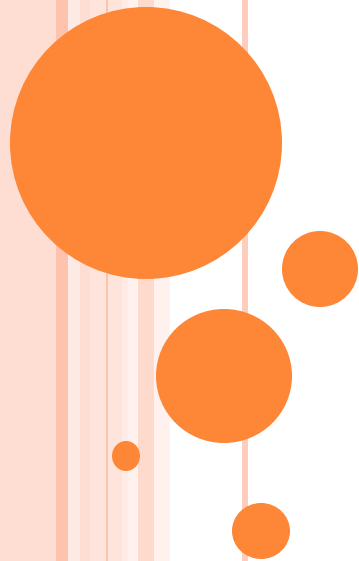


F. Y. BCA

Web Development using PHP

Chapter 3 – Functions and Strings



PRACTICAL EXERCISES

- Write PHP script to find whether the given year is leap year or not
- Write PHP Script to find addition of given n numbers.
- Write PHP script find total even and odd numbers from n numbers.
- Write PHP script to find whether input number is prime number or not.
- Write PHP script to find whether input number is Armstrong number or not.



FUCNTION – CALL BY VALUE

- The parameters passed to function are called *actual parameters* whereas the parameters received by function are called *formal parameters*.
- Call by value method copies the value of an argument into the formal parameter of that function.
- Changes made to the parameter of the main function do not affect the argument.
- Values of actual parameters are copied to function's formal parameters, and the parameters are stored in different memory locations.
- So any changes made inside functions are not reflected in actual parameters of the caller.



ACTUAL PARAMETERS & FORMAL PARAMETERS

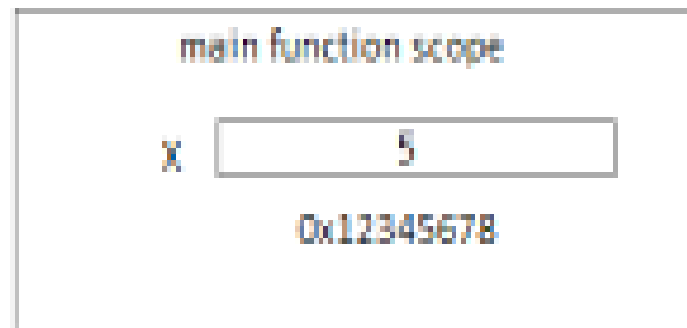
- Actual parameters are those parameters that are specified in the calling function. While on the other hand, formal parameters are those parameters that are declared in the called function.
- It is the actual value that is assigned to the process by a caller. Or in other words, we can say that it is the parameters that you determine when you invite the subroutine such as, Functions.
- A formal parameter is a variable that you specify when you determine the subroutine or function. These parameters define the list of possible variables, their positions, and their data types.



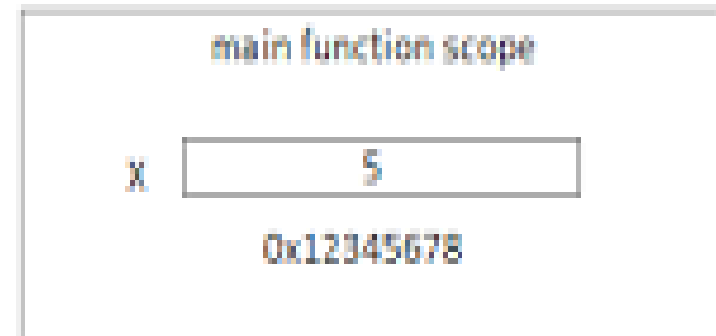
Actual Parameters	Formal Parameters
<p>The Actual parameters are the variables that are transferred to the function when it is requested.</p>	<p>The Formal Parameters are the values determined by the function that accepts values when the function is declared.</p>
<p>In actual parameters, only the variable is mentioned, not the data types.</p>	<p>In formal parameters, the data type is required.</p>
<p>Actual parameters are the values referenced in the parameter index of a subprogram call.</p>	<p>Formal parameters are the values referenced in the parameter index of a subprogram.</p>
<p>In actual parameters, there is no requirement to define datatype.</p>	<p>In formal parameters, it is mandatory to define the datatype of the receiving value.</p>
<p>These can be variables, constants, and expressions, without data types.</p>	<p>Formal parameters are variables with the data type.</p>
<p>Those parameters which are addressed in a function call are called actual parameters.</p>	<p>Parameters addressed in the function description are called formal parameters.</p>

Call by Value	Call by Reference
Copying variables pass values.	Values are passed by copying the address of the variables.
Copies are passed.	Variable is passed.
Changes do not reflect the original function.	Changes affect the variable of the function.
The actual variable can not be changed	The actual variable can be modified.
The argument is also safe from the changes.	The argument also changes with a change in the called function.
The different memory location is used to create the actual and formal arguments.	The same memory location will be used.
Used in C++, PHP, C#, etc.	Used in Java, C++, etc.

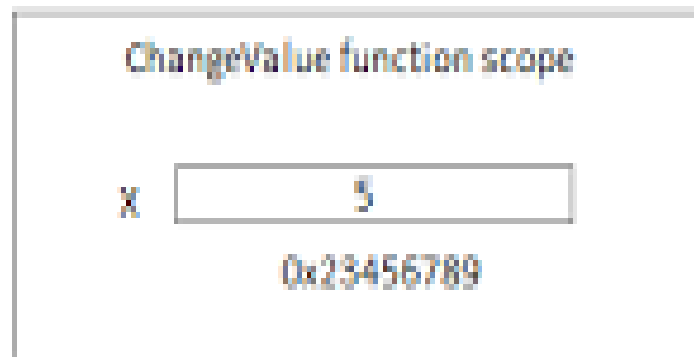
Before execution of $X=X+5$



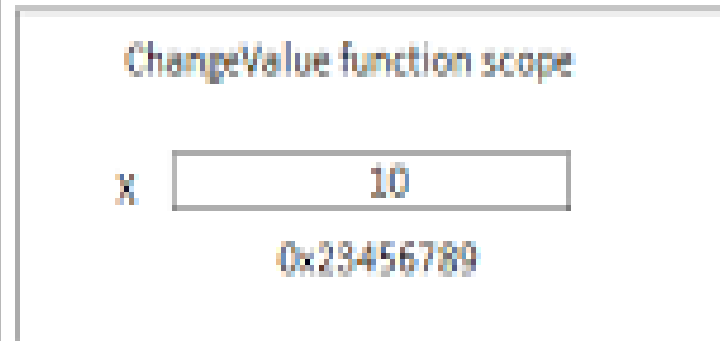
After execution of $X=X+5$



ChangeValue function scope



ChangeValue function scope



CALL BY VALUE

```
<?php
function increment($var){
    $var++;
    return $var;
}
$a = 5;
$b = increment($a);
echo $a; //Output: 5
echo $b; //Output: 6
?>
```

In this method, only values of actual parameters are passing to the function. So there are two addresses stored in memory. Making changes in the passing parameter does not affect the actual parameter.

