**HTML COMPILER USING PYTHON LANGUAGE**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**SOHA MUSKAAN SAYYAD(AP20110010602)**

**PURNIMA SINGH(AP20110010590)**

**GOWTHAMI GURRAM(AP20110010571)**

**MOUNICA NOTI(AP20110010575)**

**K.GAYATHRI(AP20110010568)**

**A.PALLAVI(AP20110010564)**

**K.MAMATHA(AP20110010595)**

**A picture containing text

Description automatically generated**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[December, 2022]**

1. **Overview of Source and target language**

**Description of source language :**

This project is basically the construction of the “HTML” compiler using python language. HTML is a language used to create web pages. It is now becoming the most useful language as full stack web development is in demand. We are designing using the most famous and user’s friendly language python. Python has many features for designing a good GUI application and has many inbuilt features. In this report we have mentioned all the keywords and the other tags and other features in the html language.

**Description of target language :**

The target language after compilation is Assembly Level Language.

1. **Conceptualization and Design of source language**

**2.1DESCRIPTION OF THE FEATURES AND TAGS:**

**2.1.1 HTML DOCUMENTS:**

1. **<!DOCTYPE html>:** this the header of all html documents and every document of html must start with the given header.
2. **<html> </html>:** all the html documents are started with the <html> and must be ended with </html>
3. **<body> </body>:** everything that we want to display on the screen of the website must be written in the body of the document.
4. **<p>** **</p>**: id used for the paragraphing purpose.
5. **<hr>**: it is a horizontal tag.
6. **<br>**: it is used for line breaks .

**2.1.2 HTML HEADINGS:**

1. **<h1></h1> <h2> <h3> <h4> <h5> <h6>**: HTML headings are used to make the website more meaningful and make the documents more structured. Html contains the headings from h1 to h6 . h1 is most important and h6 is least important.

**2.1.3 HTML FORMATTINGS:**

1. **<b>**: html bold text
2. **<strong>**: this is to define the important text.
3. **<i>**: this is used to change the style to italic text
4. **<abbr>**: this is used for abbreviation.
5. **<!-- -->**: this is used for commenting purposes.
6. **<u>**: underline tag

**2.1.4 HTML IMAGES:**

1. **<img>** : is to insert the image on the web.
2. **src** : is used in the <img> that describes the path of the image.
3. **alt** : is used for alternate text for the image

**2.1.5 HTML TABLES:**

1. **<table> </table>** :it is used for a table
2. **<tr> </tr>**: used to define the row in the table.
3. **<th> </th>**: used to define the header cell
4. **<td> </td>**: used to define the cell.
5. **<caption> </caption>**: used to define the captions in the table.

**2.1.6 HTML LISTS:**

1. **<ul>** : it used for unordered list
2. **<li> </li>**: each list starts with the given tag.
3. **<ol> </ol>**: it is used for ordered lists.
4. **<dl> </dl>**: defines the description of the list
5. **<dt></dt>**: defines terms of the description
6. **<dd></dd>**: describes the terms of the list

**2.2 BASIC SYNTAX OR THE MOST USED CODE FORMAT:**

|  |
| --- |
| **Keywords:**  <HTML></HTML>  <head>  <tittle></tittle>  </head>  <body></body>  **Headings:**  <h1></h1> <h2> </h2> <h3> </h3> <h4> </h4> <h5></h5> <h6></h6>  <div>  **Argument: style**   * </div> * <img>   **Arguments: src, width, height, alt**   * </img> * <p> * </p> * <a>   **Arguments: href**   * </a> * <br> //no closing tag * <i></i>//italic * <b></b> //bold   **Table:**  <table>  **Arguments: border, bordercolor, width, height,align**   * <caption> * <tr> bgcolor </tr> * <th></th> * <td>   **Arguments: colspan, rowspan**   * </td> * </caption> * </table>   **Form:**  <form>  <textarea>    **Argument: name, id, cols, rows**,   * </textarea> * <input>   **Arguments: type, name, value, id, required, max-length, checked**   * </input> * <select>   **Arguments: name, id, required, size, multiple**   * </select> * <fieldset><fieldset> * <textarea>   **Argument: rows, cols, name**   * </textarea>   **List:**  <ul>  **Arguments: type**   * </ul> * <li></li> * <ol>   **Arguments: type, start**   * </ol> * <dl> * <dt></dt> * <dd></dd> * </dl> |

**2.3 CFG for at least five constructs in your language:**

The below is the Context Free Grammar for HTML language.

