

# CISM 314 LAB4

LOUIS HOFF 26654156  
DIDINTLE MASIBI 26719517  
EUGINE NNTSAKO CHAUKE 26170744

May 24, 2018

## 1 AIM

The aim of this practical was to write a program that was able to parse and translate general expression that involve addition and subtraction.

## 2 INTRODUCTION

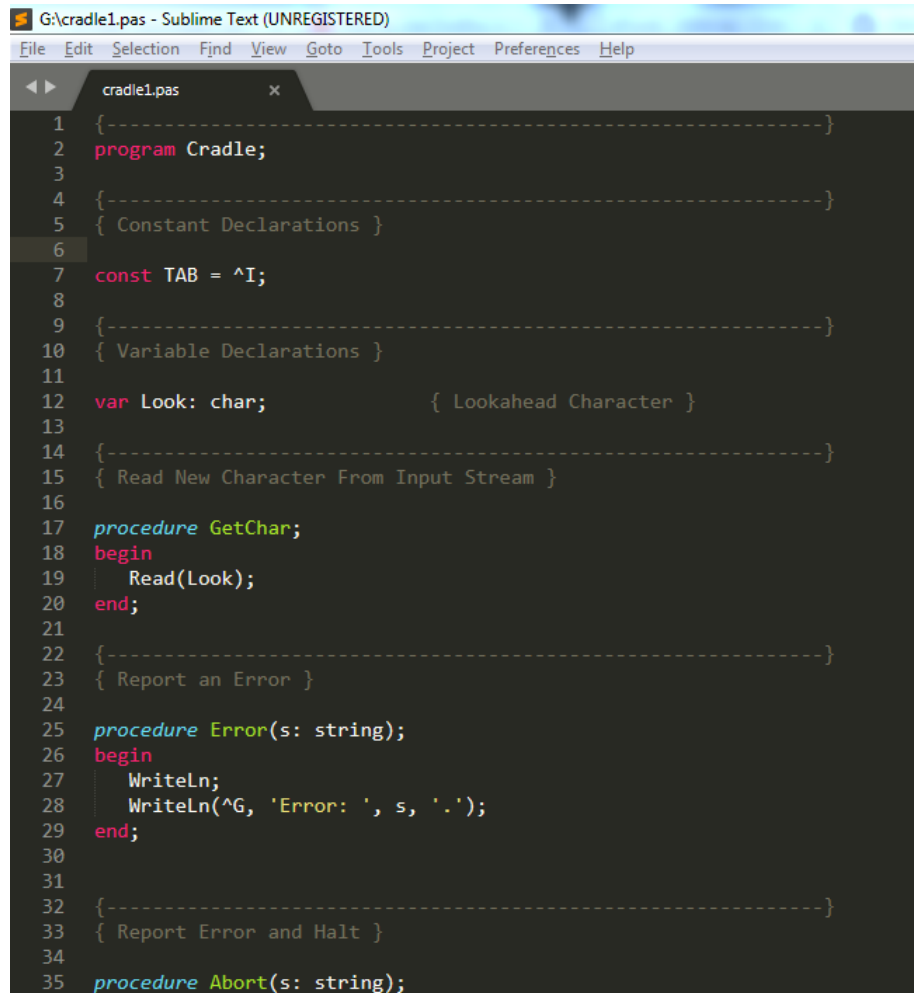
DOSBox is a program that rivals the DOS operating system which ran the majority of PCs in the late 1980s and early 1990s, with MS-DOS operating system being the most dominate. Due to the substantial demand from owners to either have the programs updated or a way to go back to MS-DOS when needed to run the older games or programs. Luckily for gamers, computer enthusiasts and owners wanting to continue using their old, trusted programs a group of developers finally came to the rescue in 2002 by creating the DOSBox emulator. The DOSBox emulator is an application that runs as a stand alone program that can be used to run thousands of old games and programs using the original files, floppy disks or CD-ROMs.

