CHAPTER IV

RESULTS AND DISCUSSION

From the experiment in chapter III, this part gives results of code algorithm part and processing algorithm part.

The Code Algorithm part has an image input condition result presented in the section of Image Segmentation Data Algorithm. Then section of Feature Extraction Data Algorithm will gives Absolute chain code and Pixel Line Length Determination results from characters A to Z.

The section of Object Classification Data Algorithm part shows the character function prototype.

The Processing Algorithm part gives experimental results in testing files through the algorithm.

4.1 The Code Algorithm part result

4.1.1 Image Segmentation Data Algorithm Result

This part contains input image condition results from the section of Image Segmentation Data Algorithm. This image specification was as well used in the part of Processing Algorithm. The result is shown in Table 4-1.

Table 4-1 The result of an input image specification used in the part of image segmentation

Image file Type	bmp
Scanning Resolution	100 dpi
Image Type	Bilevel color
Type Face	Cordia New
Font size	16 point
Paper size	A4

4.1.2 Feature Extraction Data Algorithm Result

The Absolute chain code and Pixel Line Length techniques were applied to the algorithm at the Feature Extraction Data Algorithm part. Table 4-2 show the Absolute chain code result which had been averaged from Appendix A. Each column represents the absolute code direction, taken from one boundary pixel to the next boundary pixel. For the first three columns (1-3) represents the absolute direction taken from the top of the pixel boundary to bottom on the left side of the characters. And for the last three column (4-6) was done exactly the same thing as the first three column but on the right side of the characters.

Table 4-3 show the average of Pixel line length code result, having ten columns giving the symbol of C.1 to C.10 represent the number of pixels in each line of the characters from line 1 to 10.

Table 4-2 The result of an Absolute chain code tracking process

Characters	Code No. 1	Code No. 2	Code No. 3	Code No. 4	Code No. 5	Code No. 6
A	1	1	1	3	3	3
B	2	2	2	3	2	1
C	1	2	3	3	2	1
D	2	2	2	3	2	1
E	2	2	2	1	2	3
F	2	2	2	1	2	2
G	1	2	3	3	2	1
H	2	2	2	2	2	2
I	2	2	2	2	2	2
J	2	2	2	2	2	1
K	2	2	2	1	2	3
L	2	2	2	2	2	2
M	2	2	2	2	2	2
N	2	2	2	2	2	2
O	1	2	3	3	2	1
P	2	2	2	3	2	2
Q	1	2	3	3	2	1
R	2	2	2	3	2	3
S	1	3	3	3	3	1
T	2	2	2	2	2	2

Table 4-2 The result of an Absolute chain code tracking process (continue)

Characters	Code No. 1	Code No. 2	Code No. 3	Code No. 4	Code No. 5	Code No. 6
U	2	2	2	2	2	1
V	3	3	3	1	1	1
W	2	2	2	1	1	1
X	3	2	1	1	2	3
Y	3	3	3	1	1	1
Z	1	1	1	1	2	3

Table 4-3 The result of Pixel line length code

Characters	C.1	C.2	C.3	C.4	C.5	C.6	C.7	C.8	C.9	C.10
A	5	5	8	11	14	17	17	20	23	26
B	17	20	20	20	17	17	20	23	20	17
C	14	20	23	5	5	5	5	26	23	17
D	17	20	23	23	26	26	23	23	23	17
E	20	5	5	5	17	17	5	5	5	20
F	17	5	5	5	5	14	5	5	5	5
G	14	20	26	5	5	26	29	29	26	14
H	23	23	23	23	23	23	23	23	23	23
I	5	5	5	5	5	5	5	5	5	5
J	5	5	5	5	5	5	5	17	14	14
K	23	20	17	14	11	14	17	17	20	23
L	5	5	5	5	5	5	5	5	17	17
M	29	29	29	29	29	29	29	29	29	29
N	23	23	23	23	23	23	23	23	23	23
O	11	20	26	26	29	29	29	26	26	11
P	17	17	20	20	17	5	5	5	5	5
Q	14	20	26	29	29	29	29	26	26	20
R	17	20	23	23	20	17	17	20	20	23
S	11	17	20	5	8	11	5	5	20	20
T	20	5	5	5	5	5	5	5	5	5
U	23	23	23	23	23	23	23	23	20	17

Table 4-3 The result of Pixel line length code (continue)

Characters	C.1	C.2	C.3	C.4	C.5	C.6	C.7	C.8	C.9	C.10
V	23	23	20	17	17	14	11	8	8	5
W	38	35	35	32	32	29	29	26	23	20
X	20	17	14	8	5	5	11	14	17	23
Y	23	20	17	11	8	5	5	5	5	5
Z	23	5	5	5	5	5	5	5	23	23

4.1.3 Object Classification Data Algorithm Result

This part concentrates on the function of the letter characteristic. Using the measure of the Absolute Chain Code and Pixel line length, the tolerances of acceptance to identify each characters were obtained and are summarized in section 4.1.3.1 .

4.1.3.1 A character function prototype

Feature Measure - Absolute Chain Code

- Pixel line length

(with ±3 code errors comparing)

Tolerances - The tolerances of Absolute Chain Code acceptance must be within <u>5 position</u>.

- The tolerances of Pixel line length acceptance must be within <u>9 position</u>.

Class - If the data was within the tolerance limits The character <u>identify</u>

- Otherwise, <u>unidentify</u>

4.2 The Processing Algorithm Part Results

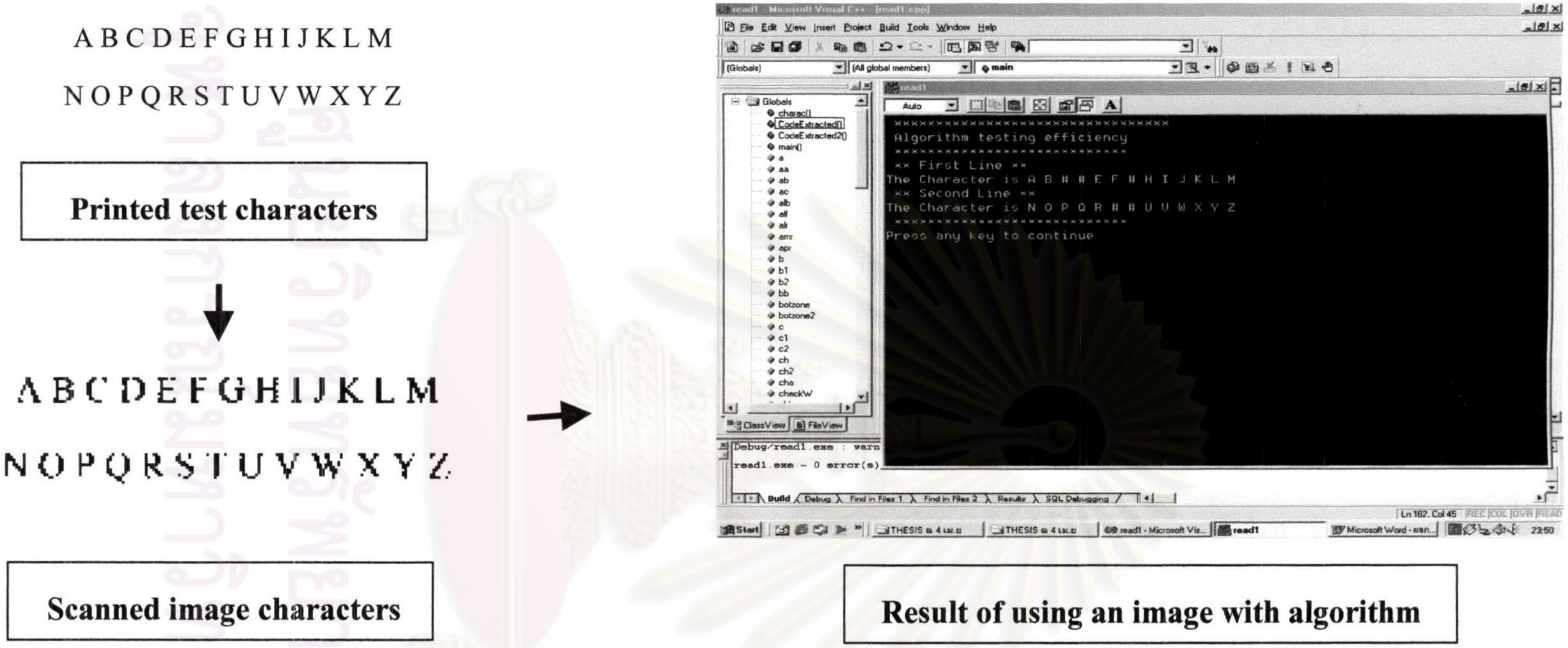
4.2.1 The font types that could be used in the algorithm

As mentioned earlier that this algorithm used a Cordia New font type as a prototype, Table 4-4 shows the ability to use the other font types with other same conditions as the Cordia New font type. (the test results of Table 4-4 was shown on Figure 4-1 to 4-9)

Table 4-4 The font types that could be use in the algorithm

Font Type
Cordia New
Angsana New
Browalia New
Dillenia UPC
Eucrosia UPC
FreesialUPC
Iris UPC
Jasmine UPC
Kodchiang UPC
Lily UPC

Figure 4-1 Result of Algorithm Type Face Test On Angsana New



A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Printed test characters

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Scanned image characters

```
 *********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is A B # # E F # H I J K L M
 ** Second Line **
The Character is N O P Q R # # U V W X Y Z
 ****************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

<u>**Testing result**</u>

The characters that cannot be translated because of insufficient scanned shape from scanner.

C,D,G,S and T

Figure 4-2 Result of Algorithm Type Face Test On Browalia New

A B C D E F G H I J K L M

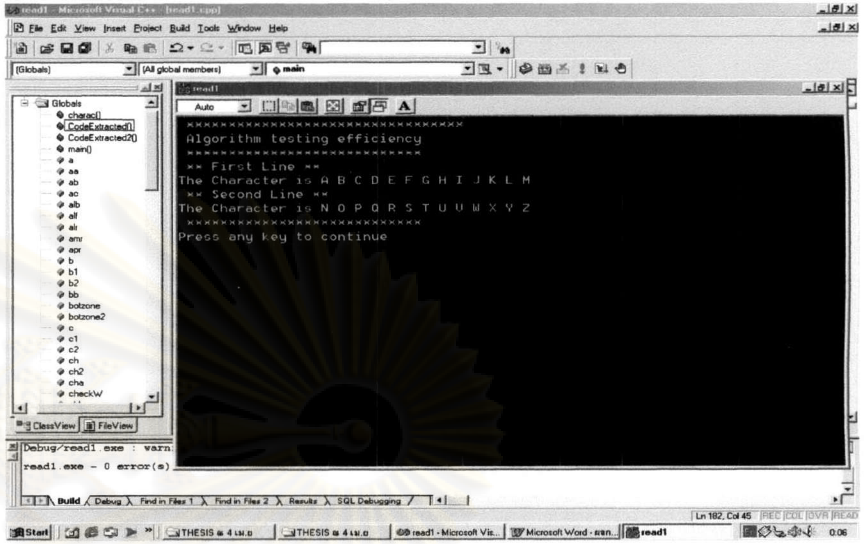
N O P Q R S T U V W X Y Z

Printed test characters

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Scanned image characters



```
****************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is A B C D E F G H I J K L M
 ** Second Line **
The Character is N O P Q R S T U U W X Y Z
 **************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

-

Figure 4-3 Result of Algorithm Type Face Test On Dillenia UPC

A B C D E F G H I J K L M

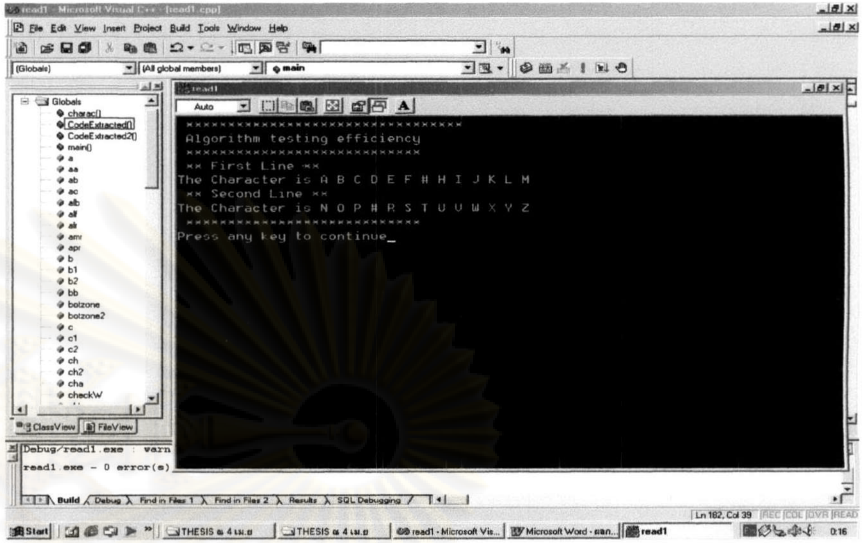
N O P Q R S T U V W X Y Z

Printed test characters

A B C D E F C H I J K L M

N O P O R S T U V W X Y Z

Scanned image characters



```
 *********************************
 Algorithm testing efficiency
 *****************************
 ** First Line **
The Character is A B C D E F # H I J K L M
 ** Second Line **
The Character is N O P # R S T U V W X Y Z
 *****************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

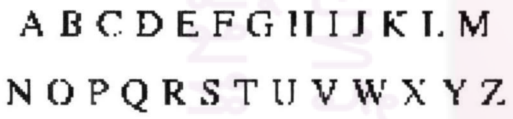
G and Q

Figure 4-4 Result of Algorithm Type Face Test On Eucrosia UPC

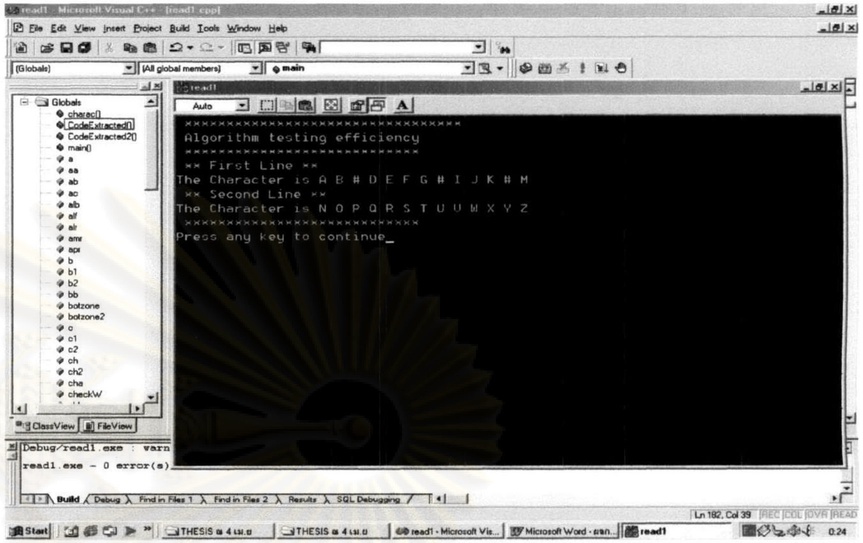
A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Printed test characters



Scanned image characters