CALL BY REFERENCE

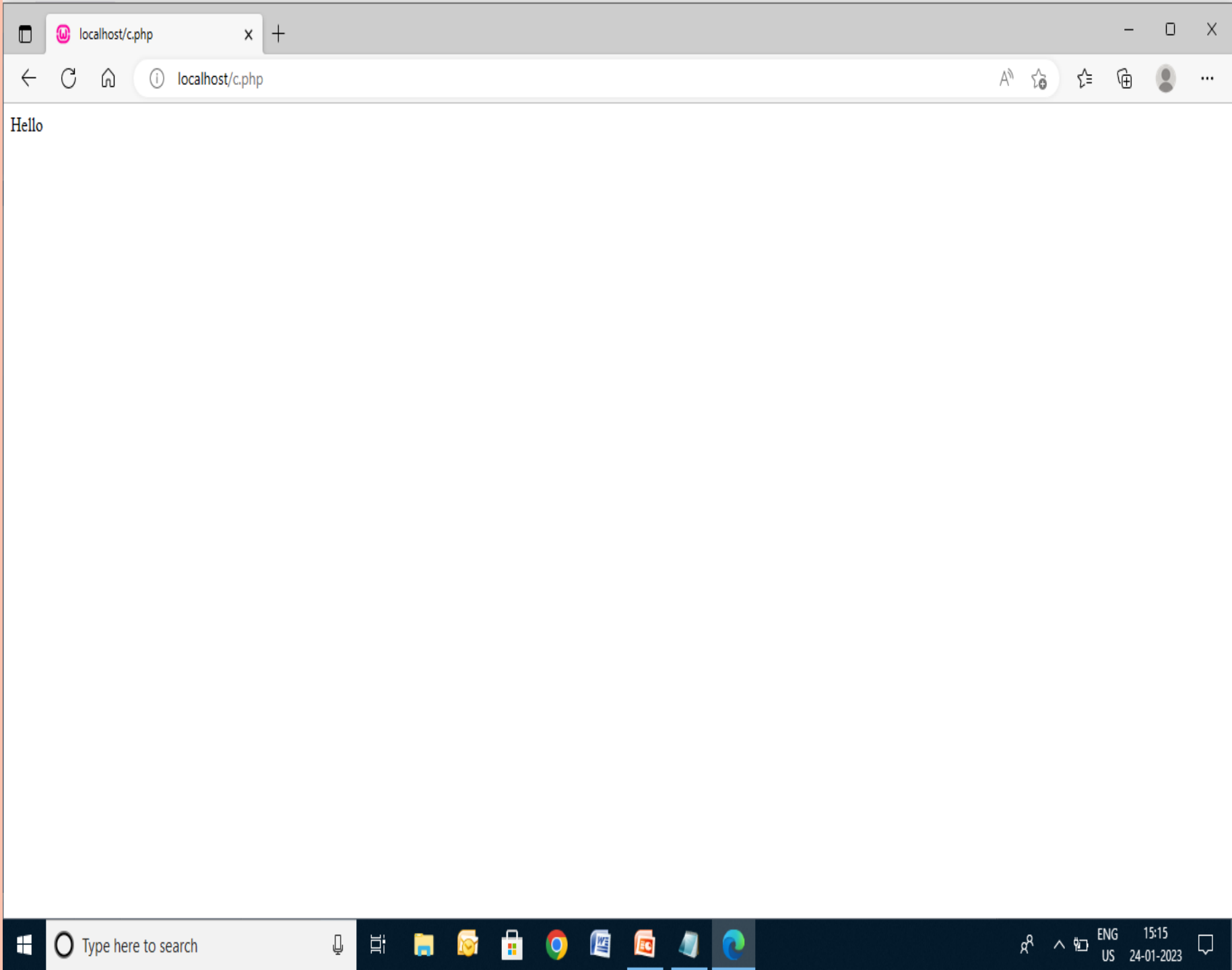
```
<?php
function increment(&$var){
    $var++;
    return $var;
}
$a = 5;
$b = increment($a);
echo $a; //Output: 6
echo $b; //Output: 6
?>
```

In this method, the address of actual parameters is passing to the function. So any change made by function affects actual parameter value.

EXAMPLE 1

```
<?php
function adder($str2)
{
    $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```





localhost/c.php



localhost/c.php



Hello



Type here to search



ENG
US

15:15
24-01-2023



EXAMPLE 2

```
<?php
function adder($str2)
{
    $str2 .= 'Call By Value';
}
$str = 'Hello ';
adder($str);
echo $str, $str2;
?>
```



Call Stack

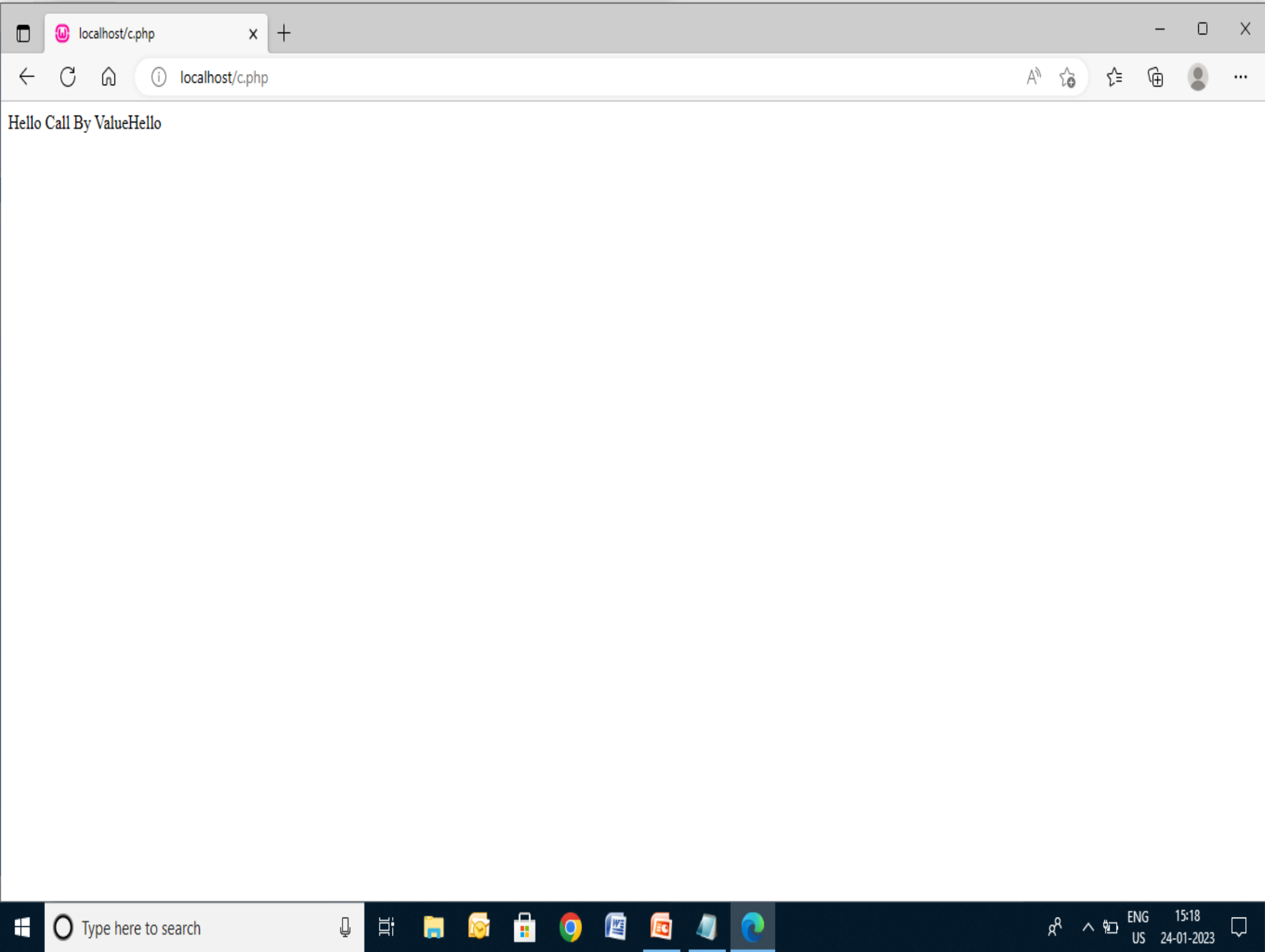
#	Time	Memory	Function	Location
1	0.0076	361768	{main}()	...\\c.php:0

EXAMPLE 3

