare integer variable readcount = 0

5. Create a nested class Reader implements Runnable

a. Override run method (Reader Logic)

```
i. wait(mutex);
ii. readcount := readcount +1;
iii. if readcount = 1 then
iv. wait(wrt);
v. signal(mutex);
vi. …
vii. reading is performed
viii. …
ix. wait(mutex);
x. readcount := readcount – 1;
xi. if readcount = 0 then signal(wrt);
xii. signal(mutex):
```

6. Create a nested class Writer implements Runnable

a. Override run method (Writer Logic)

```
i. wait(wrt);
ii. …
iii. writing is performed
iv. …
v. signal(wrt);
```

7. Create a class main

a. Create Threads for Reader and Writer

Start these thread

Input:

1. Number of Readers
2. Number of Writers

Output:

Execution of Readers and Writers

Conclusion: Implemented Reader Writer synchronization problem using semaphores in Java

Assignment No.: 05

Problem Statement:
Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority(Non Preemptive) and Round Robin (Preemptive).

Objectives:

1. To study the process management and various scheduling policies viz. Preemptive and Non preemptive.
2. To study and analyze different scheduling algorithms.

Theory :

1. Define process. Explain need of process scheduling.
2. Explain different scheduling criteria and policies for scheduling processes.
3. Explain possible process states.
4. Explain FCFS, SJF(Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive) and determine waiting time, turnaround time, throughput using eachalgorithm.

Algorithm/Flowchart:

1. FCFS

a. Input the processes along with their burst time (bt).
b. Find waiting time (wt) for all processes.
c. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e.wt[0] = 0.
d. Find waiting time for all other processes i.e. for process i ->wt[i] = bt[i-1] + wt[i-1] .
e. Find turnaround time = waiting_time + burst_time for all processes.
f. Find average waiting time = total_waiting_time / no_of_processes.
g. Similarly, find average turnaround time = total_turn_around_time / no_of_processes.

2. SJF

a. Traverse until all process gets completely executed.
b. Find process with minimum remaining time at every single time lap.
c. Reduce its time by 1.
d. Check if its remaining time becomes 0
e. Increment the counter of process completion.
f. Completion time of current process = current_time +1;
g. Calculate waiting time for each completed process.wt[i]= Completion time - arrival_time-burst_time
h. Increment time lap by one.
i. Find turnaround time (waiting_time+burst_time).

3. Priority

a. First input the processes with their burst time and priority.
b. Sort the processes, burst time and priority according to the priority.
c. Now simply apply FCFS algorithm.

4. RR

a. Create an array **rem_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array).
b. Create another array **wt[]** to store waiting times of processes.
c. Initialize this array as 0.
d. Initialize time : t = 0
e. Keep traversing the all processes while all processes are not done. Do following for i'thprocess if it is not done yet.

```
If rem_bt[i] > quantum
    t = t + quantum
    bt_rem[i] -= quantum;
Else // Last cycle for this process
    t = t + bt_rem[i];
wt[i] = t - bt[i]
bt_rem[i] = 0; // This process is over
```

Input:

1. Enter the number of processes
2. Enter burst time and arrival time of each process

Output:

1. Compute Waiting time, turnaround time, average waiting time, average turnaround time and throughput.

For each algorithm display result as follows:

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
P1				
P2				
P3				

Calculate

1. Average waiting time=
2. Average turnaround time=
3. Throughput=

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:
For simulation no dependency

Conclusion:
CPU scheduling algorithms implemented successfully

Assignment No.: 06

Problem Statement:

Write a Java/C++ program to simulate memory placement strategies

1. First Fit
2. Best Fit
3. Worst Fit
4. Next Fit

Objectives:

1. To acquire knowledge memory placement strategies
2. To be able to implement memory placement strategies

Theory:

1. Why we need memory placement strategies?
2. What is fragmentation?
3. Explain working of memory placement strategies with suitable example.

Algorithm/Flowchart:

1. First Fit algorithm/pseudo code

```
o Read all required input
o FOR i<-0 to all jobs 'js'
    ▪ FOR j<-0 to all blocks 'bs'
        o IF block[j]>=jobs[i]
            ▪ Check jth block is already in use or free
                • Continue and search next free block
            ▪ Otherwise allocate jth block to ith job
o Display all job with allocated blocks and fragmentation
```

2. First Fit algorithm/pseudo code

```
o Read all required input
o FOR i<-0 to all jobs 'js'
    ▪ SET BestInd   -1
    ▪ FOR j<-0 to all blocks 'bs'
        o IF block[j]>=jobs[i]
            ▪ IF Block is free and BestInd==-1 THEN SET
              BestInd j
            ▪ ELSEIF Block is free and block[BestInd]>block[j]
              THEN SET BestInd j
            ▪ ELSE continue with next block
                • Continue and search next free block
            ▪ IF BestInd!=-1 THEN allocate jth block to ith job
    o Display all job with allocated blocks and fragmentation
```

3. Worst Fit Algorithm/Pseudo code

```
    o Read all required input
    o FOR i<-0 to all jobs 'js'
        ▪ SET WstInd    -1
        ▪ FOR j<-0 to all blocks 'bs'
            o IF block[j]>=jobs[i]
                ▪ IF Block is free and WstInd==-1 THEN SET
                  WstInd  j
                ▪ELSEIF Block is free and block[WstInd]<block[j]
                 THEN SET WstInd  j
            ▪ ELSE continue with next block
                • Continue and search next free block
            ▪ IF WstInd!=-1 THEN allocate jth block to ith job
    o Display all job with allocated blocks and fragmentation
```

4. As above write algorithm of Next Fit strategies

Input:

- No. of jobs (js) & No. of blocks (bs)
- Job size of all jobs & Block size of all blocksFor
  Example:
  js=4 bs=5
  block[] = {100, 500, 200, 300, 600};
  jobs[] = {212, 417, 112, 426};

Output:
Sample output of Worst Fit algorithm (same way generate o/p forother algorithms)-

```
Process No. Process Size Block no.1

212 5

2 417 2
```

```
3 112 4
4 426 Not Allocated
```

Software Requirement:
1. Eclipse IDE
2. Java

Hardware Requirement:

Not specific

Conclusion: successfully implemented simulation of memory placement strategies.

Assignment No.: 07

Problem Statement:

Write a Java Program (using OOP features) to implement paging simulation using

1. FIFO
2. Least Recently Used (LRU)
3. Optimal algorithm

Objectives:

1. To study page replacement policies to understand memory management.
2. To understand efficient frame management usingreplacement policies.

