# Title -

**MINI PROJECT**

**ON**

**MINI C++ Compiler**

# 

# 

# 

# 

# 

# 

# 

# 

# TABLE OF CONTENTS

[**Title -**](#_heading=h.if2a3ryp9g3t) **1**

[**TABLE OF CONTENTS**](#_heading=h.ofxd7jlz9rpl) **2**

[**LIST OF FIGURES**](#_heading=h.bdg9vmemg6yo) **3**

[**INTRODUCTION:**](#_heading=h.44sinio) **4**

[1.1 Problem Statement:](#_heading=h.lcaziler3cgi) 4

[1.2 Project Description:](#_heading=h.mx0w5su8wjwd) 4

[**REQUIREMENTS**](#_heading=h.3j2qqm3) **7**

[2.1 Software Used :](#_heading=h.1y810tw) 7

[2.2 Algorithm:](#_heading=h.loffvbm9shr) 7

[2.3 Flowchart:](#_heading=h.fmyms5l1v4q) 9

[**TEST PLAN AND TEST CASES**](#_heading=h.v9kswkwpjkeo) **10**

[3.1 Specifications](#_heading=h.2rrc94n235cc) 10

[3.2 Test Files:](#_heading=h.mfb0ie39jhhl) 10

[**REFERENCES:**](#_heading=h.kacyhlrsuae) **12**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | LIST OF FIGURES  |  |  | | --- | --- | | Fig. 1: Flow of the program | 9 | |  |  | |  |  | |  |  | |
|  |  |

**CHAPTER 1:**

# 

# INTRODUCTION:

## **1.1 Problem Statement:**

· Construct a mini C++ type compiler.

· It should be able to strictly identify only C++ code.

· It should report an error And any C code which is acceptable in C++.

## **1.2 Project Description:**

A compiler is a computer program that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code. The C and C++ programming languages are closely related. C++ grew out of C, as it was designed to be source-and-link compatible with C. C++ was based on C and retains a great deal of the functionality. The C++ language provides mechanisms for mixing code that is compiled by compatible C and C++ compilers in the same program. As a matter of fact, C++ can run most of C code while C cannot run most C++ code.

The purpose of compatibility with C is so that C++ programs can have convenient access to the billions (trillions?) of lines of existing C code in the world.

Although, C and C++ code are almost compatible but there are still many incompatibilities or conflicts between them. The conflicts can be of two types:

1. Incompatible C feature - valid as C code but not as C++ code.

2. Incompatible C++ feature - valid as C++ code but not as C code.

In this project we focus on a different domain. Compatible C/C++ features i.e. features of C code that are valid in C++.

We aim at detecting such snippets of code in our input program and will give an error if a C code is detected, whilst if no C code could be detected then we will compile it for minor errors, i.e. a mini compiler strictly for C++.

An example of C code valid in C++ code:

|  |
| --- |
| #include <iostream>  #include <cstdio>  #include <cmath>    using namespace std;    int main()  {  int x;    // C++ style IO  cout << “Hello World, Enter a number: ” << endl;  cin >> x;    /\* C Style IO \*/  printf(“Hello World, Enter a number: ”);  scanf(“%d”, &x);    /\* A function call invalid in C++ \*  \* if not including C header files. \*/  x = sqrt(4);    return 0;  } |

In addition to detect C code inside C++ code, our Mini C++ compiler will also be able to report following errors to the user:

· Invalid variable name.

· Invalid basic arithmetic expression.

· Syntax error in While loop.

· Syntax errors in For loop.

· Syntax errors in If-Then-Else.

**CHAPTER 2:**

# REQUIREMENTS

## **2.1 Software Used :**

The following two tools are used to implement the project:

1. YACC - produces a parser

2. LEX - generates lexical analyzers

3. GCC – gnu C Compiler

4. G++ - gnu C++ Compiler

## **2.2 Algorithm:**

The project is implemented in the following steps: -

1. Read the given Input.

2. Tokenize the input using Lex rules.

3. Parse using Yacc rules.

4. Run the Algorithm described below.

Algorithm:-

Goal: detect a c code that is generally successfully compiled by a C++ compiler and accept small C++ codes.

