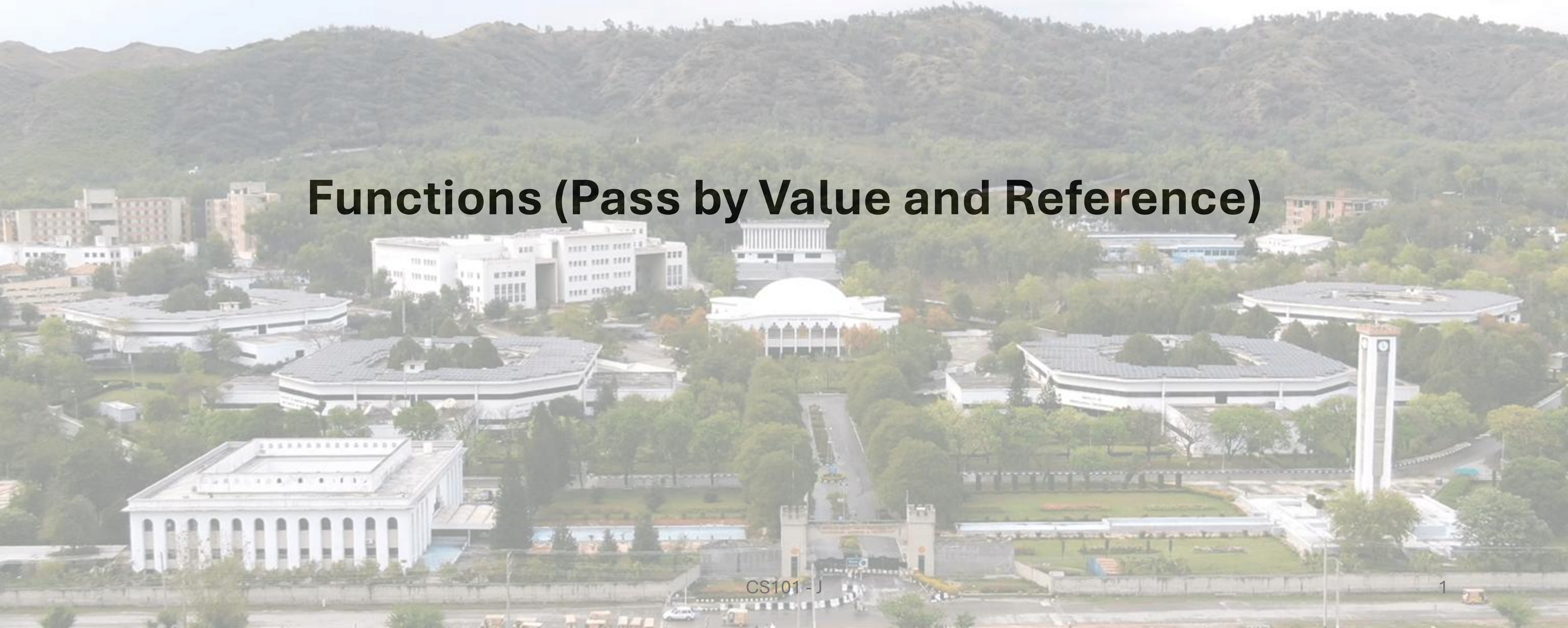


Week 9 – CS 101 –J

Functions (Pass by Value and Reference)

Functions

- In this lecture, we will study about function receiving 2 types of inputs:
  - Receiving an input value which it can modify (pass by reference)
  - Receiving an input value which it can not modify (pass by value)