Theory:

1. What is Page Simulation?
2. Explain Page Replacement Algorithm?

Algorithm/Flowchart:

FIFO :

```
Start the process
Read number of pages n
Read number of pages no
Read page numbers into an array a[i]
Initialize avail[i]=0 .to check page hit
Replace the page with circular queue, while re-placing check page availability in theframe  Place
avail[i]=1 if page is placed in the frame Count page faults
Print the results.
Stop the process.
```

LEAST RECENTLY USED

```
Start the process
Declare the size
Get the number of pages to be inserted
Get the value
Declare counter and stack
Select the least recently used page by counter value
Stack them according the selection.
Display the values
Stop the process
```

OPTIMAL ALGORTHIM:

```
Start Program
Read Number Of Pages And Frames
```

Read Each Page Value
Search For Page In The Frames
If Not Available Allocate Free Frame
If No Frames Is Free Replace The Page With The Page That Is LeastUsed 7.Print Page Number Of Page Faults
Stop process.

Input:
Number of frames.
Number of pages.
Page sequence

Output:
1. Sequence of allocation of pages in frames (for each algorithm)
2. Cache hit and cache miss ratio.

Software Requirement:
1. Fedora
2. Eclipse.
3. JDK

Hardware Requirement:

No Specific

Conclusion: Successfully implemented all page replacement policies.

Part II : Elective I

Internet of Things and Embedded Systems

Assignment No.: 01

Problem Statements:

Understanding the connectivity of Raspberry-Pi / Adriano with IR sensor. Write an application to detect obstacle and notify user using LEDs.

Objectives:

1. To understand the concept of Proximity sensor.
2. To interface Proximity sensor with Raspberry Pi model.
3. To program the Raspberry Pi model to detect the nearest object using proximity sensor and give indication through led.

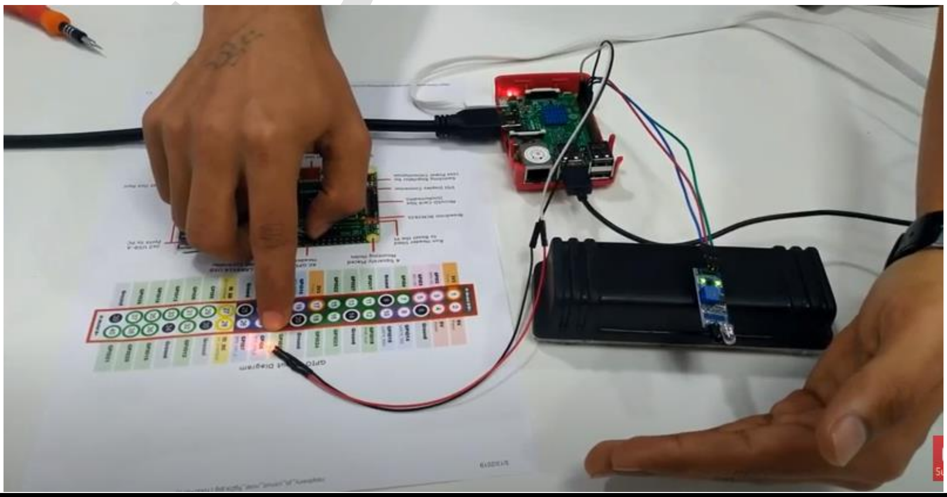
Theory:

1. What is Proximity sensor?
2. Explain Interface Diagram of Raspberry-Pi / Adriano with IR sensor
3. Explain assembling circuit steps?

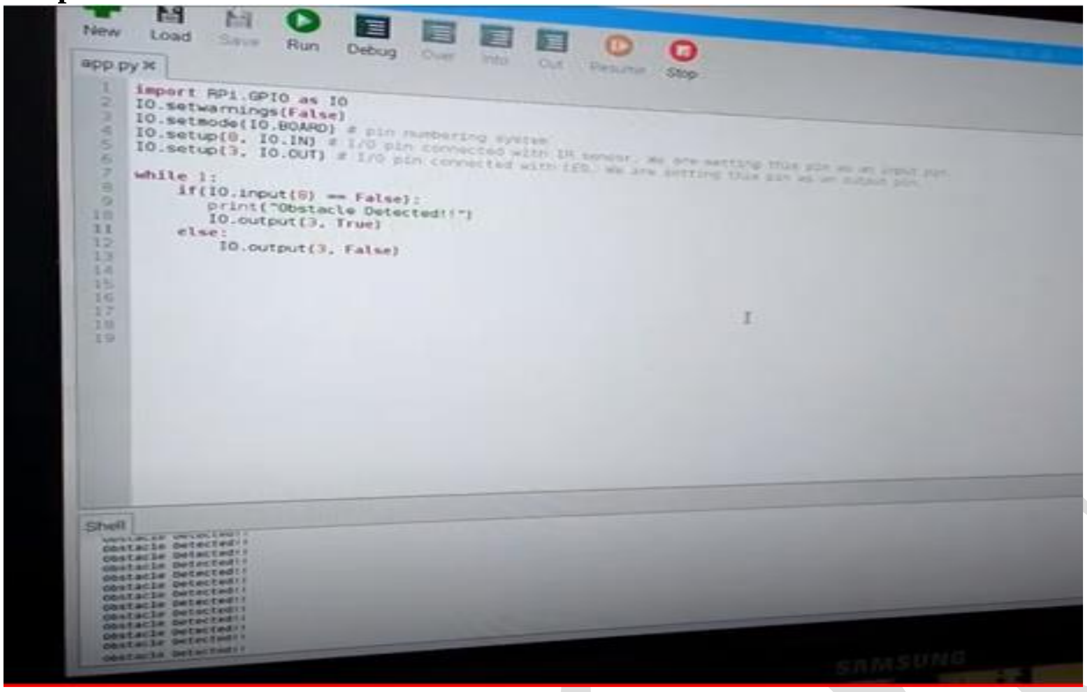
Algorithm/ Flow Chart:

1. Import GPIO and Time library
2. Set mode i.e. GPIO.BOARD
3. Set GPIO pin ‘15’ as Input
4. Set GPIO pin ‘16’ as Input
5. Read input from GPIO pin ‘15’
6. Store the input value in the variable ‘i’
7. If (i==1) then print the message as “Object is detected” and make the LED ON
8. If (i==0) then print the message as “No object detected” and make the LED OFF

Input:



Output:



```
import RPi.GPIO as IO
IO.setwarnings(False)
IO.setmode(IO.BOARD) # pin numbering system
IO.setup(8, IO.IN) # I/O pin connected with IR sensor. We are setting this pin as an input pin.
IO.setup(3, IO.OUT) # I/O pin connected with LED. We are setting this pin as an output pin.

while 1:
    if(IO.input(8) == False):
        print("Obstacle Detected!!")
        IO.output(3, True)
    else:
        IO.output(3, False)
```

```
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
Obstacle Detected!!
```

Software Requirement:
Raspbian OS (IDLE)

Hardware Requirement:
Raspberry Pi Board
Proximity sensor, Led, 330 ohm register
Monitor

Conclusion:
Successfully done the connectivity of Raspberry-Pi / Adriano with IR sensor. Tested connectivity by using LEDs to detect obstacle.

Assignment No.: 02

Problem Statements:
Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

Objectives:

1. To understand the concept of Temperature-Humidity sensor (DHT11).
2. To interface Temperature-Humidity sensor with Raspberry Pi model.
3. To program the Raspberry Pi model to measure the real time Temperature and Humidity of the Environment.