```
<?php
function adder($str2)
{
    $str2 .= 'Call By Value';
    echo $str2;

}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

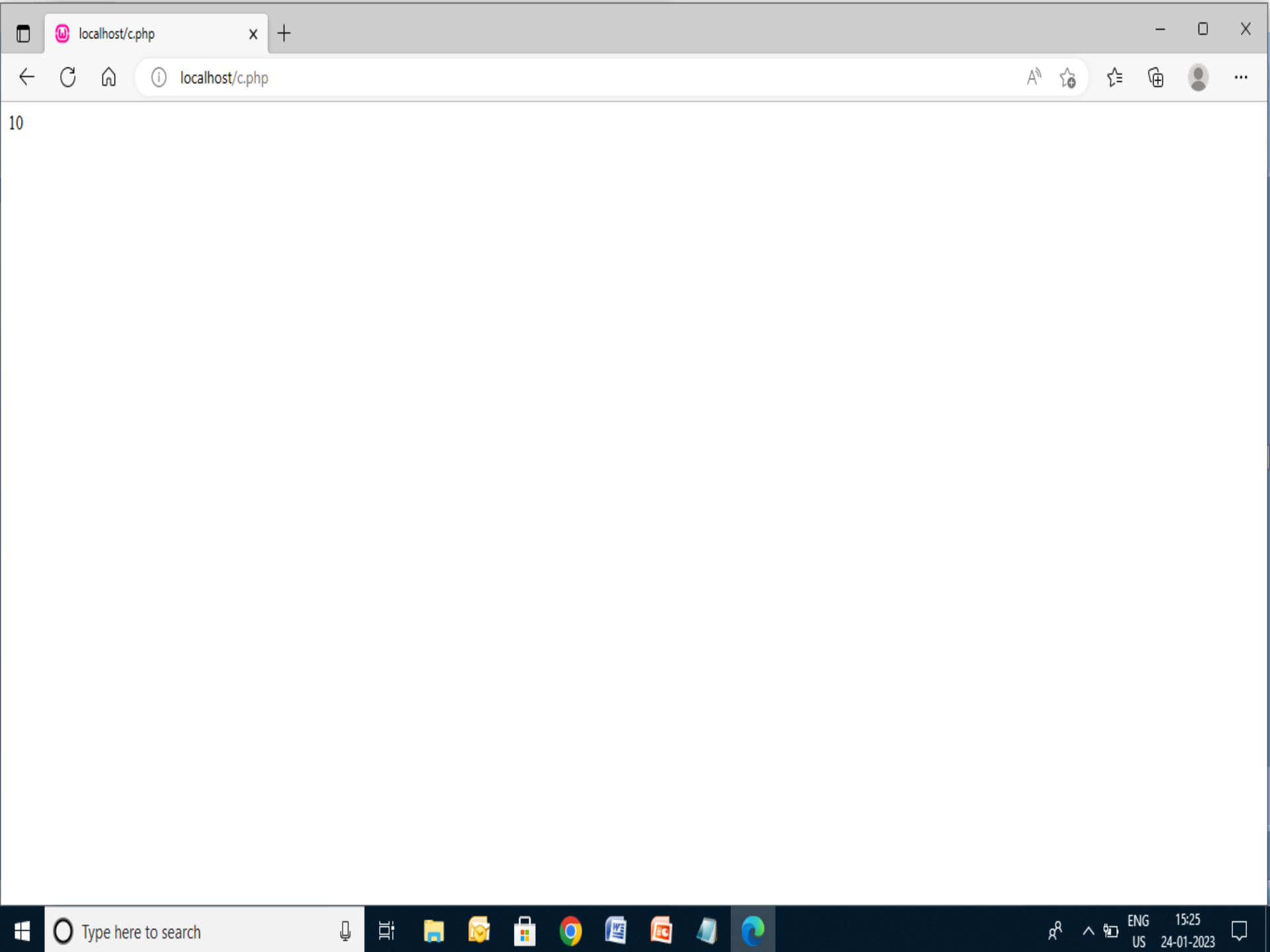




EXAMPLE 4

```
<?php
function increment($i)
{
    $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```



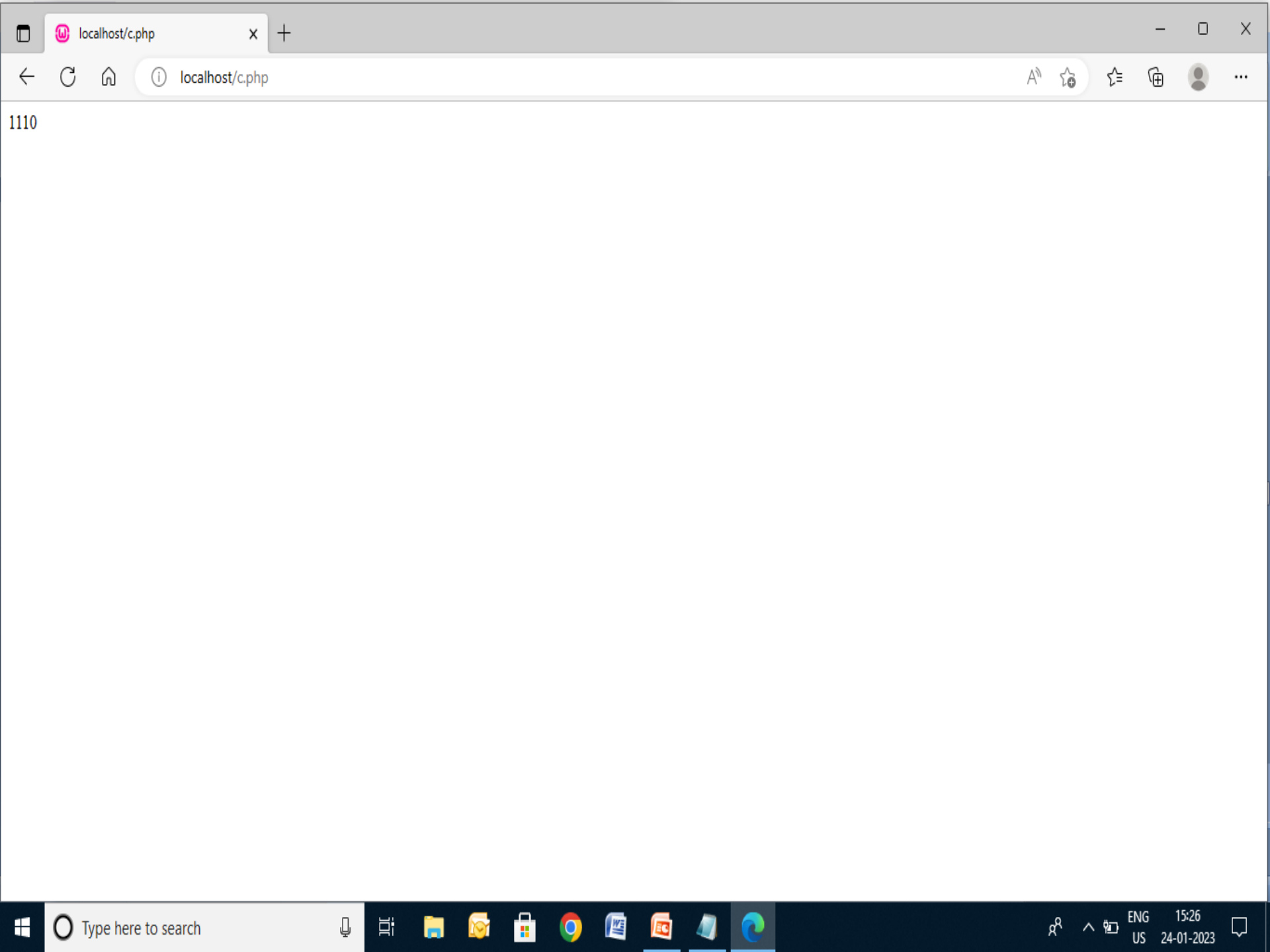


10

EXAMPLE 5