- By default, pass by value is utilized

Function – call by value

- In order to understand the term call by value, let me present you with a code

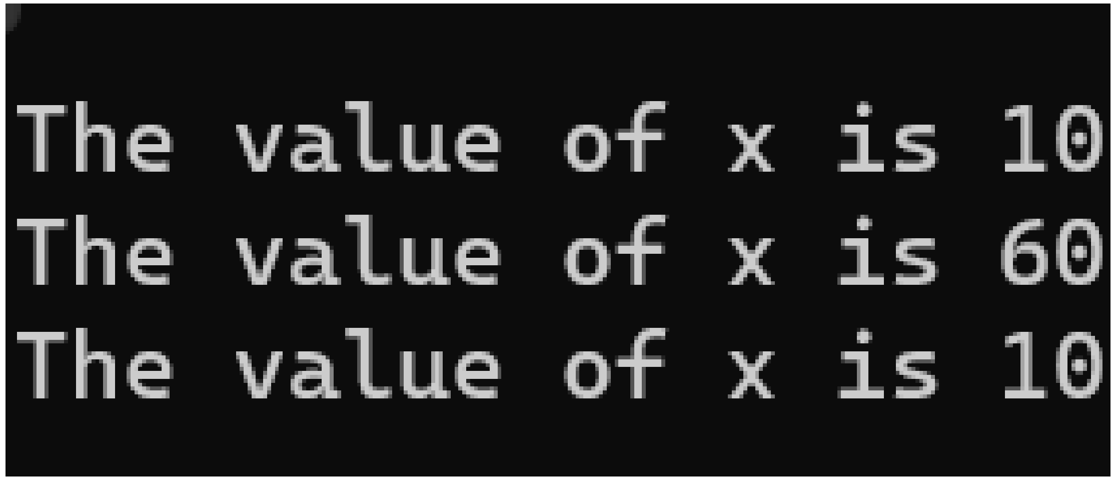
Example 1

```
#include <iostream>
using namespace std;

void f1(int x);

int main()
{
        int x;
        x=10;
        cout<<"The value of x is "<<x<<endl;
        f1(x);
        cout<<"The value of x is "<<x<<endl;
        return 0;
}
```

```
void f1(int x)
{
        x=x+50;
        cout<<"The value of x is "<<x<<endl;
}
```

Now what is the output?

Example 1



```
The value of x is 10
The value of x is 60
The value of x is 10
```

```
#include <iostream>
using namespace std;

void f1(int x);

int main()
{
        int x;
        x=10;
        cout<<"The value of x is "<<x<<endl;
        f1(x);
        cout<<"The value of x is "<<x<<endl;
        return 0;
}
```

```
void f1(int x)
{
        x=x+50;
        cout<<"The value of x is "<<x<<endl;
}
```

Now what is the output?

Function – call by value

- In the previous example, what happens to the value of variable x in function f1?

- Function f1 changes the value of x by adding 50 with it, but when the program returns to main function, the value of x is again 10

- This is called as passing by value – meaning a copy of the value of x (or variable) is passed
    - Any change in the copy of variable is not reflected in the original one

- You can compare it with photostat example (photocopy of a document)

Function – call by reference

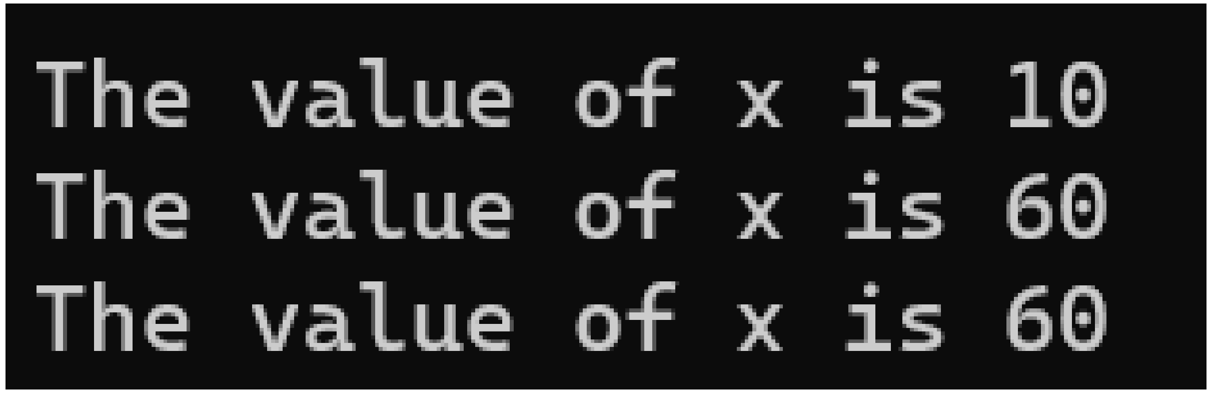
- In call by reference, the original variable is passed to function and if any change is made by the function, then it is reflected in the original variable also

- In order to pass the value of a variable by reference, we need to use & symbol

- Let us revisit the code again

Example 2

```
#include <iostream>
using namespace std;

void f1(int &x);

int main()
{
        int x;
        x=10;
        cout<<"The value of x is "<<x<<endl;
        f1(x);
        cout<<"The value of x is "<<x<<endl;
        return 0;
}
```



```
The value of x is 10
The value of x is 60
The value of x is 60
```

```
void f1(int &x)
{
        x=x+50;
        cout<<"The value of x is "<<x<<endl;
}
```

Now what is the output?