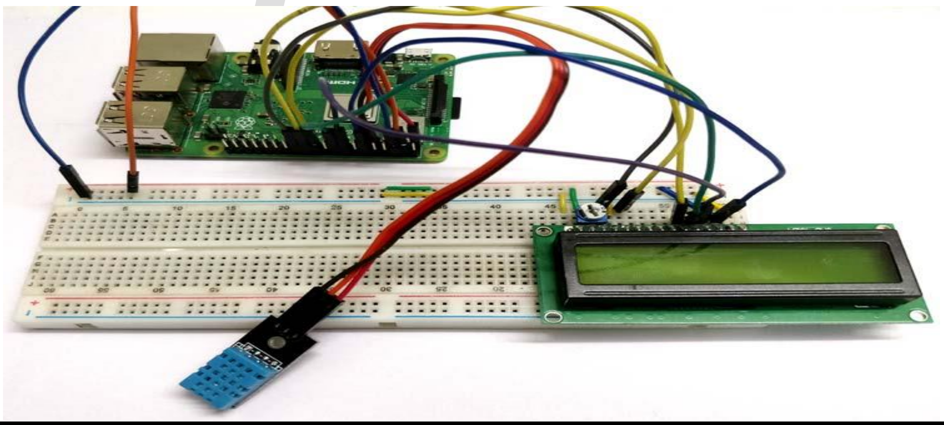
Theory:

1. What is DHT11?
2. Explain Interface Diagram of Raspberry-Pi / Adriano with Temperature-Humidity Sensor?
3. Explain assembling circuit steps?

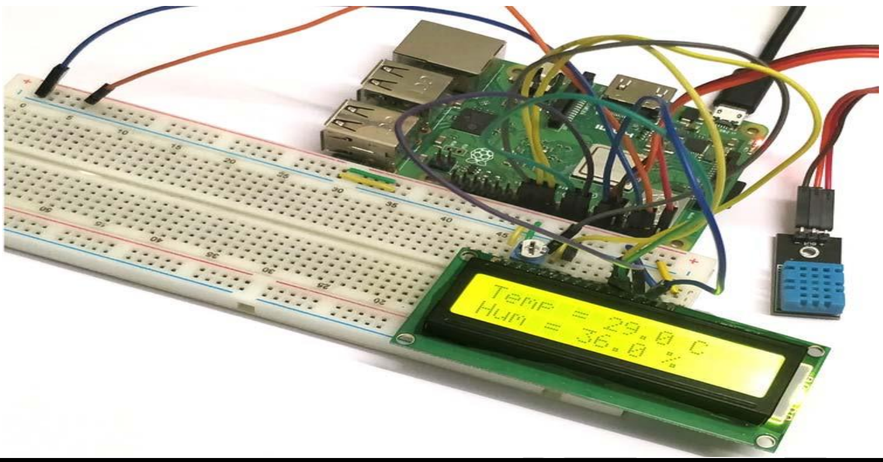
Algorithm/ Flow Chart:

1. Import GPIO, time and dht11 libraries.
2. Set all the warnings as False.
3. Set mode i.e. GPIO.BOARD
4. Read data using GPIO pin number 7 (dhtPin)
5. Write 'while loop' for displaying Temperature and Humidity values continuously
6. First Read the GPIO pin and Store the data in dhtValue Variable.
7. Print the temperature value.
8. Print the Humidity value.
9. Give delay of 1 second

Input:



Output:



Software Requirement:
Raspbian OS (IDLE)

Hardware Requirement:

1. Raspberry Pi Board module.
2. Temperature-Humidity sensor (DHT11) module.
3. Monitor

Conclusion:

1. Successfully done the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor.
2. Successfully implemented an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

Assignment No.: 03

Problem Statements:

Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.

Objectives:

1. To understand the working of Raspberry Pi Camera .
2. To interface Raspberry Pi Camera with Raspberry Pi model.
3. To program the Raspberry Pi model to control the Raspberry Pi Camera Preview.
4. To program the Raspberry Pi model to capture still images from the Raspberry Pi Camera

Theory:

Algorithm/ Flow Chart:

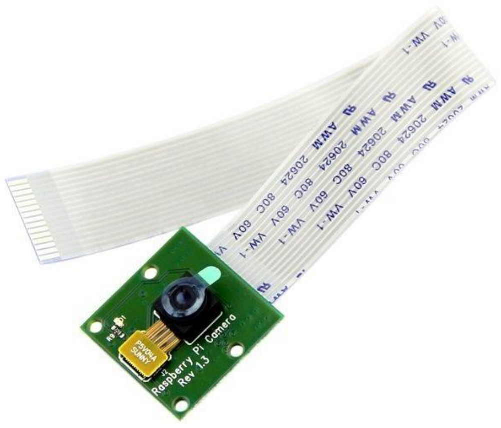
a. To program the Raspberry Pi model to control the Raspberry Pi Camera Preview:

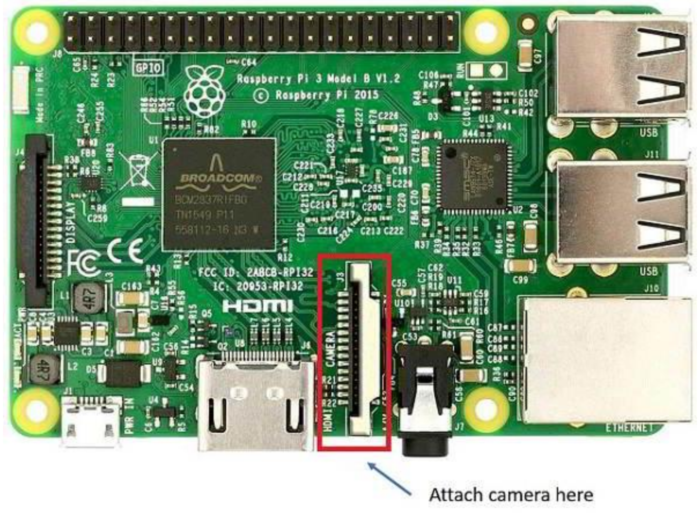
1. Import Picamera library o Import time library o Create a variable(instance) of PiCamera class
2. Display the camera preview on screen using the command “start_preview()”.
3. We can define 10 second delay to see the camera preview.
4. To stop camera preview after 10 second, use the command “stop_preview()”.

b. To program the Raspberry Pi model to capture still images from the Raspberry Pi Camera 34.

5. Import picamera library o Import time library.
6. Create a variable(instance) of PiCamera class.
7. Display the camera preview on screen using start_preview().
8. We can define 5 second delay to see the camera preview.
9. Capture the image using camera.capture(‘path of the image. extension’).
10. Then stop the camera preview using the command “stop_preview()”.

