

Programlama Dilleri

Proje 1

Sercan Bayram 05170000038

Oğuz Türk 05170000079

Gerçekleştirilen Platform ve Sürüm Adı:

Codeblocks 20.03

Programın Tanımı:

Program, code.psi adlı bir kaynak dosyasını kabul etmeli ve art arda listelenen code.lex'in tüm belirteçlerini içeren code.lex adlı bir metin dosyası üretmelidir. Program bizden temelde 8 maddenin analiz ve kontrolünü istemektedir:

1-Identifiers(Tanımlayıcılar)

Bizden bu maddede istenenler aşağıdaki gibidir:

- Maksimum tanımlayıcı boyutu 20 karakterdir. Bundan daha büyük bir tanımlayıcı kullanırsanız, bir hata mesajı vermelidir.
- Psi ++ dili büyük / küçük harfe duyarlı değildir ve tüm tanımlayıcı adları büyük harf olarak standartlaştırılmıştır.
- Tanımlayıcılar alfabetik bir karakterle (bir harf) başlar ve bir veya daha fazla harf, rakam veya _ (alt çizgi) ile başlayabilir.

Identifiers için kaynak kod: (Bu kısım diğer maddelerin de bazılarının kontrolleriyle ilişkili olduğundan içinde diğer kısımlarda bulunuyor)

```
digitstring = malloc(sizeof(char) * (digitlength)); /*integer için dinamik array oluşturulur.  
ve bu arrayle substring oluştururuz.  
Orn:"1582" , "15864ahmet" */  
  
substring(digitstring,satir,begin,digitlength);  
  
int tutac;  
  
bool flag2=false;  
  
while(tutac<digitlength){  
  
    if(isdigit(digitstring[tutac])!=false){ /* substringin her karakterinin digit olup olmama kontrolu*/  
  
        flag2=true; /*her karakteri digit degilse hatali identifier !!!
```

```

    }

    tutac++; //Orn : "9582ahmet" hatalı identifier.
}

tutac=0;

//printf("digitstring %s\n",digitstring);

if (flag2==false){

    if(digitlength>10){

        printf("ERROR!!! Integer size can't be bigger than 10: %s\n",digitstring);

        fprintf(fptr,"ERROR!!! Integer size can't be bigger than 10: %s\n",digitstring);

    }

    else{

        //printf("int const %s\n",digitstring);

        fprintf(fptr,"int const (%s)\n",digitstring);

    }

}

else{

    //printf("invalid %s\n ",digitstring);

    printf("ERROR! An identifier must be start with alphabetic character: %s\n ",digitstring);

    fprintf(fptr,"ERROR! An identifier must be start with alphabetic character: %s\n ",digitstring);

}

digitlength=0;

begin=0;

}

if(isalpha(satir[i])&&yorummu==false) //identifier olusturma kontrolu.

{
    begin=i;

    if(i==0){

        while(isalpha(satir[i]) || satir[i]=='_' || isdigit(satir[i])){

            substringlength++;

            //fprintf(fptr,"%c",satir[i]);

            i++;

        }

        if(substringlength>20){

            printf("ERROR! Please check identifier size\n");

            fprintf(fptr,"ERROR! Please check identifier size\n");

        }

    }

}

```

```

    }

    else{

        subs = malloc(sizeof(char) * (substringlength)); /* olusturulan substring keyworde esit degilse
                                                         identifier olma kontrolu.*/

        substring(subs,satir,begin,substringlength);

        substringlength=0;

        if(isKeyword(subs)==true){

            for(int i = 0; subs[i]; i++){

                subs[i] = tolower(subs[i]);

            }

            //printf("Keyword (%s)\n", subs);
            fprintf(fptr,"Keyword (%s)\n",subs);

        }

        else{

            //printf("identifier (%s)\n", subs);

            for(int i = 0; subs[i]; i++){

                subs[i] = toupper(subs[i]);

            }

            fprintf(fptr,"identifier (%s)\n",subs);

        }

        }

        substringlength=0;

    }

    else if(i>0&&(isDelimiter(satir[i-1]))){

        while(isalpha(satir[i]) || satir[i]=='_' || isdigit(satir[i])){

            substringlength++;

            //fprintf(fptr,"%c",satir[i]);

            i++;

        }

        if(substringlength>20){

