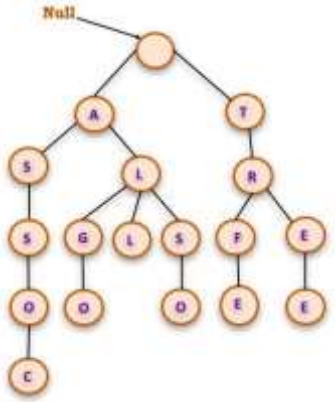



# AUTO COMPLETION OF KEYWORDS USING TRIE

A PROGRAM THAT SEARCHES YOUR KEYWORDS FOR YOU


VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# TEAM


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

# INDEX

# ACKNOWLEDGEMENT

The major part of this project comes from the wealth of knowledge given to us by the DSA faculty **Mrs.Parveen Sultana** . We thank the cooperation of the faculty in helping us decide our topic and in picking the appropriate data structure for us to materialise the idea of searching for keywords .

The huge number of websites on the internet that provide ideas for coding and explain this data structure have been very useful in helping us formulate the program that runs as we want it to.

# 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.

# ABSTRACT

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 time complexity and easy insertion and search operation.

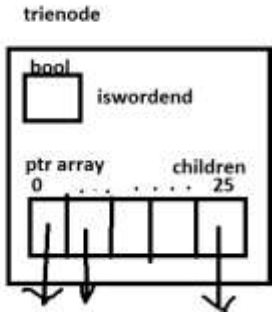
# PRINCIPLE AND METHODOLGY

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 holds the next letter and the first node is called the parent node while the latter is called the‘child’.

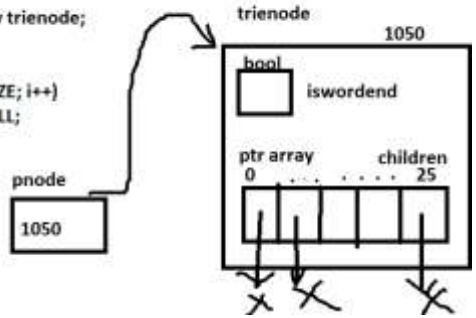
Trie finds wide application in the field of computer science. It is used in place of Hash tables as it provides O(n) time complexity (were n is the length of the search string). It has several advantages over typical binary trees and is generally used as dictionaries to search and store words.

# PROCESS MODULES

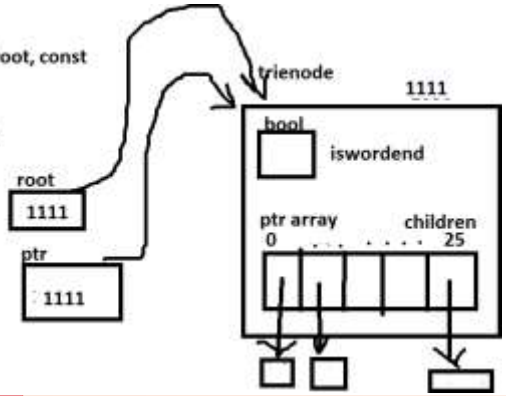
```
struct trienode
{
    struct trienode *children[ALPHABET_SIZE];
    bool iswordend;
};
```




```
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;
}
```



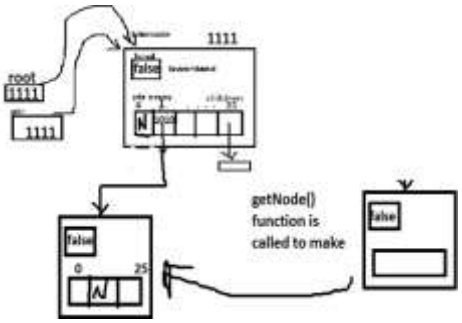

```
void inserttrie(struct trienode *root, const
string key)
{
    struct trienode *ptr = root;
```




i=0 and key[0]=b

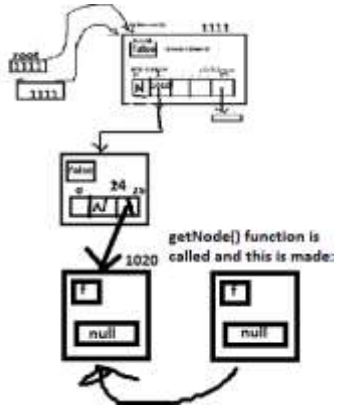
index=98-97 =1

children[1]



