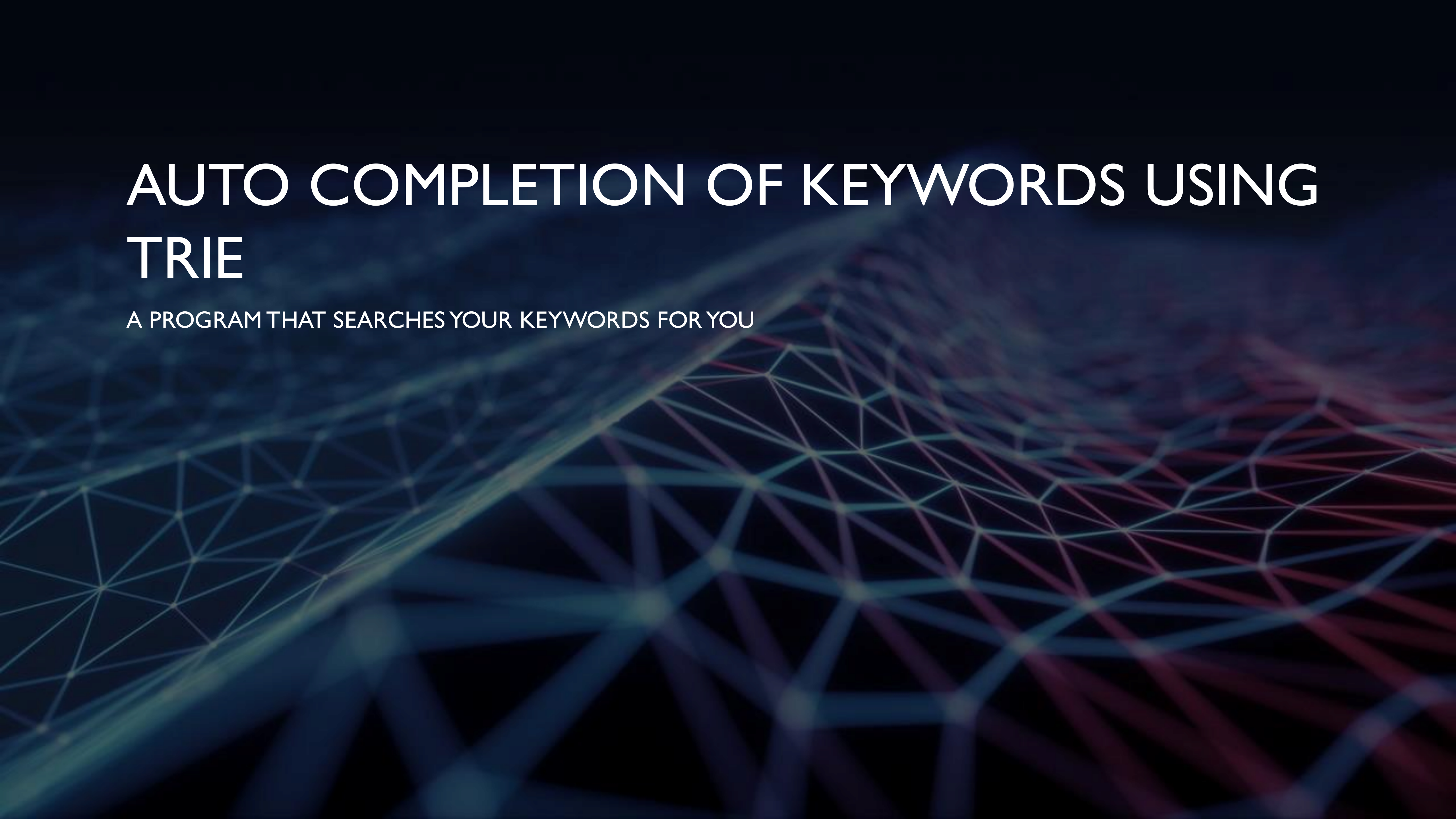


AUTO COMPLETION OF KEYWORDS USING TRIE

A PROGRAM THAT SEARCHES YOUR KEYWORDS FOR YOU

INDEX

TEAM

Neel Choksi	19BCE0990
Adwaidh Prakasan	19BCI0256
Gargi Lohia	19BCB0049
Pratham Sahay	19BCI0012

AIM AND OBJECTIVE

Through this project we aim to realize a program that successfully offers keyword recommendation for people new to programming languages like C++.Java and Python.