```
<?php
function increment($i)
{
    $i++;
    echo $i;
}
$i = 10;
increment($i);
echo $i;
?>
```





1110

FUNCTION – CALL BY REFERENCE

- In case of PHP call by reference, actual value is modified if it is modified inside the function.
- In such case, you need to use & (ampersand) symbol with formal arguments.
- The & represents reference of the variable.




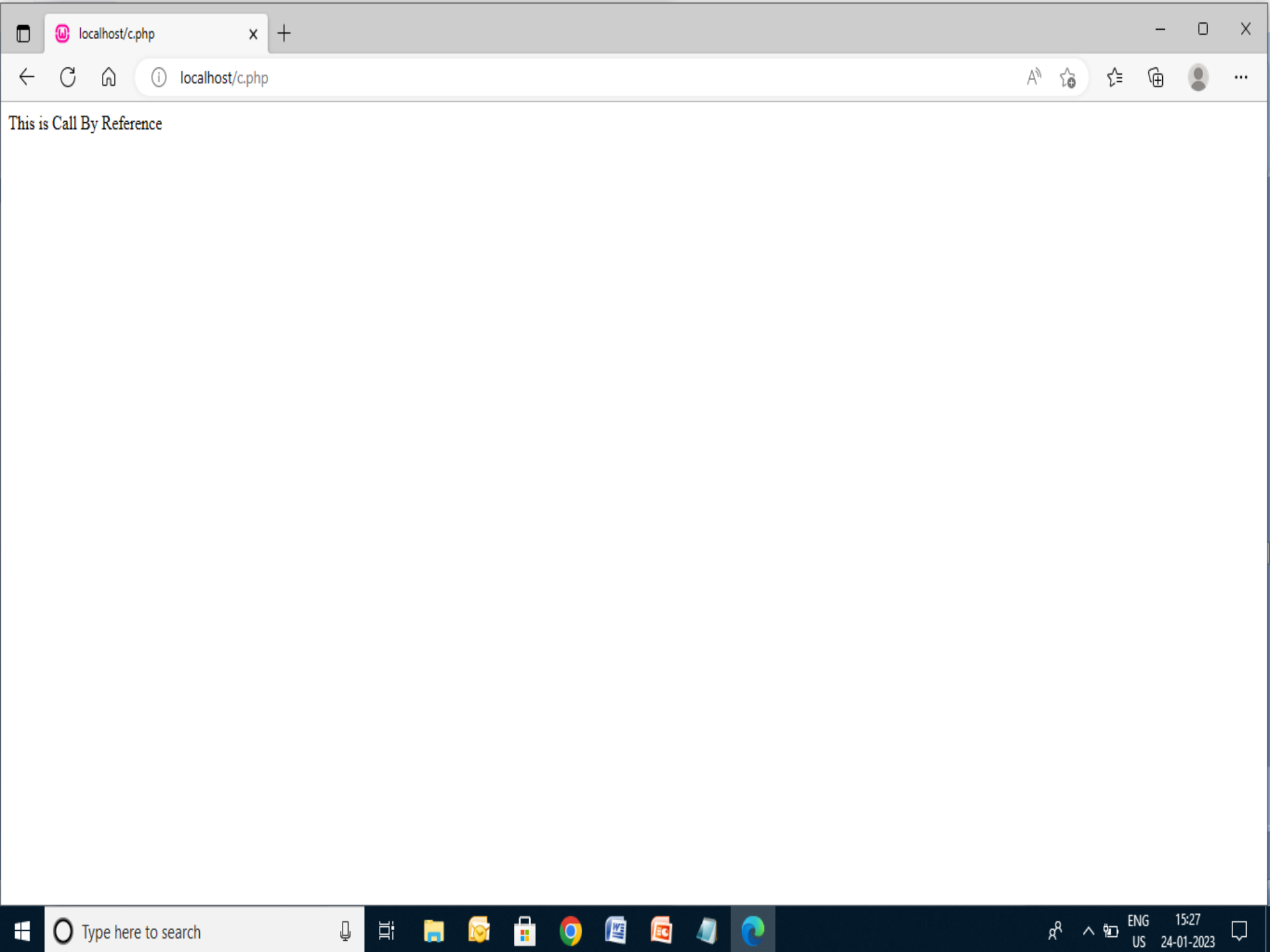
FUNCTION – CALL BY REFERENCE

EXAMPLE 6

```
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

Variable \$str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing \$str variable results 'This is Call By Reference'. It is because changes are done in the actual variable \$str.