```

```

printf("ERROR! Please check identifier size.\n");

fprintf(fp, "ERROR! Please check identifier size.\n");

}

else{
    //printf("%d\n", substringlength);

    subs = malloc(sizeof(char) * (substringlength));    // substring.
    substring(subs, satir, begin, substringlength);

    substringlength=0;
    if(isKeyword(subs)==true){
        for(int i = 0; subs[i]; i++){
            subs[i] = tolower(subs[i]);
        }

        //printf("Keyword %s\n", subs);
        fprintf(fp, "Keyword (%s)\n", subs);
    }
    else{
        //printf("identifier %s\n", subs);

        for(int i = 0; subs[i]; i++){
            subs[i] = toupper(subs[i]);
        }

        fprintf(fp, "identifier (%s)\n", subs);
    }
}
}
}

```

Bu kısımda identifier'ın max 20 karakterli olup olmadığı, bir harfle mi başladığı, digit, keyword, operatör ile ilişkili hatalı identifier durumlarının kontrolleri vs. yapılmıştır.

2-Integer Constants(Tam Sayı Sabitleri)

Bizden bu maddede istenenler aşağıdaki gibidir:

- Maksimum tamsayı boyutu 10 basamaktır. Bundan daha uzun bir tamsayı değeri kullanırsanız, program bir hata iletisi verir.
- Negatif değerler desteklenmez.

Integer Constants için kaynak kod:

```
digitstring = malloc(sizeof(char) * (digitlength)); /*integer için dinamik array olusturulur.

ve bu arrayle substring olustururuz.

Orn:"1582" , "15864ahmet" */

substring(digitstring,satir,begin,digitlength);

int tutac;

bool flag2=false;

while(tutac<digitlength){

    if(isdigit(digitstring[tutac])==false){ /* substringin her karakterinin digit olup olmama kontrolu*/

        flag2=true; /*her karakteri digit degilse hatali identifier !!!

    }

    tutac++; /*Orn : "9582ahmet" hatalı identifier.

}

tutac=0;

//printf("digitstring %s\n",digitstring);

if (flag2==false){

    if(digitlength>10){

        printf("ERROR!!! Integer size can't be bigger than 10: %s\n",digitstring);

        fprintf(fpPtr,"ERROR!!! Integer size can't be bigger than 10: %s\n",digitstring);

    }

    else{

        //printf("int const %s\n",digitstring);

        fprintf(fpPtr,"int const (%s)\n",digitstring);

    }

}
```

```

}
else{
    //printf("invalid %s\n ",digitstring);
    printf("ERROR! An identifier must be start with alphabetic character: %s\n ",digitstring);
    fprintf(fp,"ERROR! An identifier must be start with alphabetic character: %s\n ",digitstring);
}

```

Bu kısımda integer değerin doğru şekilde alınıp bastırılması ve 10'dan fazla olması engellenmiştir.

3-Operators (İşlem Operatörleri)

Bizden bu maddede istenenler aşağıdaki gibidir:

- Dilin geçerli operatörleri +, -, *, /, ++, -, := 'dir.

Operatörler için kaynak kod:

```

int flag1=0;
if (satir[i]=='+'&&flag1==0&&yorummu==false){
    if(satir[i+1]=='+'){
        fprintf(fp,"Operator(++) %c \n");
        flag1=1;
        i++;
    }
    else{
        fprintf(fp,"Operator(+) %c \n");
    }
}
flag1=0;
if (satir[i]=='-'&&flag1==0&&yorummu==false){
    if(satir[i+1]=='-'){
        fprintf(fp,"Operator(--) %c \n");
        flag1=1;
        i++; }
}

```

```

else{
    fprintf(fp, "Operator(-) %c \n");
}

}

if(satir[i]=='.' && satir[i+1]=='.' && yorummu==false){
    fprintf(fp, "Operator(=) %c \n");
}

}

if(satir[i]=='*' && yorummu==false){fprintf(fp, "Operator(*) %c\n");}
if(satir[i]=='/' && yorummu==false){fprintf(fp, "Operator(/) %c\n");}

```

Bu kısımda operatörlerin gerekli kontrolleri yapılmaktadır.