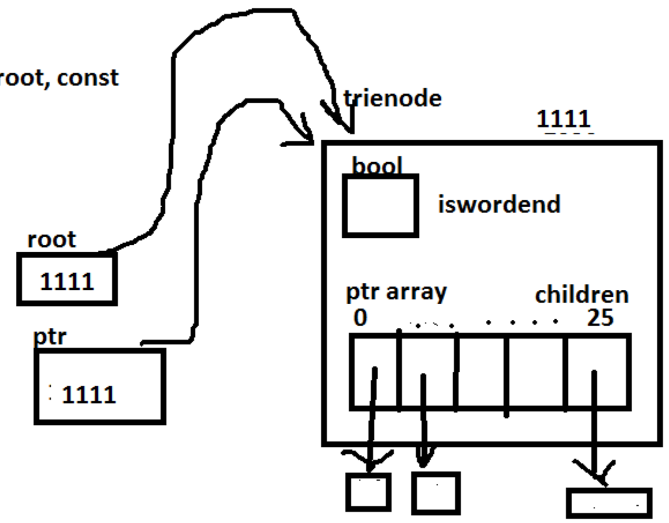
Textbooks that teach programming do help students in getting thorough with the techniques and logic to be used in programming. However, it is very common to get confused with the keywords and forget exactly what the keyword looks like. This program aims to solve this problem by displaying keywords that start with the first three letters entered by the user. To efficiently run the program, we aim to use the Trie data structure to have minimum space and time complexity.

INTRODUCTION

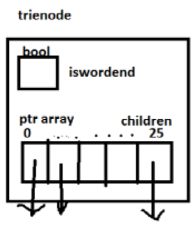
This program is designed to recommend keywords based on first three words entered by the user. It supports the keywords from FOUR programming languages- Python, C , C++ and Java.

The program first manually creates the tries along with keywords in it(keywords that are given in the program) and the uses the same tries to search for words when the user enters their data. The reason why this is better than the array implementation is because of its space and tyime complexity and easy insertion and search operation.

```
void inserttrie(struct trienode *root, const
string key)
{
        struct trienode *ptr = root;
```



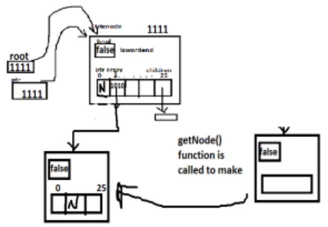
```
struct trienode
{
        struct trienode *children[ALPHABET_SIZE];
        bool iswordend;
};
```



i=0 and key[0]=b

index=98-97 =1

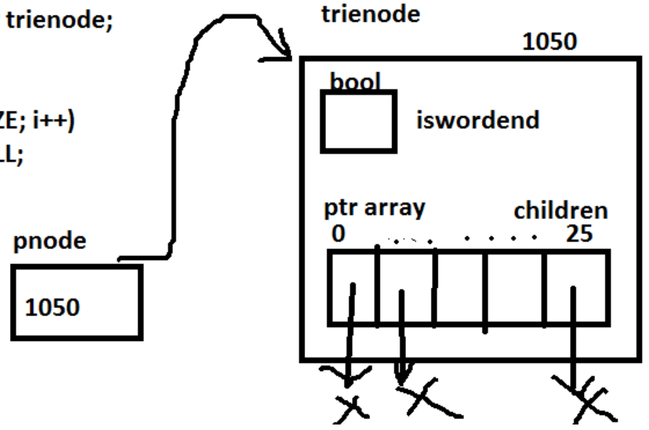
children[1]



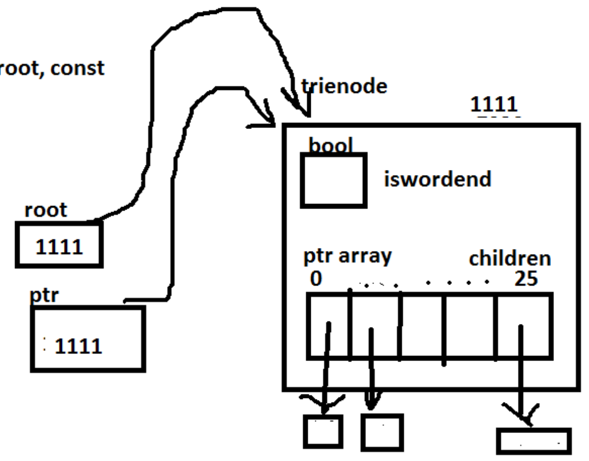
PROCESS MODULES

The data structure Trie, also called digital tree or prefix tree has been made use of in this program. It consists of a root 'node'that holds the first letter inserted into the memory. It is connected to the next node which holdsthe next letter and the first node is called the parent node while the latter is called the 'child'. Trie finds wide application in the field of computer science. It is used in place of Hash tables as it provides O(n) time complexity (were n is the length of the search string). It has several advantages over typical binary trees and is generally used as dictionaries to search and store words.

