C to C# converter

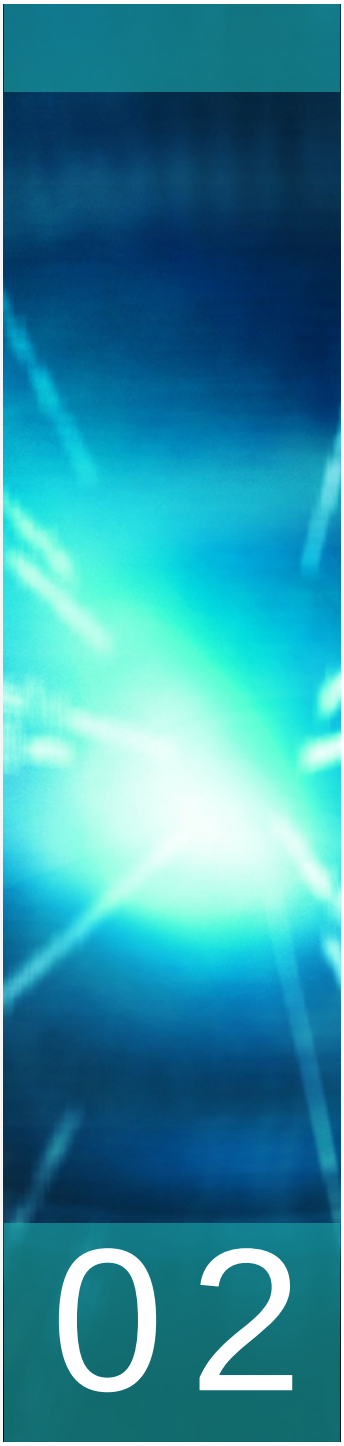
TECHNICAL AND USER MANUAL

Created By:
Raye

Technical Manual

Project Description

The following program is meant to convert a program written C code and then convert it to its C# code equivalent. The must enter the code in the variable label C_code. From the variable the program will run through a sequence that will check if the imputed code contains a while loop, printf, and scanf. The code must contain at least on of each of these parameters. If not include in this code the program will not run. The number of printf and scanf function can be any number. The user needs to input what code they want converted into the variable C_code.



Techincal Aspect

The program code contains cetain aspects that are not commonly used. Such functions are memcpy,memmove,strlen.

- Memcpy is a function in C that copies a block of memory from one location to another. It can be used to copy any type of data, including arrays, structures, and objects.
- Memmove function in C designed to handle overlapping memory blocks. It ensures that the data is copied correctly even if the source and destination memory blocks overlap.
- Strlen is a function in C calculates the length of a null-terminated string. It takes a pointer to the string as its argument and returns the number of characters in the string, excluding the null terminator.

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
int main() {//just fix the code that will be converted

    char c_code[100000] = "char input[100];\n printf(\"Enter a string : \");\nscanf(\"%s\", input);\nwhile (1) {\n print

        char* ptwhile = strstr(c_code, "while");
        if (!ptwhile) {
            printf("The code does not contain a while loop.\n");
        }
        char* ptprintf = strstr(c_code, "printf");
        if (!ptprintf) {
            printf("The code does not contain a printf.\n");
        }
        else {
            char* printf_convert = "Console.Write";
            int printf_len = strlen("printf");
            int convert_len = strlen(printf_convert);
            char* ptr = c_code;
            while ((ptr = strstr(ptr, "printf")) != NULL) {
                int offset = ptr - c_code;
                memmove(&c_code[offset + convert_len], &c_code[offset + printf_len], strlen(c_code) - offset - printf_len + 1);
                memcpy(&c_code[offset], printf_convert, convert_len);
```

Code breakedown

Lines 1 through 4 are the main setup of the code. Line 6 contains the main variable of the code. The code char c_code[10000] is where the user will input the code. Lines 8 through 14 check to make sure the string within c_code contains a "printf" and a "while"loop. Lines 16 through 24 are replacing all occurrences of the string "printf" in the given c code and replacing it with "console.write" the equivalent in C#. This is achieve by using strstr function to find the next occurrence of "printf". Memmove shifts the characters that come after the "printf" to the right between the length difference with "console.write.Then memcpy copies "console.write" string into the position of where "printf" was. Finally the pointer is update to look for the next occurrence of "printf" and the loop continues until the last "printf" is changed.