4-Brackets (Parantezler)

Bizden bu maddede istenenler aşağıdaki gibidir:

- LeftPar: (RightPar:)
- LeftSquareBracket: [RightSquareBracket:]
- LeftCurlyBracket: { RightCurlyBracket: }

Parantezler için kaynak kod:

```

if(satir[i]=='(' && yorummu==false){
    //fprintf(fp, "%s \n", leftpar);
    i++;
    if(satir[i]=='*'){
        i++;
        yorummu=true;
        flag7=true;
    }
    if (flag7==false){
        fprintf(fp, "%s \n", leftpar);
    }
}
if(satir[i]=='*'){
    //fprintf(fp, "%s \n", leftpar);
    i++;
}

```

/*yorum satiri kontrolu dosyanin sonuna kadar kontrol edilir. Eger yorummu=true kalirsa lexical error verdirilir. */


```

        if(satir[i]==' '){
            i++;
            yorummu=false;
        }
    }
    if(satir[i]==' ' && yorummu==false){ fprintf(fptr,"%s \n",rightpar);}
    if(satir[i]=='{' && yorummu==false){ fprintf(fptr,"%s \n",leftSquare);}
    if(satir[i]=='}' && yorummu==false){ fprintf(fptr,"%s \n",RightSquare);}
    if(satir[i]=='{' && yorummu==false){ fprintf(fptr,"%s \n",leftCurly);}
    if(satir[i]=='}' && yorummu==false){ fprintf(fptr,"%s \n",RightCurly);}

```

Bu kısımda parantez kontrolü bulunur. Comment kontrolüyle de ilişkili olduğu için commentle ilgili parantez ve yıldız kontrol kısmı da burada bulunur.

5-String Constants (Dize Sabitleri)

Bizden bu maddede istenenler aşağıdaki gibidir:

- Psi ++ 'nın dize sabitleri, “this is a string” deki gibi çift tırnak (ASCII kodu 34) ile sınırlandırılmıştır.
- Dize sabitleri sınırsız boyuta sahiptir
- Dize sabitleri çift tırnak karakteri içeremez. birine ulaştığınızda, dize sonlanır.
- Bir dize sabiti dosya bitmeden sona eremezse, program bir hata verilmelidir.

String Constants için kaynak kod:

```

if(satir[i]==""){
    /*string kontrolu. satir sonuna kadar kontrol edilir . eger quote kapanmadıysa
    lexical error verilir.*/

    int begin=i;
    tirnaklimi=true;
    i++;

```

```

while(i<strlen(satir)&&tirnaklimi&&yorummu==false){

    tirnaklenght++;

    i++;

    if(satir[i]==""){

        tirnaklimi=false;

        strings=malloc(sizeof(char)*tirnaklenght);
        substring(strings,satir,begin,tirnaklenght);
        //printf("%s",strings);
        fprintf(fp, "String (%s)\n",strings);
    }

}

tirnaklenght=2;
if(tirnaklimi&&yorummu==false){
    printf("Quote is missed!!!\n");
    fprintf(fp, "Quote is missed!!!\n");
}
}

```

Bu kısımda string kontrolü, satır sonuna kadar kontrol edilir. Eğer tırnak kapanmadıysa hata mesajı verilir.

6-Keywords

Bizden bu maddede istenenler aşağıdaki gibidir:

- Anahtar kelimeler
break, case, char, const, continue, do, else, enum, float, for, goto, if, int, long, record, return, static, while 'dır.

- Psi ++ dili büyük / küçük harfe duyarlı değildir ve tüm anahtar kelimeler küçük harf olarak standartlaştırılmıştır. “While” VEYA “While” VEYA “WHILE” ile aynı kelimeyi yazabilirsiniz, hepsi "while" olarak okunmalıdır.