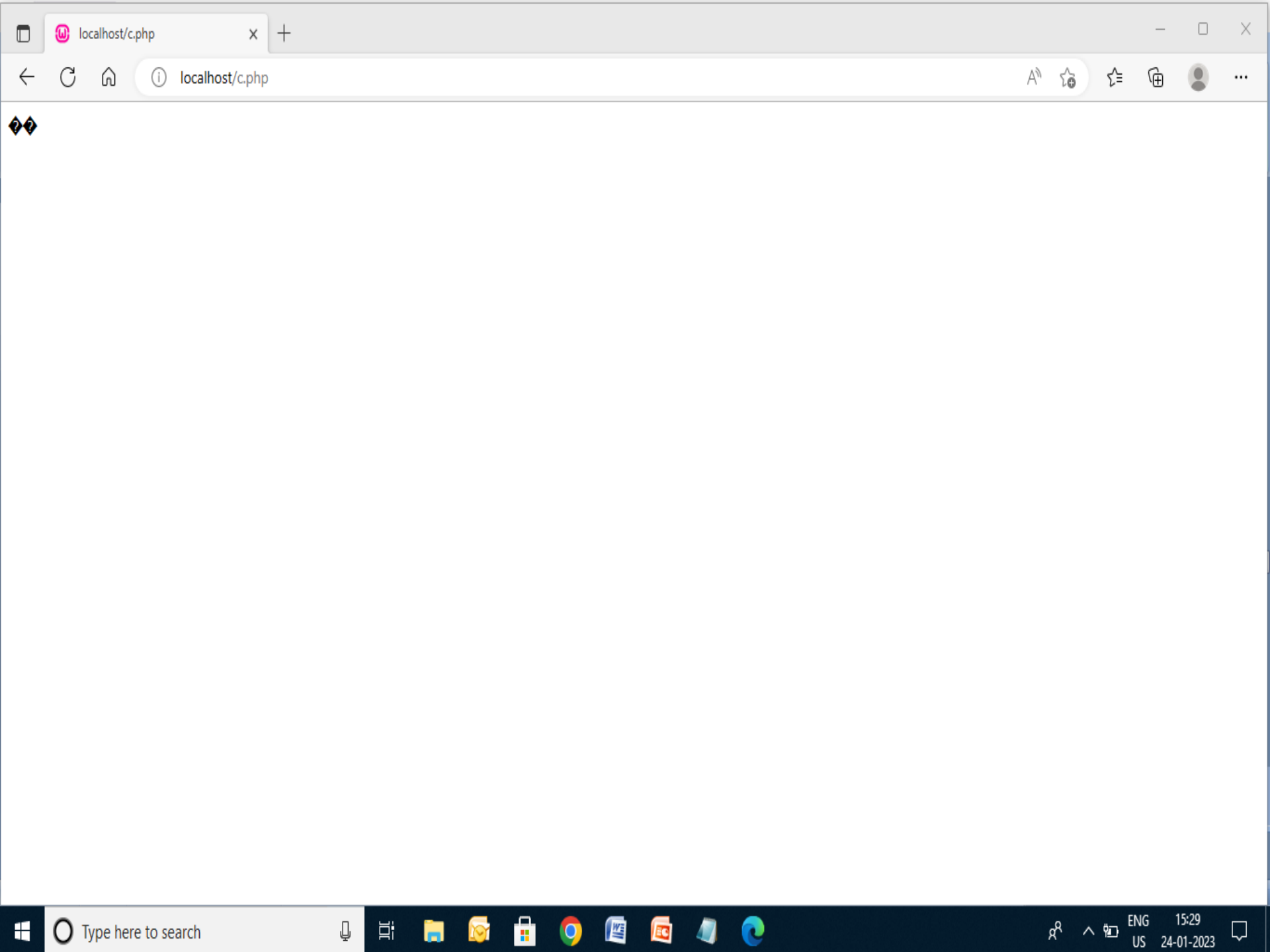


This is Call By Reference

FUNCTION – CALL BY REFERENCE EXAMPLE 8

```
<?php
function increment(&$i)
{
    $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```



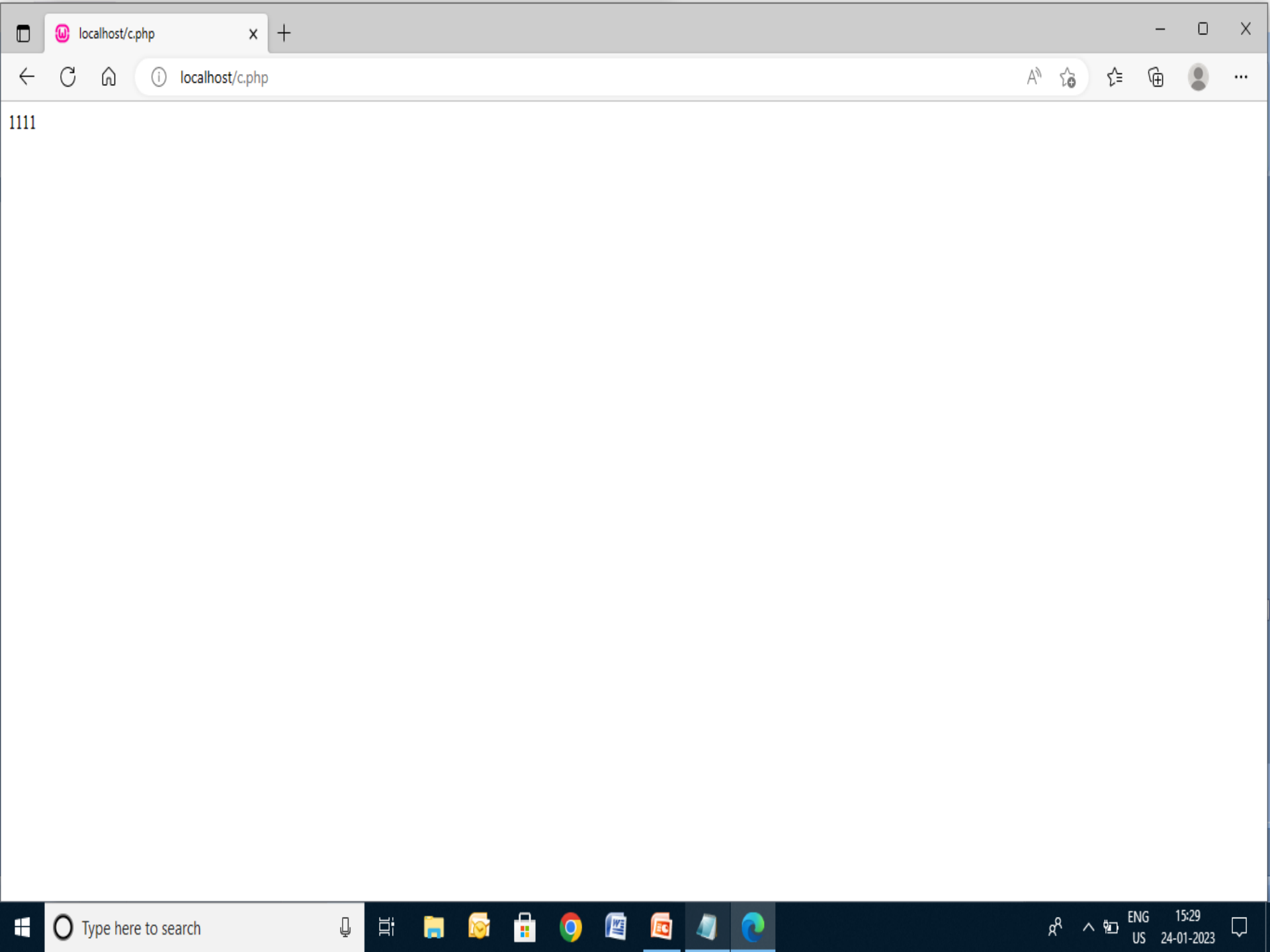


FUNCTION – CALL BY REFERENCE

EXAMPLE 7