Recursion

- Can a function call itself again?

- Yes, it can – and this process is called as recursion

- When using recursion, it is VERY IMPORTANT to write a condition, called as base case – (used to stop infinite calling of function itself)

Recursion

- There are certain problem (engineering and scientific) which requires recursion.
  - E.g. Factorial, Fibonacci Series, Tower of Hanoi, Graph Searching

- Not all problems require recursion concept

Recursion

- Example: factorial

  $n! = n * (n - 1) * (n - 2) * \dots * 1$

  - Recursive relationship ($n! = n * (n - 1)!$)

    $5! = 5 * 4!$

    $4! = 4 * 3! \dots$

  - Base case ($1! = 0! = 1$)

Recursion

```
void fun1(int n)
  {
          if (n<1)
                  return;
             cout<<n;
          fun1(n-1);
}
```

What would be the output of fun1(5)?

Recursion unrolled

What would be the output of fun1(5)?

```
fun1(5)
   5 <1 ?
  cout<<n; →                5
   fun1(4)
      4 <1 ?
      cout<<n; →            4
      fun1(3)
         3 <1 ?
         cout<<n; →         3
         fun1(2)
            2<1 ?
            cout<<n; →      2
              fun1(1)
                1 <1 ?
                cout<<n; →  1
                 fun1(0)
                    0 < 1
                      return;
```

Recursion

- Understanding the flow of statements in a recursion call is very important.

- Let us re-visit the code again and write cout after the function calls itself

Recursion with Example fun2

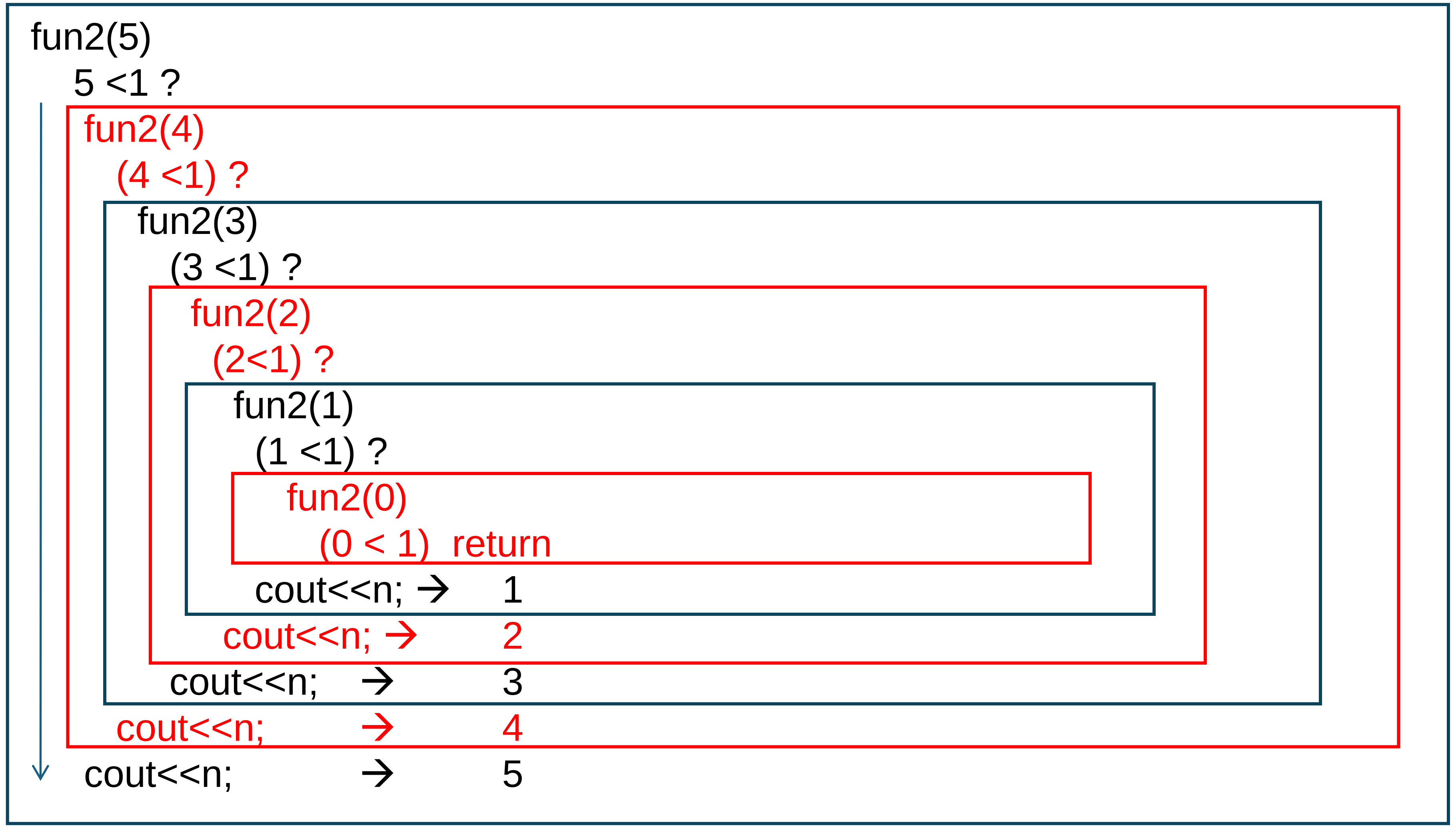
```
void fun2(int n)
    {
    if (n<1)
            return;
    fun2(n-1);
     cout<<n;
  }
```