```
 ********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is A B # D E F G # I J K # M
 ** Second Line **
The Character is N O P Q R S T U U W X Y Z
 ***************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

C,H and L

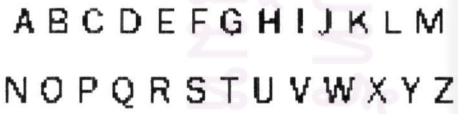
Figure 4-5 Result of Algorithm Type Face Test On Freesial UPC

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Printed test characters

↓

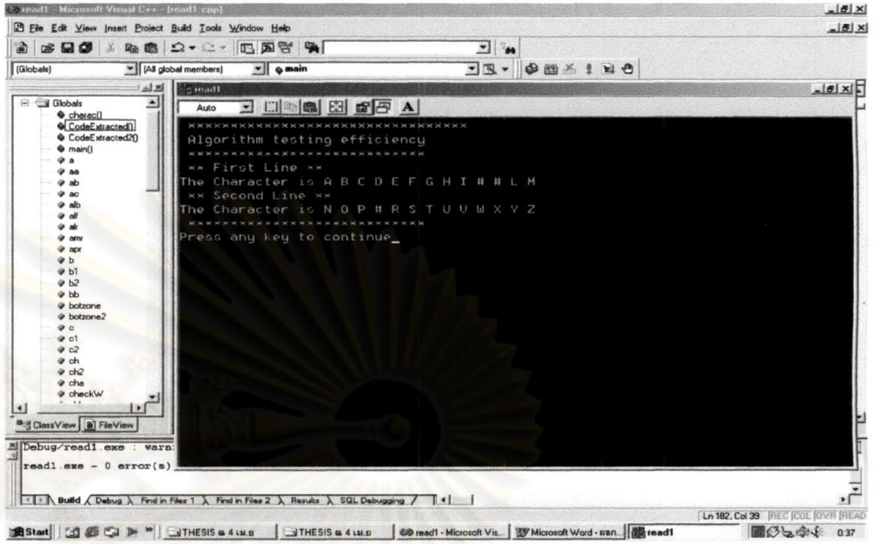


A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Scanned image characters

→



```
 *********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is A B C D E F G H I # # L M
 ** Second Line **
The Character is N O P # R S T U V W X Y Z
 ****************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

J,K and Q

Figure 4-6 Result of Algorithm Type Face Test On Iris UPC

A B C D E F G H I J K L M

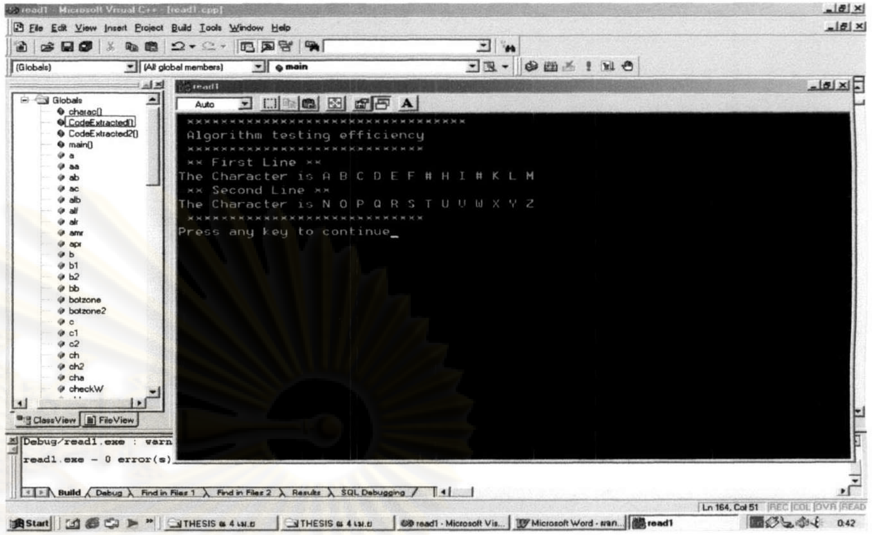
N O P Q R S T U V W X Y Z

Printed test characters

A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

Scanned image characters



```
 ××××××××××××××××××××××××××××××××××
 Algorithm testing efficiency
 ×××××××××××××××××××××××××××
 ×× First Line ××
The Character is A B C D E F # H I # K L M
 ×× Second Line ××
The Character is N O P Q R S T U V W X Y Z
 ×××××××××××××××××××××××××××
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

G and J

Figure 4-7 Result of Algorithm Type Face Test On Jasmine UPC

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

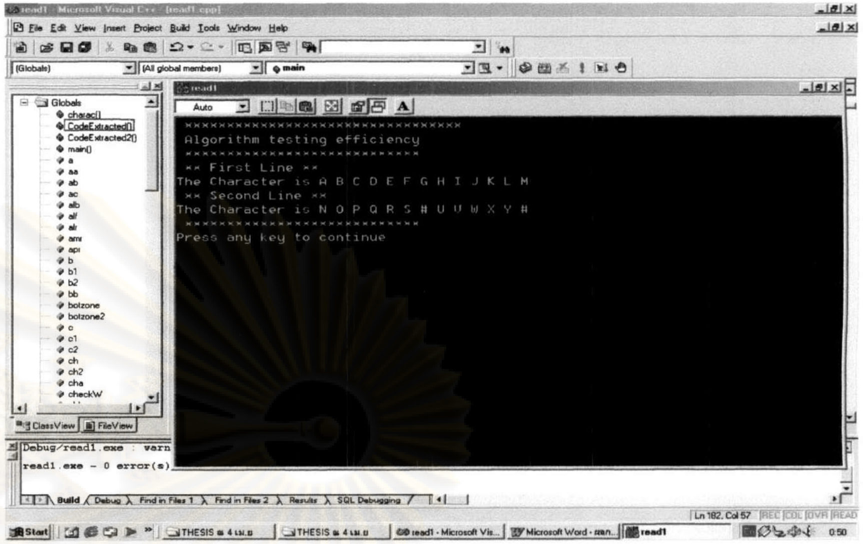
Printed test characters

↓

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Scanned image characters

→



```
 *********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is A B C D E F G H I J K L M
 ** Second Line **
The Character is N O P Q R S # U U W X Y #
 ****************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

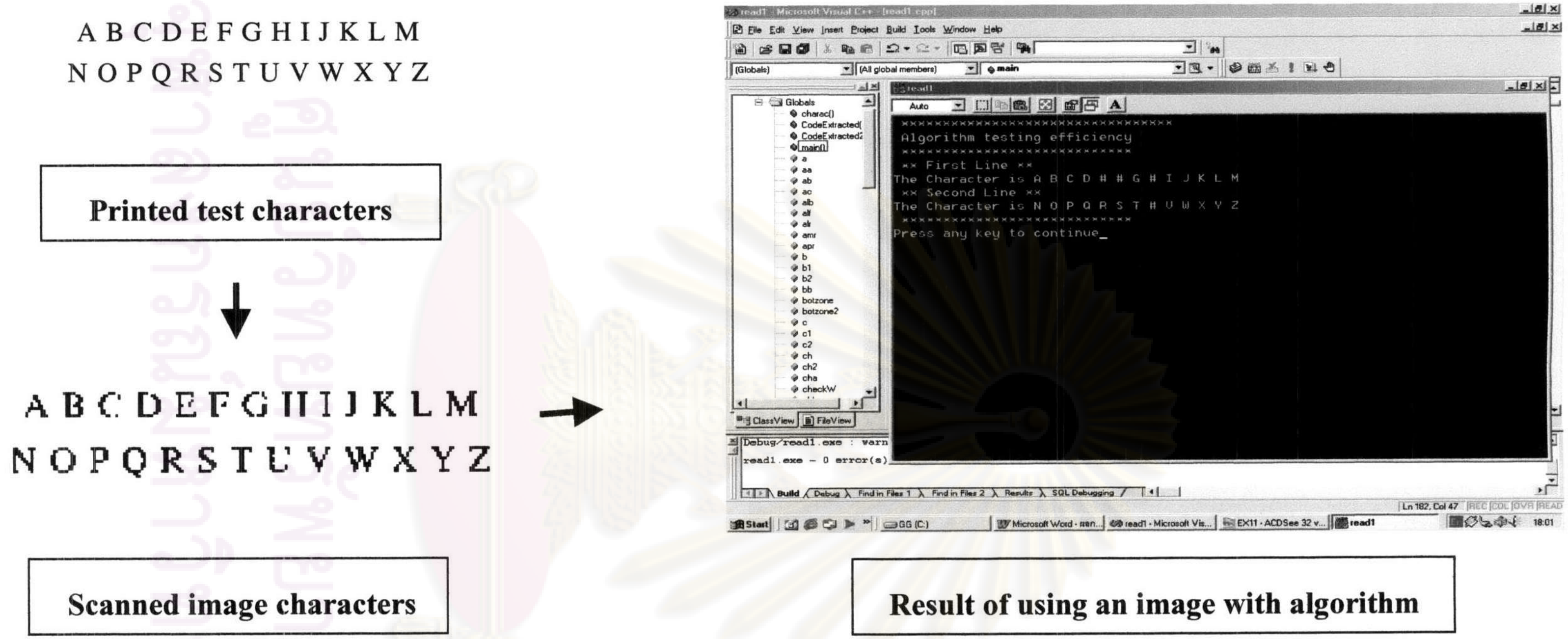
Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

T and Z

Figure 4-8 Result of Algorithm Type Face Test On Kodchiang UPC



A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Printed test characters

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Scanned image characters

```
 *********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is A B C D # # G # I J K L M
 ** Second Line **
The Character is N O P Q R S T # U W X Y Z
 ***************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

E,F,H and U

Figure 4-9 Result of Algorithm Type Face Test On Lily UPC

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

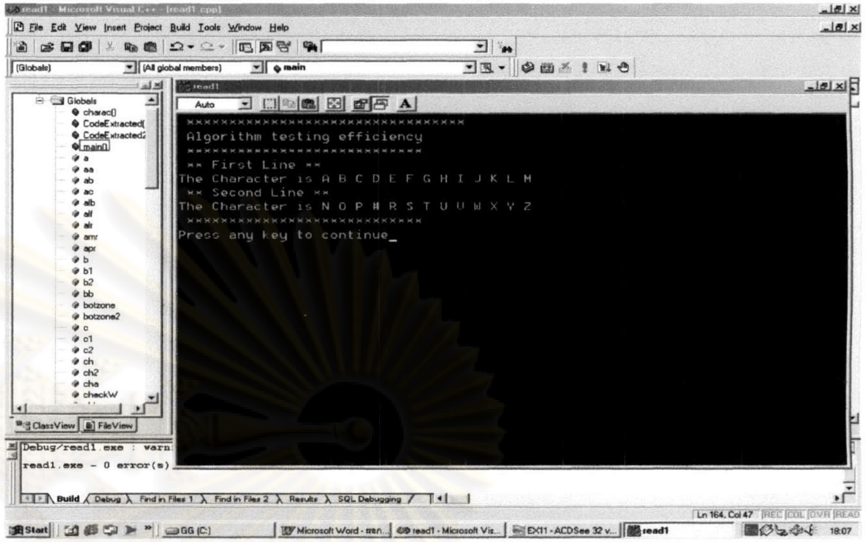
Printed test characters

↓

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

Scanned image characters

→



```
 *********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is A B C D E F G H I J K L M
 ** Second Line **
The Character is N O P # R S T U U W X Y Z
 ****************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result

The characters that cannot be translated because of insufficient scanned shape from scanner.

Q

4.2.2 The Processing Algorithm Test Result

Testing the efficiency of an algorithm to be used with an image which contains characters is a very important part of the entire processing procedure using an algorithm. A document contains 16 identical characters on the page which have been scanned and changed to digital image form. The result of an algorithm operation means the characters have been translated and compared with the whole characters on the document page [the original had 16 characters]. All of the pages were translated in a similar fashion. The algorithm had the ability to correctly translate characters in the image with a working efficiency of 80%. This means, of course, that there is an 80% probability for each character to be properly translated.

The test results mainly depended on the acceptance tolerance in the characters function and the quality of the image tested.(the test results shown on figure 4-10 to 4-35)

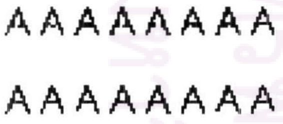
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z

(Test result on character A)

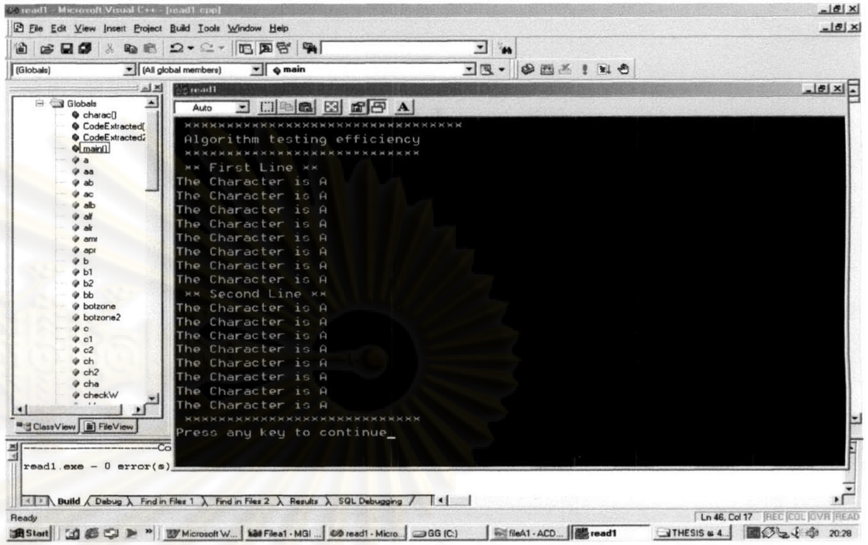
A A A A A A A A

A A A A A A A A

Printed test characters



Scanned image characters



```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Algorithm testing efficiency
xxxxxxxxxxxxxxxxxxxxxxxxxxx
xx First Line xx
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
xx Second Line xx
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
The Character is A
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters A. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

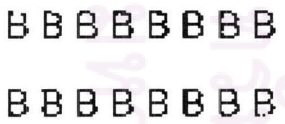
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character B)