Anahtar kelimeler için kaynak kod:

```
if(isKeyword(subs)==true){
    for(int i = 0; subs[i]; i++){
        subs[i] = tolower(subs[i]);
    }

    //printf("Keyword (%s)\n", subs);
    fprintf(fp, "Keyword (%s)\n", subs);

}
else{
    //printf("identifier (%s)\n", subs);
    for(int i = 0; subs[i]; i++){
        subs[i] = toupper(subs[i]);
    }
    fprintf(fp, "identifier (%s)\n", subs);
}
}
substringlength=0;
}

else if(i>0&&(isDelimiter(satir[i-1]))){
    while(isalpha(satir[i]) || satir[i]=='_' || isdigit(satir[i])){

        substringlength++;
        //fprintf(fp, "%c", satir[i]);
        i++;
    }
    if(substringlength>20){

        printf("ERROR! Please check identifier size.\n");

        fprintf(fp, "ERROR! Please check identifier size.\n");

    }

    else{
        //printf("%d\n", substringlength);
        subs = malloc(sizeof(char) * (substringlength)); // substring.
        substring(subs, satir, begin, substringlength);
        substringlength=0;
        if(isKeyword(subs)==true){
            for(int i = 0; subs[i]; i++){
                subs[i] = tolower(subs[i]);
            }
            //printf("Keyword %s\n", subs);
            fprintf(fp, "Keyword (%s)\n", subs);
        }
        else{
            //printf("identifier %s\n", subs);
```

```

        for(int i = 0; subs[i]; i++){
            subs[i] = toupper(subs[i]);
        }
        fprintf(fp, "identifier (%s)\n", subs);
    }
}
}

```

Bu kısımda kendi oluşturduğumuz isKeyword fonksiyonu kullanılır ve bu kelimelerin identifier olup olmadığının kontrolü yapılır.

7-End of Line(Satır Sonu)

Bizden bu maddede istenenler aşağıdaki gibidir:

- “ ; ” karakteri satır sonu anlamına gelmektedir.

Satır sonu için kaynak kod:

```
if(satir[i]!=';' && yorummu==false){fprintf(fp, "%s\n", endlOfLine)
```

8-Comments(Yorumlar)

Bizden bu maddede istenenler aşağıdaki gibidir:

- (* Ve *) arasındaki herhangi bir şey bir yorumdur.
- Bir yorum dosya bitmeden sona erdirilemezse, program bir hata verilmelidir.
- Yorumlar tıpkı boş alan gibidir

Yorumlar için kaynak kod:

```
if(satir[i]!=';' && yorummu==false){fprintf(fp, "%s\n", endlOfLine);} //end of line;
```

```
    bool flag7=false;
```

```
    if(satir[i]!='(' && yorummu==false){
```

*/*yorum satiri kontrolu dosyanin sonuna kadar kontrol edilir. eger yorummu=true kalirsa lexical error verilir. */*

```
        //fprintf(fp, "%s \n", leftpar);
```

```
        i++;
```

```

if(satir[i]=='*'){
    i++;
    yorummu=true;
    flag7=true;

}
if (flag7==false){
    fprintf(fpPtr,"%s \n",leftpar);
}
}
if(satir[i]=='*'){
    //fprintf(fpPtr,"%s \n",leftpar);
    i++;
    if(satir[i]==''){
        i++;
        yorummu=false;

    }

}
}

```

“Yorummu” bir bool state’ dir ve comment içindeki diğer durumların işlenmemesi için bütün durumlarda kontrolü yapılır.İfadenin comment içinde olup olmadığını belirler.

