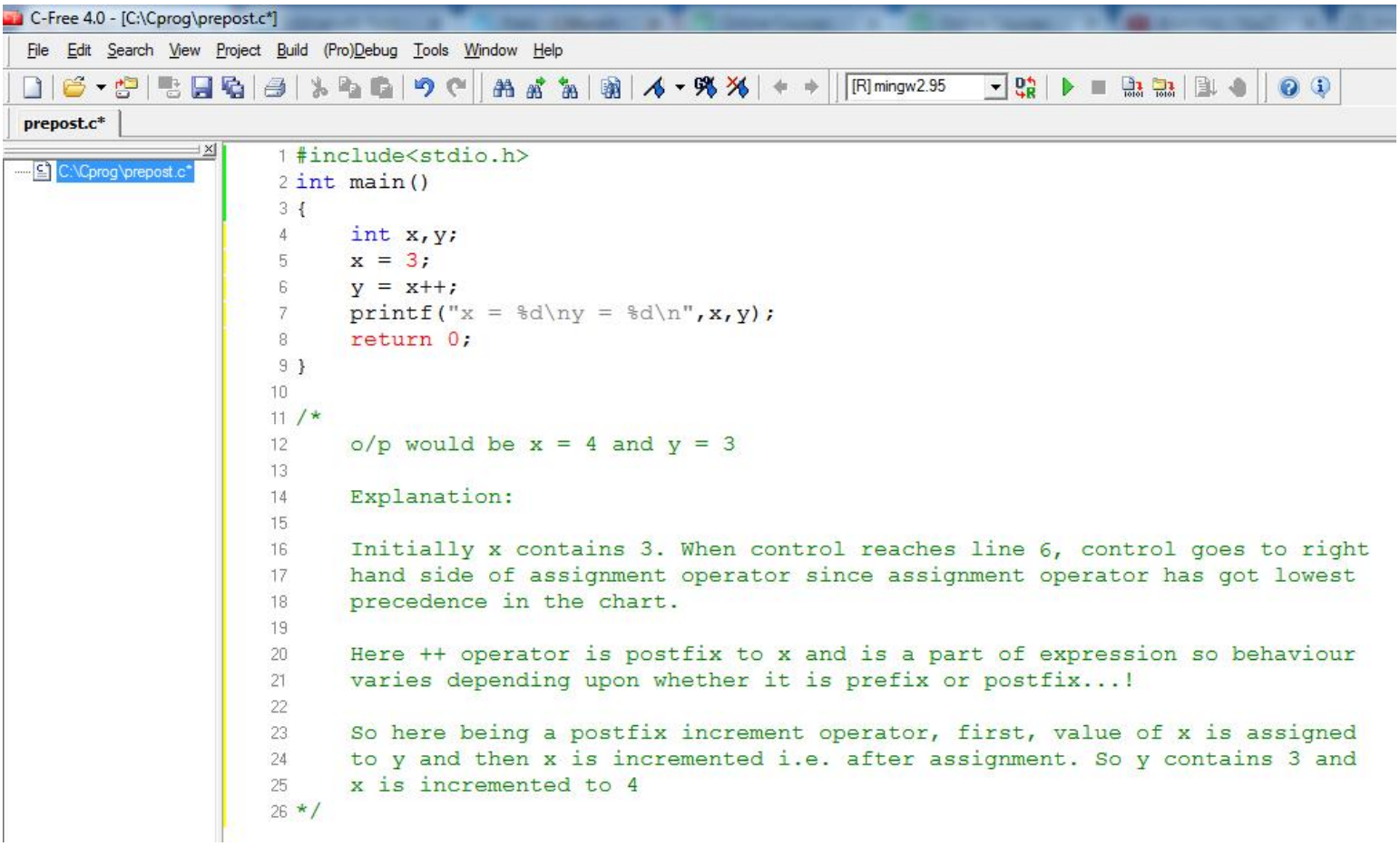