```
28      char* ptscanf = strstr(c_code, "scanf");
29      if (!ptscanf) {
30          printf("The code does not contain a scanf.\n");
31      }
32      else {
33
34          char* scanf_convert = "Console.Read";
35          int printf_len = strlen("scanf");
36          int convert_len = strlen(scanf_convert);
37          char* ptr = c_code;
38          while ((ptr = strstr(ptr, "scanf")) != NULL) {
39              int offset = ptr - c_code;
40              memmove(&c_code[offset + convert_len], &c_code[offset + printf_len], strlen(c_code) - offset - printf_len + 1);
41              memcpy(&c_code[offset], scanf_convert, convert_len);
42              ptr += convert_len;
43          }
44              //
45          char* s_remove = "{0}";
46          const int printf_len_const = strlen("\"%s\"");
47          const int convert_len_const = strlen(s_remove);
48          char* ptr_replace = c_code;
49          while ((ptr_replace = strstr(ptr_replace, "\"%s\"")) != NULL) {
50              int offset = ptr_replace - c_code;
51              memmove(&c_code[offset + convert_len_const], &c_code[offset + printf_len_const], strlen(&c_code[offset + printf_len_const]) + 1);
52              memcpy(&c_code[offset], s_remove, convert_len_const);
53              ptr_replace += convert_len_const;
54
```

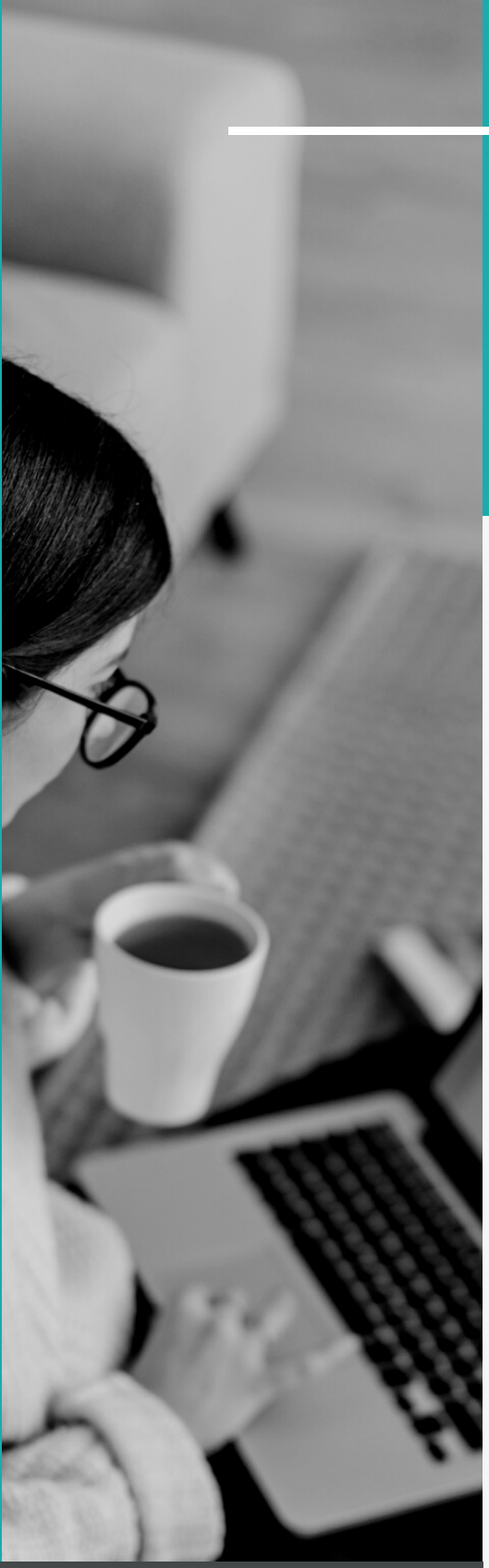
Code breakedown

Lines 28 through 31 check to make sure the string within c_code contains a "scanf" Lines 34 through 42 are replacing all occurrences of the string "scanf" in the given c code and replacing it with "console.read" the equivalent in C#. This is achieve by using the same process that was used to replace "printf". Lines 45 through 53 are replacing the "%s" that is used with the "scanf" to {0} .This is also done similarly to how replacing "printf" was done.