Başlangıçta tanımlanıp bütün kod boyunca kullanılan ifadeler ve dosya işlemleri:

```

char*substring(char*destination,const char *source,int start,int length){ /* diziden substring almamizi saglar*/
    strncpy(destination, (source + start), length);
    return destination;
}

```

```

bool isKeyword(char* str)      /*Keyword Kontrolu*/
{
    if (!strcasecmp(str, "if") || !strcasecmp(str, "else") ||
        !strcasecmp(str, "while",str) || !strcasecmp(str, "do") ||
        !strcasecmp(str, "break") ||
        !strcasecmp(str, "continue") || !strcasecmp(str, "int")
        || !strcasecmp(str, "double") || !strcasecmp(str, "float")
        || !strcasecmp(str, "return") || !strcasecmp(str, "char")
        || !strcasecmp(str, "case")
        || !strcasecmp(str, "sizeof") || !strcasecmp(str, "for")
        || !strcasecmp(str, "const")
        || !strcasecmp(str, "enum")
        || !strcasecmp(str, "long") || !strcasecmp(str, "static")
        || !strcasecmp(str, "record") || !strcasecmp(str, "goto"))
        return (true);
    return (false);
}

```

```

bool isDelimiter(char ch)     /*Delimiter Kontrolu*/
{
    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == ';' || ch == ':' || ch == '>' ||
        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}')
        return (true);
    return (false);
}

```

```

int i;

FILE *fp;

FILE*fptr;

char* leftpar="LeftPar";

char * endofLine="EndOfLine";

char* rightpar="RightPar";

char*leftSquare="LeftSquareBracket";

char*RightSquare="RightSquareBracket";

```

```
char*leftCurly="LeftCurlyBracket";
char*RightCurly="RightCurlyBracket";
int begin=0;

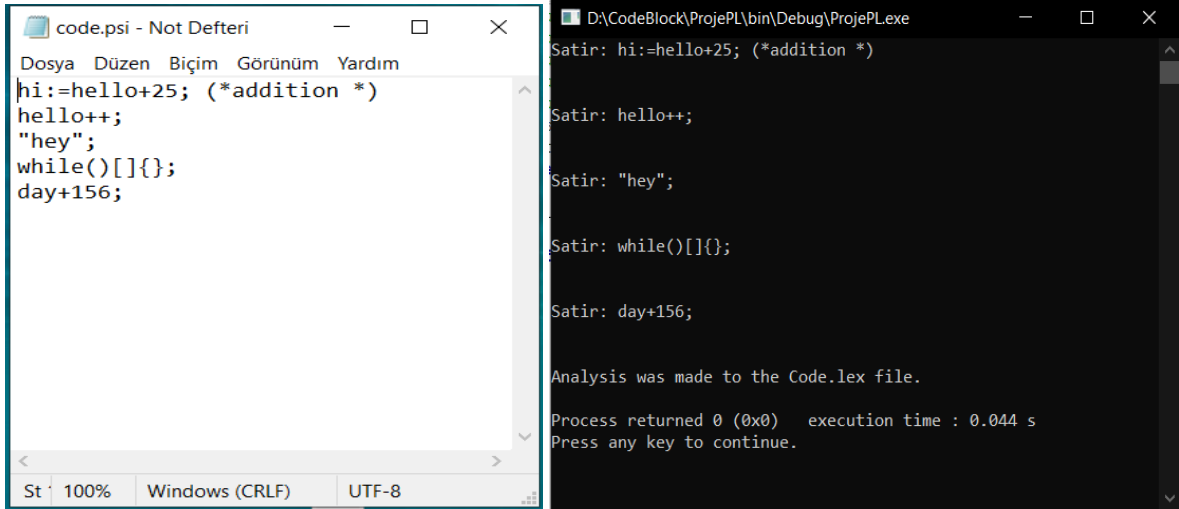
char*strings;
char *subs;
char *digitstring;
int bufferLength = 255;
char satir[bufferLength];
bool tirnaklimi=false;
bool yorummu=false;
int substrlength=0;
int tirnaklenght=2;

fp = fopen("C:/Users/SERCANBAYRAM/Desktop/code.psi.txt", "r");
fptr=fopen("C:/Users/SERCANBAYRAM/Desktop/code.lex.txt","w"); //okunacak ve yazılacak dosyaların
acilmasi...

//while(fscanf(fp, "%s", satir)!=EOF){
if ((fp) == NULL){
printf("dosya okumada hata var");
}
else{
while(fgets(satir, bufferLength, fp)) { //dosyayi bu while dongusuyle satir satir okuruz...

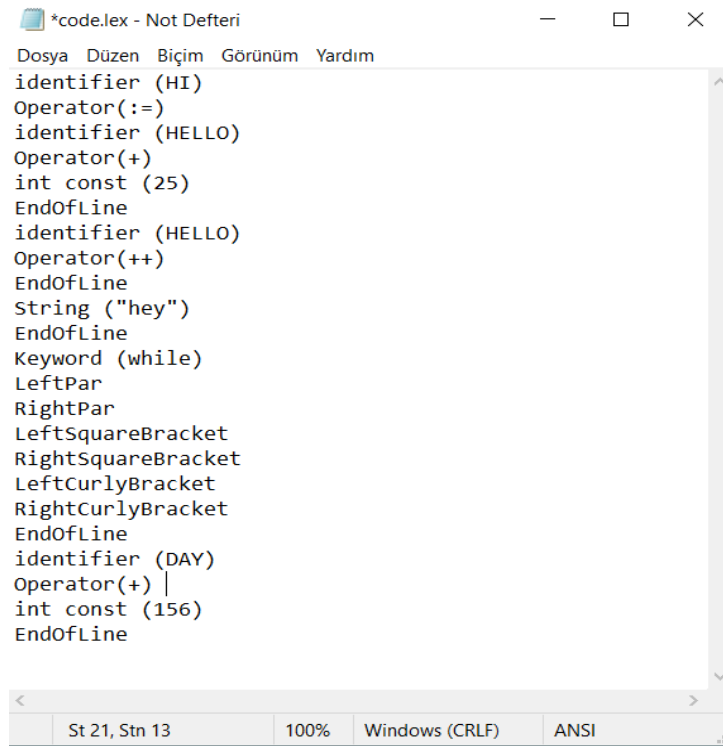
printf("Satir: %s \n", satir );
int digitlength=0;
for(i=0;i<strlen(satir);++i){ //satirin icinde karakter karakter donmemizi saglayan for dongusu..
```