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

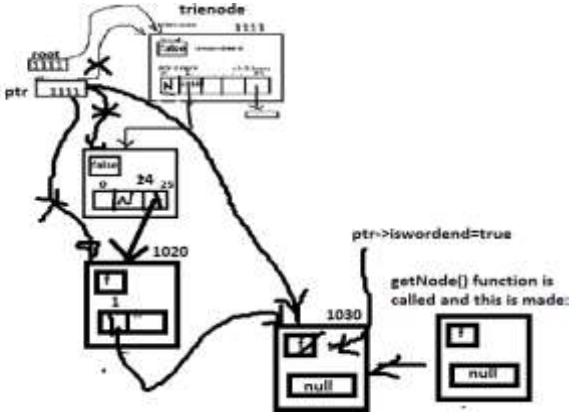
children[14]



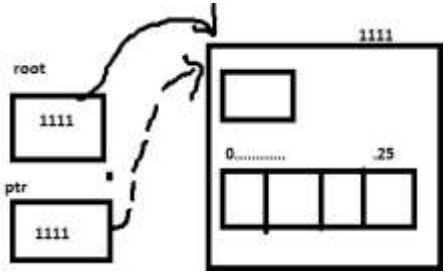

i=2 key[2]=b

index=98-97=1

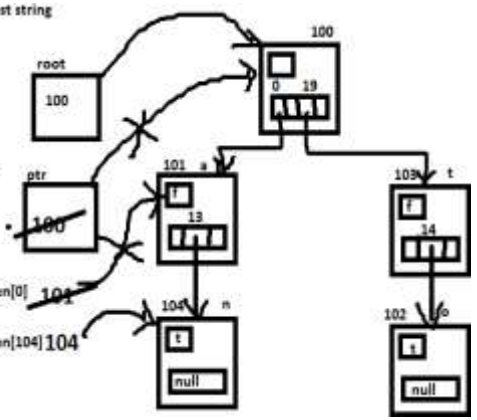
children[1]

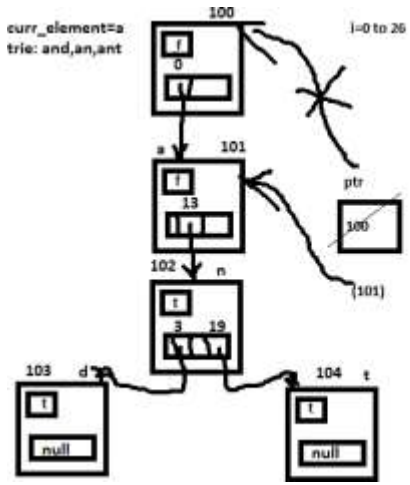



```
bool lastnode(struct trienode* root)
{
   struct trienode* ptr=root;
        for (int i = 0; i < ALPHABET_SIZE; i++)
                if (ptr->children[i])
                        return 0;
        return 1;

}
```




```
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
```






at

i=13

curr_element

a n 102

autosuggestfunction(ptr->children[13],an)

curr_element o/p: an

i=3

a n d 103

autosuggestfunction(ptr->children[3],and)

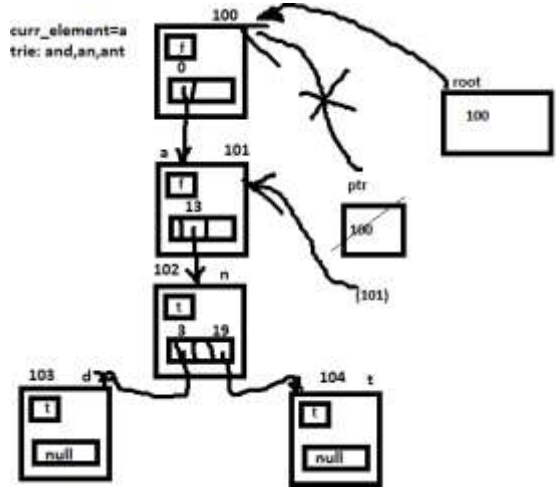
curr_element o/p: and

a n

i=19

autosuggestfunction(ptr->children[19],ant)

o/p: ant




```
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;
}
```

# CODE

The next few slides detail the code following the visualization of certain parts of the code that explain the working of those segments. It not only reveals when the functions are called ,but details what happens to the data structure at every such step