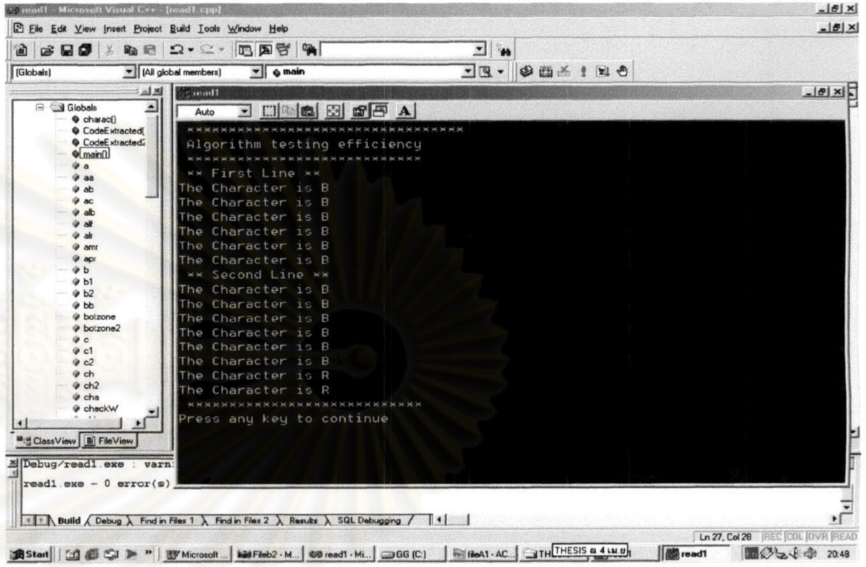
B B B B B B B B

B B B B B B B B

Printed test characters



Scanned image characters



```
 ********************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is B
The Character is B
The Character is B
The Character is B
The Character is B
The Character is B
 ** Second Line **
The Character is B
The Character is B
The Character is B
The Character is B
The Character is B
The Character is B
The Character is R
The Character is R
 **************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters B. It can read out **12** characters, as displayed. This gives the extracting algorithm efficiency of **75%.**

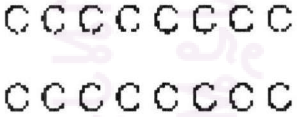
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character C)

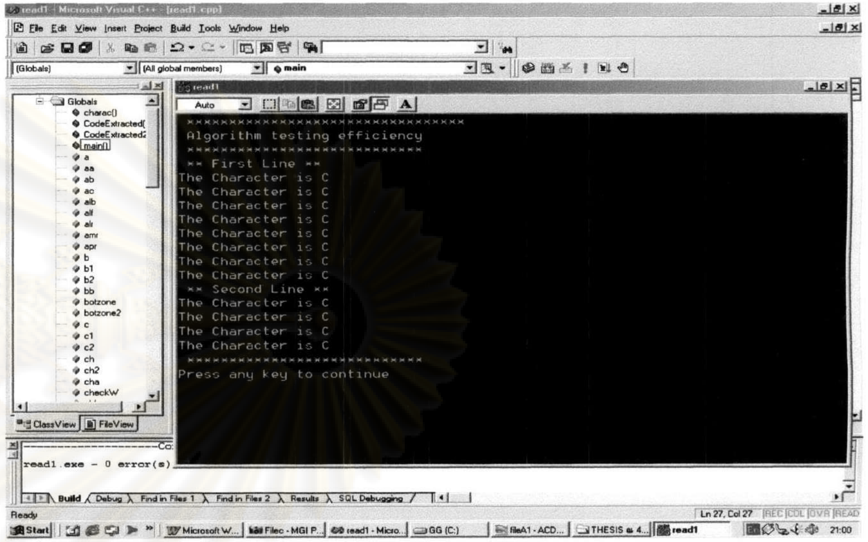
C C C C C C C C

C C C C C C C C

Printed test characters



Scanned image characters



```
 ********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is C
The Character is C
The Character is C
The Character is C
The Character is C
The Character is C
The Character is C
The Character is C
 ** Second Line **
The Character is C
The Character is C
The Character is C
The Character is C
 ***************************
Press any key to continue
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters C. It can read out **12** characters, as displayed. This gives the extracting algorithm efficiency of **75%.**

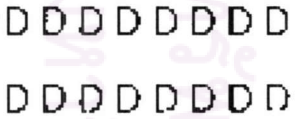
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character D)

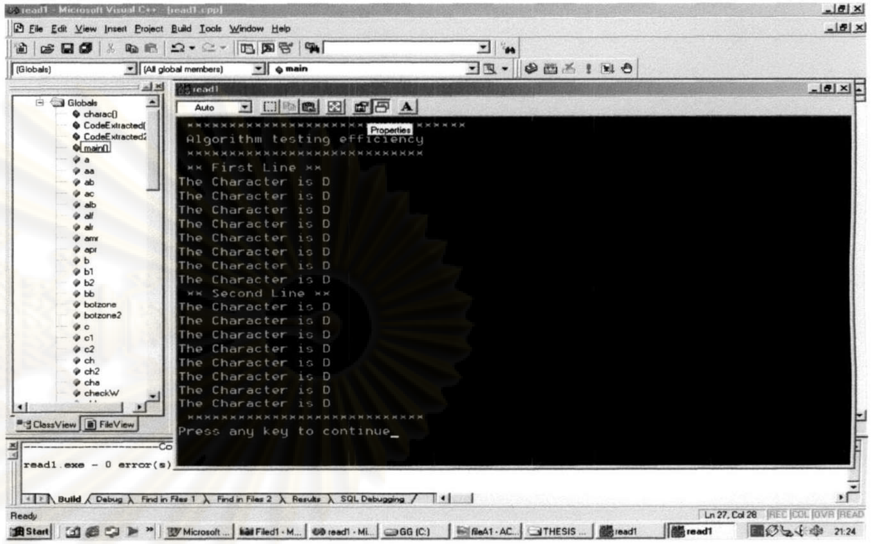
D D D D D D D D

D D D D D D D D

Printed test characters



Scanned image characters



```
 *************************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
 ** Second Line **
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
The Character is D
 ****************************
Press any key to continue_
```

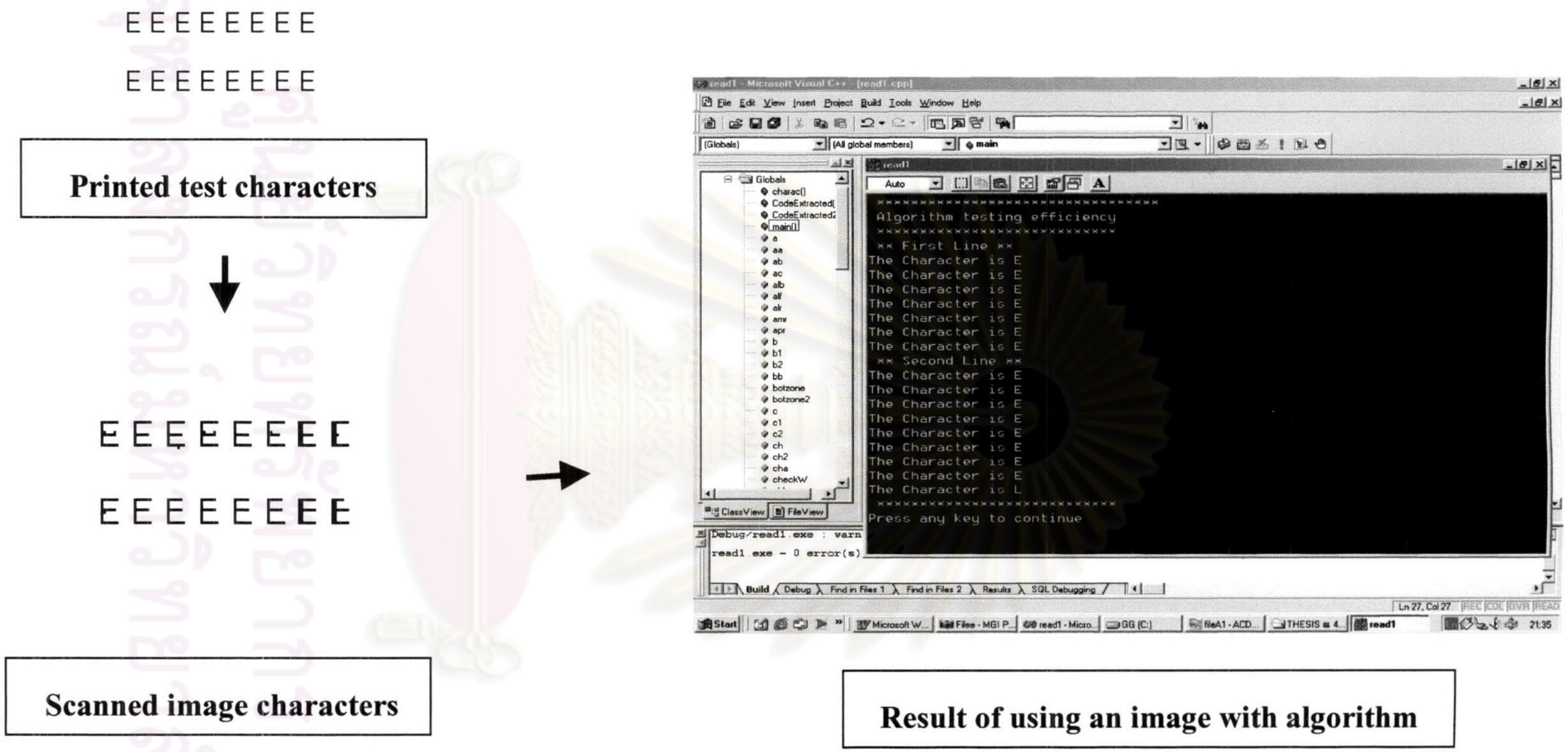
```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters D. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character E)



E E E E E E E E

E E E E E E E E

Printed test characters

E E E E E E E E

E E E E E E E E

Scanned image characters

```
 *******************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
 ** Second Line **
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
The Character is E
The Character is L
 ****************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters E. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

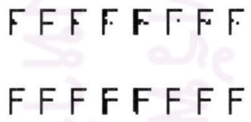
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character F)

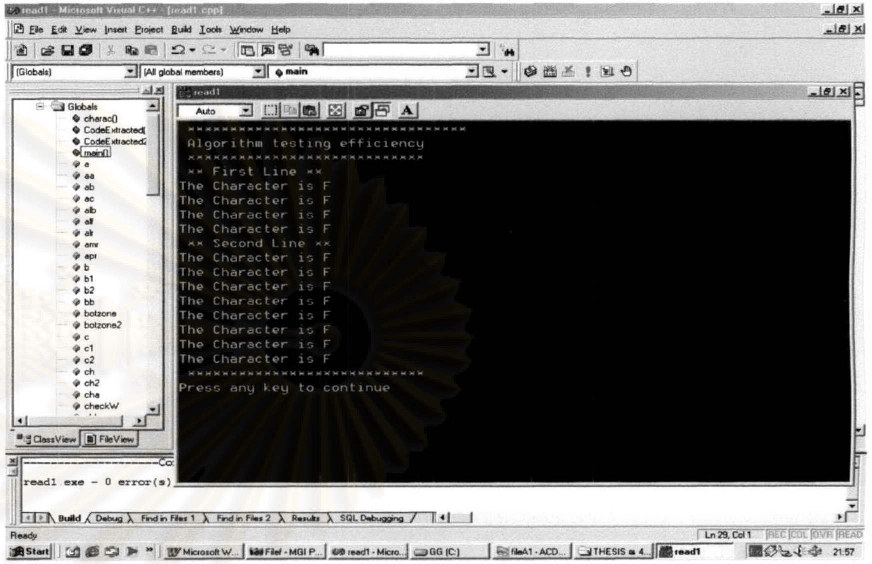
F F F F F F F F

F F F F F F F F

Printed test characters



Scanned image characters



```
******************************
 Algorithm testing efficiency
*************************
 ** First Line **
The Character is F
The Character is F
The Character is F
The Character is F
 ** Second Line **
The Character is F
The Character is F
The Character is F
The Character is F
The Character is F
The Character is F
The Character is F
The Character is F
*************************
Press any key to continue
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters F. It can read out **12** characters, as displayed. This gives the extracting algorithm efficiency of **75%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character G)

G G G G G G G G

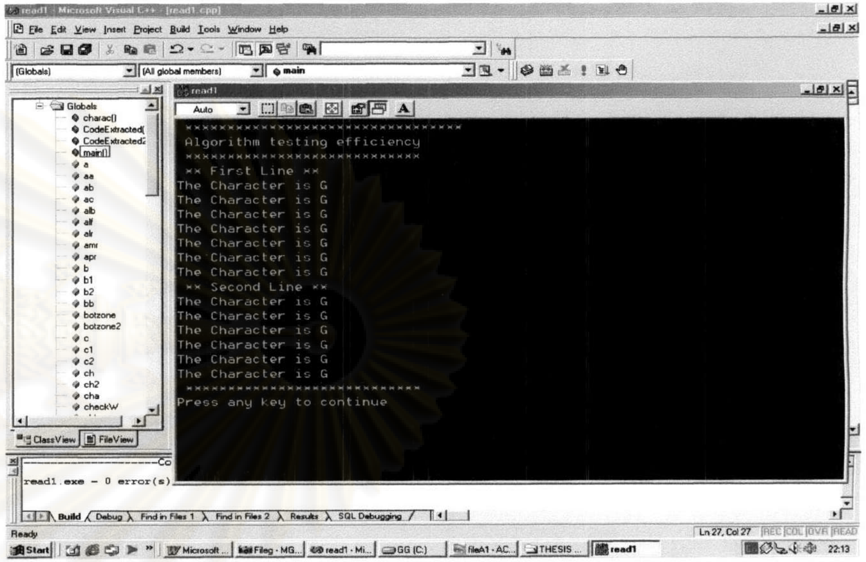
G G G G G G G G

Printed test characters

G G G G G G G G

G G G G G G G G

Scanned image characters



```
 ******************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is G
The Character is G
The Character is G
The Character is G
The Character is G
The Character is G
The Character is G
 ** Second Line **
The Character is G
The Character is G
The Character is G
The Character is G
The Character is G
The Character is G
 **************************
Press any key to continue
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters G. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

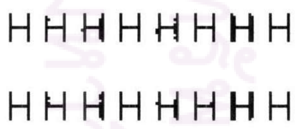
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character H)