Input:




Attach camera here

Output:

Accessing the Raspberry Pi camera w/ Python + OpenCV
Raspberry Pi + camera

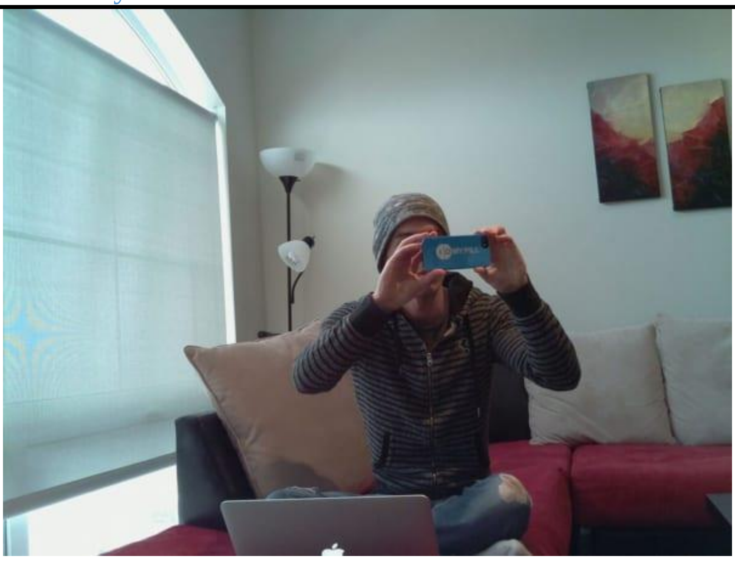


Image Captured by using Raspberry-pi Camera.

Software Requirement:

1. Raspbian OS
2. IDLE IDE

Hardware Requirement:

1. Raspberry Pi Board module.
2. Pi-Camera module.
3. Monitor

Conclusion:

1. Successfully done the connectivity of Raspberry-Pi /Beagle board with camera.
2. Successfully implemented the application to capture and store the image.

Assignment No.: 04

Problem Statements:
Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.

Objectives:

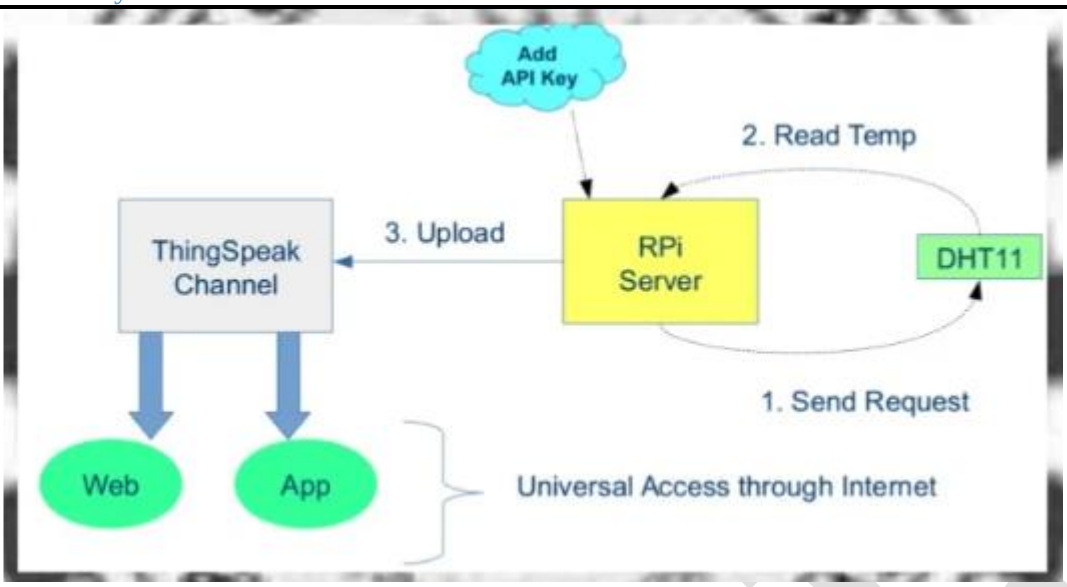
1. To understand creation of a small application to deployed on cloud.
2. To understand steps required to publish information on cloud using ThingSpeak

Theory:

1. Explain IOT platform?
2. Explain IOT cloud platform?

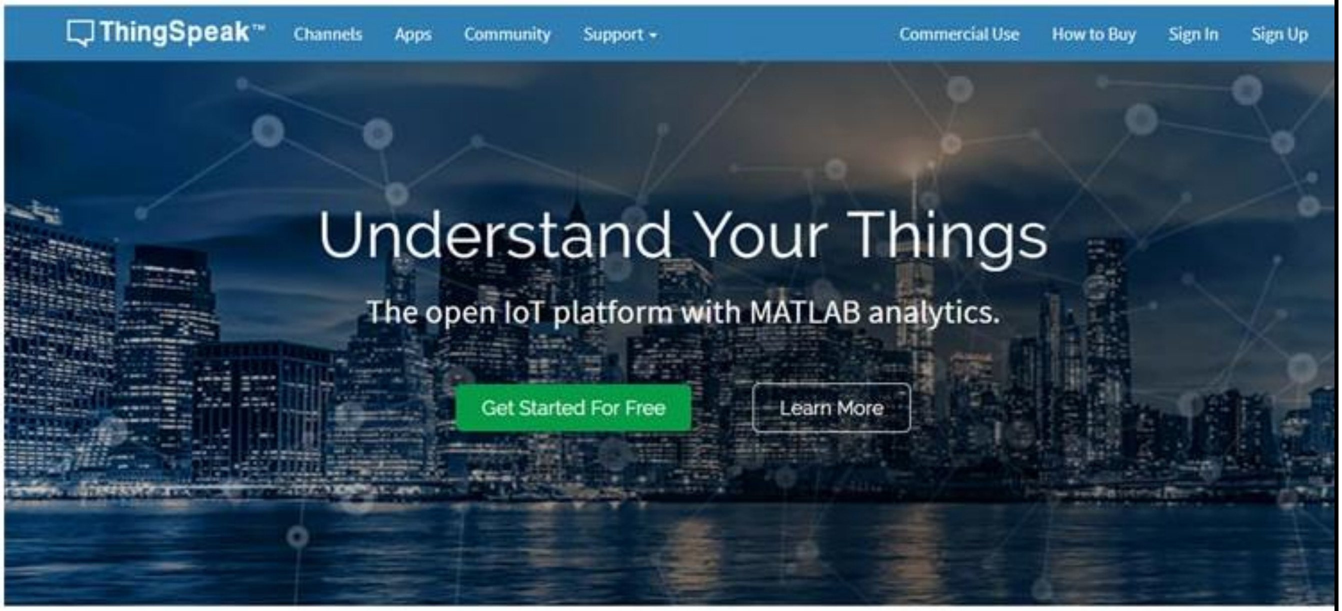
Algorithm/ Flow Chart:
In this Practical we are going to upload sensed temperature value on ThingSpeak.

```
import httplib,urllib
import time,Adafruit_DHT
key='8QPPQALZTUZUZ7IS'
while True:
   h,t=Adafruit_DHT.read_retry(11,4)
   print "temp:",t
   param=urllib.urlencode({'field1':t,'key':key})
   headers={"content-typZZe":"application/x-www-form-urlencoded","Accept":"text/plain"}
   conn=httplib.HTTPConnection("api.thingspeak.com:80")
   try:
      conn.request("POST","/update",param,headers)
      response=conn.getresponse()
      print response.status,response.reason
      data=response.read()
      conn.close()
   except:
      print "connection Failed"
```

