Ahsanullah University of Science &Technology

Department of Computer Science & Engineering

Course No : CSE4130
Course Title : Formal Languages and Compilers Lab
Assignment No : 04

Date of Performance : 09/01/2023

Date of Submission : 31/01/2023

Submitted To : Mr. Aminur Rahman & Mr. Al Hasib Mahamud

Submitted By-
Group: A_2
Name : Asif Iftekher Fahim
Id : 190104027
Section : A

```
 #include<stdio.h>
#include<string.h>

int id(char lex[])
{
   int i = 0,l,s;
   if(isalpha(lex[i]) || (lex[i]=='_'))
   {
      s=1;
      i++;
   }
   else
      s=0;
   l=strlen(lex);
   if(s)
   {
      for(; i<l; i++)
      {
         if(isalpha(lex[i]) || (lex[i]=='_') || isdigit(lex[i]))
         {
            s=1;
         }
         else
         {
            s=0;
            break;
         }
      }
   }
   return s;
}

void kwIdChecker(char inp[],int inpi)
{
   FILE *ptr;
   char c,word[20],kword[20],keyW[20]=" ",idW[20]=" ",linec[20];
   int j=0,kwf,idf,wf;
   char scope[20]="global";
   ptr = fopen("space.txt","w");
   for(int i=1; i<inpi; i++)
   {
      if(inp[i]=='(' || inp[i]==';' || inp[i]==',' || inp[i]==')' || inp[i]=='=' || inp[i]=='<')
         fprintf(ptr," %c ",inp[i]);
      else
         fputc(inp[i],ptr);
   }
   fclose(ptr);
   inpi=0;
```

```
ptr= fopen("space.txt","r");
while((c=fgetc(ptr))!=EOF)
{
   inp[inpi] = c;
   inpi++;
}
fclose(ptr);
for(int i=0; i<inpi; i++)
{
   if(inp[i]=='\n')
   {
      strcpy(keyW," ");
      strcpy(idW," ");
      idf=0;
      kwf=0;
      i++;
      j=0;
      while(inp[i] != ' ')
      {
         linec[j] = inp[i];
         i++;
         j++;
      }
      linec[j] = '\0';
      continue;
   }
   if(inp[i]=='}')
   {
      strcpy(scope,"global");
   }
   j=0;
   while(inp[i] != ' ')
   {
      word[j] = inp[i];
      i++;
      j++;
   }
   word[j] = '\0';
   wf = 1;

   ptr = fopen("keyword.txt","r");
   while(fscanf(ptr,"%s ",kword)==1)
   {
      if(!strcmp(kword,word))
      {
         if(kwf && !strcmp(word,keyW) && strcmp(word,"else"))
         {
            printf("\nDuplicate keyword detected at line %s",linec);
```

```
        }
        strcpy(keyW,word);
        kwf = 1;
        wf=0;
      }
    }
    fclose(ptr);
    if(id(word) && wf)
    {
      strcpy(idW,word);
      idf=1;
    }
    if(idf && kwf && strcmp(idW," ") && strcmp(keyW," ") && keyW[0]!='e' &&
keyW[0]!='r' )
    {
      i++;
      j=0;
      while(inp[i] != ' ')
      {
        word[j] = inp[i];
        i++;
        j++;
      }
      word[j] = '\0';
      if(lookUp(idW,scope))
      {
        printf("\nDuplicate identifier detected at line %s",linec);
      }
      else if(!strcmp(word,"("))
      {

        insert(idW,"func",keyW,scope);
        strcpy(scope,idW);
      }
      else
      {

        insert(idW,"var",keyW,scope);
      }
      idf = 0;
      kwf = 0;
    }
    else if(idf && strcmp(idW," "))
    {
      if(!lookUp(idW,scope))
      {
        printf("\nUnknown identifier detected at line %s",linec);
```

```
            }
            idf = 0;
        }


    }
}

void commentRemove(FILE *fp)
{
    FILE *f;
    f= fopen("comment.txt","w");
    int putfi=1,parf=0,prif=1;
    char c,cb;
    while((c=getc(fp))!=EOF)
    {
        if(c=='(')
            parf = 1;
        if(c=='"' && parf)
            prif = 0;
        if(c== ')')
        {
            prif = 1;
            parf = 0;
        }
        if(prif)
        {
            if (c == cb && c==' ')
                putfi = 0;
            if(c=='/')
            {
                while(c=getc(fp))
                {
                    if(c=='/')
                    {
                        while((c=getc(fp) )!= '\n');
                        break;
                    }
                    else if(c=='*')
                        while((c=getc(fp) )!= '/');
                    putfi=0;
                    break;
                }
            }
        }
        if(putfi)
            fputc(c,f);
        putfi=1;
        cb = c;
    }
    fclose(f);

  }
int forCheck(char inp[],int k)
{
    k--;
    while(inp[k]==' ' && inp[k]!='(')
    {
        k--;
    }

    if(inp[k]=='(')
    {
        if(inp[k-3]=='f' && inp[k-2]=='o' && inp[k-1]=='r')
            return 1;
        else
            return 0;
    }
    else
        return 0;
}
void duplicateErrors(char inp[],int inpi)
{
    char linec[20],linecB[20],lineP[20],lineD[20],lineD2[20];
    int i=0, ifflag=0;
    int j=0, countB=0,countP=0,k;