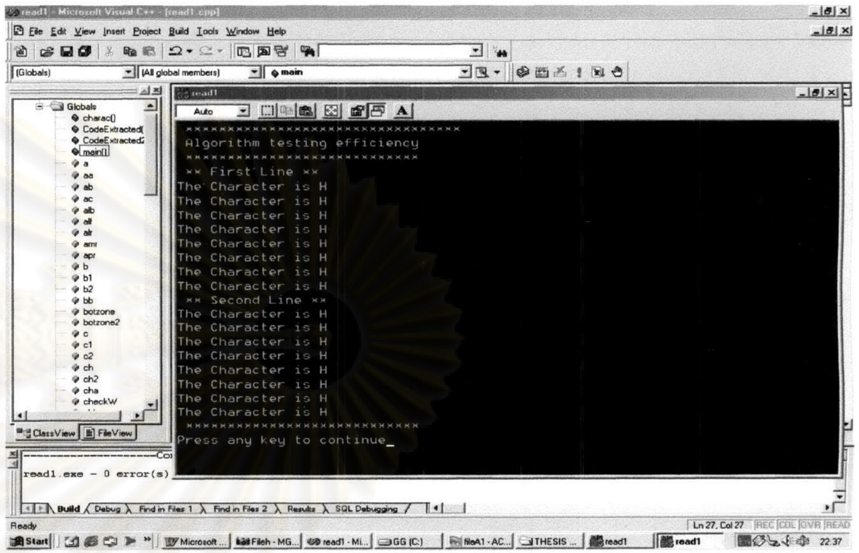
H H H H H H H H

H H H H H H H H

Printed test characters



Scanned image characters



```
 *********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
 ** Second Line **
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
The Character is H
 ***************************
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters H. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

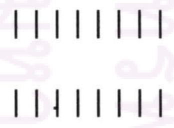
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character I)

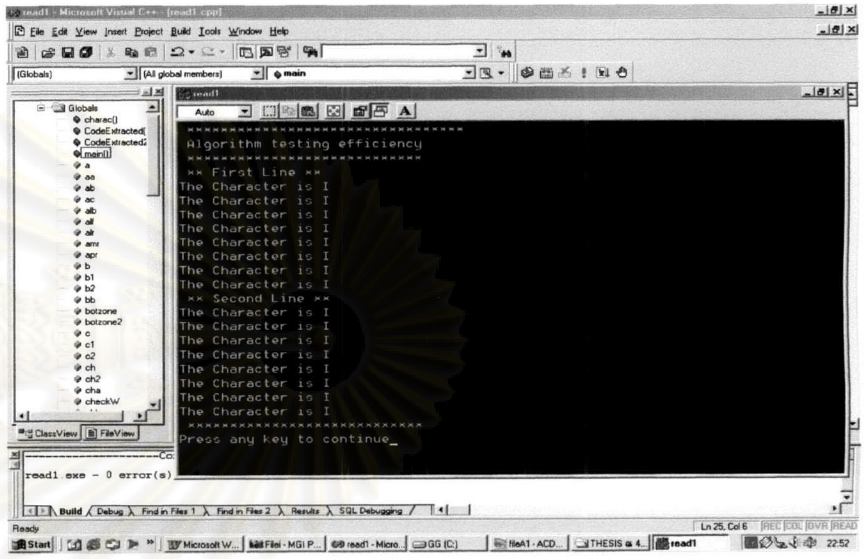
IIIIIIII

IIIIIIII

Printed test characters



Scanned image characters



```
 ******************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
 ** Second Line **
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
The Character is I
 ***************************
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters I. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

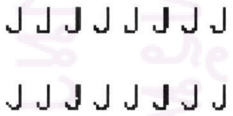
(Test result on character J)

J J J J J J J J

J J J J J J J J

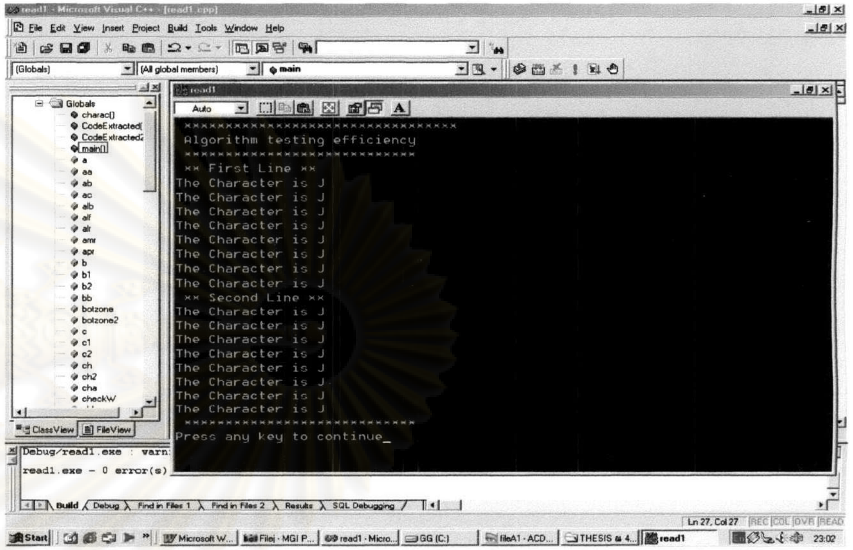
Printed test characters

↓



Scanned image characters

→



```
******************************
Algorithm testing efficiency
*************************
** First Line **
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
** Second Line **
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
The Character is J
*************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters J. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

Figure **4-10** Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character K)

K K K K K K K K

K K K K K K K K

Printed test characters

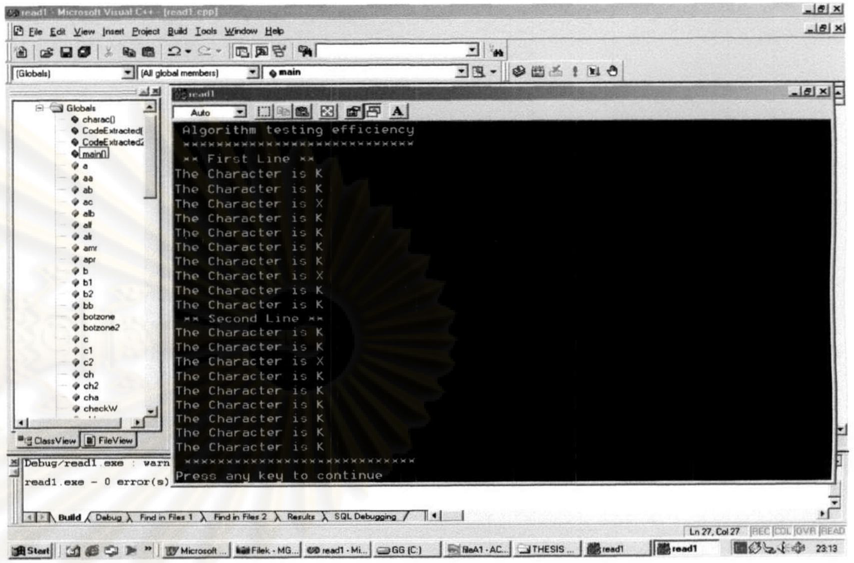
↓

K K K K K K K K

K K K K K K K K

Scanned image characters

→



```
Algorithm testing efficiency
****************************
** First Line **
The Character is K
The Character is K
The Character is X
The Character is K
The Character is K
The Character is K
The Character is K
The Character is X
The Character is K
The Character is K
** Second Line **
The Character is K
The Character is K
The Character is X
The Character is K
The Character is K
The Character is K
The Character is K
The Character is K
The Character is K
****************************
Press any key to continue
```

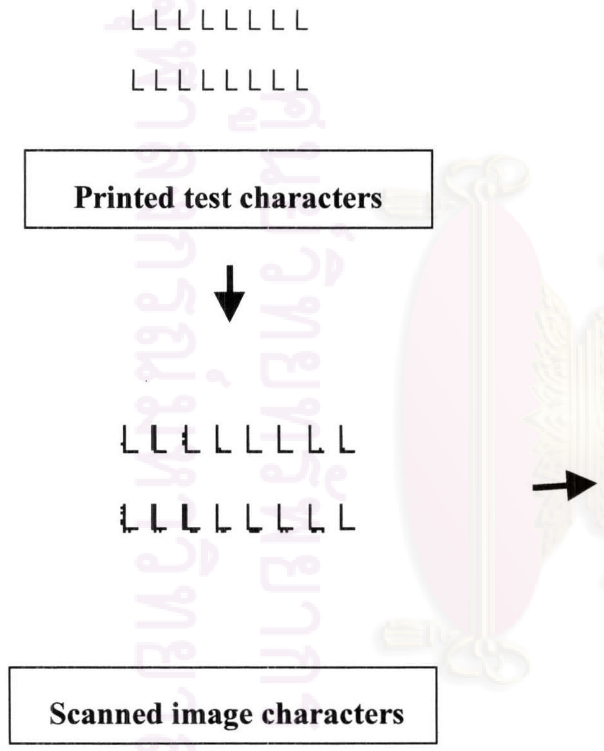
```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

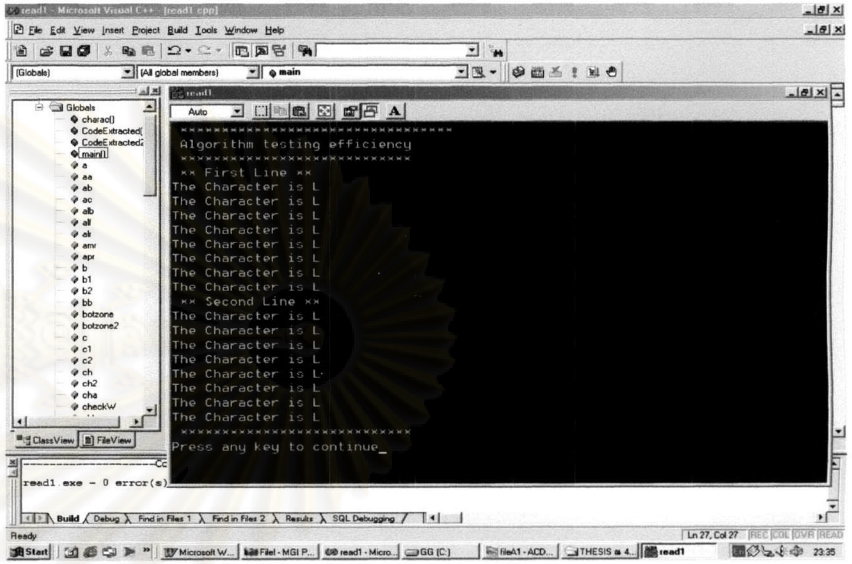
Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters K. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character L)





```
 ****************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
 ** Second Line **
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
The Character is L
 **************************
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Testing result of the scanning process, using algorithm on image, which contain 16 characters L. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

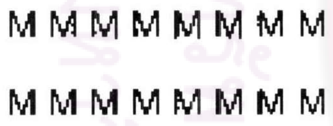
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character M)

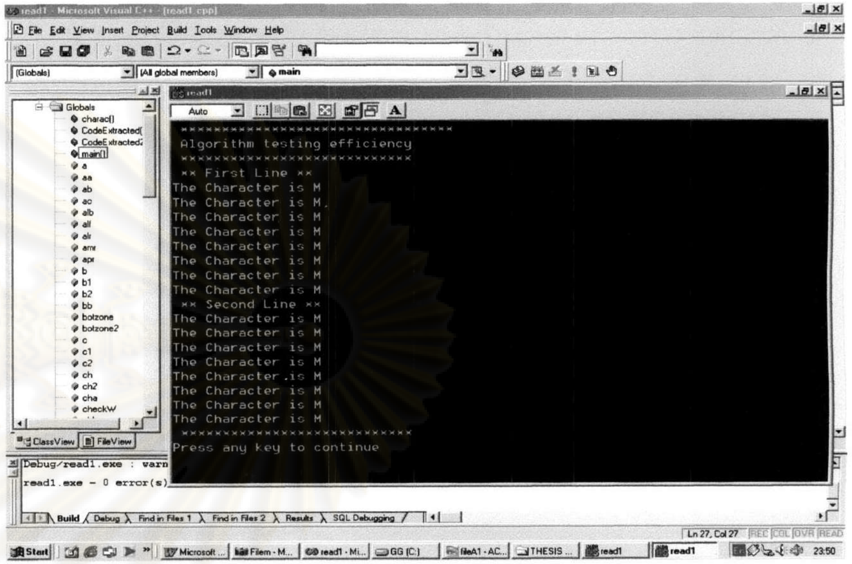
M M M M M M M M

M M M M M M M M

Printed test characters



Scanned image characters



```
*********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
 ** Second Line **
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
The Character is M
 ***************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters M. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

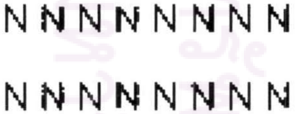
(Test result on character N)

N N N N N N N N

N N N N N N N N

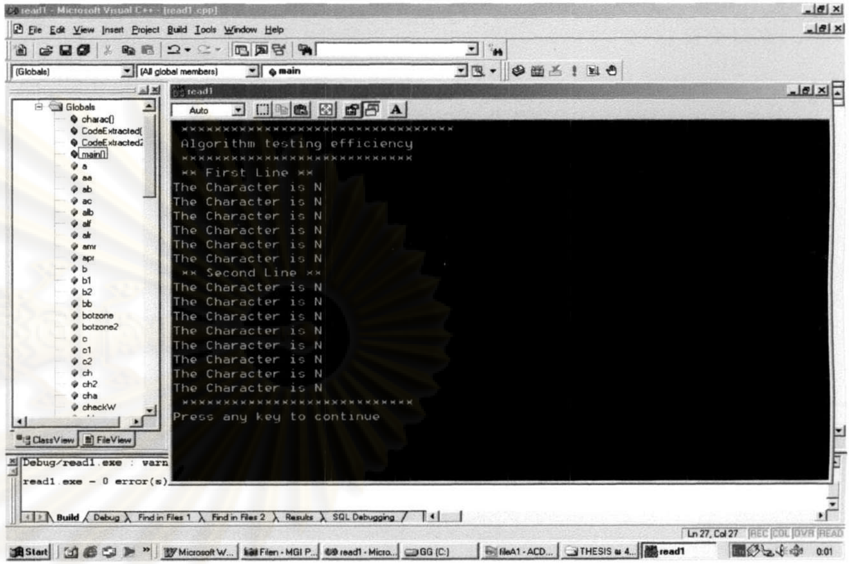
Printed test characters

↓



Scanned image characters

→



```
××××××××××××××××××××××××××××××××
 Algorithm testing efficiency
××××××××××××××××××××××××××××
 ×× First Line ××
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
 ×× Second Line ××
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
The Character is N
××××××××××××××××××××××××××××
Press any key to continue
```

```
Debug/read1.exe : varn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters N. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

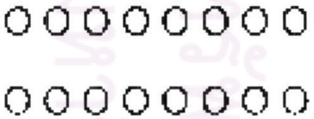
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character O)

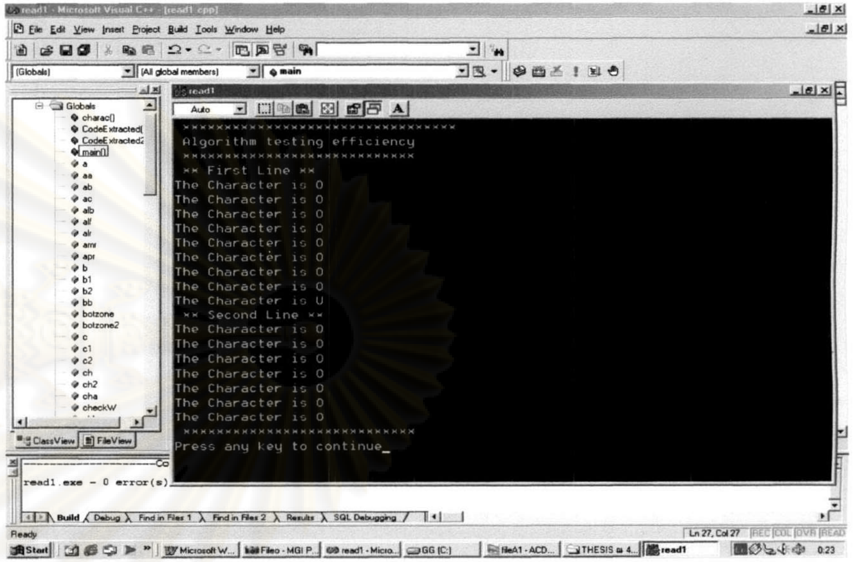
O O O O O O O O

O O O O O O O O

Printed test characters



Scanned image characters