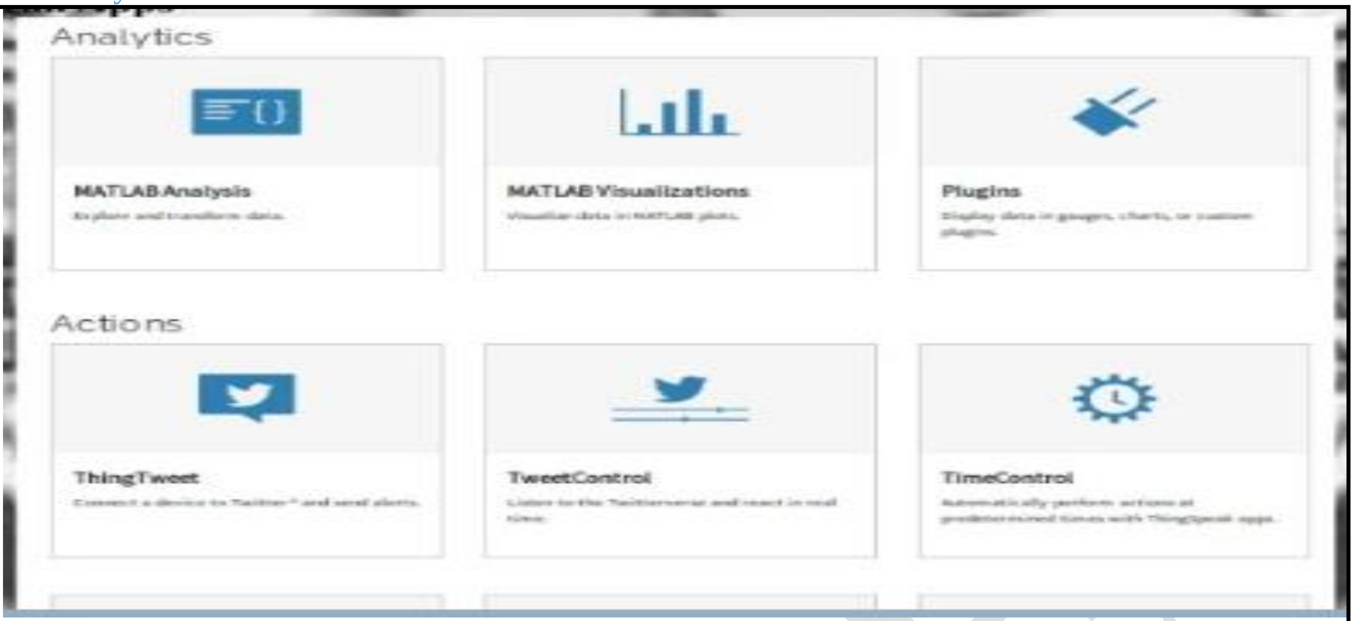


Steps for configuration of ThingSpeak:

3. **Open ThingSpeak Link**



4. **ThingSpeak APP**



Analytics

MATLAB Analysis
Explore and transform data.

MATLAB Visualizations
Visualize data in MATLAB plots.

Plugins
Display data in gauges, charts, or custom plugins.

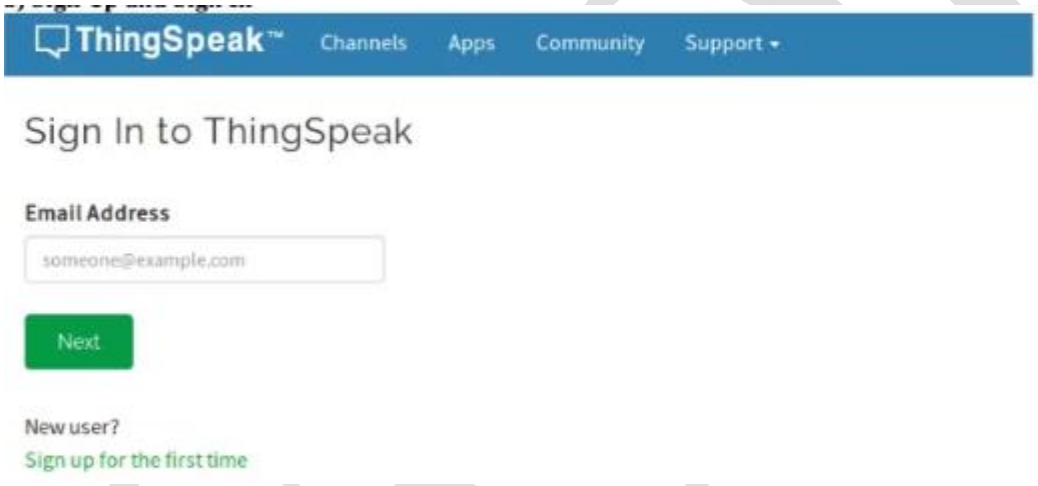
Actions

ThingTweet
Connect a device to Twitter® and send alerts.

TweetControl
Listen to the Twitterverse and react in real time.

TimeControl
Automatically perform actions at predetermined times with ThingSpeak apps.