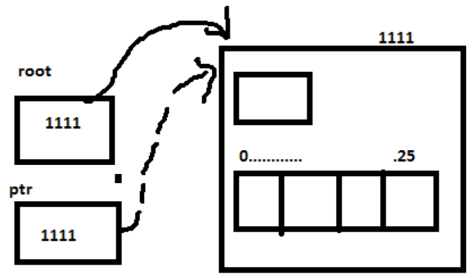
```
struct trienode *getNode(void)
{
        struct trienode *pnode = new trienode;
        pnode->iswordend = false;

        for (int i = 0; i < ALPHABET_SIZE; i++)
                pnode->children[i] = NULL;

        return pnode;
}
```



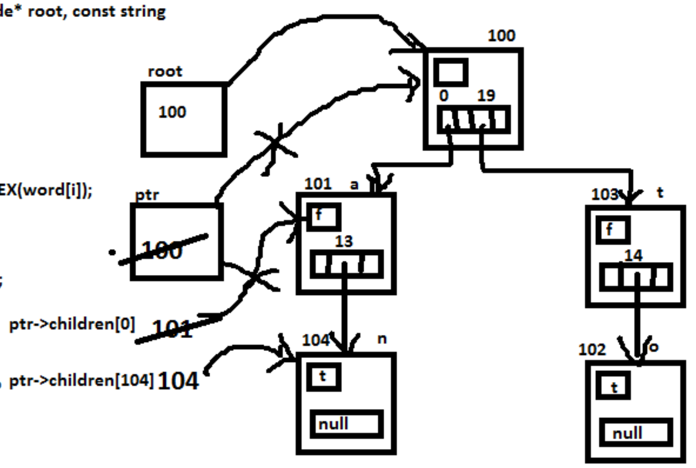
```
void inserttrie(struct trienode *root, const
string key)
{
        struct trienode *ptr = root;
```



```
bool lastnode(struct trienode* root)
{
   struct trienode* ptr=root;
         for (int i = 0; i < ALPHABET_SIZE; i++)
                  if (ptr->children[i])
                           return 0;
         return 1;

}
```



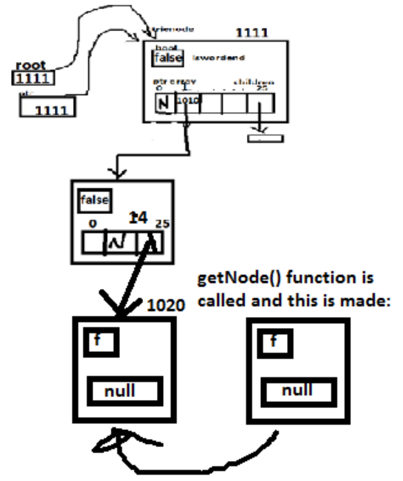
```
bool searchtrie(struct trienode* root, const string
word)
{
   int size=word.length();
   struct trienode* ptr;
   ptr=root;
   for(int i=0; i<size;i++)
   {
      int index=CHAR_TO_INDEX(word[i]);

      if(!ptr->children[index])
         return false;
      ptr=ptr->children[index];
```



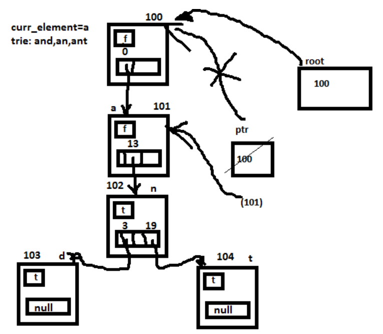
```
      //consider a trie: an, to
      and word: an
```

i=1 therfore key[1]=o
index =111-97=14

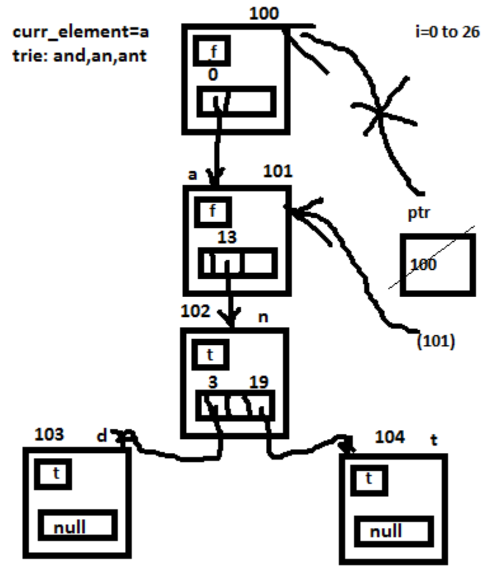
children[14]