```
 ********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is U
 ** Second Line **
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
The Character is O
 ***************************
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters O. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

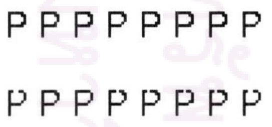
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character P)

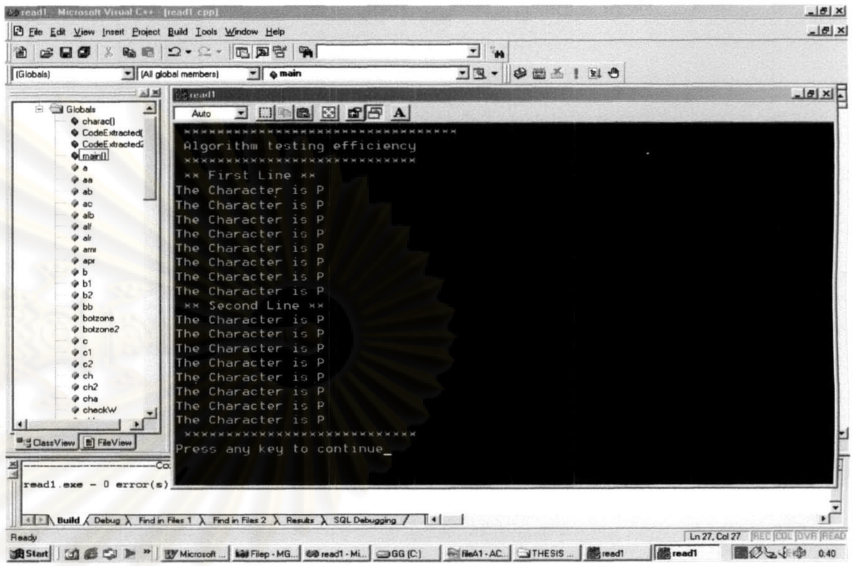
P P P P P P P P

P P P P P P P P

Printed test characters



Scanned image characters



```
 ************************************
 Algorithm testing efficiency
 *****************************
 ** First Line **
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
 ** Second Line **
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
The Character is P
 *****************************
Press any key to continue_
```

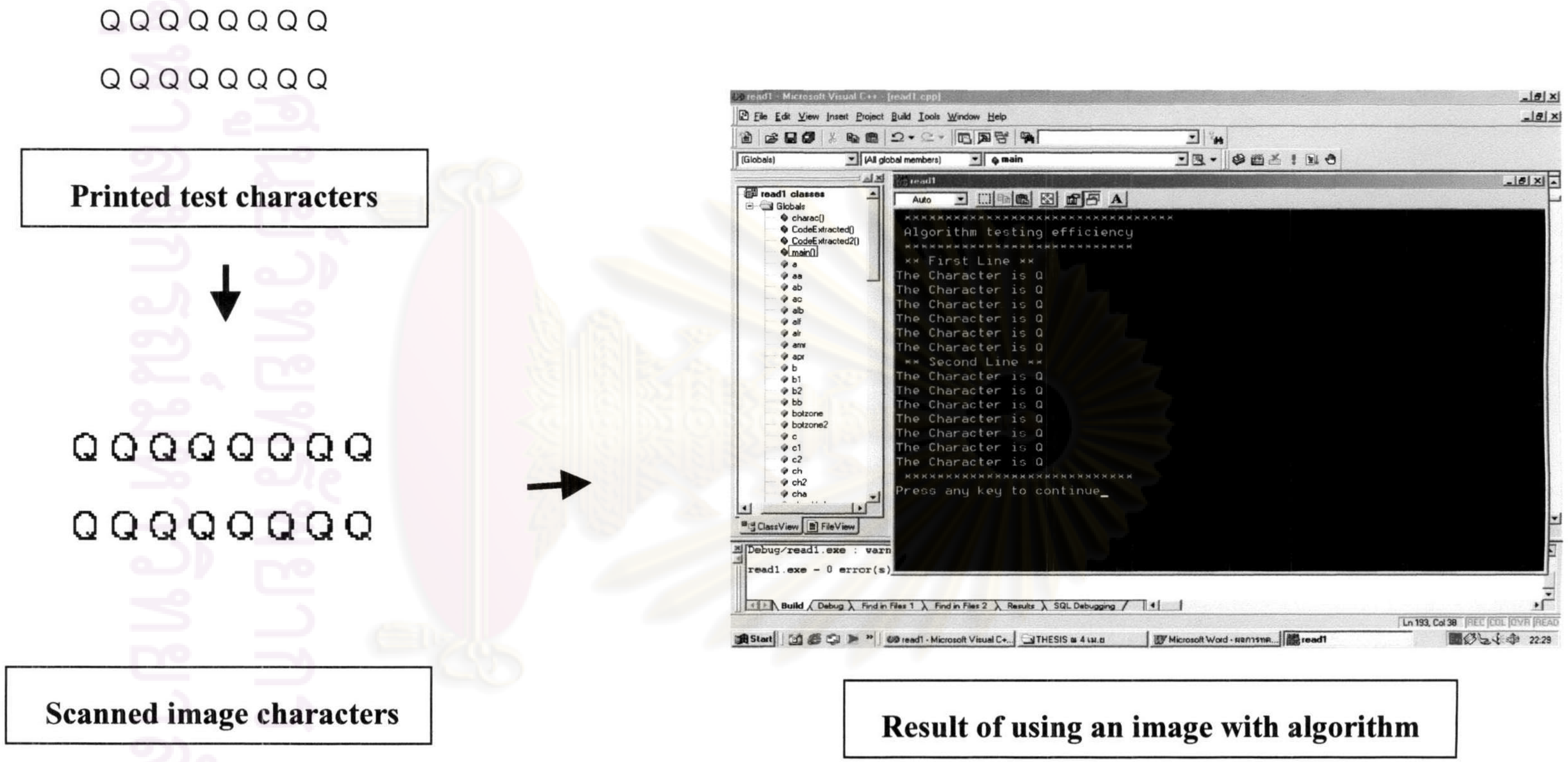
```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters P. It can read out **16** characters, as displayed. This gives the extracting algorithm efficiency of **100%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character Q)



Q Q Q Q Q Q Q Q

Q Q Q Q Q Q Q Q

Printed test characters

Scanned image characters

```
****************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is Q
The Character is Q
The Character is Q
The Character is Q
The Character is Q
The Character is Q
 ** Second Line **
The Character is Q
The Character is Q
The Character is Q
The Character is Q
The Character is Q
The Character is Q
The Character is Q
 **************************
Press any key to continue_
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters Q. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

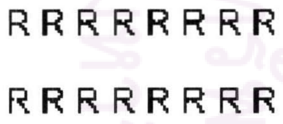
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character R)

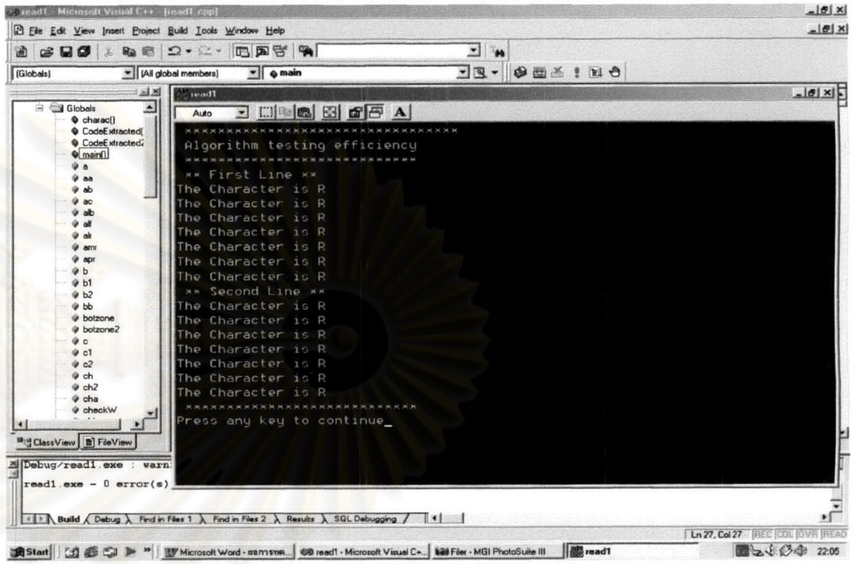
R R R R R R R R

R R R R R R R R

Printed test characters



Scanned image characters



```
****************************
Algorithm testing efficiency
****************************
** First Line **
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
** Second Line **
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
The Character is R
****************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters R. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

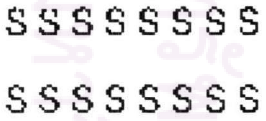
Figure **4-10** Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character S)

S S S S S S S S

S S S S S S S S

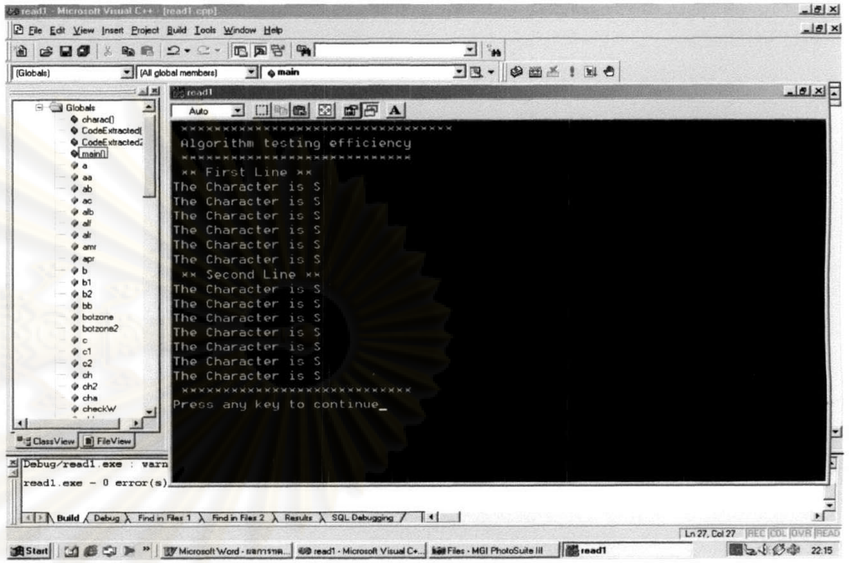
Printed test characters



S S S S S S S S

S S S S S S S S

Scanned image characters



```
*******************************
Algorithm testing efficiency
***************************
** First Line **
The Character is S
The Character is S
The Character is S
The Character is S
The Character is S
The Character is S
** Second Line **
The Character is S
The Character is S
The Character is S
The Character is S
The Character is S
The Character is S
The Character is S
**************************
Press any key to continue_
```

```
Debug/read1.exe : varn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters S. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character T)

T T T T T T T T

T T T T T T T T

Printed test characters

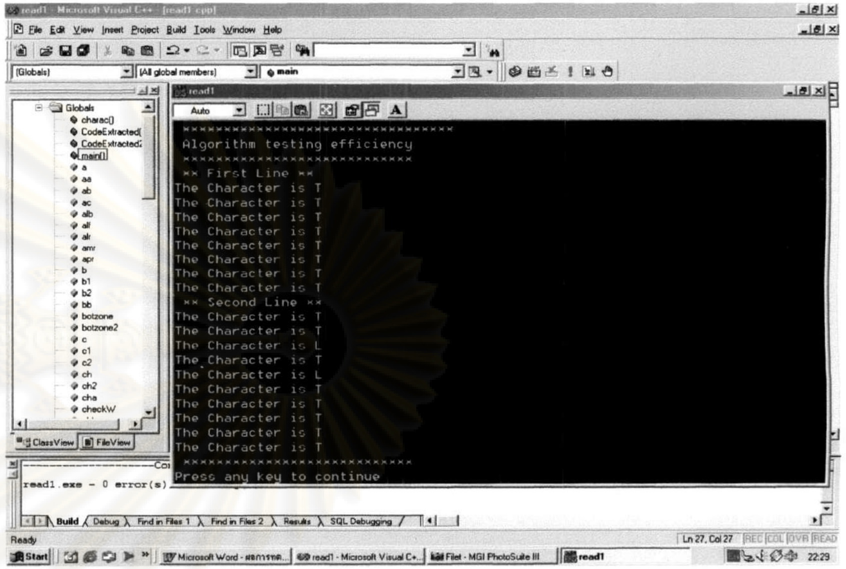
↓

T T T T T T T T

T T T T T T T T

Scanned image characters

→



```
****************************
Algorithm testing efficiency
**************************
** First Line **
The Character is T
The Character is T
The Character is T
The Character is T
The Character is T
The Character is T
The Character is T
The Character is T
** Second Line **
The Character is T
The Character is T
The Character is L
The Character is T
The Character is L
The Character is T
The Character is T
The Character is T
The Character is T
**************************
Press any key to continue
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters T. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

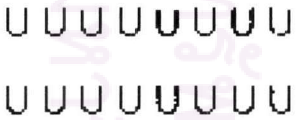
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character U)

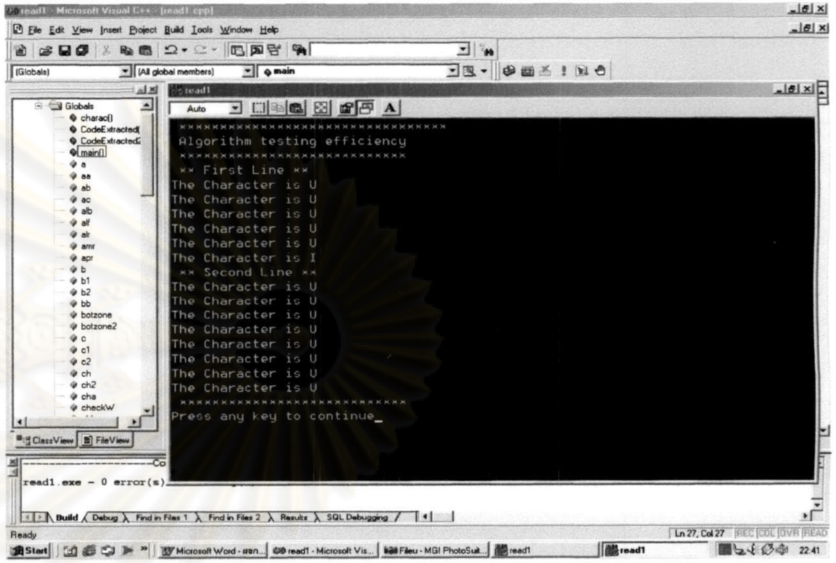
U U U U U U U U

U U U U U U U U

Printed test characters



Scanned image characters