```
<?php
function increment(&$i)
{
    $i++;
    echo $i;
}
$i = 10;
increment($i);
echo $i;
?>
```





1111

Call by Value

While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”.

In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function.

With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function.

Call by Reference

While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as “Call By References.

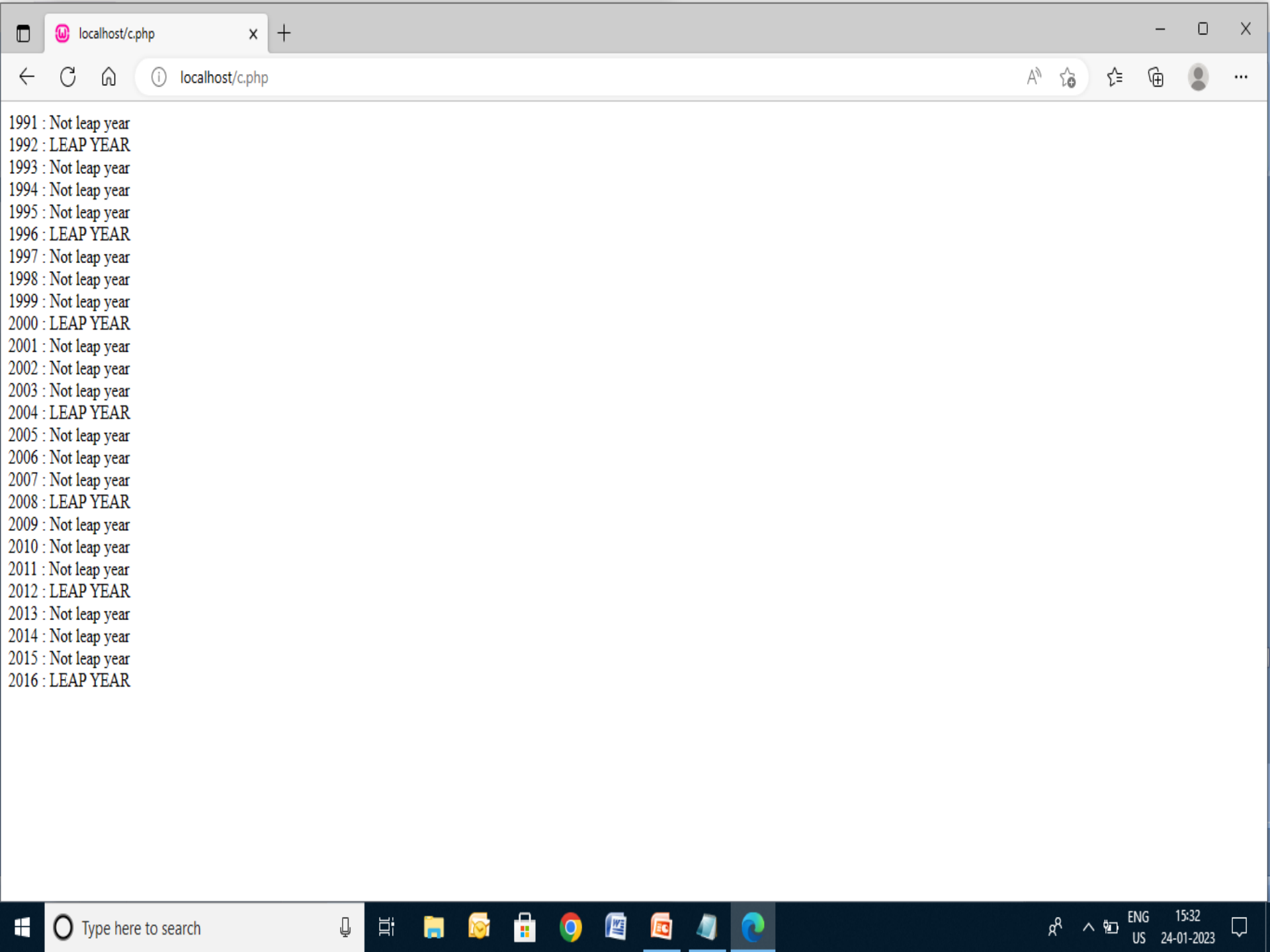
In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function.

With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate them.

EXAMPLE 9 // LEAP YEAR

```
<?php
function isLeap($year)
{
    return (date('L',mktime(0,0,0,1,1,$year))==1);
}
//For testing
for($year=1991; $year<=2016; $year++)
{
    If (isLeap($year))
    {
        echo "$year: LEAP YEAR<br />\n";
    }
    else
    {
        echo "$year: Not leap year<br />\n";
    }
}
?>
```



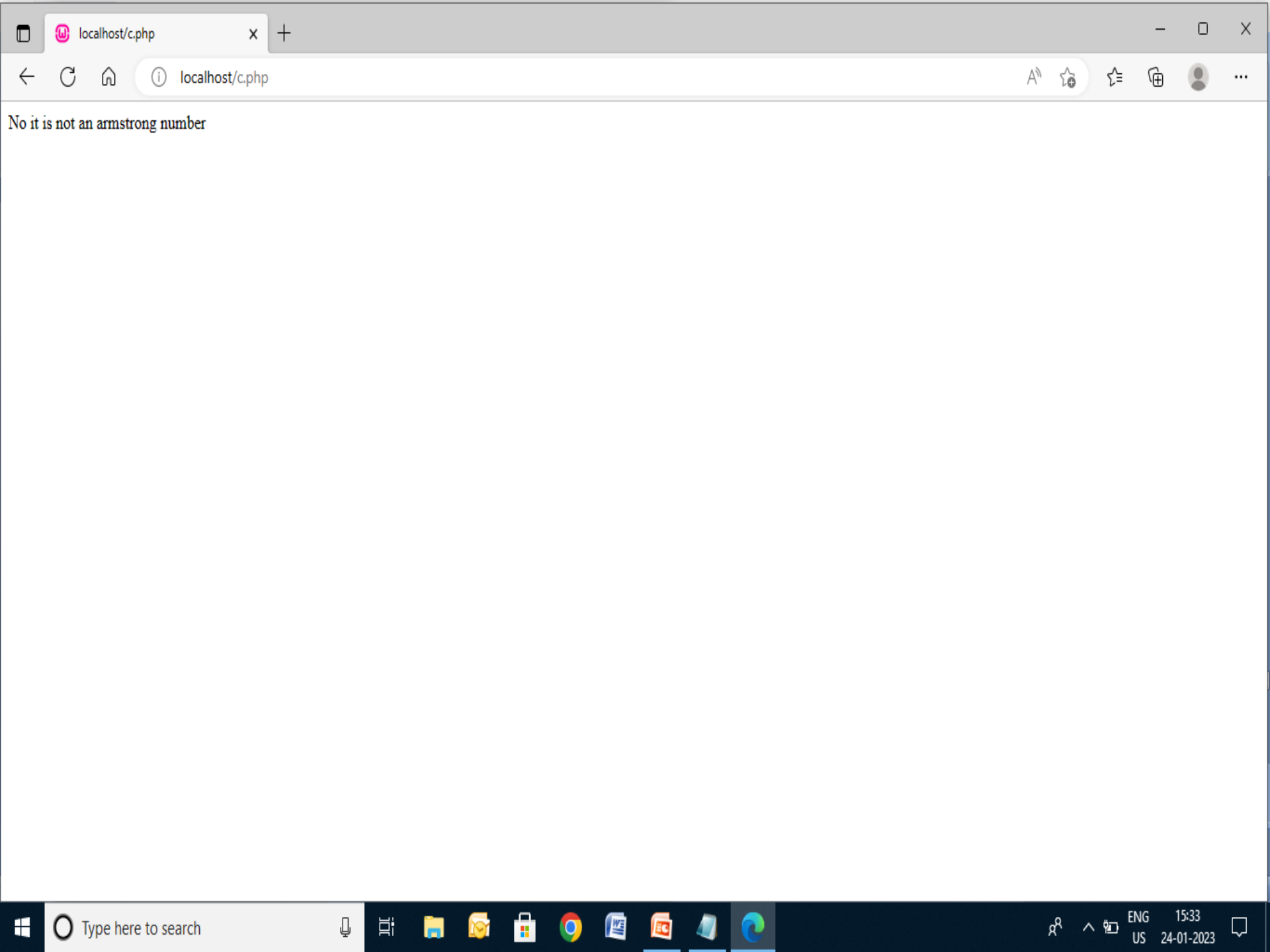


1991 : Not leap year
1992 : LEAP YEAR
1993 : Not leap year
1994 : Not leap year
1995 : Not leap year
1996 : LEAP YEAR
1997 : Not leap year
1998 : Not leap year
1999 : Not leap year
2000 : LEAP YEAR
2001 : Not leap year
2002 : Not leap year
2003 : Not leap year
2004 : LEAP YEAR
2005 : Not leap year
2006 : Not leap year
2007 : Not leap year
2008 : LEAP YEAR
2009 : Not leap year
2010 : Not leap year
2011 : Not leap year
2012 : LEAP YEAR
2013 : Not leap year
2014 : Not leap year
2015 : Not leap year
2016 : LEAP YEAR

EXAMPLE 10 // ARMSTRONG