```
58      char orginalc_code[10000] = "char input[100];\n printf(\"Enter a string : \");\nscanf(\"%s\", input);\nwhile (1)
59      printf("%s\n", orginalc_code);
60      printf("////////////////////////////////////////////////////////////////////////////////////////////////////////////
61      printf("C code to C# code coversion  \n");
62      printf("using System;\nclass Program {\nstatic void Main(string[] args) {%s\n", c_code);
63      return 0;
64  }
```

Code breakedown for final part

Lines 58 shows the orginal code that was inputed. Lines 59 throught 62 print out the orginal code and the new converted code.



Grammar

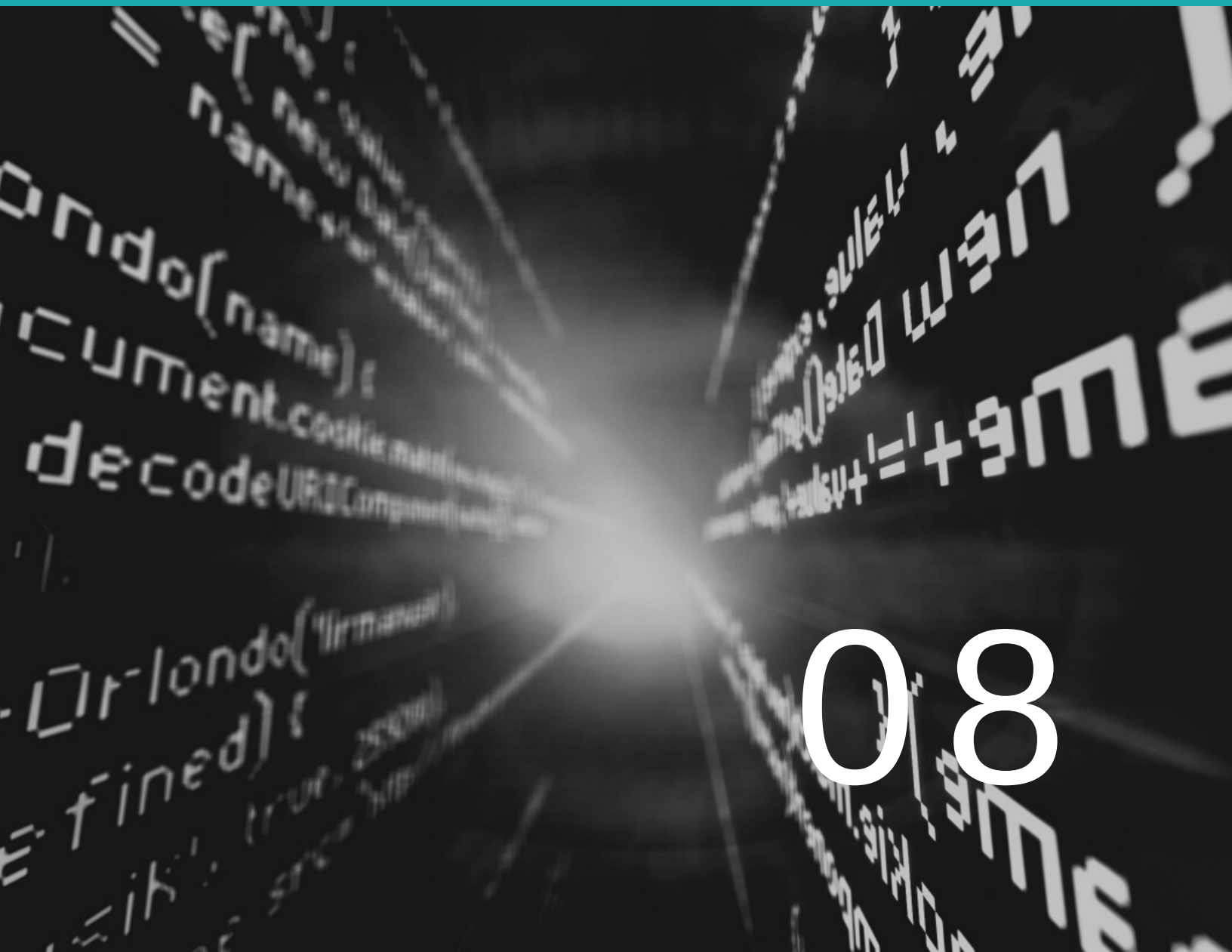
The main program its self doesn't use grammar. where the grammar comes in is with the code the user which to convert. But this makes the program more versatile so its not constrained to a certain grammar structure. Example of such grammar that can be used are:

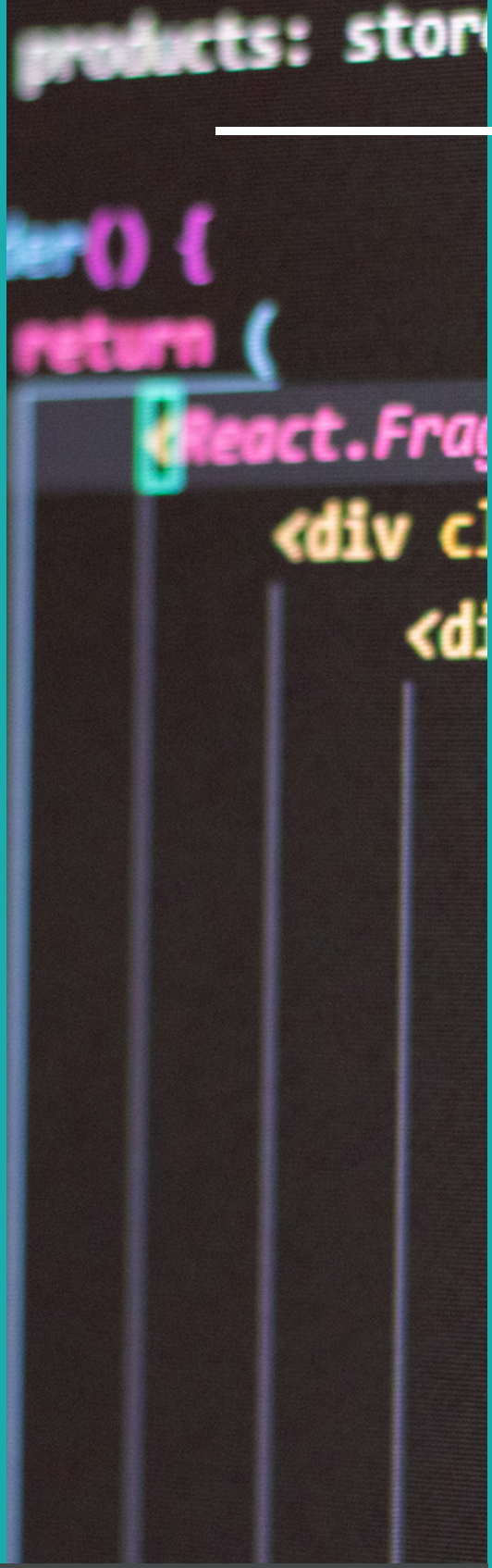
ENBF Rules for program input
program> ::= <statement>+
<statement> ::= <while-loop> | <other-statement>
<while-loop> ::= "while" "(" <condition> ")" "{" <statement>+ "}"
<condition> ::= <expression> <comparison-operator> <expression>
<expression> ::= <variable> | <value> | <printf> <operator> <expression>
<comparison-operator> ::= "==" | "!=" | "<" | ">" | "<=" | ">="