Örnek Ekran Görüntüleri:



```
code.psi - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
hi:=hello+25; (*addition *)
hello++;
"hey";
while()[]{};
day+156;
St 100% Windows (CRLF) UTF-8
```

```
D:\CodeBlock\ProjePL\bin\Debug\ProjePL.exe
Satir: hi:=hello+25; (*addition *)
Satir: hello++;
Satir: "hey";
Satir: while()[]{};
Satir: day+156;
Analysis was made to the Code.lex file.
Process returned 0 (0x0)   execution time : 0.044 s
Press any key to continue.
```



```
*code.lex - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
identifier (HI)
Operator(:=)
identifier (HELLO)
Operator(+)
int const (25)
EndOfLine
identifier (HELLO)
Operator(++ )
EndOfLine
String ("hey")
EndOfLine
Keyword (while)
LeftPar
RightPar
LeftSquareBracket
RightSquareBracket
LeftCurlyBracket
RightCurlyBracket
EndOfLine
identifier (DAY)
Operator(+)
int const (156)
EndOfLine
St 21, Stn 13 100% Windows (CRLF) ANSI
```

Yukarıdaki görseller tüm durumların hatasız olduğu bir örnektir.

The image displays three windows related to a C++ program and its execution:

- code.psi - Not Defteri**: A code editor showing the source code:

```
hi:=hello+25; (*addition *)
hello++;
"hey;
while()[]{};
day+156;
```
- D:\CodeBlock\ProjePL\bin\Debug\ProjePL.exe**: A debugger window showing the execution flow and the error message:

```
Satir: hi:=hello+25; (*addition *)
Satir: hello++;
Satir: "hey;
Quote is missed!!!
Satir: while()[]{};
Satir: day+156;
Analysis was made to the Code.lex file.
Process returned 0 (0x0) execution time : 0.455 s
Press any key to continue.
```
- *code.lex - Not Defteri**: A window showing the output of the lexer, listing tokens and the error message:

```
identifier (HI)
Operator(:=)
identifier (HELLO)
Operator(+)
int const (25)
EndOfLine
identifier (HELLO)
Operator(++ )
EndOfLine
Quote is missed!!!
Keyword (while)
LeftPar
RightPar
LeftSquareBracket
RightSquareBracket
LeftCurlyBracket
RightCurlyBracket
EndOfLine
identifier (DAY)
Operator(+)
int const (156)
EndOfLine
```