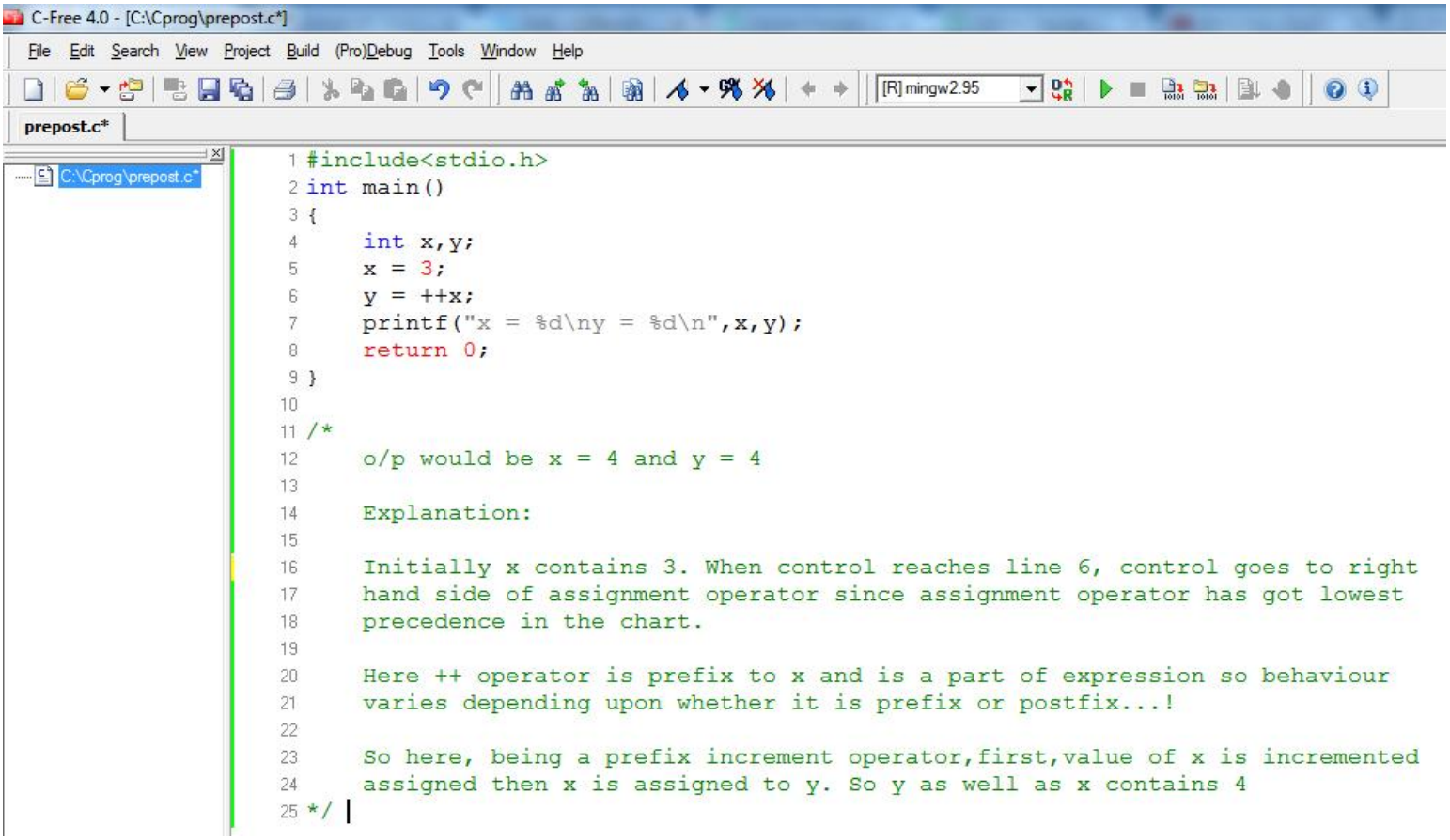