```
<?php
$num=123;
$total=0;
$x=$num;
while($x!=0)
{
    $rem=$x%10;
    $total=$total+$rem*$rem*$rem;
    $x=$x/10;
}
if($num==$total)
{
    echo "Yes it is an Armstrong number";
}
else
{
    echo "No it is not an armstrong number";
}
?>
```



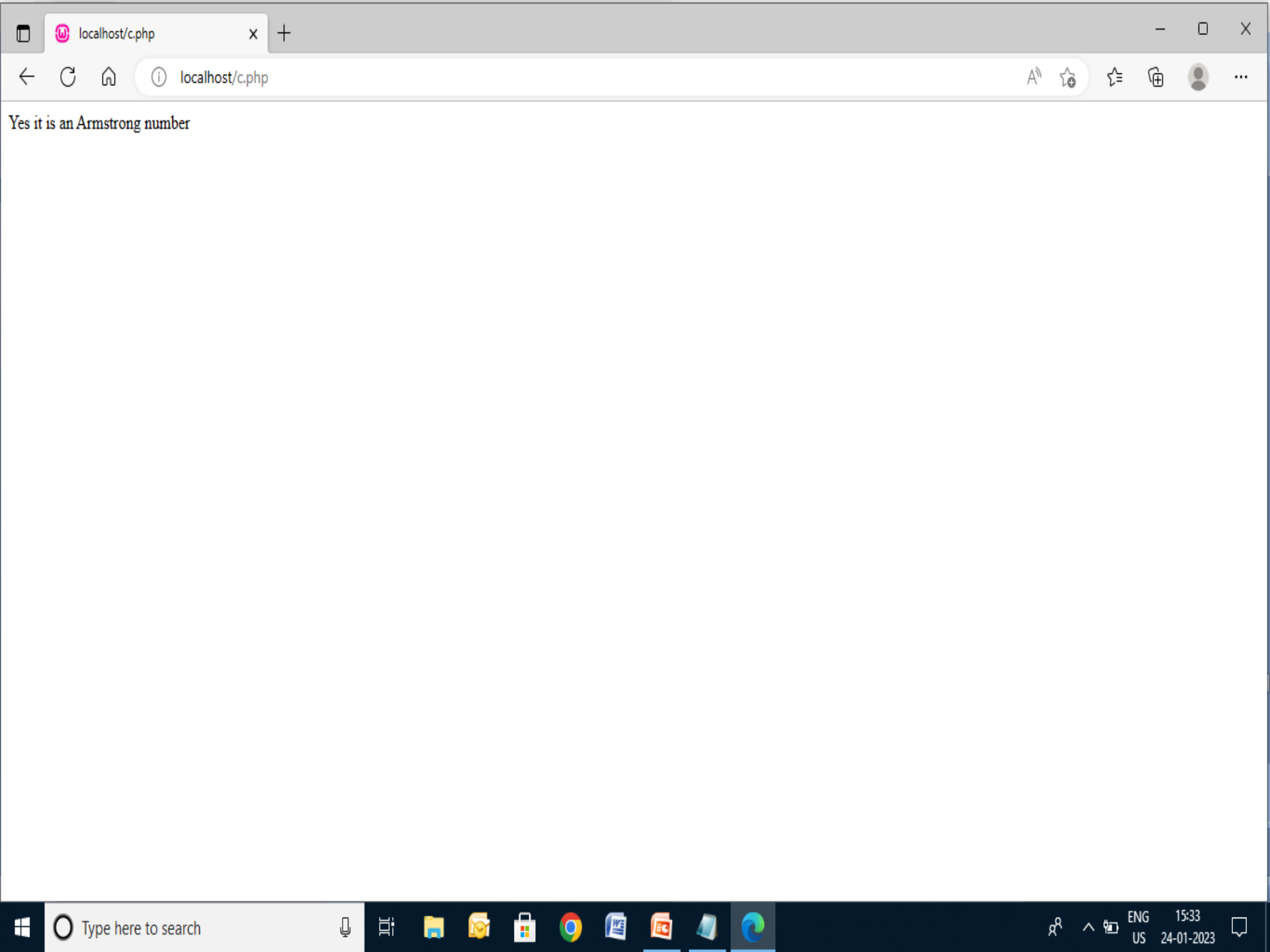


No it is not an armstrong number

EXAMPLE 10 (A) // ARMSTRONG

