Name of Teacher: Miss Radhika M. Patil

Class: B.Sc. Computer Science (Entire)- II **Semester : 3**

Course Title: Object Oriented Programming using C++

Unconditional Control Statements

- Unconditional control statements do not need to satisfy any condition. They immediately move control from one part of the program to another part.
- The break, continue, goto, and return statements unconditionally transfer program control within a function. C++ has four statements that perform an unconditional branch :

1. break
2. continue
3. return
4. goto

Of these, you may use return and goto anywhere in the program whereas break and continue are used inside smallest enclosing like loops etc

1. break statement

A break statement terminates the execution of the loop and the control is transferred to the statement immediately following the loop. i.e., the break statement is used to terminate loops or to exit from a switch.

- It can be used within for, while, do-while, or switch statement.
- The break statement is written simply as break;
- In case of inner loops, it terminates the control of inner loop only.

Program:

```
void main()
{
     int i;
     for(i=0;i<100;i++)
     {
         if(i= =10)
             break;
         cout<<i<<“  “;
     }
    cout<<“\n Loop terminated”;
  getch();
}
```

Output:

0 1 2 3 4 5 6 7 8 9
Loop terminated

2. continue statement

- The continue statement is used to bypass the remainder of the current pass through a loop.
- The loop does not terminate when a continue statement is encountered. Instead, the remaining loop statements are skipped and the computation proceeds directly to the next pass through the loop.
- The continue statement can be included within a while, do-while, for statement.
- It is simply written as “continue”.
- The continue statement tells the compiler “Skip the following Statements and continue with the next Iteration”.
- In “while” and “do” loops continue causes the control to go directly to the test – condition and then to continue the iteration process.
- In the case of “for” loop, the updation section of the loop is executed before test-condition, is evaluated.

Program:

```
void main()
{
     int i;
     for(i=10;i>0;i--)
     {
          if(i= =5)
               continue;
          cout<<i<<“  “;
     }
    cout<<“\n Loop terminated”;
  getch();
}
```

Output:

10 9 8 7 6 4 3 2 1
Loop terminated

3. return statement

return statement causes immediate termination of function in which it appears and transfers the control to the calling function.

Program

```
void main()
{
    int x,y;
    cout<<“\n Enter x and y:”;
    cin>>x>>y;
    int sum;
    sum=add(x,y);
    cout<<“\n Sum=“<<sum;
getch()  ;
}

int add(int x,int y)
{
   return(x+y);
}
```

Output:

```
Enter x and y: 10 20
Sum= 30
```

goto Statement

- The goto statement can transfer the program control anywhere in the program.
- The target destination of a goto statement is marked by a label. the target label and goto must appear in the same function.

Here is the syntax of the goto statement in C++:

Syntax:

```
goto label;                    label:
.....                          statement;
.....          OR              .....
.....
label:
statement;
```

where label is a user supplied identifier and can appear either before or after goto. For example, the following code fragment :

Program

```
// goto loop example
#include <iostream.h>
#include <conio.h>
 void main ()
{
    int n=10;
      mylabel:
         cout << n << ", ";
         n--;
      if (n>0)
         goto  mylabel;
      cout << "Exited!"<<endl;
   getch();
}
```

Output:

```
10   9   8   7   6   5    4    3   2    1 Exited
```

THANK YOU…