<operator> ::= "+" | "-" | "*" | "/"
<variable> ::= <letter>+
<value> ::= <digit>+
<other-statement> ::= ...
<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
<digit> ::= "0" | "1" | ... | "9"

ENBF Rules for Output
program> ::= <statement>+
<statement> ::= <while-loop> | <other-statement>
<while-loop> ::= "while" "(" <condition> ")" "{" <statement>+ "}"
<condition> ::= <expression> <comparison-operator> <expression>
<expression> ::= <variable> | <value> | <console.write> <operator> <expression>
<comparison-operator> ::= "==" | "!=" | "<" | ">" | "<=" | ">="
<operator> ::= "+" | "-" | "*" | "/"
<variable> ::= <letter>+
<value> ::= <digit>+
<other-statement> ::= ...
<letter> ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
<digit> ::= "0" | "1" | ... | "9"

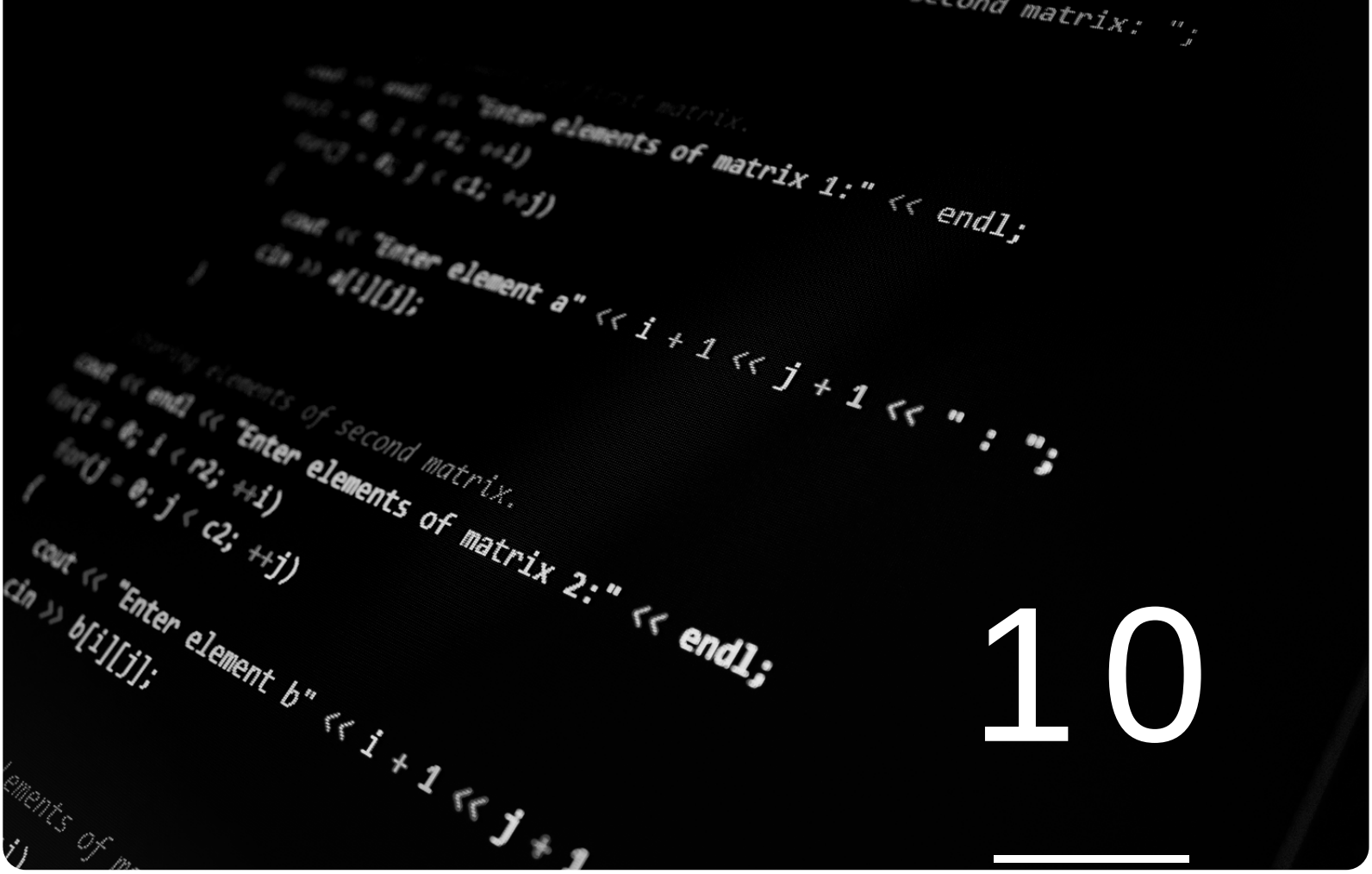
Grammar for code example input
<program> ::= <function> <main>
<function> ::= "void" "grammar" "(" <parameter> ")" "{" <statement> "}"
<parameter> ::= "int" <identifier>
<main> ::= "int" "main" "(" ")" "{" <statement> "return" <expression> ";}"
<statement> ::= <if-statement> | <while-statement> | <expression> ";"
<if-statement> ::= "if" "(" <expression> ")" "{" <statement> "}"
<while-statement> ::= "while" "(" <expression> ")" "{" <statement> "}"
<expression> ::= <identifier> <assignment-operator> <value> | <function-call>
<assignment-operator> ::= "="
<value> ::= <integer>
<function-call> ::= <identifier> "(" <argument-list> ")"
<argument-list> ::= <expression> | <expression> "," <argument-list>
<identifier> ::= <letter> | <identifier> <letter> | <identifier> <digit>
<integer> ::= <digit> | <integer> <digit>
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<letter> ::= "a" | "b" | "c" | ... | "x" | "y" | "z" | "A" | "B" | "C" | ... | "X" | "Y" | "Z"

User Manual





USER INSTRUCTIONS.....



Program Code limitations

The converter is limited to only the programing languages of C and C#.
The code is limited to only replacing the "printf" "scanf", and "%s" . The program will not replace "%s" if there it looks similar to "% s" or " %s". The convert is not capable of moving or rearranging the code. The program is only able to convert code that has been rearrange into a string.

Code example that uses grammar(file cto Cscharpconverter)

The code within picture lowest on the left hand side shows the correct syntax of what the C# code look like. While the first picture shows the code that was enter then after the conversion. I show the output below it. The only thing the user would need to do now is change the char to string. An move the "input " to in front of console.Read. Thus having proper syntax.