```
 ********************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is I
 ** Second Line **
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
 ***************************
Press any key to continue_
```

```
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters U. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character V)

V V V V V V V V

V V V V V V V V

Printed test characters

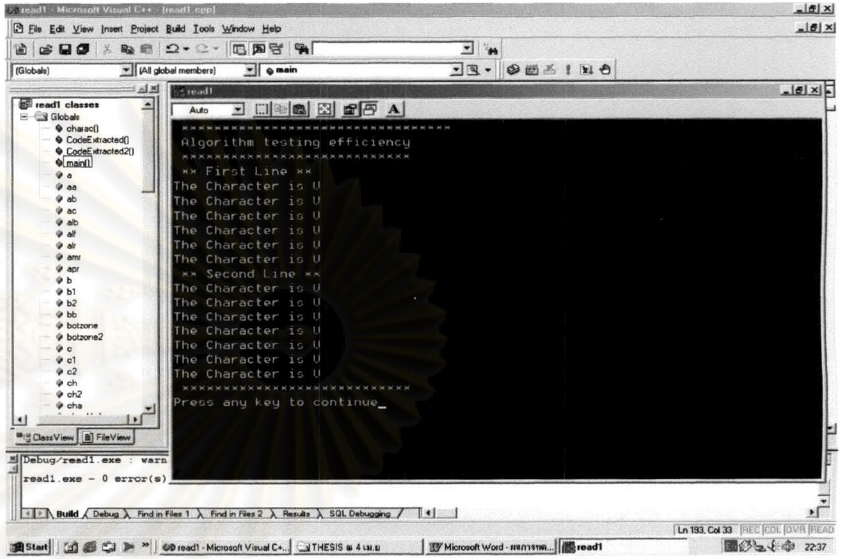
↓

V V V V V V V V

V V V V V V V V

Scanned image characters

→



```
 ********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
 ** Second Line **
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
The Character is U
 ****************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters V. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

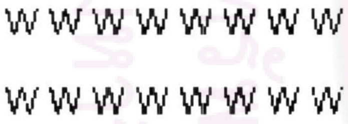
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character W)

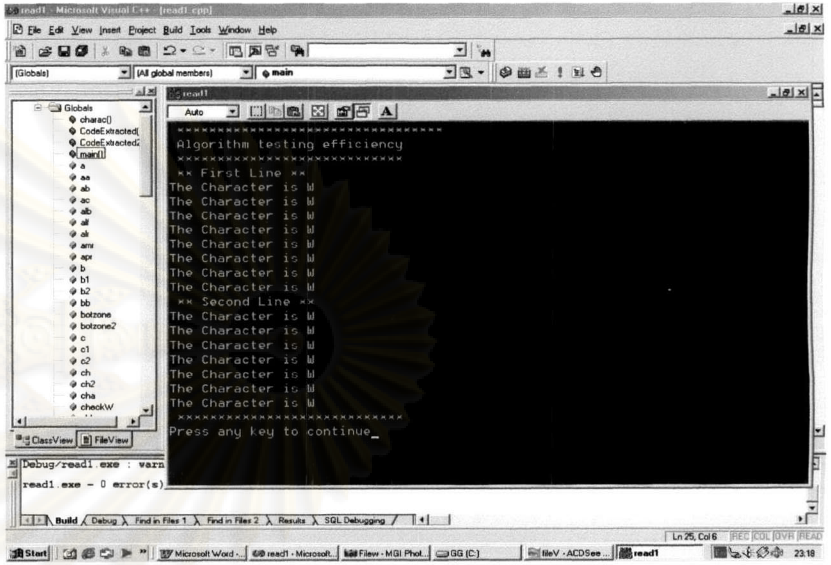
W W W W W W W W

W W W W W W W W

Printed test characters



Scanned image characters



```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

```
 *********************************
 Algorithm testing efficiency
 ****************************
 ** First Line **
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
 ** Second Line **
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
The Character is W
 ****************************
Press any key to continue_
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters W. It can read out **15** characters, as displayed. This gives the extracting algorithm efficiency of **94%.**

Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character X)

XXXXXXXX

XXXXXXXX

Printed test characters

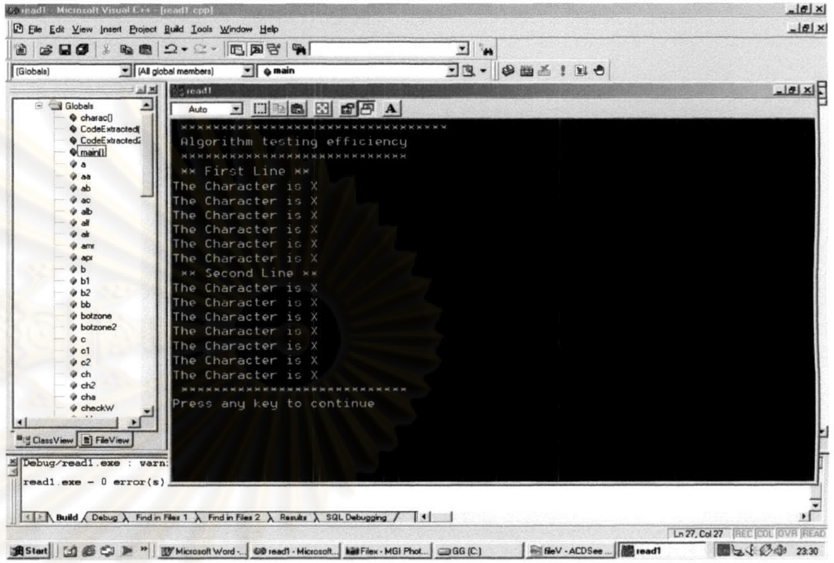
↓

XXXXXXXX

XXXXXXXX

Scanned image characters

→



```
****************************
Algorithm testing efficiency
**************************
 ** First Line **
The Character is X
The Character is X
The Character is X
The Character is X
The Character is X
The Character is X
 ** Second Line **
The Character is X
The Character is X
The Character is X
The Character is X
The Character is X
The Character is X
The Character is X
**************************
Press any key to continue
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters X. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

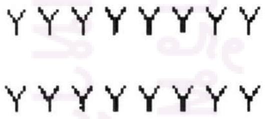
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character Y)

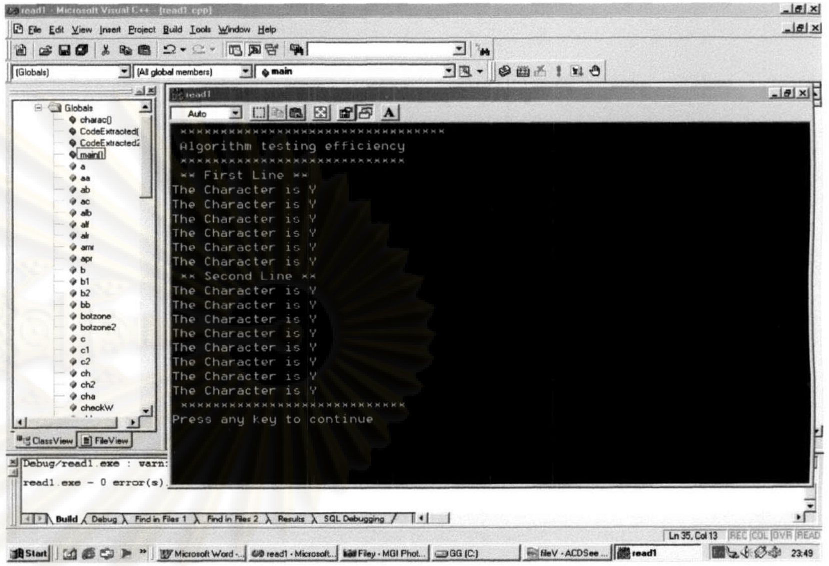
Y Y Y Y Y Y Y Y

Y Y Y Y Y Y Y Y

Printed test characters



Scanned image characters



```
 ****************************
 Algorithm testing efficiency
 **************************
 ** First Line **
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
 ** Second Line **
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
The Character is Y
 **************************
Press any key to continue
```

```
Debug/read1.exe : warn:
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters Y. It can read out **14** characters, as displayed. This gives the extracting algorithm efficiency of **88%.**

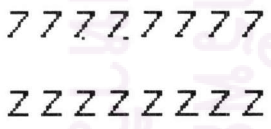
Figure 4-10 Algorithm Processing Test Efficiency On Character A to Z (continue)

(Test result on character Z)

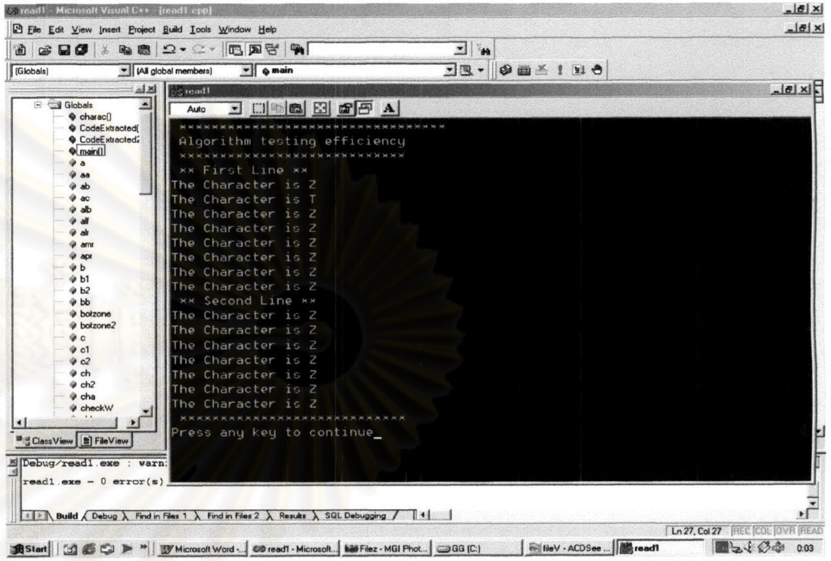
Z Z Z Z Z Z Z Z

Z Z Z Z Z Z Z Z

Printed test characters



Scanned image characters



```
 *******************************
 Algorithm testing efficiency
 ***************************
 ** First Line **
The Character is Z
The Character is T
The Character is Z
The Character is Z
The Character is Z
The Character is Z
The Character is Z
The Character is Z
 ** Second Line **
The Character is Z
The Character is Z
The Character is Z
The Character is Z
The Character is Z
The Character is Z
The Character is Z
 ***************************
Press any key to continue_
```

```
Debug/read1.exe : warn
read1.exe - 0 error(s)
```

Result of using an image with algorithm

Testing result of the scanning process, using algorithm on image, which contain 16 characters Z. It can read out **13** characters, as displayed. This gives the extracting algorithm efficiency of **81%.**

Table 4-5 The result of algorithm processing test efficiency

Characters	The whole characters on the document page	The characters that could be translated	Translated efficiency
A	16	16	100%
B	16	12	75%
C	16	12	75%
D	16	16	100%
E	16	14	88%
F	16	12	75%
G	16	13	81%
H	16	16	100%
I	16	16	100%
J	16	16	100%
K	16	13	81%
L	16	16	100%
M	16	16	100%
N	16	14	88%
O	16	14	88%
P	16	16	100%
Q	16	13	81%
R	16	14	88%
S	16	13	81%
T	16	14	88%
U	16	13	81%
V	16	13	81%

Table 4-5 The result of algorithm processing test efficiency (continue)

Characters	The whole characters on the document page	The characters that could be translated	Translated efficiency
W	16	15	94%
X	16	13	81%
Y	16	14	88%
Z	16	13	81%

Figure 4-11 An Example Of Algorithm Test Results

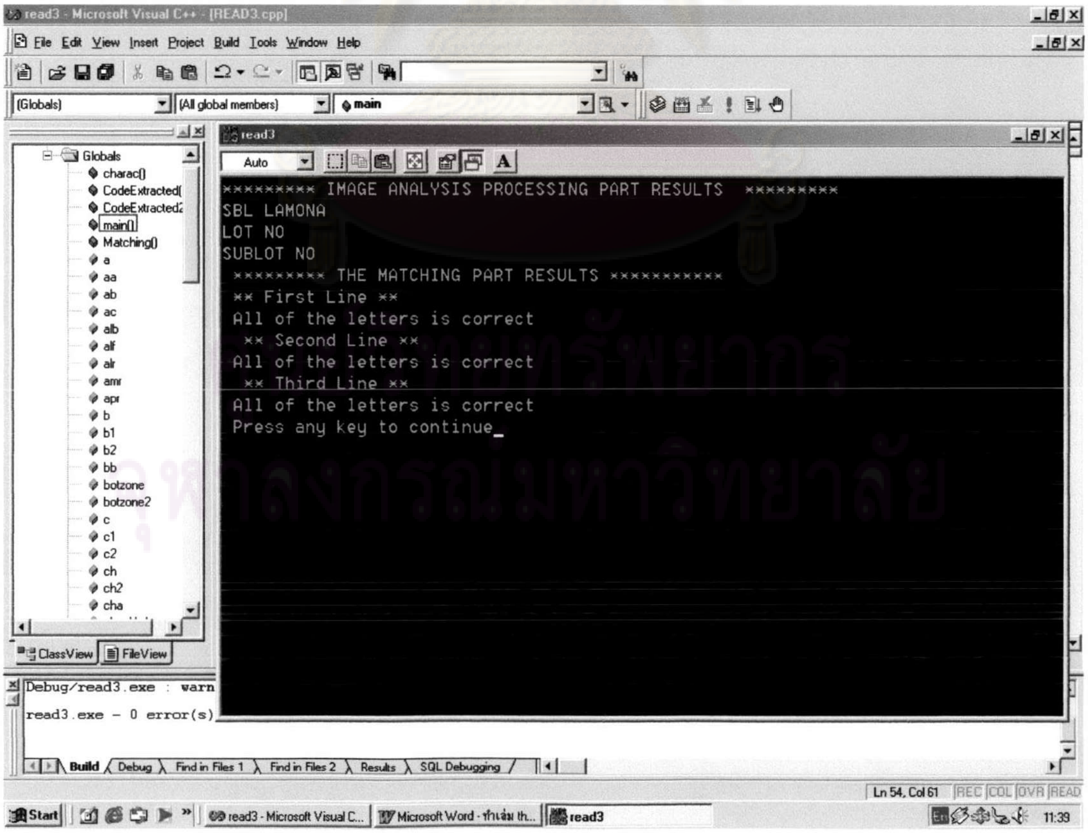
Image Test File Information (Test sheet No.1)

Image File Name : WB-001 Image Dimension : (Widths x Lengths) 386x229 Pixel

Image Size : 259.5 KB No Of Character Lines: 3 Total Of Character : 22

Characters Image Files (WB-001.bmp)	Original Text Files (WB-001.txt)
SBL LAMONA	SBL LAMONA
LOT NO	LOT NO
SUBLOT NO	SUBLOT NO

Image Analysis Processing And Matching Algorithm