What would be the output of fun2(5)?

Recursion unrolled

What would be the output of fun2(5)?

```
void fun2(int n){
    if (n<1) return;
      fun2(n-1);
      cout<<n;
   }
```



```
fun2(5)
   5 <1 ?
   fun2(4)
     (4 <1) ?
     fun2(3)
       (3 <1) ?
       fun2(2)
         (2<1) ?
         fun2(1)
           (1 <1) ?
           fun2(0)
             (0 < 1)  return
           cout<<n; →     1
         cout<<n; →       2
       cout<<n;   →       3
     cout<<n;     →       4
   cout<<n;       →       5
```

Factorial: Example Using Recursion

$n! = n * (n – 1) * (n – 2) * … * 1$

- Recursive relationship ($n! = n * (n – 1)!$)

  $5! = 5 * 4!$

  $4! = 4 * 3!…$

- Base case ($1! = 0! = 1$)

- C++ code for `factorial` function

```
unsigned long factorial(unsigned long n )
{
  if ( n <= 1)   // base case
        return 1;
  else         // recursive case
        return n * factorial(n -1);
}
```

Factorial: Non-recursive implementation

```
unsigned long factorial(unsigned long n )
{
for (int i = n-1; i >0; i--)
                  n = n*i;


return n;
}

int main()
{
    cout << factorial(5);
     return 0;
}
```

Recursion vs. Iteration

- Repetition
  - Iteration: explicit loop
  - Recursion: repeated function calls

- Termination
  - Iteration: loop condition fails
  - Recursion: base case recognized

- Both can have infinite loops

Inline functions

- If a function has very statements, as 1 line code or 2 line codes, then we can also use inline functions for that
- For example:

```
inline double sqr(double x )
   { return x * x; }
```

- What are the advantages of inline function?
  - The compiler does not call the function, rather it pastes the code when the function is called

Inline functions

- Inline functions are not used if
  - A function contains a loop
  - A function uses static variables
  - A function uses recursion
  - A function contains switch statements

- Use inline function is a function has say for example: one cout statement, and one maths statement (like square or cube or max or min)

Function default parameters

- If a function has 2 input parameters, is it necessary to ALWAYS pass 2 input parameters?

- No, -- you can use some default values of variables also (in case nothing is passed in input)
- **`int fun (int x = 1, int y = 2);`**

Function overloading

- Can you write 2 functions with same name, and same number of arguments?

- Yes, you can – and this is sometimes good also

- We call this as function overloading

Function overloading

```
int square (int x)
{
 int y;
 y= x*x;
return y;
}
float square (float x)
{
 float y;
 y= x*x;
return y;
}
```

How much more course to cover before Final Exam?

- Final Exam will be of 3 hours and have 40% weight

- We will be having 2 more quizzes, and 3 assignments
- Total 5 quizzes, 5 assignments

- Topics Remaining: Arrays, and Pointers