```
void grammar(int n) {
if (n > 0) {
 printf("%d bottles of beer on the wall, %d bottles of beer., n, n");
 printf("Take one down, pass it around, ");
grammar(n - 1); }
else {
printf(" no more bottles of beer on the wall.");
}
}
 int main() {
 int num;
printf("Enter a number : ");
scanf(" %d", &num);
while (num > 0) {
 grammar(num);
 printf("
")
printf("Enter another number : ");
scanf(" %d", &num);
}
printf("Goodbye!");}
return 0}
```

```
using System;
class Program {
static void Main(string[] args) {void grammar(int n) {
if (n > 0) {
 Console.Write("%d bottles of beer on the wall, %d bottles of beer., n, n");
 Console.Write("Take one down, pass it around, ");
grammar(n - 1); }
else {
Console.Write(" no more bottles of beer on the wall.");
}
}
 int main() {
 int num;
Console.Write("Enter a number : ");
Console.Read(" %d", &num);
while (num > 0) {
 grammar(num);
 Console.Write("
")
Console.Write("Enter another number : ");
Console.Read(" %d", &num);
}
Console.Write("Goodbye!");}
return 0}
```

```
using System;

class Program {
    static void Grammar(int n) {
        if(n > 0) {
            Console.WriteLine($"{n} bottles of beer on the wall, {n} bottles of beer.");
            Console.Write("Take one down, pass it around, ");
            Grammar(n-1);
        } else {
            Console.WriteLine("no more bottles of beer on the wall.");
        }
    }