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
#include<vector>
using namespace std;
#define ALPHABET_SIZE (26)
#define CHAR_TO_INDEX(c) ((int)c - (int)'a')
struct trienode
{
struct trienode *children[ALPHABET_SIZE];
bool iswordend;
};
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, conststring key)
{
struct trienode *ptr = root;
for (int i = 0; i< key.length(); i++)
{
int index = CHAR_TO_INDEX(key[i]);
if (!ptr->children[index])
ptr->children[index] = getNode();
ptr = ptr->children[index];
}
ptr->iswordend = true;
}
```

```
 return pnode;
}
void inserttrie(struct trienode *root,
const string key)
{
struct trienode *ptr = root;
for (int i = 0; i< key.length(); i++)
{
int index = CHAR_TO_INDEX(key[i]);
if (!ptr->children[index])
ptr->children[index] = getNode();
ptr = ptr->children[index];
}
ptr->iswordend = true;
bool searchtrie(struct trienode* root,
const string word)
{
int
size=word.lengt
h();
struct trienode*
ptr;
ptr=root;
for(int i=0; i<size;i++)
{
int index=(int)word[i]-(int)'a';
```

```
if(!ptr->children[index])
return false;
ptr=ptr->children[index];
}
return(ptr!=NULL && ptr->iswordend);
}
bool lastnode(struct trienode* root)
{
for (int i = 0; i < ALPHABET_SIZE; i++)
if (root->children[i])
return 0;
return 1;
}
void autosuggestfunction(struct trienode* root, string
current_element)
{
if (root->iswordend)
{
cout << current_element;
```

```
cout << endl;
}
if (lastnode(root))
return;
for (int i = 0; i < ALPHABET_SIZE; i++)
{
if (root->children[i])
{
current_element.push_back(97 + i);
autosuggestfunction(root->children[i],
current_element);
current_element.pop_back();
}
}
}
int displayautosuggestion(trienode* root,const string
user)
{
struct trienode* ptr = root;
int i,n =user.length();
for (i= 0; i<n;i++)
{
```

```
cout << endl;
}
if (lastnode(root))
return;
for (int i = 0; i < ALPHABET_SIZE; i++)
{
if (root->children[i])
{
current_element.push_back(97 + i);
autosuggestfunction(root->children[i],
current_element);
current_element.pop_back();
}
}
}
int displayautosuggestion(trienode* root, const
string user)
{
struct trienode* ptr = root;
int i,n = user.length();
for (i= 0; i<n;i++)
{
int index = CHAR_TO_INDEX(user[i]);
if (!ptr->children[index])
return 0;
ptr = ptr->children[index];
```

```
autosuggestfunction(ptr, prefix);

return 1;

}

}

int main()

{

cout<<"******* WELCOME *******"<<endl;

struct trienode* root = getNode();

string keyword1[] = { "false", "class", "finally", "is", "return", "None", "continue",  "for", "lambda",  "try", "true", "def", "from", "nonlocal", "while", "and", "del",
        "global",  "not", "with", "as", "elif", "if", "or", "yield", "assert", "else", "import", "pass", "break",  "except",  "in", "raise" };
string keyword2[] = { "auto", "break", "case", "char", "const", "continue", "default", "do", "double", "else", "enum",  "extern",
        "float", "for", "goto", "if", "int", "long", "register", "return", "short", "signed", "sizeof", "static", "struct", "switch",
        "typedef", "union", "unsigned", "void", "volatile", "while" }; //C keywords

string keyword3[] = { "asm", "else", "new", "this", "auto", "enum", "operator", "throw", "bool", "explicit", "private", "true",
        "break", "export", "protected", "try", "case", "extern", "public", "typedef", "catch", "false", "register", "typeid", "char",
        "float", "typename", "class", "for", "return", "union", "const", "friend", "short", "unsigned", "goto", "signed", "using",
        "continue", "if", "sizeof", "virtual", "default", "inline", "static", "void", "delete", "int", "volatile", "do", "long", "struct",
        "double", "mutable", "switch", "while", "namespace", "template"}; //C++ keywo"rds

string keyword4[] = { "abstract", "assert", "boolean", "break", "byte", "case", "catch", "char", "class", "const", "continue",
        "default", "do", "double", "else",
        "enum", "extends", "final", "finally", "float", "for", "goto", "if", "implements", "import", "instanceof", "int", "interface",
        "long", "native", "new", "package",
        "private", "protected", "public", "return", "short", "static", "strictfp", "super", "switch", "synchronized", "this", "throw",
        "throws", "transient", "try",
        "void", "volatile", "while" }; //java keywords

int ch=1;

while(ch!=3)