C-Free 4.0 - [C:\Cprog\prepost.c*]
File Edit Search View Project Build (Pro)Debug Tools Window Help
[R] mingw2.95
prepost.c*
C:\Cprog\prepost.c*
#include<stdio.h>
int main()
{
int x,y;
x = 3;
y = x++;
printf("x = %d\ny = %d\n",x,y);
return 0;
}
/*
o/p would be x = 4 and y = 3
Explanation:
Initially x contains 3. When control reaches line 6, control goes to right hand side of assignment operator since assignment operator has got lowest precedence in the chart.
Here ++ operator is postfix to x and is a part of expression so behaviour varies depending upon whether it is prefix or postfix...!
So here being a postfix increment operator, first, value of x is assigned to y and then x is incremented i.e. after assignment. So y contains 3 and x is incremented to 4
*/


C-Free 4.0 - [C:\Cprog\prepost.c*]
File Edit Search View Project Build (Pro)Debug Tools Window Help
[R] mingw2.95
prepost.c*
C:\Cprog\prepost.c*
#include<stdio.h>
int main()
{
int x,y;
x = 3;
y = ++x;
printf("x = %d\ny = %d\n",x,y);
return 0;
}
/*
o/p would be x = 4 and y = 4
Explanation:
Initially x contains 3. When control reaches line 6, control goes to right hand side of assignment operator since assignment operator has got lowest precedence in the chart.
Here ++ operator is prefix to x and is a part of expression so behaviour varies depending upon whether it is prefix or postfix...!
So here, being a prefix increment operator,first,value of x is incremented assigned then x is assigned to y. So y as well as x contains 4
*/

C-Free 4.0 - [C:\Cprog\prepost.c*]

File Edit Search View Project Build (Pro)Debug Tools Window Help

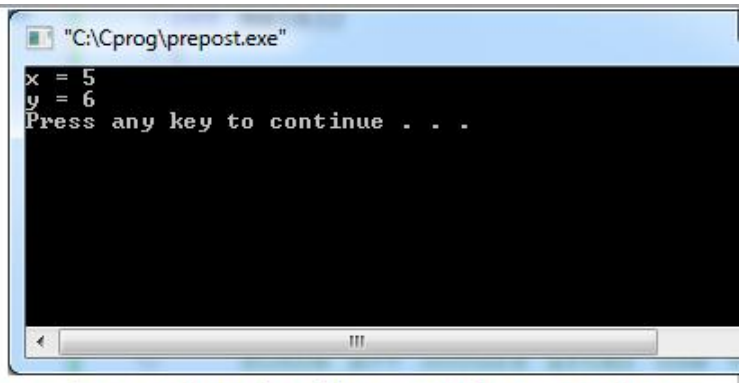
[R] mingw2.95

prepost.c*

C:\Cprog\prepost.c*

```c
#include<stdio.h>
int main()
{
    int x,y;
    x = 3;
    y = ++x;
    printf("x = %d\ny = %d\n",x,y++);
    return 0;
}

/*
    Here again o/p would be x = 4 and y = 4 though y in the printf statement
    is incremented.

    Explanation:

    Initially x contains 3. When control reaches line 7, Here ++ operator
    is postfix to y.

    So first,value of y would be printed on the o/p window and then y is
    incremented. Ultimately memory location of y would contain 5.
    It would be confirmed if you try to print contents of y in the next
    statement... (try it out)
*/
```

```c
#include<stdio.h>
int main()
{
    int x,y;
    x = 3;
    y = x++ + x++;
    printf("x = %d\ny = %d\n",x,y);
    return 0;
}

/*
    Since x++ occurs after the varibale x, its value is first used to evaluate
    the expression (line 6) and then x is incremented twice. Thus x + x would
    result into 6 and then the result would be assigned to z. After this, the
    first x would increment the value of x to 4, followed by second x++, which
    would further increment x to 5
*/
```



```c
#include<stdio.h>
int main()
{
    int x,y;
    x = 3;
    y = x++ + ++x;
    printf("x = %d\ny = %d\n",x,y);
    return 0;
}

/*
    This one is bit tricky, follow it carefully. We might be led to believe
    that while evaluating z, what would be added is 3 and 4. But this is
    definitely wrong. This is because while calculating z, the very first
    operation that is performed is ++x, which increments the value of x to 4.
    So by the time the addition (x+x) is performed, x has already become 4.
    Thus the addition  4+4 would be performed and not 3+4. After this the
    result of the addition 4+4 is assigned to the varibale x. and then
    x is incremented to 5 (because of x++)
*/
```