G **→** <N>G<\N>|NLT|epsilon

NLT **→** <br>|epsilon

N → html|head|tittle|body|HEADING|div|img|p|a|FORMATTING|HTML| HEAD|TITTLE|BODY|DIV|IMG|P|TABLE|LIST| |TABLE|FORM|charnum|epsilon

TABLE **→** table|TableTags

TableTags **→**  TD|TH|TR|CAPTION|tr|th|td|charnum|caption

LIST **→** list|ListTags

ListTags **→** ul|li|ol|dl|dt|dd|UL|LI|OL|DL|DT|DD|charnum|epsilon

FORM **→**  form|FORMTAGS

FORMTAGS **→**  textarea|input|select|fieldset|TEXTARE|INPUT|SELECT|FIELDSET|charnum|epsilon

HEADING **→** JK

J **→** h1|h2|h3|h4|h5|h6|charnum|epsilon

K **→** H1|H2|H3|H5|H6|charnum|epsilon

FORMATTING **→** CAPS|SMALL

CAPS **→**  I|U|STRONG|ABBR|B

SMALL **→**  i|u|b|strong|abbr

letter **→** a|b|...z|A|B|...Z|epsilon

digit **→** 0|1|...|9|epsilon

charnum **→**  letter|digit|

Comment **→←** charnum **→**

**2.4 Design of Parser**

**2.4.1.Table**

|  |
| --- |
| G -> < N > G < \ N >  G -> ''  N -> TABLE  TABLE -> table  TABLE -> TableTags  TableTags -> TD  TableTags -> TH  TableTags -> TR  TableTags -> tr  TableTags -> th  TableTags -> td |

| **Non-Terminals** | **FOLLOW** | **FIRST** |
| --- | --- | --- |
| G | {$,<} | {<,''} |
| N | {>} | {table,TD,TH,TR,tr,th,td,''} |
| TABLE | {>} | {table,TD,TH,TR,tr,th,td} |
| TableTags | {>} | {TD,TH,TR,tr,th,td} |

**Parsing with LL(1) parser:**

**2.4.2.Table:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Non-**  **Terminal** | **table** | **TD** | **TH** | **TR** | **tr** | **th** | **td** | **<** | **>** | **\** | **$** |
| G |  |  |  |  |  |  |  | G ->  <N>G<\N>  G -> '' |  |  | G -> '' |
| N | N -> TABLE | N -> TABLE | N -> TABLE | N -> TABLE | N -> TABLE | N -> TABLE | N->TABLE |  |  |  |  |
| TABLE | TABLE -> table | TABLE -> TableTags | TABLE -> TableTags | TABLE -> TableTags | TABLE -> TableTags | TABLE -> TableTags | TABLE -> TableTags |  |  |  |  |
| TableTags |  | TableTags -> TD | TableTags -> TH | TableTags -> TR | TableTags -> tr | TableTags -> th | TableTags->td |  |  |  |  |

There are double entries in the table thus the grammar can’t be parsed by the LL(1) Parser.

