C++ Programming – Lecture 3

Recursion

- A function that calls itself is called a recursive function
- Any function, including main( ) can become a recursive function
- Recursive call always leads to an infinite loop. So a provision must be made to get outside this infinite loop
- The provision is done by making the recursive call either in the if block or in the else block
- If recursive call is made in the if block, else block should contain the end condition logic
- If recursive call is made in the else block, if block should contain the end condition logic
- Fresh set of variables are born during each function call - normal call and recursive call
- Variables die when control returns from a function
- Recursive function may or may not have a return statement
- Recursion is an alternative for loop in logics which are expressible in the form of themselves
- Recursive calls are slower than an equivalent while / for / do-while loop
- Understanding how a recursive function is working becomes easy if you make several copies of the same function on paper and then perform a dry run of the program
- Understanding how a recursive function is working becomes easy if you run the program in debug mode and follow the control flow by single-stepping through the program

Pointers

- Pointers are variables which hold addresses of other variables
- Address, Reference, Memory Location, Cell number are same
- & - Address of operator, * - Value at address or Indirection operator
- &, * - Pointer operators
- & can be used only with a variable
- * can be used with variable, constant or expression
- variable is same as *&variable

- Example of pointer usage :

  ```
  int i = 10  ;  int *j ;  int **k ;
  j = &i ;   k = &j ;
  cout <<  I << *j << **k ;
  ```

  Here j is an integer pointer. k is a pointer to an integer pointer
- Even if a is a 4-byte variable, &a, gives address of first out of these 4 bytes
- Using an integer ptr – Use * to reach integer
- Using a pointer to an integer pointer – Use ** to reach integer

References

- References are constant pointers that get automatically dereferenced
- A reference can be tied with only one variable
- A variable may have multiple references
- A reference to a reference is not possible
- An array of references is not allowed