{

cout<<"Please choose the operation from
below"<<endl;

cout<<"1.For searching a keyword in
Trie"<<endl;

cout<<"2.For Auto Suggestion of Keyword using
Trie"<<endl;

cout<<"3.For exit"<<endl;
```

```
cout<<":";
cin>>ch;
if(ch==1)
{
int var;
cout << "1. For Python keywords" << endl;
cout << "2. For C keywords" <<endl;
cout << "3. For C++ keywords" <<endl;
cout << "4. For Java keywords"<<endl;
cout<<":";
cin>>var;
switch (var)
{
case 1:
{
for ( int i = 0; i < sizeof (keyword1) /sizeof (keyword1[0]); i++)
{
inserttrie (root, keyword1[i]);
}
break;
}
case 2:
{
for ( int j = 0; j < sizeof (keyword2) / sizeof (keyword2[0]); j++)
{
inserttrie (root, keyword2[j]);
}
break;
}
case 3:
{
for ( int k = 0; k < sizeof (keyword3) / sizeof (keyword3[0]); k++)
{
```

```
inserttrie(root, keyword3[k]);
}
 break;
 }
 case 4:
 {
 for ( int l = 0; l < sizeof (keyword4) / sizeof
(keyword4[0]); l++)
 {
 inserttrie(root, keyword4[l]);
 }
 break;
 }
 }
 string user;
 cout<<"Enter the keyword you want to confirm in the the
preferred language:";
 cin>>user;
 int q=searchtrie(root,user);
 if(q==1)
 {
 cout<<"The Keyword exist in the choosen
language"<<endl;
 }
```

```
for ( int j = 0; j < sizeof (keyword2) / sizeof (keyword2[0]); j++)
{
inserttrie (root, keyword2[j]);
}
break;
}
case 3:
{
for ( int k = 0; k < sizeof (keyword3) / sizeof (keyword3[0]); k++)
{
inserttrie(root, keyword3[k]);
}
break;
}
case 4:
{
for ( int l = 0; l < sizeof (keyword4) / sizeof (keyword4[0]); l++)
{
inserttrie(root, keyword4[l]);
}
break;
}
}
string user;
cout<<"Enter the prefix of the keyword in your preffered language:";
```