**Parsing with SLR Parser:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SLR CLOSURE TABLE** | | |  |
| **GOTO** | **Kernel** | **State** | **Closure** |
|  | {G -> .< N > G < \ N >} | 0 | {G -> .< N > G < \ N >} |
| goto(0, <) | {G -> <.N > G < \ N >} | 1 | {G -> <.N > G < \ N >;  N -> .TABLE;  TABLE -> .table; TABLE -> .TableTags; TableTags -> .TD; TableTags -> .TH; TableTags -> .TR; TableTags -> .tr; TableTags -> .th; TableTags -> .td} |
| goto(1, N) | {G -> < N.> G < \ N >} | 2 | G -> < N.> G < \ N >} |
| goto(1, TABLE) | {N -> TABLE.} | 3 | {N -> TABLE.} |
| goto(1, table) | {TABLE -> table.} | 4 | {TABLE -> table.} |
| goto(1, TableTags) | {TABLE -> TableTags.} | 5 | {TABLE -> TableTags.} |
| goto(1, TD) | {TableTags -> TD.} | 6 | {TableTags -> TD.} |
| goto(1, TH) | {TableTags -> TH.} | 7 | {TableTags -> Th.} |
| goto(1, TR) | {TableTags -> TR.} | 8 | {TableTags -> TR.} |
| goto(1, tr) | {TableTags -> tr.} | 9 | {TableTags -> tr.} |
| goto(1, th) | {TableTags -> th.} | 10 | {TableTags -> th.} |
| goto(1, td) | {TableTags -> td.} | 11 | {TableTags -> td.} |
| goto(2, >) | {G -> < N >.G < \ N >} | 12 | {G -> < N >.G < \ N >; G -> .< N > G < \ N >; G -> .} |
| goto(12, G) | {G -> < N > G.< \ N >} | 13 | {G -> < N > G.< \ N >} |
| goto(12, <) | {G -> <.N > G < \ N >} | 1 |  |
| goto(13, <) | {G -> < N > G <.\ N >} | 14 | {G -> < N > G <.\ N >} |
| goto(14, \) | {G -> < N > G < \.N >} | 15 | {G -> < N > G < \.N >;  N -> .TABLE;  TABLE -> .table; TABLE -> .TableTags; TableTags -> .TD; TableTags -> .TH; TableTags -> .TR; TableTags -> .tr; TableTags -> .th; TableTags -> .td} |
| goto(15, N) | {G -> < N > G < \ N.>} | 16 | {G -> < N > G < \ N.>} |
| goto(15, TABLE) | {N -> TABLE.} | 3 |  |
| goto(15, table) | {TABLE -> table.} | 4 |  |
| goto(15, TableTags) | {TABLE -> TableTags.} | 5 |  |
| goto(15, TD) | {TableTags -> TD.} | 6 |  |
| goto(15, TH) | {TableTags -> TH.} | 7 |  |
| goto(15, TR) | {TableTags -> TR.} | 8 |  |
| goto(15, tr) | {TableTags -> tr.} | 9 |  |
| goto(15, th) | {TableTags -> th.} | 10 |  |
| goto(15, td) | {TableTags -> td.} | 11 |  |
| goto(16, >) | {G -> < N > G < \ N >.} | 17 | {G -> < N > G < \ N >.} |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LR Table** | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **State** | **ACTION** | | | | | | | | | | | **GOTO** | | | |
| < | > | \ | table | TD | TH | TR | tr | th | td | $ | G | N | TABLE | TableTags |
| 0 | s1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  | s4 | s6 | s7 | s8 | s9 | s10 | s11 |  |  | 2 | 3 | 5 |
| 2 |  | S12 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  | r2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  | r3 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  | r4 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  | r5 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  | r6 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  | r7 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  | r8 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  | r9 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 11 |  | r10 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 | s1/r1 |  |  |  |  |  |  |  |  |  | r1 | 13 |  |  |  |
| 13 | S14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  | s15 |  |  |  |  |  |  |  |  |  |  |  |  |
| 15 |  |  |  | s4 | s6 | s7 | s8 | s9 | s10 | s11 |  |  | 16 | 3 | 5 |
| 16 |  | s17 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 17 | acc |  |  |  |  |  |  |  |  |  | acc |  |  |  |  |

As there are conflicting entries in the table thus the grammar is not suitable for “SLR Parser”.

2.LIST

|  |
| --- |
| G -> < N > G < \ N >  G -> ''  N -> LIST  LIST -> list  LIST -> ListTags  ListTags -> ul  ListTags -> li  ListTags -> ol  ListTags -> dl  ListTags -> dt  ListTags -> dd  ListTags -> UL  ListTags -> LI  ListTags -> OL  ListTags -> DL  ListTags -> DT  ListTags -> DD  ListTags -> '' |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Non-Terminal** | **list** | **ul** | **li** | **ol** | **dl** | **dt** | **dd** | **UL** | **LI** | **OL** | **DL** | **DT** | **DD** | **<** | **>** | **\** | **br** | **$** |
| LIST | LIST -> list | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags | LIST -> ListTags |  | LIST -> ListTags |  |  | LIST -> ListTags |
| ListTags |  | ListTags -> ul | ListTags -> li | ListTags -> ol | ListTags -> dl | ListTags -> dt | ListTags -> dd | ListTags -> UL | ListTags -> LI | ListTags -> OL | ListTags -> DL | ListTags -> DT | ListTags -> DD |  | ListTags -> '' |  |  | ListTags -> '' |
| G |  |  |  |  |  |  |  |  |  |  |  |  |  | G ->  <N>G<\N> G -> '' |  |  |  |  |
| N | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' | N -> LIST N' |  | N ->LIST N’  N -> N’ |  |  |  |
| N’ |  |  |  |  |  |  |  |  |  |  |  |  |  |  | N'->N' N'->'' |  |  |  |