    static void Main() {
        int num;
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        while(num > 0) {
            Grammar(num);
            Console.WriteLine();
            Console.Write("Enter another number: ");
            num = int.Parse(Console.ReadLine());
        }
        Console.WriteLine("Goodbye!");
    }
}
```

Grammar for code example input

Grammar for code example input

<program> ::= <function> <main>

<function> ::= "void" "grammar" "(" <parameter> ")" "{" <statement> "}"

<parameter> ::= "int" <identifier>

<main> ::= "int" "main" "(" ")" "{" <statement> "return" <expression> ";}"

<statement> ::= <if-statement> | <while-statement> | <expression> ";"

<if-statement> ::= "if" "(" <expression> ")" "{" <statement> "}"

<while-statement> ::= "while" "(" <expression> ")" "{" <statement> "}"

<expression> ::= <identifier> <assignment-operator> <value> | <function-call>

<assignment-operator> ::= "="

<value> ::= <integer>

<function-call> ::= <identifier> "(" <argument-list> ")"

<argument-list> ::= <expression> | <expression> "," <argument-list>

<identifier> ::= <letter> | <identifier> <letter> | <identifier> <digit>

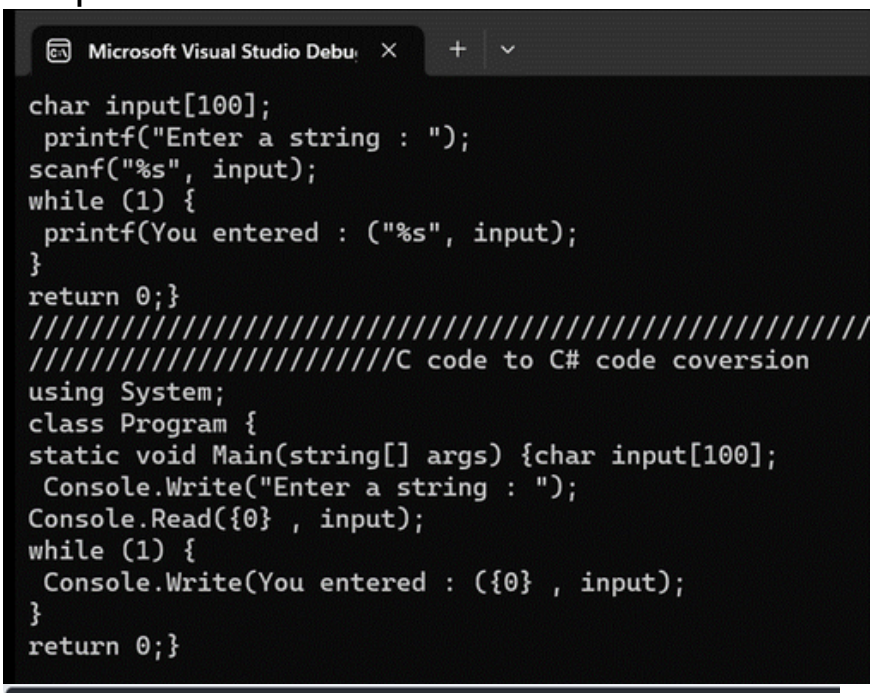
<integer> ::= <digit> | <integer> <digit>