```
cin>>user;
int check = displayautosuggestion(root, user);
if (check == -1)
cout << "No other keyword with this prefix"<<endl;
else if (check == 0)
cout << "No keyword with this prefix"<<endl;
}
else
{
break;
}
cout<<endl;
}
cout<<"**********THANK YOU**********";
return 0;
}
```

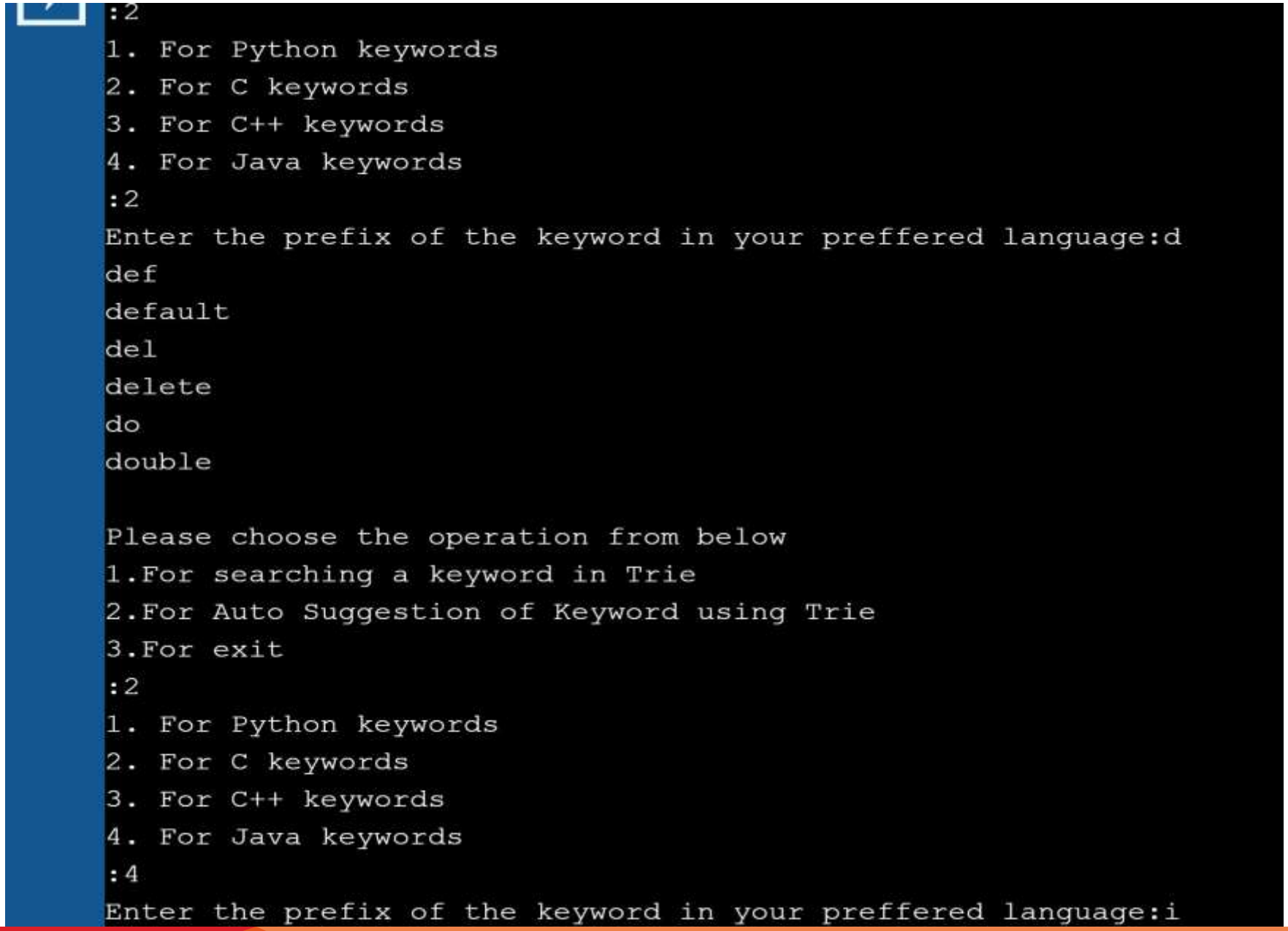
# OUTPUT

```
******* WELCOME *******
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
:4
Enter the keyword you want to confirm in the the preferred language:this
The Keyword exists 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
:2
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:3
Enter the prefix of the keyword in your preffered language:l
long
```

```
******* WELCOME *******
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
:1
Enter the keyword you want to confirm in the the preferred language:finally
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
:2
Enter the keyword you want to confirm in the the preferred language:extern
```

```
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
```

```
Enter the keyword you want to confirm in the the preferred language:print
The Keyword dosen't 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
:2
1. For Python keywords
2. For C keywords
3. For C++ keywords
4. For Java keywords
:1
Enter the prefix of the keyword in your preffered language:f
false
finally
float
for
friend
from

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
:2
Enter the prefix of the keyword in your preffered language:d
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 language:i

```
Enter the prefix of the keyword in your preffered language:i
if
implements
import
in
inline
instanceof
int
interface
is

Please choose the operation from below
1.For searching a keyword in Trie
2.For Auto Suggestion of Keyword using Trie
3.For exit
:3
**********THANK YOU**********

...Program finished with exit code 0
Press ENTER to exit console.
```

```
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
:3
Enter the prefix of the keyword in your preffered language:c
case
catch
char
class
const
continue

Please choose the operation from below
1.For searching a keyword in Trie
2.For Auto Suggestion of Keyword using Trie
3.For exit
:3
**********THANK YOU**********

...Program finished with exit code 0
Press ENTER to exit console.
```

# 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.

# BIBLOGRAPHY


https://en.wikipedia.org/wiki/Trie

https://www.geeksforgeeks.org/trie-insert-and-search/

https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/tutorial/