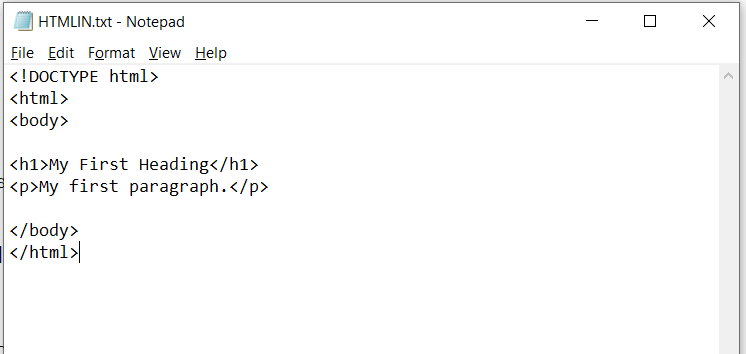
|  |  |  |
| --- | --- | --- |
| **Non Terminals** | **FOLLOW** | **FIRST** |
| LIST | {$,>} | {list,ul,li,ol,dl,dt,dd,UL,LI,OL,DL,DT,DD,''} |
| ListTags | {$,>} | {ul,li,ol,dl,dt,dd,UL,LI,OL,DL,DT,DD,''} |
| G | {<} | {<,''} |
| N | {>} | {list,ul,li,ol,dl,dt,dd,UL,LI,OL,DL,DT,DD,''} |
| N’ | { > } | {''} |