<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<letter> ::= "a" | "b" | "c" | ... | "x" | "y" | "z" | "A" | "B" | "C" | ... | "X" | "Y" | "Z"

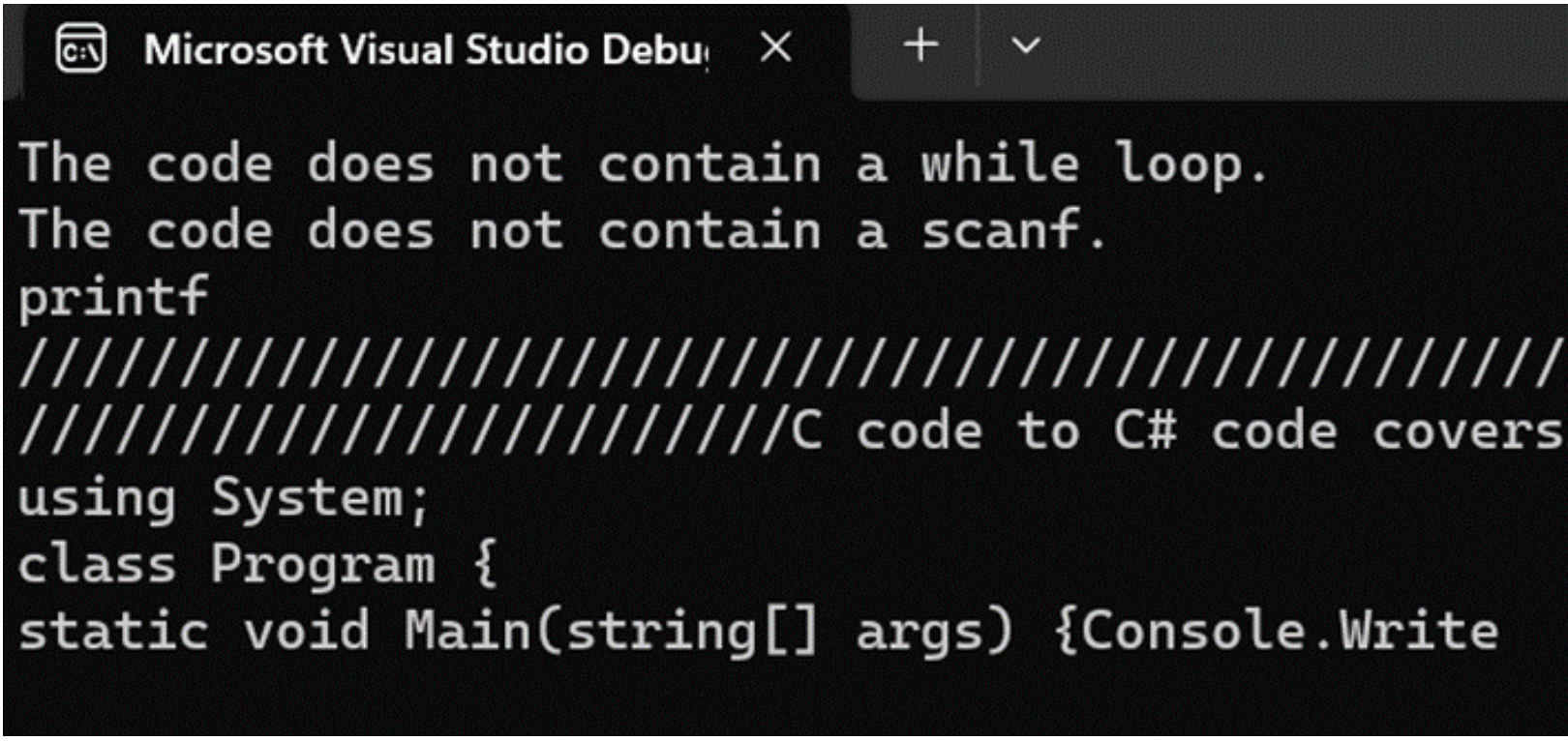
Code example without grammar(file tester)

The code within the second picture shows the correct syntax of what the C# code look like. While the first picture shows the code that was enter then after the conversion. I show the output below it. The only thing the user would need to do now is change the char to string. An move the "input " to in front of console.Read. Thus having proper syntax.



The following Picture shows code that was inputed and converted and below it shows the proper syntax for C# code

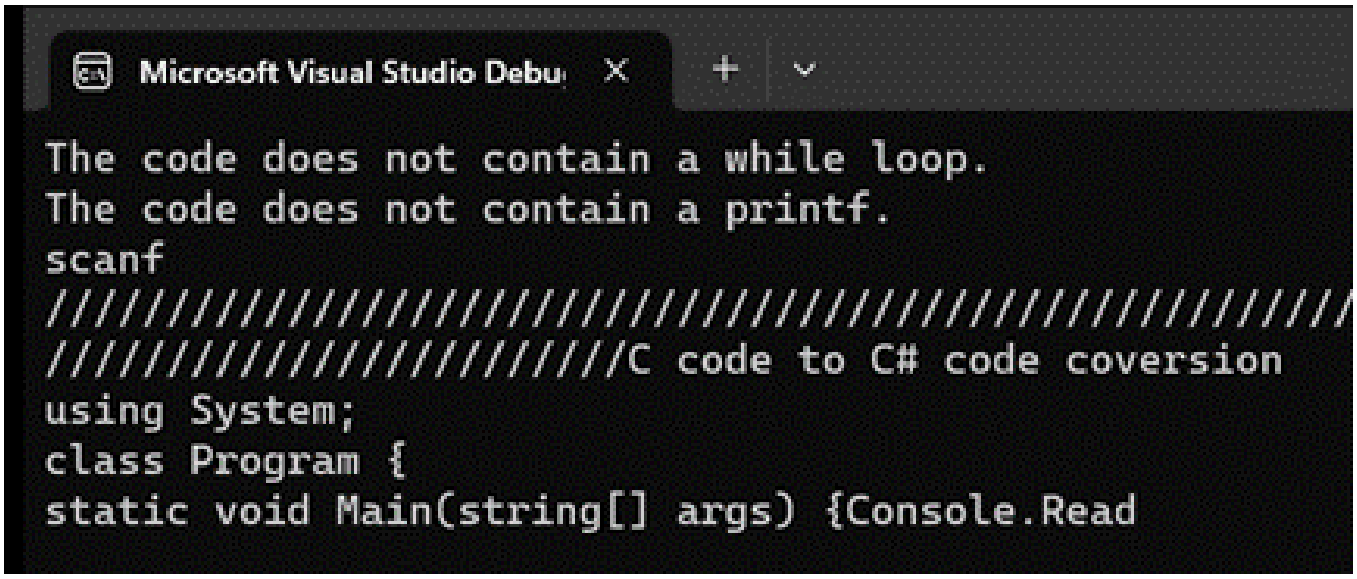
```
1  using System;
2
3  class Program {
4      static void Main(string[] args) {
5          string input;
6          Console.Write("Enter a string: ");
7          input = Console.ReadLine();
8          while (true) {
9              Console.WriteLine("You entered: {0}", input);
10         }
11     }
12 }
```



incorrect code input(file ex1.c)

The code is an example of what would happen if only "printf" was present.

The following Picture shows code that was imputed and converted.



incorrect code input

The code is an example of what would happen if only "scanf" was present.

The following Picture shows code that was imputed and converted