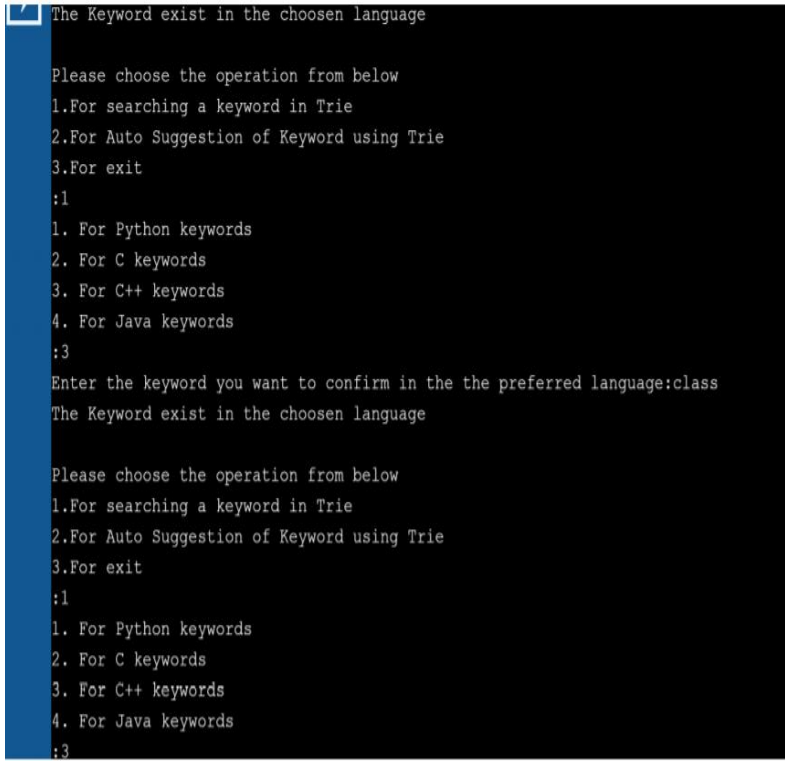
getNode() function is called and this is made:



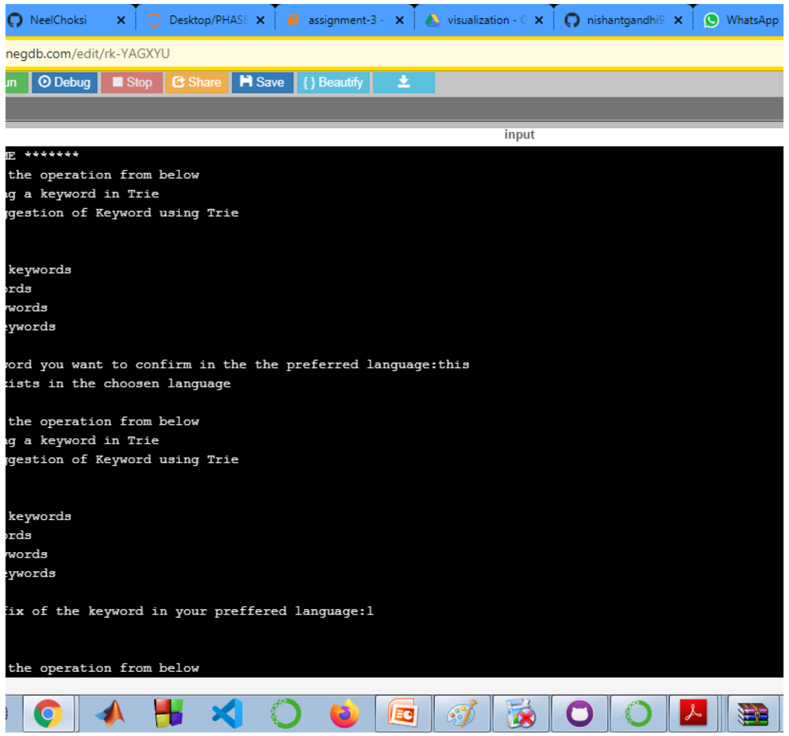
```
user =a   n=1

index =97-97=0


isword=false
islast=false
if(false&& false){ //not entered here}
if(!f)
{
        string prefix =a;
        autosuggestfunction(ptr,a)
        return 1;
}
```