Yukarıdaki görseller String Constants durumunda tırnağın kapanmama durumundaki hata örneğidir.

```
code.psi - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
hi:=hello+25; (*addition *)
hello++;
"hey"; (*comment|
while()[]{};
day+156;

100% Windows (CRLF) UTF-8

D:\CodeBlock\ProjePL\bin\Debug\ProjePLexe
Satir: hi:=hello+25; (*addition *)
Satir: hello++;
Satir: "hey"; (*comment
Satir: while()[]{};
Satir: day+156;
Comment ERROR!!!
Analysis was made to the Code.lex file.
Process returned 0 (0x0) execution time : 0.020 s
Press any key to continue.
```

```
*code.lex - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
identifier (HI)
Operator(:=) Ę
identifier (HELLO)
Operator(+
int const (25)
EndOfLine
identifier (HELLO)
Operator(++
EndOfLine
String ("hey")
EndOfLine
Comment ERROR!!!
|

100% Windows (CRLF) ANSI
```

Yukarıdaki görseller comment bırakırken (* açıp dosya sonuna kadar bir yerde kapatmamama hatasının örneğidir.

The image shows two windows side-by-side. The left window, titled 'code.psi - Not Defteri', contains the following code:

```
Dosya Düzen Biçim Görünüm Yardım
hi:=hello+25sss; (*addition *)
hello++;
"hey"; (*comment*)
while()[]{ };
day+156aa;
```

The right window, titled 'D:\CodeBlock\ProjePL\bin\Debug\ProjePL.exe', shows the execution output:

```
Satir: hi:=hello+25sss; (*addition *)
ERROR! An identifier must be start with alphabetic character: 25sss
Satir: hello++;
Satir: "hey"; (*comment*)
Satir: while()[]{ };
Satir: day+156aa;
ERROR! An identifier must be start with alphabetic character: 156aa
Analysis was made to the Code.lex file.
Process returned 0 (0x0)   execution time : 0.023 s
Press any key to continue.
```

The image shows a window titled '*code.lex - Not Defteri' displaying the lexical analysis output for the code. The output is as follows:

```
Dosya Düzen Biçim Görünüm Yardım
identifier (HI)
Operator(:=)
identifier (HELLO)
Operator(+)
ERROR! An identifier must be start with alphabetic character: 25sss
EndOfLine
identifier (HELLO)
Operator(++ )
EndOfLine
String ("hey")
EndOfLine
Keyword (while)
LeftPar
RightPar
LeftSquareBracket
RightSquareBracket
LeftCurlyBracket
RightCurlyBracket
EndOfLine
identifier (DAY)
Operator(+)
ERROR! An identifier must be start with alphabetic character: 156aa
EndOfLine
```

Yukarıdaki görseller Invalide Identifier hatalarından birinin örneğidir.

```
code.psi - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
hi:=hello+2514256321452; (*addition *)
hello++;
"hey"; (*comment*)
WHILE;
day;

D:\CodeBlock\ProjePL\bin\Debug\ProjePL.exe
Satir: hi:=hello+2514256321452; (*addition *)
ERROR!!! Integer size can't be bigger than 10: 2514256321452
Satir: hello++;
Satir: "hey"; (*comment*)
Satir: WHILE;
Satir: day;
Analysis was made to the Code.lex file.
Process returned 0 (0x0) execution time : 0.020 s
Press any key to continue.
```

```
*code.lex - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
identifier (HI)
Operator(:=)
identifier (HELLO)
Operator(+
ERROR!!! Integer size can't be bigger than 10: 2514256321452
EndOfLine
identifier (HELLO)
Operator(++
EndOfLine
String ("hey")
EndOfLine
Keyword (while)
EndOfLine
identifier (DAY)
EndOfLine

St 16, Stn 1 100% Windows (CRLF) ANSI
```

Yukarıdaki görsellerde Integer değerinin 10'dan fazla girilmemesi gerektiğinin bir örneğidir.