```
********* IMAGE ANALYSIS PROCESSING PART RESULTS *********
SBL LAMONA
LOT NO
SUBLOT NO
 ********* THE MATCHING PART RESULTS ***********
 ** First Line **
 All of the letters is correct
  ** Second Line **
 All of the letters is correct
  ** Third Line **
 All of the letters is correct
 Press any key to continue_
```

```
Debug/read3.exe : warn
read3.exe - 0 error(s)
```

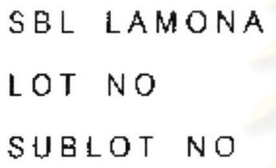
Figure 4-11 An Example Of Algorithm Test Results (continue)

Image Test File Information (Test sheet No.2)

Image File Name : WB-001 Image Dimension : (Widths x Lengths) 386x229 Pixel

Image Size : 259.5 KB No Of Characters Lines: 3 Total Of Characters : 22

Characters Image Files (WB-001.bmp)



SBL LAMONA

LOT NO

SUBLOT NO

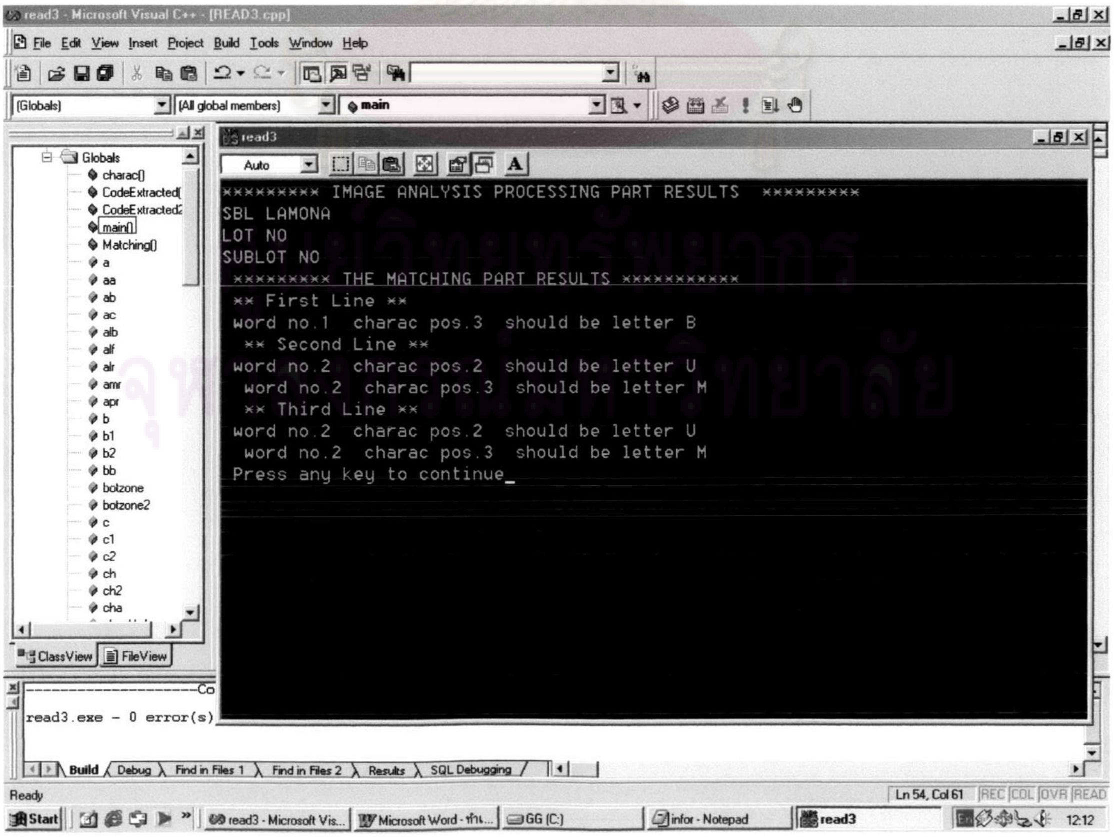
Original Text Files (WB-001-1.txt)

SBB LAMONA

LOT NUM

SUBLOT NUM

Image Analysis Processing And Matching Algorithm



```
********* IMAGE ANALYSIS PROCESSING PART RESULTS  *********
SBL LAMONA
LOT NO
SUBLOT NO
 ********* THE MATCHING PART RESULTS ***********
 ** First Line **
 word no.1  charac pos.3  should be letter B
  ** Second Line **
 word no.2  charac pos.2  should be letter U
  word no.2  charac pos.3  should be letter M
  ** Third Line **
 word no.2  charac pos.2  should be letter U
  word no.2  charac pos.3  should be letter M
 Press any key to continue_
```

```
--------------------Co
read3.exe - 0 error(s)
```

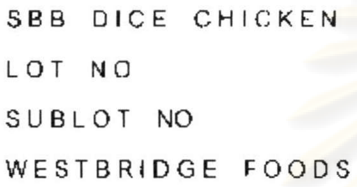
Figure 4-11 An Example Of Algorithm Test Results (continue)

Image Test File Information (Test sheet No.3)

Image File Name : WB-002 Image Dimension : (Widths x Lengths) 386x229 Pixel

Image Size : 259.5 KB No Of Characters Lines: 4 Total Of Characters : 42

Characters Image Files (WB-002.bmp)

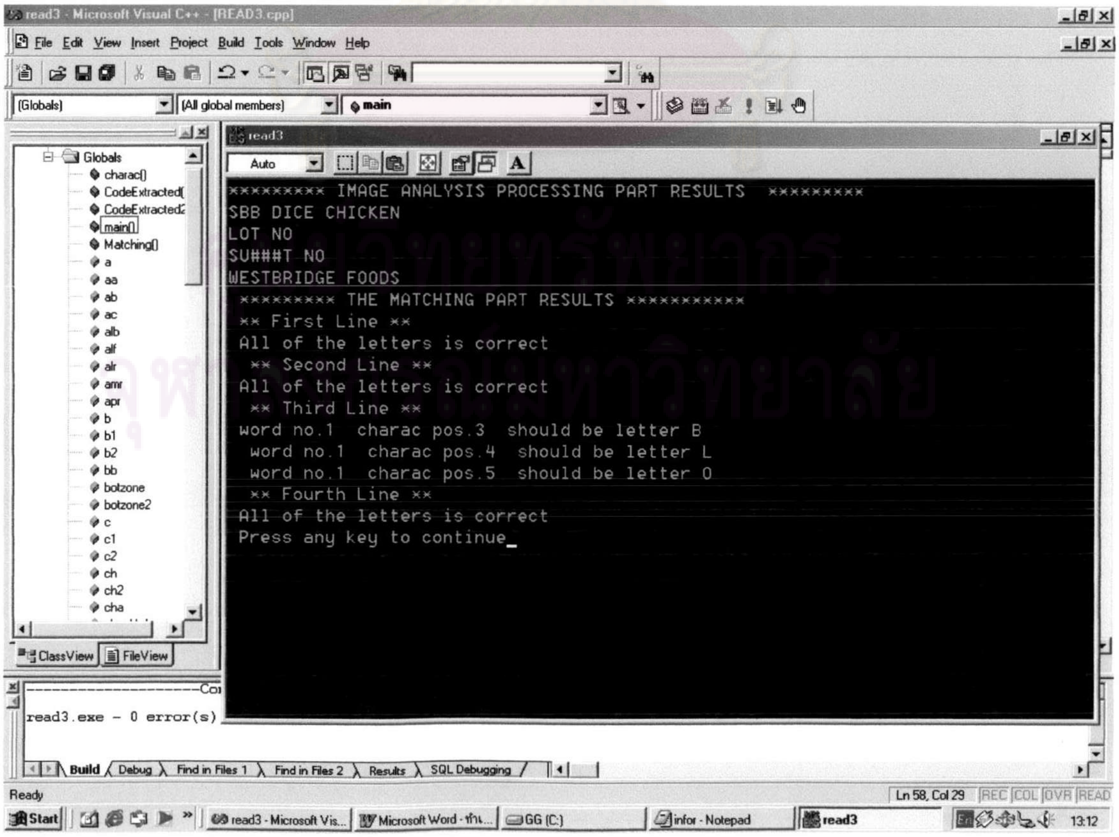


SBB DICE CHICKEN
LOT NO
SUBLOT NO
WESTBRIDGE FOODS

Original Text Files (WB-002.txt)

SBB DICE CHICKEN
LOT NO
SUBLOT NO
WESTBRIDGE FOODS

Image Analysis Processing And Matching Algorithm



```
********* IMAGE ANALYSIS PROCESSING PART RESULTS  *********
SBB DICE CHICKEN
LOT NO
SU###T NO
WESTBRIDGE FOODS
 ********* THE MATCHING PART RESULTS ***********
 ** First Line **
 All of the letters is correct
  ** Second Line **
 All of the letters is correct
  ** Third Line **
 word no.1  charac pos.3  should be letter B
  word no.1  charac pos.4  should be letter L
  word no.1  charac pos.5  should be letter O
  ** Fourth Line **
 All of the letters is correct
 Press any key to continue_
```

```
-------------------Co
read3.exe - 0 error(s)
```

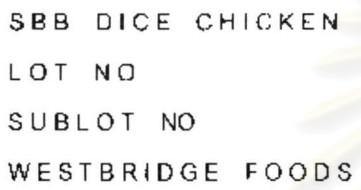
Figure 4-11 An Example Of Algorithm Test Results (continue)

Image Test File Information (Test sheet No.4)

Image File Name : WB-002 Image Dimension : (Widths x Lengths) 386x229 Pixel

Image Size : 259.5 KB No Of Characters Lines: 4 Total Of Characters : 42

Characters Image Files (WB-002.bmp)



SBB DICE CHICKEN
LOT NO
SUBLOT NO
WESTBRIDGE FOODS

Original Text Files (WB-002-1.txt)

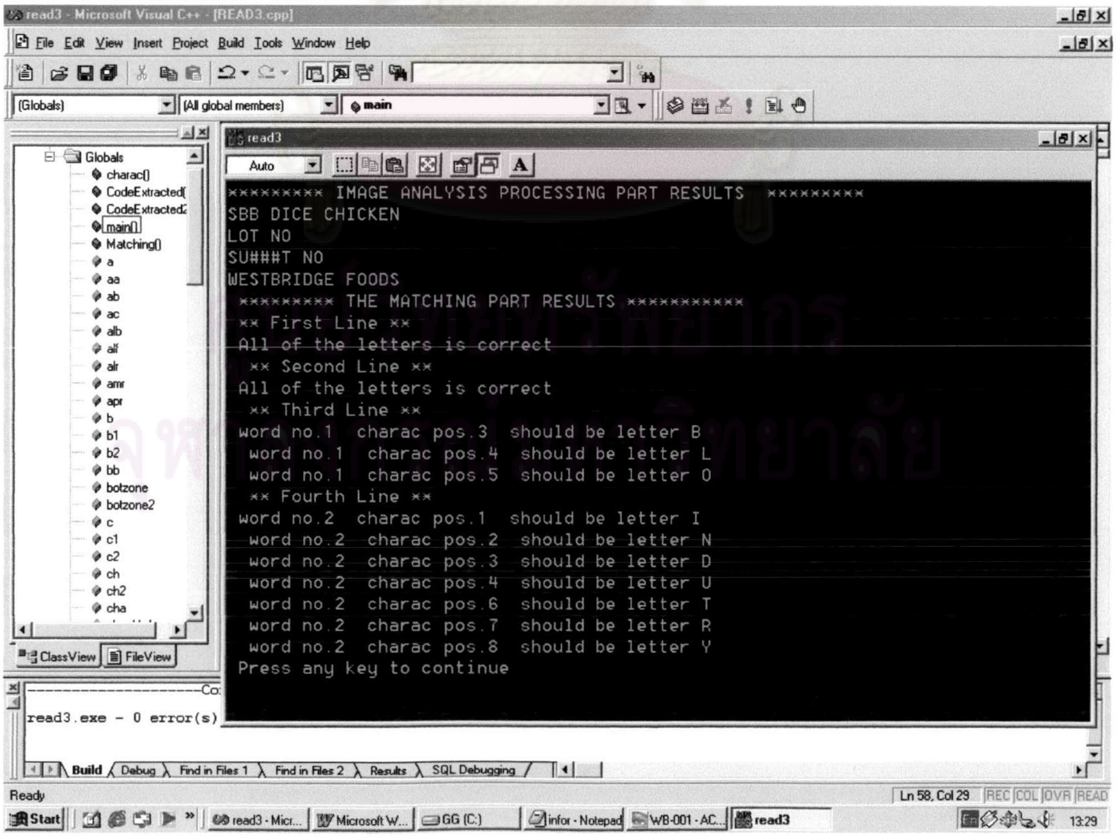
SBB DICE CHICKEN

LOT NO

SUBLOT NO

WESTBRIDGE FOODS INDUSTRY

Image Analysis Processing And Matching Algorithm



```
********* IMAGE ANALYSIS PROCESSING PART RESULTS  *********
SBB DICE CHICKEN
LOT NO
SU###T NO
WESTBRIDGE FOODS
 ********* THE MATCHING PART RESULTS ***********
 ** First Line **
 All of the letters is correct
  ** Second Line **
 All of the letters is correct
  ** Third Line **
 word no.1  charac pos.3  should be letter B
  word no.1  charac pos.4  should be letter L
  word no.1  charac pos.5  should be letter O
  ** Fourth Line **
 word no.2  charac pos.1  should be letter I
  word no.2  charac pos.2  should be letter N
  word no.2  charac pos.3  should be letter D
  word no.2  charac pos.4  should be letter U
  word no.2  charac pos.6  should be letter T
  word no.2  charac pos.7  should be letter R
  word no.2  charac pos.8  should be letter Y
 Press any key to continue
```