## 3 PROCEDURE(figure1 - figure6)

## 4 From figure 7 to figure 10: Running cradle.pas file

## 5 Conclusion

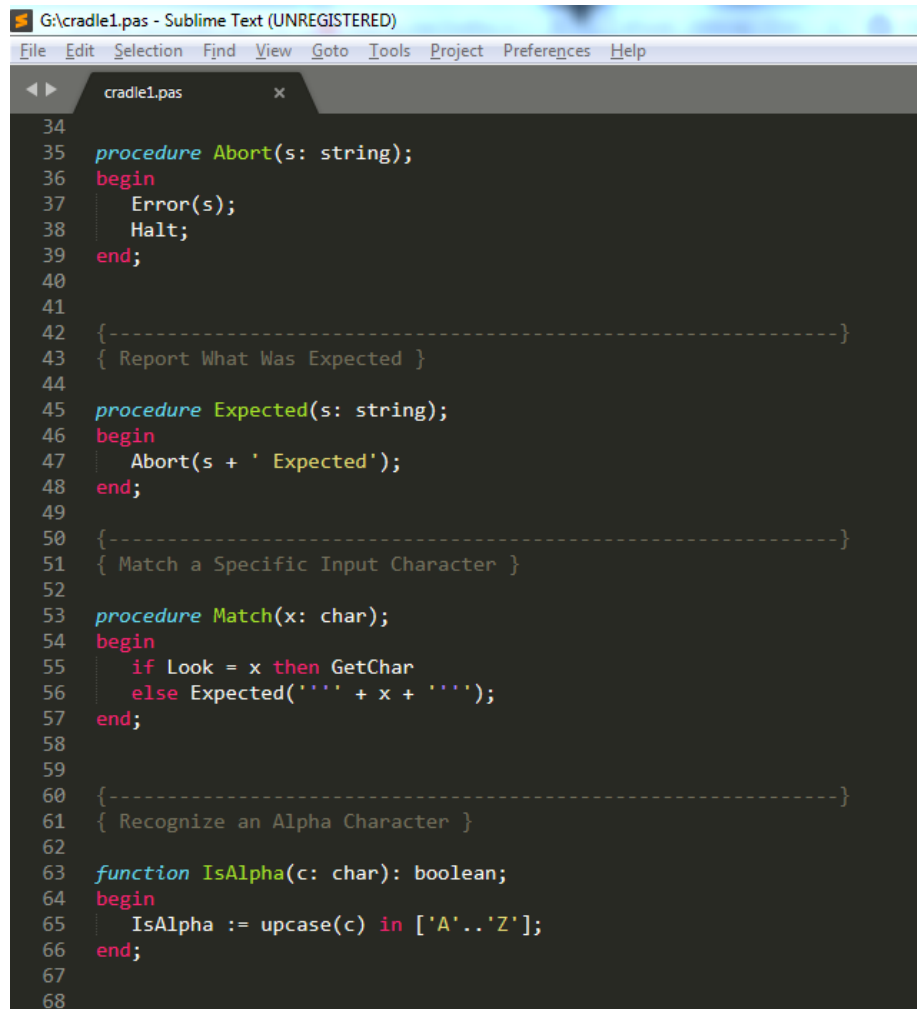
In this practical we were able to complete the program successfully and all the pictures indicate the parsing and translating general expressions that involved addition and subtraction as expected.



The image shows a screenshot of a Sublime Text editor window. The title bar reads "G:\cradle1.pas - Sublime Text (UNREGISTERED)". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The editor tab is labeled "cradle1.pas". The code is written in Pascal and is as follows:

```
1  {-----}
2  program Cradle;
3
4  {-----}
5  { Constant Declarations }
6
7  const TAB = ^I;
8
9  {-----}
10 { Variable Declarations }
11
12 var Look: char;           { Lookahead Character }
13
14 {-----}
15 { Read New Character From Input Stream }
16
17 procedure GetChar;
18 begin
19     Read(Look);
20 end;
21
22 {-----}
23 { Report an Error }
24
25 procedure Error(s: string);
26 begin
27     WriteLn;
28     WriteLn(^G, 'Error: ', s, '.');
29 end;
30
31
32 {-----}
33 { Report Error and Halt }
34
35 procedure Abort(s: string);
```

Figure 1: Program 1



The image shows a screenshot of a Sublime Text editor window. The title bar reads "G:\cradle1.pas - Sublime Text (UNREGISTERED)". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The editor tab is labeled "cradle1.pas". The code is written in Pascal and is as follows:

```
34
35 procedure Abort(s: string);
36 begin
37     Error(s);
38     Halt;
39 end;
40
41
42 {-----}
43 { Report What Was Expected }
44
45 procedure Expected(s: string);
46 begin
47     Abort(s + ' Expected');
48 end;
49
50 {-----}
51 { Match a Specific Input Character }
52
53 procedure Match(x: char);
54 begin
55     if Look = x then GetChar
56     else Expected('' + x + '');
57 end;
58
59
60 {-----}
61 { Recognize an Alpha Character }
62
63 function IsAlpha(c: char): boolean;
64 begin
65     IsAlpha := upcase(c) in ['A'..'Z'];
66 end;
67
68
```

Figure 2: Program 2

```
G:\cradle1.pas - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

cradle1.pas x
70
71 { Recognize a Decimal Digit }
72
73 function IsDigit(c: char): boolean;
74 begin
75     IsDigit := c in ['0'..'9'];
76 end;
77
78
79 {-----}
80 { Get an Identifier }
81
82 function GetName: char;
83 begin
84     if not IsAlpha(Look) then Expected('Name');
85     GetName := UpCase(Look);
86     GetChar;
87 end;
88
89
90 {-----}
91 { Get a Number }
92
93 function GetNum: char;
94 begin
95     if not IsDigit(Look) then Expected('Integer');
96     GetNum := Look;
97     GetChar;
98 end;
99
100
101 {-----}
102 { Output a String with Tab }
```