3.Sign Up and Sign In

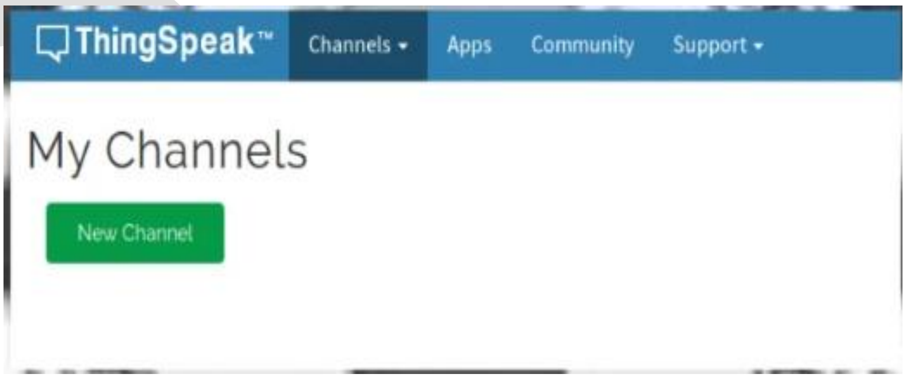


Sign In to ThingSpeak

Email Address
someone@example.com

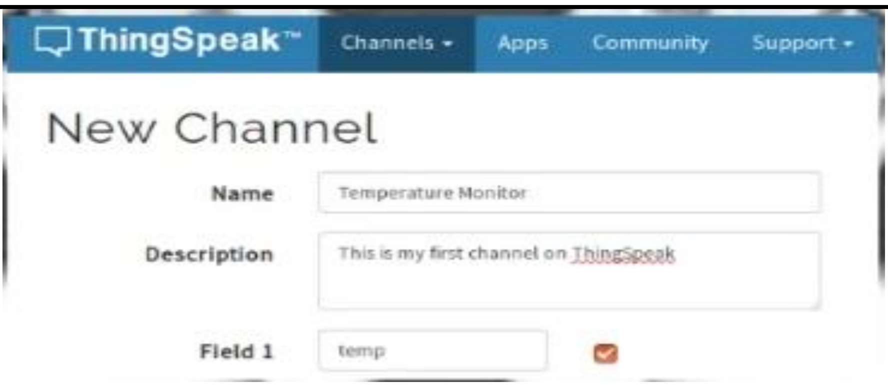
New user?
Sign up for the first time

4. After Sign In



My Channels

5. Create New Channel



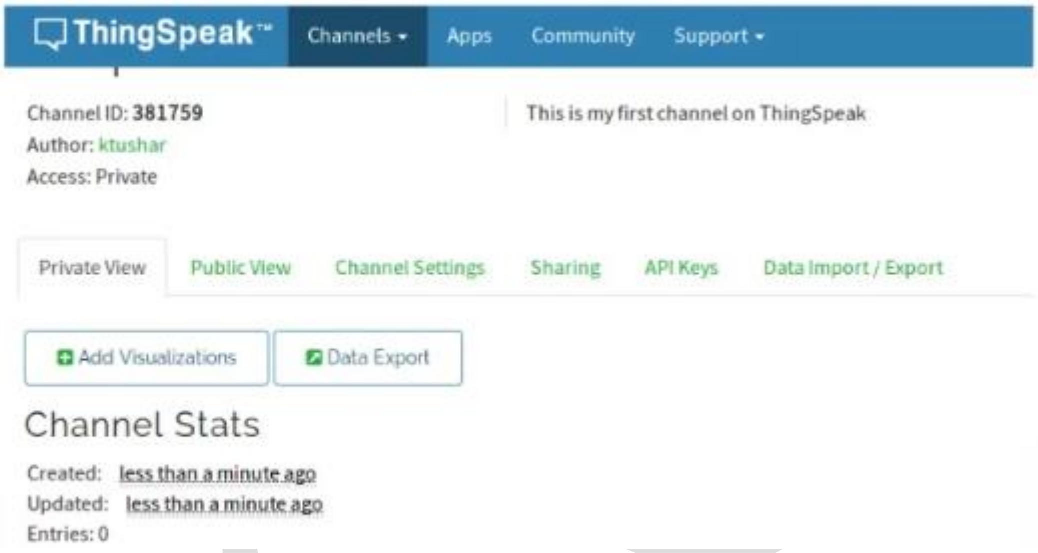
New Channel

Name: Temperature Monitor

Description: This is my first channel on ThingSpeak

Field 1: temp ☑

6. **First Time Channel View**



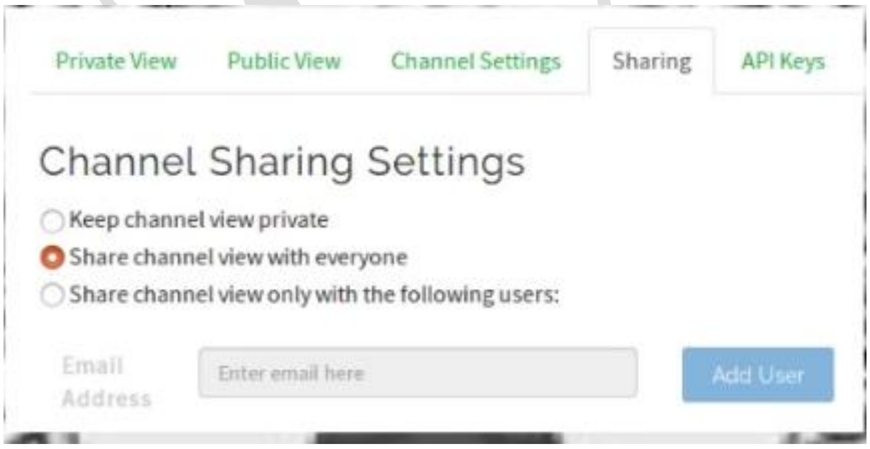
Channel ID: 381759
Author: ktushar
Access: Private

This is my first channel on ThingSpeak

Channel Stats

Created: less than a minute ago
Updated: less than a minute ago
Entries: 0

7. **Make Our Channel Public**

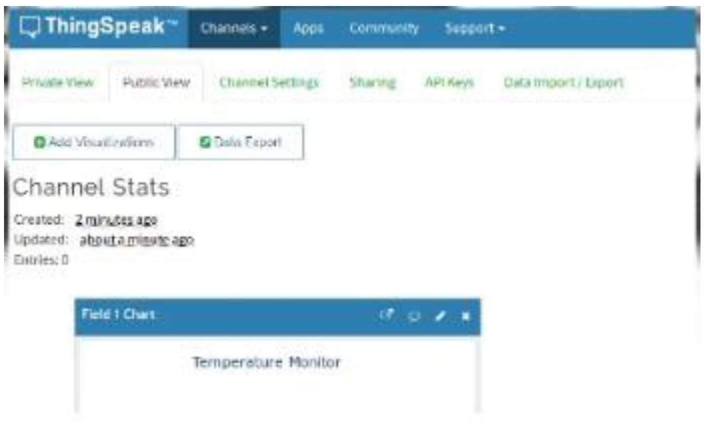


Channel Sharing Settings

○ Keep channel view private
● Share channel view with everyone
○ Share channel view only with the following users:

Email Address: Enter email here

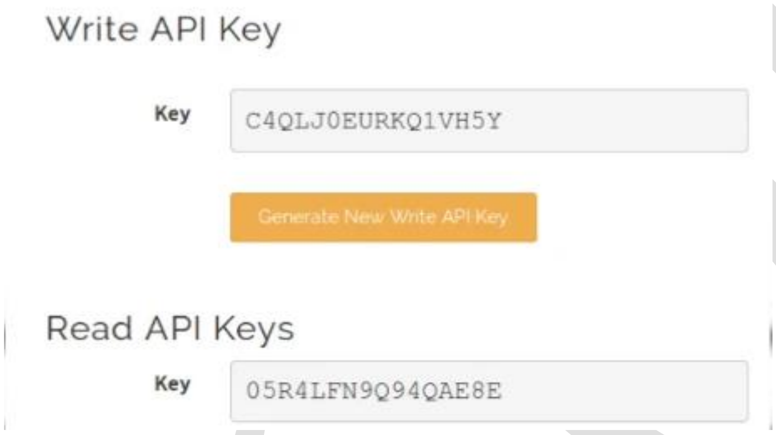
8. **Now public view will be available**



Channel Stats

Created: 2 minutes ago
Updated: about a minute ago
Entries: 0

9. **API key for R/W Operation**


Write API Key

Key C4QLJ0EURKQ1VH5Y

Read API Keys

Key 05R4LFN9Q94QAE8E

Input:
Follow all above steps to deploy application on cloud.