Steps:

1. Detect for header files (generally all c codes have **.h** header files)

2. Detect C language functions and keywords that are compatible with C++ compilers.

3. If(c code detected )

Then return ERROR and STOP.

Else goto step 4.

4. Check for error like

o Invalid variable name.

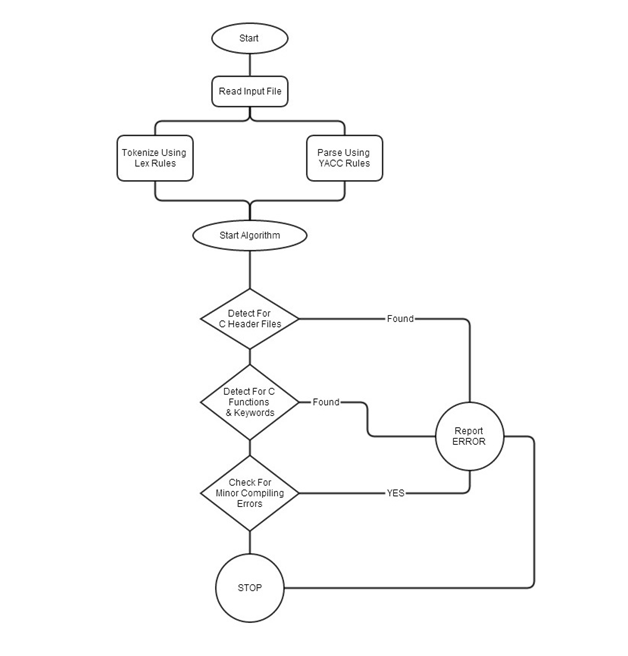
o Invalid basic arithmetic expression.

o Syntax error in While loop.

o Syntax errors in For loop.

o Syntax errors in If-Then-Else.

## 2.3 Flowchart:



## 

Fig 1. Flow of the Program

CHAPTER 3:

# TEST PLAN AND TEST CASES

## 3.1 Specifications

The Compiler will be able to check the following errors:

* Invalid basic arithmetic expression.
* Invalid variable name.
* Syntax errors in If-Then-Else.
* Syntax errors in For loop.
* Syntax error in While loop.

## 3.2 Test Files:

Test File for syntax error for “ if ” loop:

|  |
| --- |
|  |

fig 3.2.1: Test program for checking syntax for “if” loop

Test file for invalid variable:

|  |
| --- |
|  |

fig 3.2.2 Test program for invalid variable error

Test file for while loop:

|  |
| --- |
|  |

fig 3.2.3 Test program for while loop

Test File for Invalid Arithmetic Operators

|  |
| --- |
|  |

fig 3.2.4 Test program for invalid arithmetic operators

# 

# 

# 

# 

# 

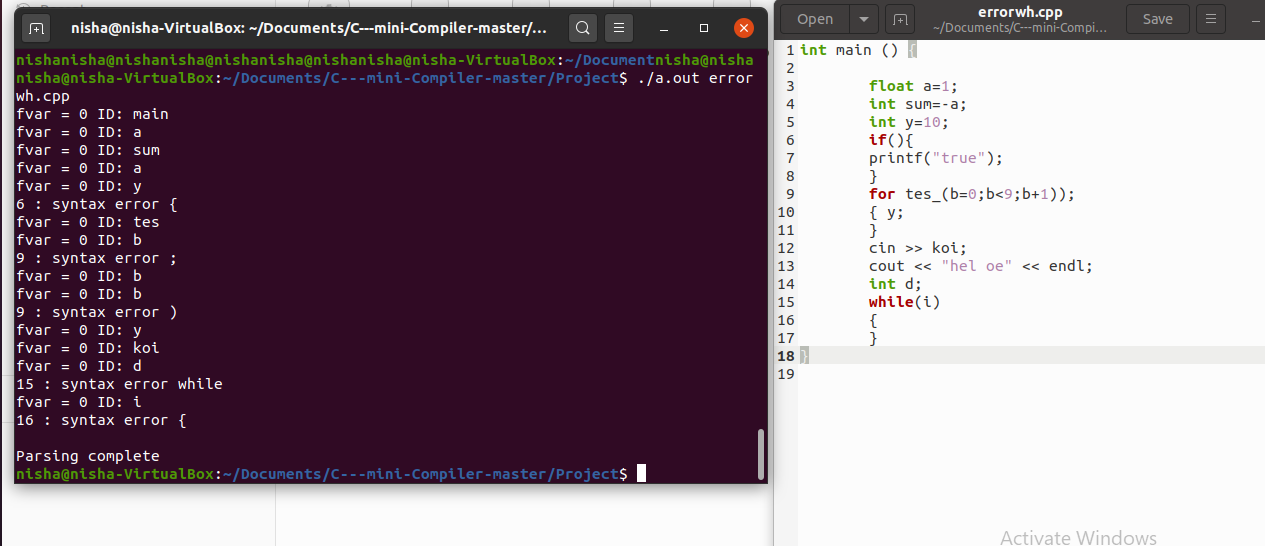
# RESULTS

# 

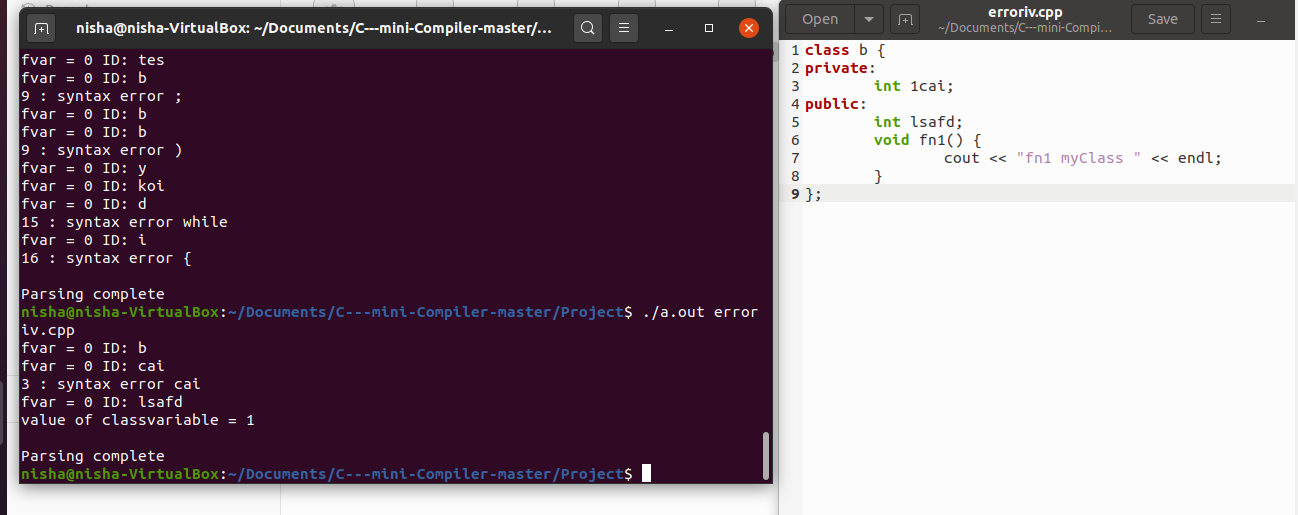
# 

# 

Output for error in arithmetic operator



Output for while loop



# 

# Output for invalid variable

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# REFERENCES:

1. http://epaperpress.com/lexandyacc/download/LexAndYaccTutorial.pdf - Lex and Yacc Tutorial by Tom Nieman

2. Compilers – Principles, Techniques and Tools By Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman

3. https://www.tutorialspoint.com/compiler\_design/compiler\_design\_semantic\_analysis .htm - Compiler Design Semantic Analysis by TutorialsPoint