Figure 3: Program 3

```
G:\cradle1.pas - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

cradle1.pas x
103
104 procedure Emit(s: string);
105 begin
106     Write(TAB, s);
107 end;
108 {-----}
109 { Output a String with Tab and CRLF }
110
111 procedure EmitLn(s: string);
112 begin
113     Emit(s);
114     WriteLn;
115 end;
116
117 procedure Term;
118 begin
119     EmitLn('MOVE #' + GetNum + ',D0')
120 end;
121
122 {-----}
123
124 {-----}
125 { Recognize and Translate an Add }
126 procedure Add;
127 begin
128     Match('+');
129     Term;
130     EmitLn('ADD D1,D0');
131 end;
132 {-----}
```

Figure 4: Program 4

```

133 { Recognize and Translate a Subtract }
134 procedure Subtract;
135 begin
136     Match('-');
137     Term;
138     EmitLn('SUB D1,D0');
139 end;
140
141 {-----}
142 { Parse and Translate an Expression }
143 procedure Expression;
144 begin
145     Term;
146     EmitLn('MOVE D0,D1');
147     case Look of
148     '+' : Add;
149     '-' : Subtract;
150     else Expected('Addop');
151     end;
152 end;
153
154 {-----}
155 {-----}
156 { Initialize }
157
158 procedure Init;
159 begin
160     GetChar;
161 end;
162 {-----}
163 { Main Program }
164

```

Figure 5: Program 5

```

{ Main Program }

begin
    Init;
    Expression;
end.
{-----}