Output:
Publisher can see their information on cloud.

Software Requirement:

1. Raspbian OS-IDLE
2. Account at ThingSpeak

Hardware Requirement:

1. Raspberry Pi Board Module.
2. DHT-11 Sensor

Conclusion:
Successfully done deployment of application on cloud.

Part II: Elective I

Human Computer Interface

Assignment No.: 01

Problem Statements:
Design a paper prototype for selected Graphical User Interface.

Objectives:
1. To study how to design paper prototype for selected graphical user interface.
2. To study the paper prototyping design for different types of design purpose.
3. To study What is Paper prototyping?

Theory:

1. Explain paper prototype for selected graphical user interface.
2. Explain the design of different types of design purpose with all steps.
3. Explain the term paper prototyping.

Algorithm/ Flow Chart:

1. **Gather stationery** – pens, pencils, markers, paper, card, Post-its, scissors, tape, glue, rulers and suitable stencils. About Rs.100/$10/£10/€10 is typically enough to cover this. You can use graph paper to help guide ideas. Colored paper is great for representing buttons.
2. **Get building –** Just go ahead and see where you go. As you get your ideas down on paper, you can think about them more concretely. Later, you can get insights about improving them.
3. **Make one sketch per screen.**
4. **Move quickly** – If you spend too long making your prototype, you'll get attached to it. Don't waste time erasing: You want to get ideas down rather than revise them and potentially miss insights from the raw version.
5. **Remember what to test** – Build with that purpose in mind, but stay aware of other factors.
6. **Prototype for small screens first –**When you go **mobile-first**, you can prioritize your content better.
7. **Remember the users** – As you'll test your prototype against users' behaviors and needs, consider their expectations as you build.

Conclusion: Thus, we have studied designing of a paper prototype for selected Graphical User Interface

Assignment No.: 02

Problem Statements: Implement GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.

Objectives:

1. To study the design and implementation of GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.
2.

Theory:

1. Explain what is GOMS (Goals, Operators, Methods and Selection rules.
2. Explain what is modeling technique to model users behavior.

Algorithm/ Flow Chart:

A GOMS analysis of a task follows the familiar top-down decomposition approach. The model is developed top-down from the most general user goal to more specific subgoals, with primitive operators finally at the bottom. The methods for the goals at each level are dealt with before going down to a lower level. The recipe presented here thus follows a top-down, breadth-first expansion of methods.

In overview, start by describing a method for accomplishing a top-level goal in terms of high-level operators. Then choose one of the high-level operators, replace it with an Accomplish_goal operator for the corresponding goal, and then write a method for accomplishing that goal in terms of lower-level operators. Repeat with the other level operators. Then descend a level of analysis, and repeat the process for the lower-level operators. Continue until the methods have arrived at enough detail to suit the design needs, or until the methods are expressed in terms of primitive operators. So, as the analysis proceeds, high-level operators are replaced by goals to be accomplished by methods that involve lower-level operators.

It is important to perform the analysis breadth-first, rather than depth-first. By considering all of the methods that are at the same level of the hierarchy before getting more specific, similar methods are more likely to be noticed, which is critical to capturing the procedural consistency of the user interface.

Step 1: Choose the top-level user's goals

The top-level user's goals are the first goals that will be expanded in the top-down decomposition. It is worthwhile to make the topmost goal, and the first level of subgoals, very high-level to capture any important relationships within the set of tasks that the system is supposed to address. An example for a text editor is revise document, while a lower-level one would be delete text. Starting with a set of goals at too low a level entails a risk of missing the methods involved in going from one type of task to another. As an example of very high-level goals, consider the goal of produce document in the sense of ”publishing” - getting a document actually distributed to other people. This will involve first creating it, then revising it, and then getting the final printed version of it. In an environment that includes a mixture of ordinary and desktop publishing facilities, there may be some important subtasks that have to be done in going from one to the other of the major tasks, such as taking a document out of an ordinary text editor and loading it into a page-layout editor, or combining the results of a text and a graphics editor. If only one of these applications is under design, say the page-layout editor, and the analysis start only with goals that correspond to page-layout functions, the analysis may miss what the user has to do to integrate the use of the page-layout editor in the rest of the environment. As a lower-level example, many Macintosh applications combine deleting and inserting text in an especially convenient way. The goal of change word has a method of its own; i.e., double click on the word and then type the new word. If the analysis starts with revise document it is possible to see that one kind of revision is changing a piece of text to another, and so this especially handy method might well be noticed in the analysis. But if the analysis starts with goals like insert text and delete text the decision has already been made about how revisions will be done, and so it is more likely to miss a case where a natural goal for the user has been well-mapped onto the software directly, instead of going through the usual functions.

Step 2. Write the Top-Level Method Assuming a Unit-Task Control Structure

Unless there is reason to believe otherwise, assume that the overall task has a unit-task type of control structure. This means that the user will accomplish the topmost goal (the overall task) by doing a series of smaller tasks one after the other. The smaller tasks correspond to the set of top-level goals chosen in Step 1. For a system such as a text editor, this means that the topmost goal of edit document will be accomplished by a unit-task method similar to that described by Card, Moran, and Newell, (1983). One way to describe this type of method in GOMSL is as follows:

```
Method_for_goal: Edit Document
Step. Store First under <current_task_name>.
Step Check_for_done.
Decide: If <current_task_name> is None, Then
Delete<current_task> ; Delete <current_task_name>;
Return_with_goal_accomplished.
Step. Get_task_item_whose Name is<current_task_name> and_store_under <current_task>.
Step. Accomplish_goal: Perform Unit_task.
Step. Store Next of <current_task>under<current_task_name> ;
Goto Check_for_done.
```

The goal of performing the unit task typically is accomplished via a selection rule set, which dispatches control to the appropriate method for the unit task type, such as:

```
Selection_rules_for_goal: Perform Unit_task
If Type of <current_task> is move,
Then Accomplish_goal: Move Text.
If Type of <current_task> is delete, Then Accomplish_goal: Erase Text.
If Type of is copy, Then Accomplish_goal: Copy Text.
//... etc. ...
Return_with_goal_accomplished.
```

This type of control structure is common enough that the above method and selection rule set can be used as a template for getting the GOMS model started. The remaining methods in the analysis will then consist of the specific methods for these subgoals, similar to those described in the extended example below. In this example the task type maps directly to a goal whose name is a near-synonym of the type, but this is not always the case. A good exercise is to consider the typical VCR, which has at least three modes for recording a broadcast program; the selection rule for choosing the recording method consists not of tests for a simple task types like "one button recording", but rather the conditions under which each mode can or should be applied. For example, if the user is present at the beginning of the program, but will not be present at the end, and the length of the program is known, then the one-button recording method should be used.

