```cpp
#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

enum statetype{ newtoken, resword, variable, integer, real, statedelimiter, 
laststate};
enum chartype{ letter, digit, period, chardelimiter, blank, pod, eoln, illegal, 
lastchartype};

const int maxstring = 17;

statetype stringtostatetype(string s){
    statetype answer;
    if(s=="NewToken")
        answer = newtoken;
    else if(s=="ReservedWord")
        answer = resword;
    else if(s=="Variable")
        answer = variable;
    else if(s=="Integer")
        answer = integer;
    else if(s=="Real")
        answer = real;
    else if(s=="Delimiter")
        answer = statedelimiter;
    else
        answer = laststate;
    return answer;
}

string statetypetostring(statetype s){
    string answer;
    if(s==newtoken)
        answer = "NewToken";
    else if(s==resword)
        answer = "ReservedWord";
    else if(s==variable)
        answer = "Variable";
    else if(s==integer)
        answer = "Integer";
    else if(s==real)
        answer = "Real";
    else if(s==statedelimiter)
        answer = "Delimiter";
    else
        answer = "Not Valid";
    return answer;
}

chartype stringtochartype(string s){
    chartype answer;
    if(s=="Letter")
        answer = letter;
    else if(s=="Digit")
        answer = digit;
    else if(s=="Period")
        answer = period;
    else if(s=="Delimiter")
        answer = chardelimiter;
    else if(s=="Blank")
        answer = blank;
    else if(s=="Pod")
        answer = pod;
    else if(s=="EOLN")
        answer = eoln;
    else if(s=="Illegal")
        answer = illegal;
    else
        answer = lastchartype;
    return answer;
}

string chartypetostring(chartype s){
    string answer;
    if(s==letter)
        answer = "Letter";
    else if(s==digit)
        answer = "Digit";
    else if(s==period)
        answer = "Period";
    else if(s==chardelimiter)
        answer = "Delimiter";
    else if(s==blank)
        answer = "Blank";
    else if(s==pod)
        answer = "Pod";
    else if(s==eoln)
        answer = "EOLN";
    else if(s==illegal)
        answer = "Illegal";
    else
        answer = "Not Valid";
    return answer;
}

template <class Bryn>
void swapme(Bryn &first, Bryn &second) {
Bryn temp = first;
first = second;
second = temp;
}

void AlphaBubSort(string array[]) {
for (int y = 0; y < maxstring-1; y++){
    for (int b = 0; b < maxstring - 1; b++) {
        if (array[b] > array[b+1] && array[b+1] != "") { swapme(array[b], 
        array[b+1]); } } }
}