```

Figure 6: Program 6

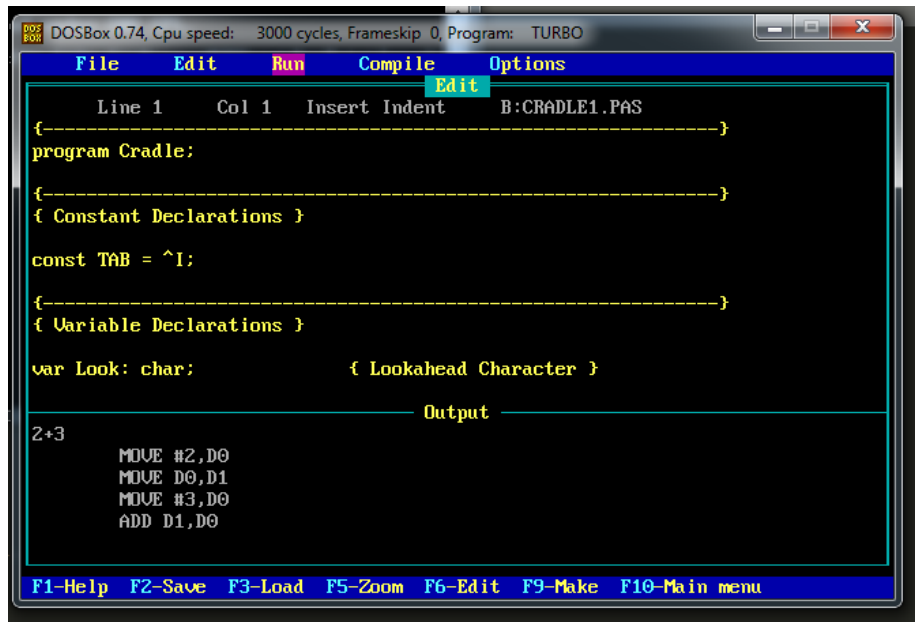


Figure 7: Program 7

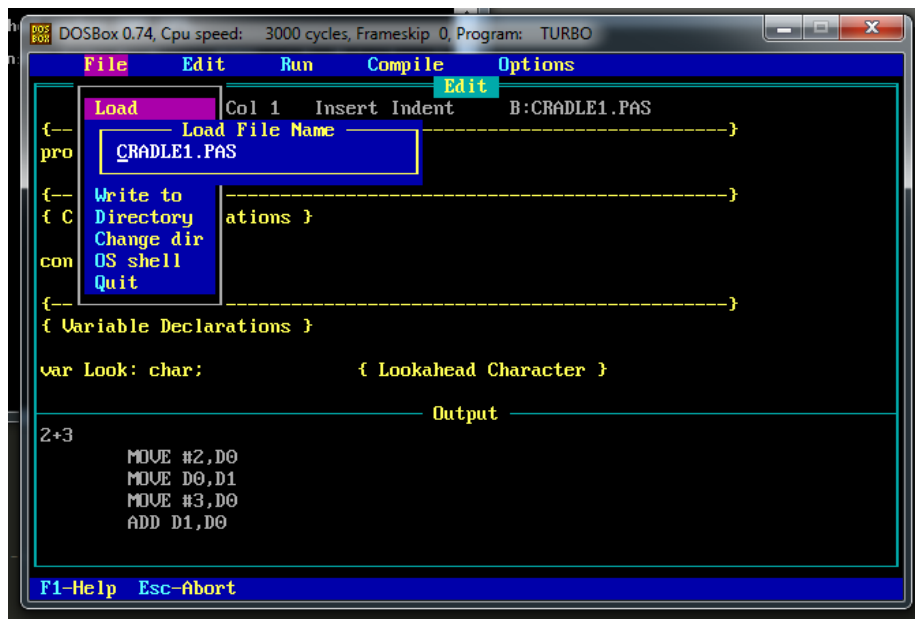


Figure 8: Program 8

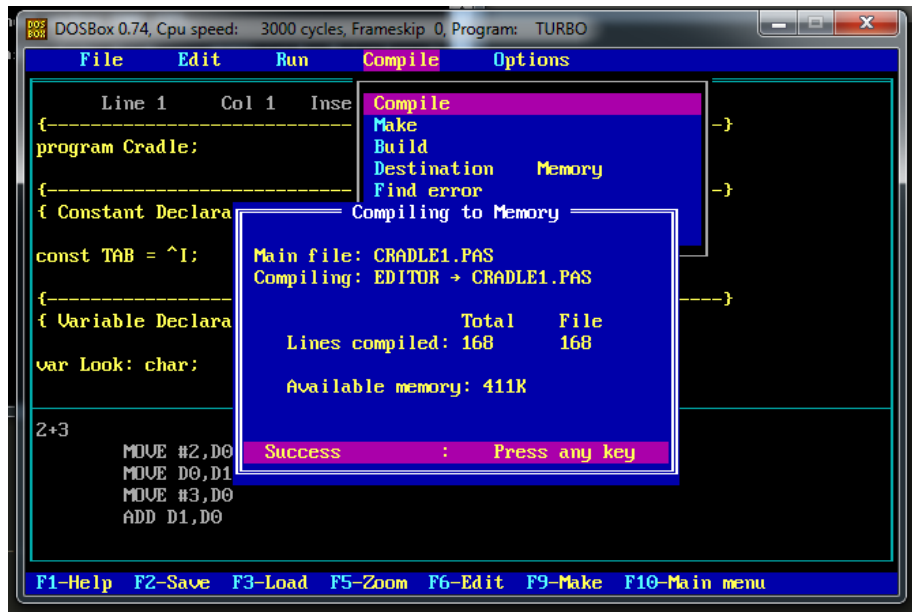


Figure 9: Program 9

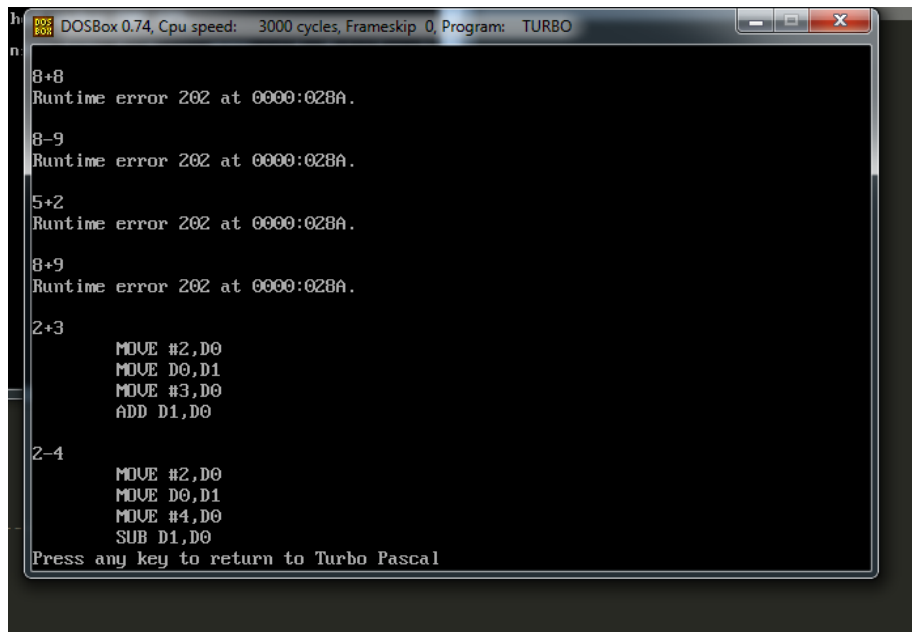


Figure 10: Program 10