```
<?php
$num=153;
$total=0;
$x=$num;
while($x!=0)
{
    $rem=$x%10;
    $total=$total+$rem*$rem*$rem;
    $x=$x/10;
}
if($num==$total)
{
    echo "Yes it is an Armstrong number";
}
else
{
    echo "No it is not an armstrong number";
}
?>
```





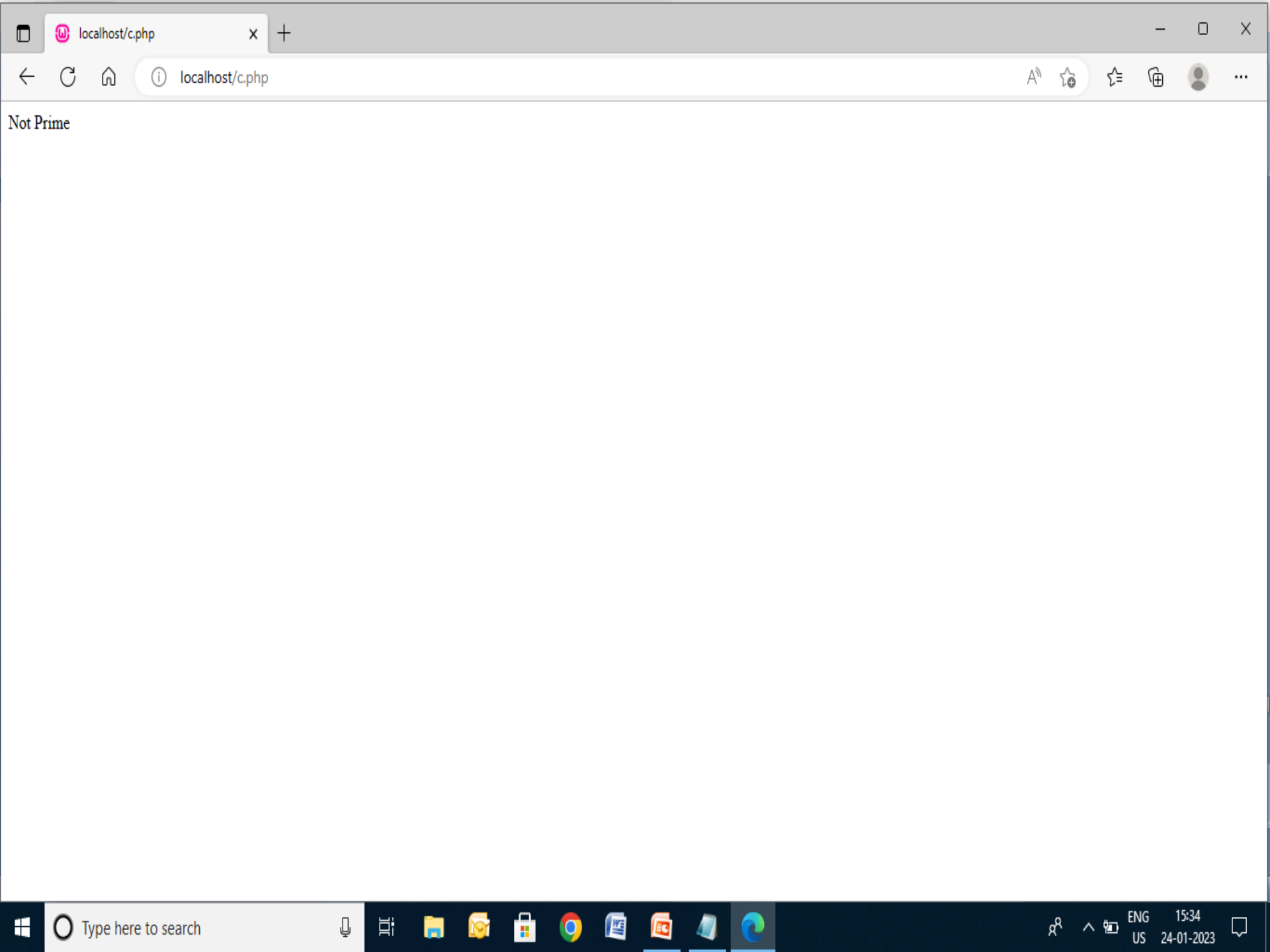
Yes it is an Armstrong number

EXAMPLE 11 PRIME

```
<?php
function primeCheck($number)
{
    if ($number == 1)
        return 0;
    for ($i = 2; $i <= $number/2; $i++)
    {
        if ($number % $i == 0)
            return 0;
    }
    return 1;
}
```

```
$number = 10;
$flag = primeCheck($number);
if ($flag == 1)
    echo "Prime";
else
    echo "Not Prime"
?>
```





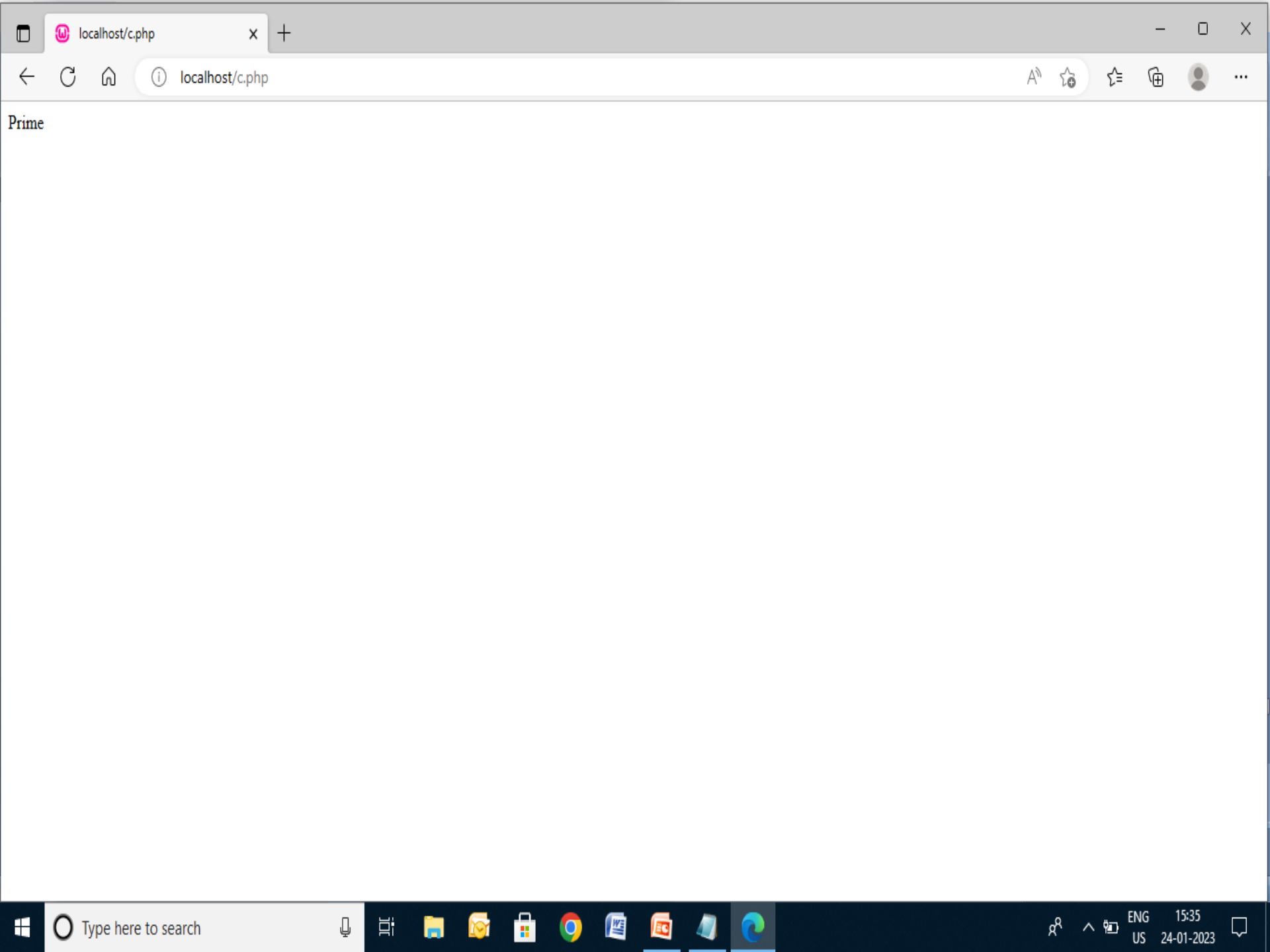
Not Prime

EXAMPLE 11 (A) PRIME

```
<?php
function primeCheck($number)
{
    if ($number == 1)
        return 0;
    for ($i = 2; $i <= $number/2; $i++)
    {
        if ($number % $i == 0)
            return 0;
    }
    return 1;
}
```

```
$number = 11;
$flag = primeCheck($number);
if ($flag == 1)
    echo "Prime";
else
    echo "Not Prime"
?>
```





Prime

Thank You