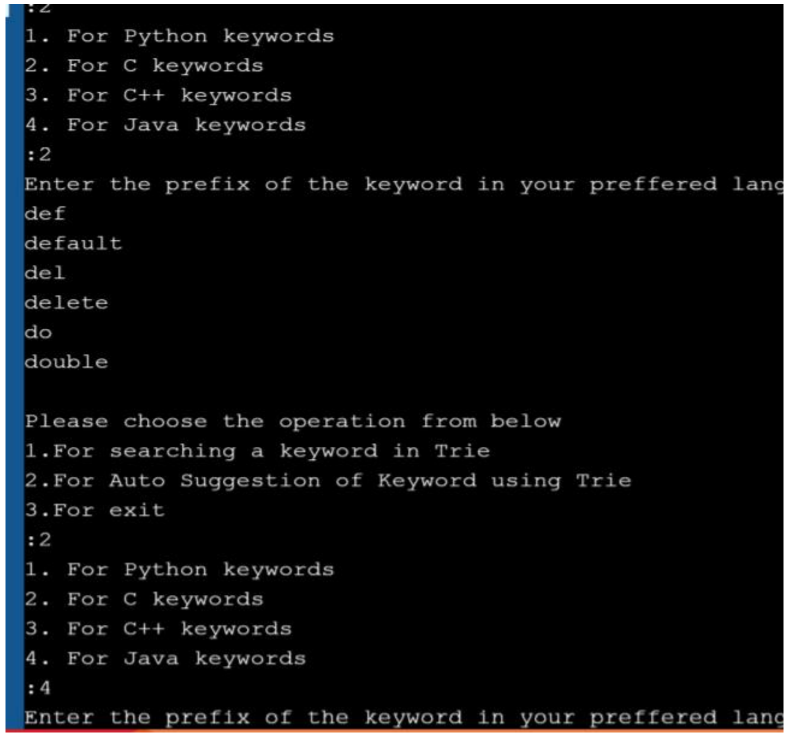
```
at      curr_element
i=13    a n                                102
        autosuggestfunction(ptr->children[13],an)
        ----------------------------------------------
        curr_element                              o/p: an

i=3      a n d
                                            103
          autosuggestfunction(ptr->children[3],and)
         ----------------------------------------------
         curr_element                             o/p: and
          a n

i=19     autosuggestfunction(ptr->children[19],ant)
                                                  o/p: ant
```



```
The Keyword exist in the choosen language

Please choose the operation from below
1.For searching a keyword in Trie
2.For Auto Suggestion of Keyword using Trie
3.For exit
:1
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:3
Enter the keyword you want to confirm in the the preferred language:class
The Keyword exist in the choosen language

Please choose the operation from below
1.For searching a keyword in Trie
2.For Auto Suggestion of Keyword using Trie
3.For exit
:1
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:3
```



```
E *******
 the operation from below
g a keyword in Trie
gestion of Keyword using Trie


 keywords
rds
words
ywords

ord you want to confirm in the the preferred language:this
ists in the choosen language

 the operation from below
g a keyword in Trie
gestion of Keyword using Trie


 keywords
rds
words
ywords

ix of the keyword in your preffered language:1


 the operation from below
```



```
:2
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:2
Enter the prefix of the keyword in your preffered lang
def
default
del
delete
do
double

Please choose the operation from below
1.For searching a keyword in Trie
2.For Auto Suggestion of Keyword using Trie
3.For exit
:2
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:4
Enter the prefix of the keyword in your preffered lang
```

OUTPUT SLIDES:

THESE SLIDES SHOW THE OUT PUT WHEREIN THE USER HAS ENTERED THE KEY DEF AND THE PROGRAM DISPLAYS KEYWORDS. THE OTHER TWO SLIDES SHOW SIMILAR FUNCTIONING OF THE PROGRAM.

CONCLUSION:

THROUGH THIS PROJECT WE LEARNT EFFICIENT USE OF TRIES AND HAVE COME TO UNDERSTAND THE NUANCES OF CODING IN C++. WE HAVE ALSO LEARNT HOW TO EFFICIENTLY USE MEMORY AND HEADER FILES TO STORE CERTAIN PROGRAM FUNCTIONS AND HAVE THE PROGRAM RUN SMOOTHLY WITHOUT IT TAKING A LOT OF SPACE.