    while(inp[i] != ' ')
    {
        linec[j] = inp[i];
        i++;
        j++;
    }
    linec[j] = '\0';
    for(i = 0; i<inpi; i++)
    {
        if(inp[i] == '\n')
        {
            if(countP>0)
            {
              printf("\nUnbalanced '(' at line %s",lineP);
                countP = 0;
            }
            i++;
            j=0;
            while(inp[i] != ' ')
            {
            linec[j] = inp[i];
            i++;
            j++;
          }
          linec[j] = '\0';
        }
        else if(inp[i] == '{')
        {
          countB++;
          strcpy(linecB,linec);
        }
        else if(inp[i] == '}')
          countB--;
        else if(inp[i] == '(')
        {
          countP++;
          ///printf("\n+++%s",linec);
          strcpy(lineP,linec);
        }
        else if(inp[i]==';')
        {
          k=i;
          k++;
          while(inp[k]!=';' && inp[k]==' ')
            k++;
          if(inp[k]==';')
            if(!forCheck(inp,i))
            {
              if(strcmp(lineD2,linec))
              {
                printf("\nDuplicate token at line %s",linec);
                strcpy(lineD2,linec);
              }
            }
        }
        else if(inp[i] == ')')
          countP--;
        else if(inp[i] == 'i' && inp[i+1] == 'f' && inp[i+2] == '(')
          ifflag++;
        else if(inp[i] == ' ' && inp[i+1] == 'e' && inp[i+2] == 'l' && inp[i+3] == 's' &&
    inp[i+4] == 'e' && inp[i+5] == ' ')
          ifflag--;

        if(countB<0 && strcmp(lineD,linec))
        {
          printf("\nUnbalanced '}' at line %s",linec);
          strcpy(lineD,linec);
          countB=0;
        }
        else if(countB < 0)
            countB = 0;

        if(countP<0 && strcmp(lineP,linec))
        {
            printf("\nUnbalanced ')' at line %s",linec);
            strcpy(lineP,linec);
            countP=0;
        }
        else if(countP < 0)
            countP = 0;
        if(ifflag<0)
        {
            printf("\nUnmatched 'else' at line %s",linec);
            ifflag=0;
        }
    }
    if(countB)
        printf("\nUnbalanced '{' at line %s",linecB);

}

void addLineNum(FILE *fp, char inp[],int inpi)
{
    int linen=1;
    fprintf(fp,"%d ",linen);
    for(int i = 0; i<inpi; i++)
    {
        while(inp[i]!='\n')
        {
            fputc(inp[i],fp);
            i++;
        }
        fputc(inp[i],fp);
        linen++;
        fprintf(fp,"%d ",linen);
    }
}


int tal = 0;
struct symbolTable
{
    char name[20][20];
    char idType[20][20];
    char dataType[20][20];
    char scope[20][20];
```

```
} sTable;

void insert(char n[], char iT[], char dT[], char scp[])
{
    strcpy(sTable.name[taI],n);
    strcpy(sTable.idType[taI],iT);
    strcpy(sTable.dataType[taI],dT);
    strcpy(sTable.scope[taI],scp);
    taI++;
}

int lookUp(char n[],char scp[])
{
    for(int i = 0; i<taI; i++)
        if(!strcmp(sTable.name[i],n))
        {

            if(!strcmp(sTable.idType[i],"func"))
                return (i+1);
            else if(!strcmp(sTable.scope[i],scp) || !strcmp(scp,"global") ||
!strcmp(sTable.scope[i],"global"))
            {
                return (i+1);
            }
        }
    return 0;
}
void display()
{
    printf("\nSl\tName\tId TYPE\tData Type\tScope\tValue\n-------------------------------------
--------------------\n");
    for(int i = 0; i<taI ; i++)

printf("%d\t|%s\t|%s\t|%s\t\t|%s\t\n",(i+1),sTable.name[i],sTable.idType[i],sTable.d
ataType[i],sTable.scope[i]);
}



int main()
{
    FILE *ptr,*fp2;
    int inpi=0;
    char inp[1000],c;
    int j,countB=0;
    printf("Input File:\n");
    ptr = fopen("input.txt","r");
    while((c=getc(ptr))!=EOF)
    {
        printf("%c",c);
    }
    fclose(ptr);
    printf("\n-----------------------------------------------\n");

    ptr = fopen("input.txt","r");
    commentRemove(ptr);
    fclose(ptr);

    ptr = fopen("comment.txt","r");
    while((c=getc(ptr))!=EOF)
    {
        inp[inpi]=c;
        inpi++;
    }
    fclose(ptr);

    ptr = fopen("line.txt","w");
    addLineNum(ptr,inp,inpi);
    fclose(ptr);

    ptr= fopen("line.txt","r");
    inpi=0;
    printf("Output 1:\n");
    while((c=getc(ptr))!=EOF)
    {
        printf("%c",c);
        inp[inpi]=c;
        inpi++;
    }
    fclose(ptr);

    printf("\n------------------------------\nErrors:");
    duplicateErrors(inp,inpi);
    kwIdChecker(inp,inpi);
    return 0;
}
```