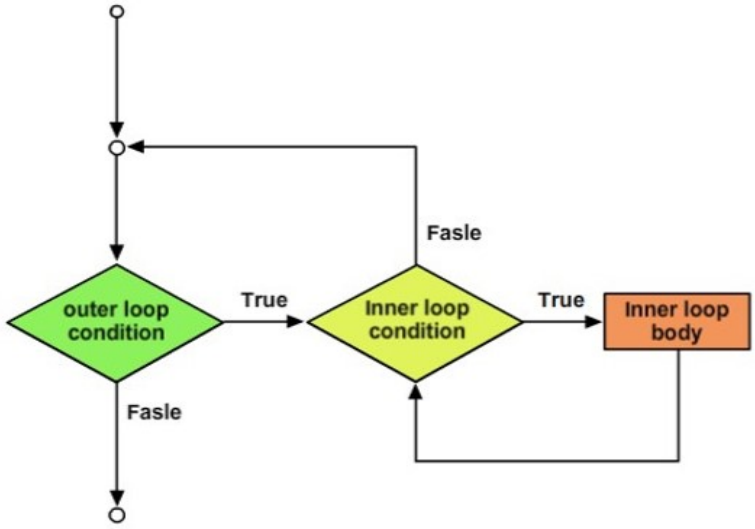
Programming and Problem Solving through C Language
O Level / A Level

Chapter -4 : Conditional Statements and Loops

Nested Loop - Introduction

- A nested loop is a loop within a loop, an inner loop within the body of an outer one.
- Structure of nested loop



- What happens is that the first pass of the outer loop triggers the inner loop, which executes to completion.
- Then the second pass of the outer loop triggers the inner loop again.
- This repeats until the outer loop finishes.
- **A break within the inner loop , interrupt the inner loop only.**
- When the user "nest" two loops, the outer loop takes control of the number of complete repetitions of the inner loop.

The syntax for a **nested for loop** statement −

```
for ( init; condition; increment ) {

   for ( init; condition; increment ) {
      statement(s);
   }
   statement(s);
}
```

The syntax for a **nested while loop** statement −

```
while(condition) {

   while(condition) {
      statement(s);
   }
   statement(s);
}
```

The syntax for a **nested do...while loop** statement −

```
do {
   statement(s);

   do {
      statement(s);
   }while( condition );

}while( condition );
```

Example-1 :

```
main()
{
   for (int i=0; i<=10; i++)
   {
      for (int j=0; j<=10; j++)
      {
         printf("%d, %d",i ,j);
      }
   }
}
```

Example-2 :

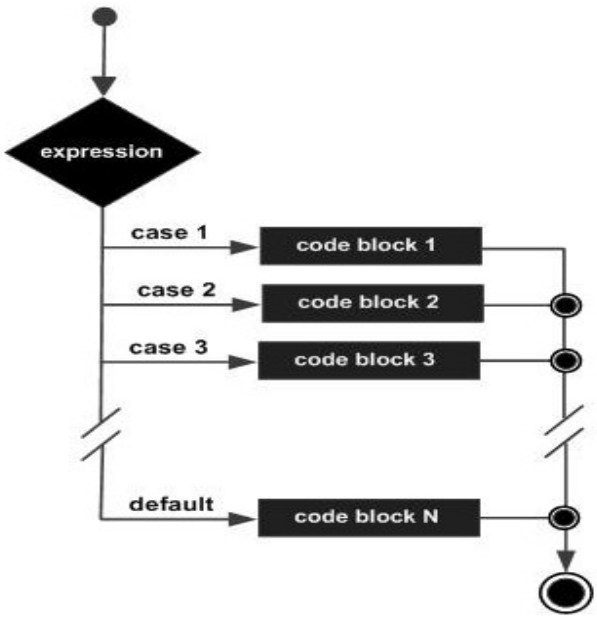
The following program uses a nested for loop to find the prime numbers from 2 to 100 −

```
#include <stdio.h>
void main () {
   /* local variable definition */
   int i, j;

   for(i = 2; i<100; i++) {
      for(j = 2; j <= (i/j); j++)
      if(!(i%j)) break; // if factor found, not prime
      if(j > (i/j)) printf("%d is prime\n", i);
   }
}
```

Switch Statement

- Switch statement, allows the program to execute different statements based on an expression that can have more than two values.
- When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.



Syntax

```
switch(expression) {

   case constant-expression  :
      statement(s);
      break; /* optional */

   case constant-expression  :
      statement(s);
      break; /* optional */

   /* you can have any number of case statements */
   default : /* Optional */
   statement(s);
}
```

Example

```
#include <stdio.h>

void  main () {

   /* local variable definition */
   char grade = 'B';

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }
   printf("Your grade is  %c\n", grade );
}
```

Output

```
Well done
Your grade is B
```

Goto Statement

- The goto statement is one of C′s unconditional jump, or branching statements.
- When program execution reaches a goto statement, execution immediately jumps, or branches , to the location specified by the goto statement.
- This statement is unconditional because execution always branches when a goto statement is encountered; the branch doesn′t depend on any program conditions.
- A goto statement and its target must be in the same function, but they can be in different blocks.
- A break statement, a continue statement, or a function call can eliminate the need for a goto statement.

Example

```
int main()
{
  int age;
  Vote:
    printf("you are eligible for voting");

  NoVote:
    printf("you are not eligible to vote");

  printf("Enter you age:");
  scanf("%d", &age);
  if(age>=18)
      goto Vote;
  else
      goto NoVote;

  return 0;
}
```

Assignement

1. Write a program to print using nested loop

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

2. Write a program to print the table of 1 to 10.
3. Write a program to print using nested loop

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```