Step 3. Recursively Expand the Method Hierarchy

This step consists of writing a method for each goal in terms of high-level operators, and then replacing the high-level operators with another goal/method set, until the analysis has worked down to the finest grain size desired. First, draft a method to accomplish each of the current goals. Simply list the series of steps the user has to do. Each step should be a single natural unit of activity; heuristically, this is just an answer to the question "how would a user describe how to do this?" Make the steps as general and highlevel as possible for the current level of analysis. A heuristic is to consider how a user would describe it in response to the instruction "don't tell me the details yet." Define new high-level operators, and bypass complex psychological processes as needed. Make a note of the analyst-defined operators and task description information as it is developed. Make simplifying assumptions as needed, such as deferring the consideration of possible shortcuts that experienced users might use. Make a note of these assumptions in comments in the method. If there is more than one method for accomplishing the goal, draft each method and then draft the selection rule set for the goal. A recommendation: defer consideration of minor alternative methods until later; especially for alternative "shortcut" methods. After drafting all of the methods at the current level, examine them one at time. If all of the operators in a method are primitives, then this is the final level of analysis of the method, and nothing further needs to be done with this method. If some of the operators are high-level, non-primitive operators, examine each one and decide whether to provide a method for performing it. The basis for the decision is whether additional detail is needed for design purposes. For example, early in the design of a specialized text-entry device, it might not be decided whether the system will have a mouse or cursor keys. Thus it will not be possible to describe cursor movement and object selection below the level of high-level operators. In general, it is a good idea to expand as many high-level operators as possible into primitives at the level of keystrokes, because many important design problems, such as a lack of consistent methods, will show up mainly at this level of detail. Also, the time estimates are clearest and most meaningful at this level. For each operator to be expanded, rewrite that step in the method (and in all other methods using the operator) to replace the operator with an Accomplish_goal operator for the corresponding goal.

For example, suppose the current method for copying selected text is:
Method_for_goal: Copy Selection
Step 1. Select Text.
Step 2. Issue Command using Copy.
Step 3. Return_with_goal_accomplished.

To descend a level of analysis for the Step 1 operator Select Text, rewrite the method as:

Method_for_goal: Copy Selection
Step 1. Accomplish_goal: Select Text.
Step 2. Issue Command using Copy.
Step 3. Return_with_goal_accomplished.
Then provide a method for the goal of selecting the text. This process should be repeated until all of the methods consist only of operators that are either primitive operators, or higher-level operators that will not be expanded.

Step 4. Document and Check the Analysis
After the methods and auxiliary information has been written out to produce the complete GOMSL model, list the any analyst-defined operators used, along with a brief description of each one, and the assumptions and judgment calls made during the analysis. Then, choose some representative task instances, and check on the accuracy of the model either by hand or with the GLEAN tool, to verify that the methods generate the correct sequence of overt actions, and correct and recheck if necessary. Examine the judgment calls and assumptions made during the analysis to determine whether the conclusions about design quality and the performance estimates would change radically if the judgments or assumptions were made differently. This sensitivity analysis will be very important if two designs are being compared that involved different judgments or assumptions; less important if these were the same in the two designs. It may be desirable to develop alternate GOMS models to capture the effects of different judgment calls to systematically evaluate whether they have important impacts on the design.

Conclusion: Thus we have studied Implementation of GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.

Assignment No.: 03

Problem Statements:
Design a User Interface in Python.

Objectives:

1. To study and design user interface in python
2. To study how user interface can design in python.

Theory:

1. What is User Interface?
2. What is Python?

Algorithm/ Flow Chart:

A graphical user interface (GUI) is a desktop interface that allows you to communicate with computers. They carry out various activities on desktop computers, laptops, and other mobile devices. Text-Editors and other graphical user interface applications build, read, download, and erase various types of files. You can also play games such as Sudoku, Chess, and Solitaire through these apps. Google Chrome, Firefox, and Microsoft Edge are examples of graphical user interface (GUI) Internet browsers.

Python has a variety of libraries, but these four stand out, especially in terms of GUI.

- Tkinter
- Kivy
- Python QT
- wxPython

Tkinter is the first option for a lot of learners and developers because it is quick and convenient to use. Tkinter is a Python library that can be used to construct basic graphical user interface (GUI) applications. In Python, it is the most widely used module for GUI applications.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Conclusion: Thus, we have studied designing of a User Interface in Python.

Assignment No.: 04

Problem Statements:
To redesign existing Graphical User Interface with screen complexity.

Objectives:
1. To Study and redesign existing Graphical User Interface with screen complexity.
2. To identify and Implementation/redesign existing Graphical User Interface with screen complexity.

Theory:
1.Explain Graphical User interface
2.Explain Screen complexity.

Algorithm/ Flow Chart:

1. To be able to establish a foundation for the design of a GUI different design methods can be applied to the process.
2. One prominent group of methods belongs to the user-centered design process . In this design process the focus lies on the users of said design, the goal is to create highly usable and accessible products by involving the user throughout the process.
3. The key for this process to work is iteration, to take the user's thoughts and opinions into consideration, revaluating the steps of the process based on the user's feedback and present the results for the users to evaluate again.
4. A general user-centered design process consists of four separate phases. The goal of the first phase is to try to understand in which context the users use the system. This then leads to the second phase in which the designer identifies and specifies the user's requirements. This initiates the third phase and the design work of the process in which the development of the solution starts, based on the previous established requirements from phase two. The fourth and final phase is the evaluation phase where the design outcome is evaluated against and with the users to determine how well the design fits the user's context and satisfies the user's needs.
5. Depending on the results of the evaluation, the phases of the process are repeated until a satisfactory result is achieved. The following sections will describe in more detail some of the methods that can be used in a user-centered design process for each phase, from the user's context to a completed design.

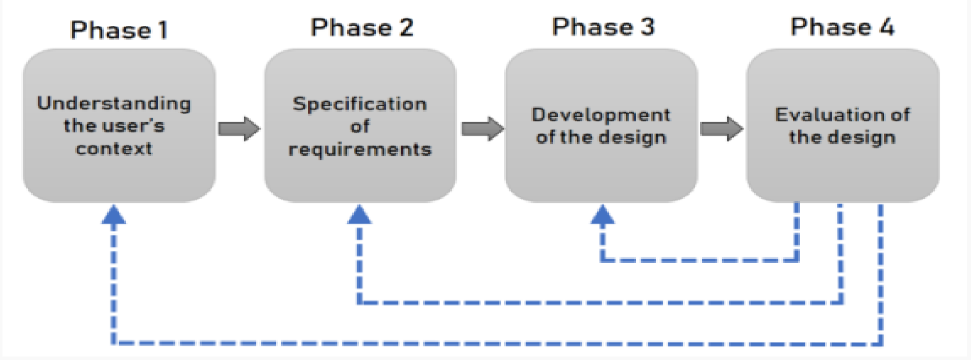

Phase 1
Understanding the user's context
Phase 2
Specification of requirements
Phase 3
Development of the design
Phase 4
Evaluation of the design

Fig. The user-centered redesign process

Conclusion: Thus, we have studied redesigning of Graphical User Interface.