```

```cpp
chartype getchartype(char ch){
chartype answ;
if (ch >= 'A' && ch <= 'Z') answ = letter;
else if (ch >= 'a' && ch <= 'z') answ = letter;
else if (ch >= '0' && ch<='9') answ = digit;
else if (ch == '$' || ch=='%') answ = pod;
else if (ch == '.') answ = period;
else if (ch == ' ') answ = blank;
else if (ch == '@') answ = eoln;
else if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '(' || ch == ')' 
|| ch=='=') answ = chardelimiter;
else if (ch == ',' || ch == '^' || ch == '"' || ch == '&' || ch == '>' || ch == 
'<') answ = chardelimiter;
else answ = illegal;
return answ;
}

void readreserved(string reserves[]){
ifstream resinf;
resinf.open("reserve.dat");
for (int i=0; i<maxstring;i++) {
    resinf >> reserves[i] >> ws;}
}

void writereserved(string reserves[], ofstream &outf){
AlphaBubSort(reserves);
outf << setw(5) << right << " " << "Reserved Words Table" << endl;
for (int i =0; i < maxstring; i++) {outf << reserves[i]<< endl;}
}

void readprog1(string ProgString[], int &numprog){
ifstream inf;
inf.open("prog1.bas");
numprog=0;
while (!inf.eof()) {
    for (int k = 0; k < maxstring; k++){
        numprog++;
        getline(inf,ProgString[k]); } }
}

void writeprog1(string ProgString[], ofstream &outf, int numprog){
outf << setw(7) << right << " " << "Prog1.bas Table" << endl;
for (int j=0;j<numprog;j++) outf << ProgString[j]<< endl;
}

void readaction(int Action[laststate][lastchartype]){
    ifstream inf;
    inf.open("action.dat");
    for (int i= newtoken;i< laststate; i++)
        for (int j=letter; j<lastchartype;j++)
            inf >> Action[i][j];
}

void writeaction(ofstream &outf, int Action[laststate][lastchartype]){
    outf << endl << endl;
    outf << setw(50) << "Action Table" << endl;
    outf << setw(13) << " ";
    for (int i=letter;i<lastchartype; i++) { outf << left << setw(8) <<
    chartypetostring((chartype)i) << " "; }
    outf << endl;
    for (int k= newtoken;k< laststate; k++) {
        outf << left << setw(16) << statetypetostring((statetype)k);
        for (int j=letter; j<lastchartype;j++){ outf << setw(9) << Action[k][j]; }
         outf << endl;  }
    outf << endl;
}

void readexplain(string ExplainString[], int &numexplain){
ifstream inf;
inf.open("explain.dat");
numexplain = 0;
while (!inf.eof()) {
    for (int y = 0; y < maxstring; y++){
         getline(inf,ExplainString[y]);
         numexplain++; }}
}

void writeexplain(string ExplainString[], ofstream &outf, int numexplain){
    outf << endl << endl;
    outf << setw(25) << right << " " << "Explanations Table" << endl;
    for (int m=0;m<numexplain;m++) outf << ExplainString[m]<< endl;
}

void readstate(statetype FSM[laststate][lastchartype]){
string str;
ifstream inf;
inf.open("state.dat");
for (int k= newtoken;k< laststate; k++) {
    for (int j=letter; j<lastchartype;j++){
        inf >> str >> ws;
        FSM[k][j]=stringtostatetype(str); } }
}

void writestate(ofstream &outf, statetype FSM[laststate][lastchartype]){
outf << setw(65) << "State Table" << endl;
outf << setw(12) << " " ;
for (int i=letter;i<lastchartype; i++) {
    outf << left << setw(13) << "|"+chartypetostring((chartype)i);}
    outf << endl;
    outf << right << setw(117) << setfill('-') << " ";
    outf << setfill(' ') << endl;
for (int k= newtoken;k< laststate; k++) {
    outf << left << setw(12) << statetypetostring((statetype)k) << "|";
    for (int j=letter; j<lastchartype;j++){
        outf << left << setw(12) << statetypetostring(FSM[k][j]) << "|"; }
        outf << endl; }
}

void printtoken(string token, statetype state, ofstream &outf){
outf << right << setw(22) << token;
outf << setw(15) << statetypetostring(state) << endl;
}
```

```

void searchreserves(string reserves[], string token, statetype &state){
bool found = false;
    for(int i=0; i<maxstring; i++){
        if (token == reserves[i])
            found = true; }
if (found == false) state=variable;
}

void doactions(char ch, string &token, statetype &state, chartype cct, ofstream 
&outf, int actiontodo, string reserves[]){
if (actiontodo==1) {
    token += ch; }
else if (actiontodo==2) {
    searchreserves(reserves, token, state);
    printtoken(token, state, outf);
    token = ""; }
else if (actiontodo==3) {
    printtoken(token, state, outf);
    token = ""; }
else if (actiontodo==4) {
    printtoken(token, state, outf);
    outf << "Improper Usage:";
    outf << setw(7) << ch;
    outf << endl;
    token = ""; }
else if (actiontodo==5) {
    outf << "Improper Usage:";
    outf << setw(7) << ch;
    outf << endl;}
else if (actiontodo==6) {/*Continue*/}
else if (actiontodo==7) {
    outf << "Illegal Character:";
    outf << setw(4) << ch;
    outf << endl;}
else if (actiontodo==8) {
    searchreserves(reserves, token, state);
    printtoken(token, state, outf);
    token="";
    token+=ch;}
else if (actiontodo==9) {
    printtoken(token, state, outf);
    token = "";
    token += ch; }
else if (actiontodo==10) {
    token += ch;
    state = variable;
    printtoken(token, state, outf);
    token = "";
}
else if (actiontodo==11) {
    searchreserves(reserves, token, state);
    printtoken(token,state,outf);
    outf <<"Illegal Character:";
    outf << setw(4) << ch;
    outf << endl;}
```

```
else if (actiontodo==12){
    token += ch;
    printtoken(token,state,outf);
    token="";}
else if (actiontodo==13){
    printtoken(token,state,outf);
    outf << "Illegal Character:";
    outf << setw(4) << ch;
    outf << endl;
    token = "";}
else cout << "ERROR in ActionToDo";
}

void scanner(ofstream &outf, string reserves[],statetype 
FSM[laststate][lastchartype],int Action[laststate][lastchartype] ){
ifstream inf;
inf.open("prog1.bas");
string line;
char ch;
chartype cct;
statetype state = newtoken;
string token;
outf << endl << endl;
outf << right << setw(11) <<" " << "SCANNER RESULTS:" << endl;
outf << right <<"Error?";
outf << right << setw(16) << "TOKEN";
outf << right << setw(15) << "TOKEN-TYPE" << endl;
while (!inf.eof()) {
    getline(inf,line);
    line += '@';
    int actiontodo;
    int length = line.length();
    for (int i=0; i<length;i++){
        ch = line[i];
        cct = getchartype(ch);
        actiontodo = Action[state][cct];
        doactions(ch, token, state, cct, outf, actiontodo, reserves);
        state = FSM[state][cct]; } }
}

int main() {
int Action[laststate][lastchartype];
statetype FSM[laststate][lastchartype];
int numprog;
int numexplain;

ofstream outf;
outf.open("outputfile.txt");

string reserves[maxstring];
string ProgString[maxstring];
string ExplainString[maxstring];

readreserved(reserves);
writereserved(reserves, outf);

```

```cpp
    readprog1(ProgString, numprog);
    writeprog1(ProgString, outf, numprog);

    readaction(Action);
    writeaction(outf, Action);

    readexplain(ExplainString, numexplain);
    writeexplain(ExplainString, outf, numexplain);

    readstate(FSM);
    writestate(outf,FSM);

    scanner(outf, reserves,FSM, Action);

    system("pause");
}

```