```
--------------------Co
read3.exe - 0 error(s)
```

Figure 4-11 An Example Of Algorithm Test Results (continue)

Image Test File Information (Test sheet No.5)

Image File Name : WB-003 Image Dimension : (Widths x Lengths) 386x229 Pixel

Image Size : 259.5 KB No Of Characters Lines: 5 Total Of Characters : 50

Characters Image Files (WB-003.bmp)	Original Text Files (WB-003.txt)
FRIED DRUMTYPE	FRIED DRUMTYPE
PRODUCTION DATE	PRODUCTION DATE
EXPIRY DATE	EXPIRY DATE
LOT NO	LOT NO
SUBLOT NO	SUBLOT NO

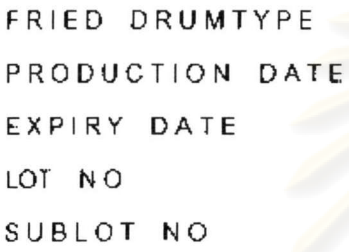
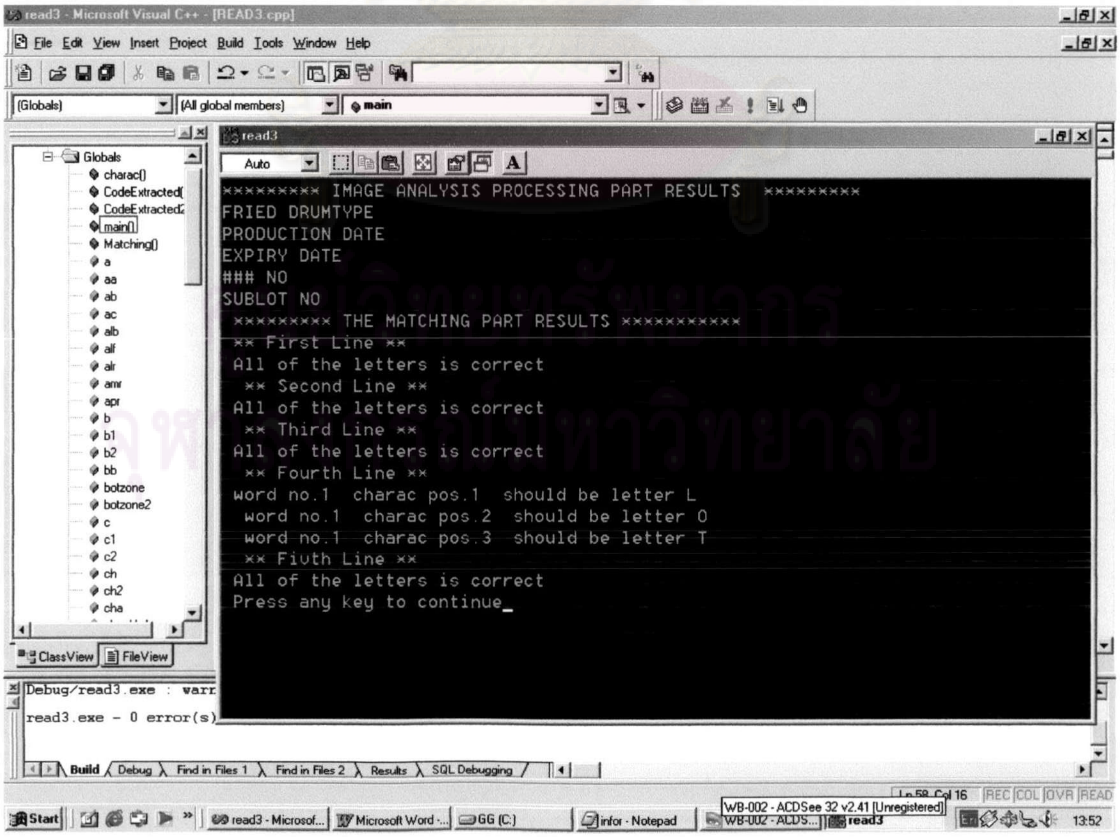


Image Analysis Processing And Matching Algorithm

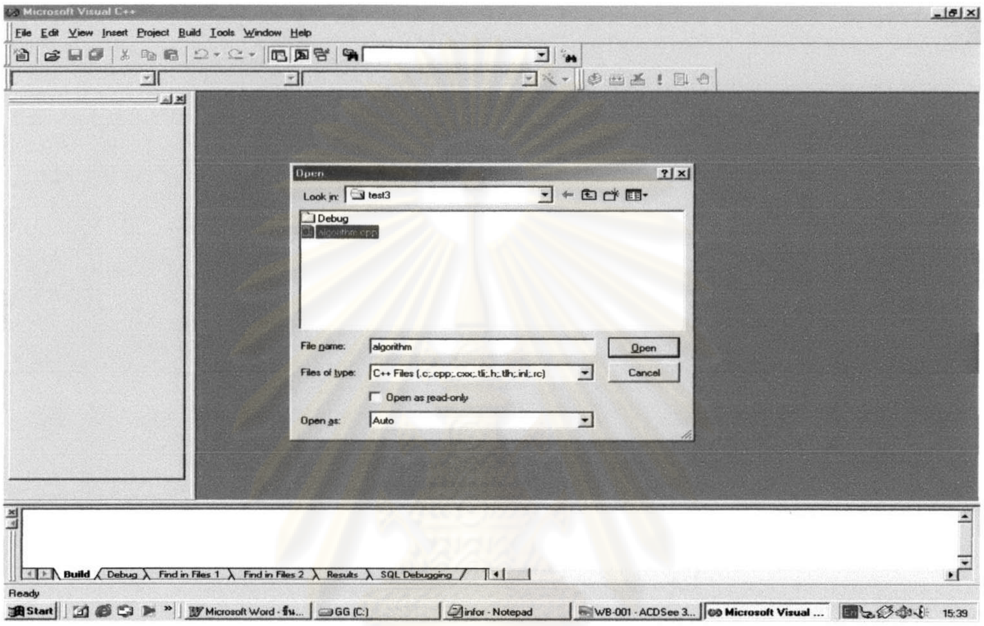


```
********* IMAGE ANALYSIS PROCESSING PART RESULTS *********
FRIED DRUMTYPE
PRODUCTION DATE
EXPIRY DATE
### NO
SUBLOT NO
 ********* THE MATCHING PART RESULTS ***********
 ** First Line **
 All of the letters is correct
  ** Second Line **
 All of the letters is correct
  ** Third Line **
 All of the letters is correct
  ** Fourth Line **
 word no.1  charac pos.1  should be letter L
  word no.1  charac pos.2  should be letter O
  word no.1  charac pos.3  should be letter T
  ** Fivth Line **
 All of the letters is correct
 Press any key to continue_
```

```
Debug/read3.exe : warr
read3.exe - 0 error(s)
```

4.3 User Manual Instruction

1. Open Microsoft Visual C++



Look in: test3

Debug
algorithm.cpp

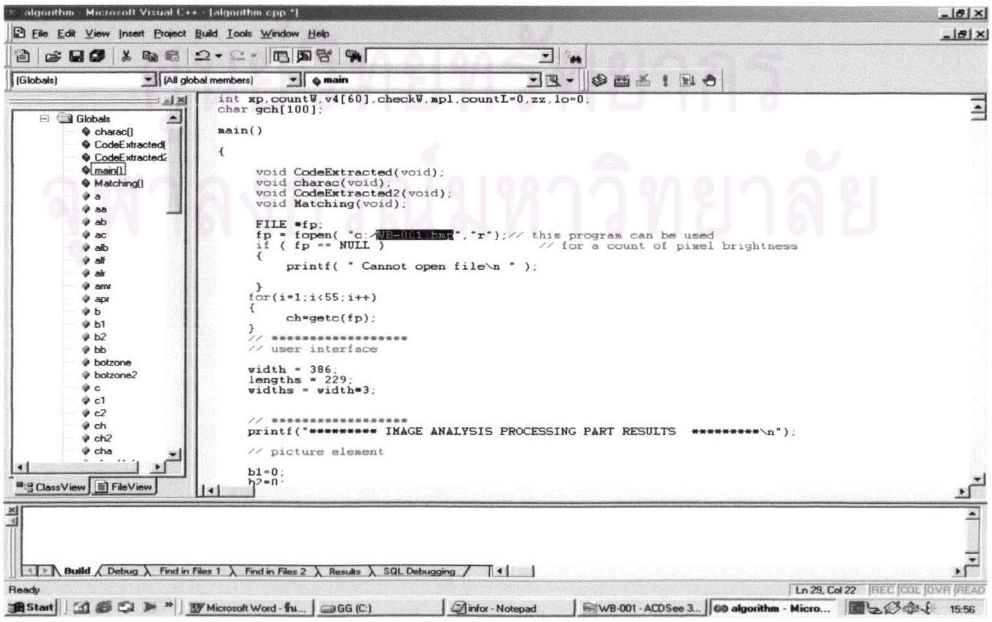
File name: algorithm

Files of type: C++ Files (.c;.cpp;.cxx;.tli;.h;.tlh;.inl;.rc)

□ Open as read-only

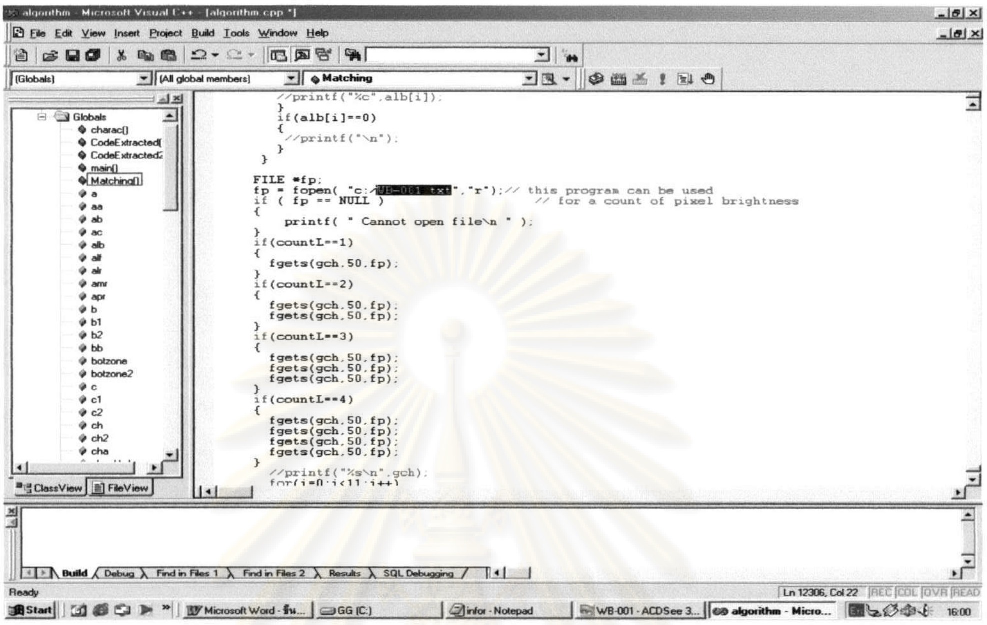
Open as: Auto

3. Click >Input file name (.bmp) > Input Image size



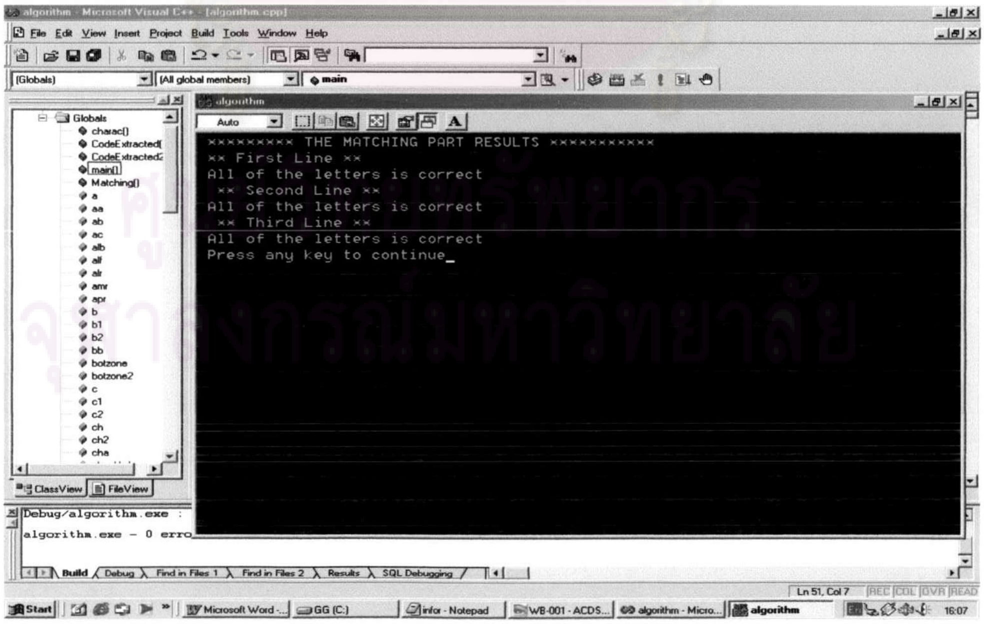
```
int xp,countW,v4[60],checkW,mpl,countL=0,zz,lo=0;
char gch[100];

main()

{

     void CodeExtracted(void);
     void charac(void);
     void CodeExtracted2(void);
     void Matching(void);

     FILE *fp;
     fp = fopen( "c:/WB-001.bmp","r");// this program can be used
     if ( fp == NULL )                         // for a count of pixel brightness
     {
         printf( " Cannot open file\n " );

     }
    for(i=1;i<55;i++)
    {
         ch=getc(fp);
    }
    // ******************
    // user interface

    width = 386;
    lengths = 229;
    widths = width*3;


    // ******************
    printf("********* IMAGE ANALYSIS PROCESSING PART RESULTS  *********\n");

    // picture element

    b1=0;
    b2=0;
```

4. Input file name (.txt) for matching



```
     //printf("%c",alb[i]);
    }
    if(alb[i]==0)
    {
     //printf("\n");
    }
  }

FILE *fp;
fp = fopen( "c:/WB-001.txt","r");// this program can be used
if ( fp == NULL )                    // for a count of pixel brightness
{
    printf( " Cannot open file\n " );
}
if(countL==1)
{
  fgets(gch,50,fp);
}
if(countL==2)
{
  fgets(gch,50,fp);
  fgets(gch,50,fp);
}
if(countL==3)
{
  fgets(gch,50,fp);
  fgets(gch,50,fp);
  fgets(gch,50,fp);
}
if(countL==4)
{
  fgets(gch,50,fp);
  fgets(gch,50,fp);
  fgets(gch,50,fp);
  fgets(gch,50,fp);
}
  //printf("%s\n",gch);
  for(i=0;i<11;i++)
```

5. Click > Build > Execute > Display



```
********* THE MATCHING PART RESULTS ***********
** First Line **
All of the letters is correct
 ** Second Line **
All of the letters is correct
 ** Third Line **
All of the letters is correct
Press any key to continue_
```

```
Debug/algorithm.exe :
algorithm.exe - 0 erro
```