1. **Implementation:**

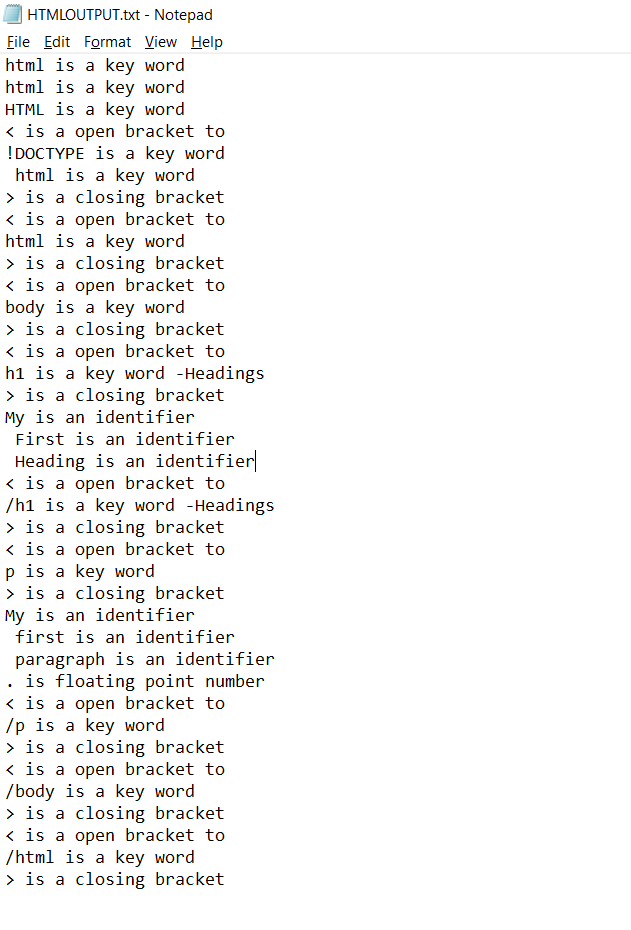
**3.1 Lexical Analyzer**

|  |
| --- |
| %{  #include<stdio.h>  #include<string.h>  int count=0,count1=0,n=0,i=0,j=0,cnt=0;  char st[10][10];  int lookup(char [10][10],char \*,int );  FILE \*yyin,\*yyout;  %}  %%  \n {count=0;n++;}.;  TITLE|title|head|DOCTYPE|BODY|body|HTML|html|div|DIV|IMG|img|p|P|a|A|list|table|forms {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word\n",yytext);}  TD|TH|TR|CAPTION|tr|th|td {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word -Table Tags\n",yytext);}  ul|li|ol|dl|dt|dd|UL|LI|OL|DL|DT|DD {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word -List Tags\n",yytext);}  textarea|input|select|fieldset|TEXTARE|INPUT|SELECT|FIELDSET {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word -Form Tags\n",yytext);}  h1|h2|h3|h4|h5|h6|H1|H2|H3|H5|H6 {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word -Headings\n",yytext);}  I|U|STRONG|ABBR|B|i|u|b|strong|abbr {if(count!=1 && count1!=1)fprintf(yyout,"%s is a key word -Formattings\n",yytext);}  [0-9]\* {if(count!=1 && count1!=1)fprintf(yyout,"%s is the number \n",yytext);}  [0-9]\*\.[0-9]\* {if(count!=1 && count1!=1)fprintf(yyout,"%s is floating point number\n",yytext);}  [a-zA-Z]\*[a-zA-Z0-9]\* {if(count!=1 && count1!=1)  {fprintf(yyout,"%s is an identifier\n",yytext);  if (!lookup(st,yytext,i))  {strcpy(st[i++],yytext);  cnt++;}}}  "<" {if(count!=1 && count1!=1) fprintf(yyout,"%s is a open bracket to\n",yytext);}  ">" {if(count!=1 && count1!=1) fprintf(yyout,"%s is a closing bracket\n",yytext);}  "!-- --" {if(count!=1 && count1!=1) fprintf(yyout,"%s is a comment \n",yytext);}  %%  int yywrap()  {  return 1;  }  int lookup(char st[10][10],char \*id,int n)  {  for(j=0;j<n;j++)  if(!strcmp(st[j],id))  return 1;  return 0;  }  int main(int argc, char \*argv[])  {  yyin=fopen("C:\\Users\\IMRAN\\Documents\\HTMLIN.txt","r");  yyout=fopen("C:\\Users\\IMRAN\\Documents\\HTMLOUTPUT.txt","a");  yylex();      printf("the contents of symbol table are :\n");  for(j=0;j<i;j++)  printf("%s\n",st[j]);  getch();  fclose(yyin);  fclose(yyout);  return 0;  } |

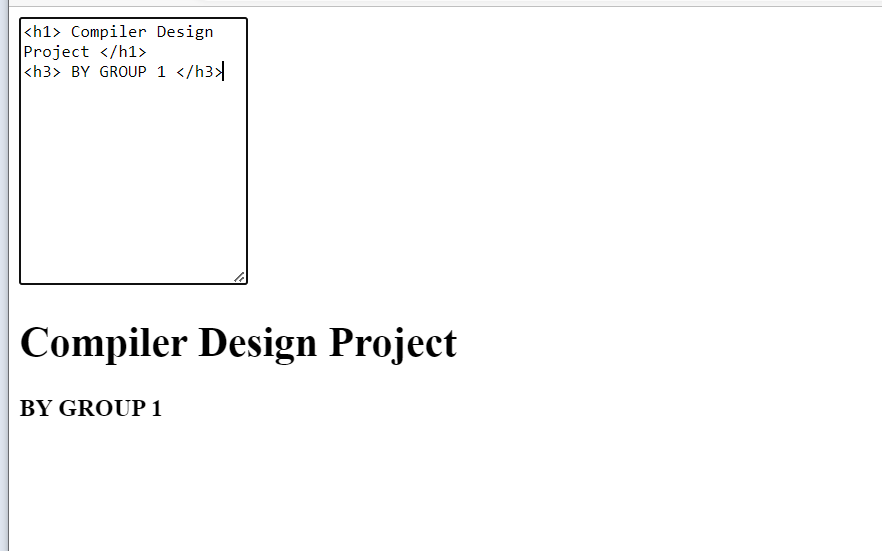
**INPUT:**



**OUTPUT:**



3.2 An interface, preferably web based for accepting your source program and display target program (Optional)



1. **Your insights on how you design a compiler/interpreter for the chosen language, what are the issues you have faced and how you resolved them**

* This project is basically aimed at to design the compiler for a HTML language using python as a support language.
* In the flow of developing this project there were many issues.
* For creating the CFGs as HTML is a waste language thus we couldn’t create grammar for the whole language thus we created it only for tables, list, form and included only few basic things as described.
* The main problem was created during the paring of the grammar that was created. while parsing the grammar was not accepted by any of the parsers. In the LL(1) parser one cell is containing more than one grammar production that is violating the rule of the parser.
* If we assume the language to be parsed successfully then for the intermediate code generation, as HTML is a scripting language all the statements are command scripts. Thus, as there are no variables and expressions available hence we can’t create an intermediate code for the language.

**5) CONCLUSION AND RESULTS:**

We are going to design the compiler using python with the rules that are used to design a compiler. Finally, we have included all the features that we have mentioned. In second report we are going to implement this using the procedure of the Compiler designing subject and we are going to use some GUI’s of python to make and look it better.