Code in 10 days

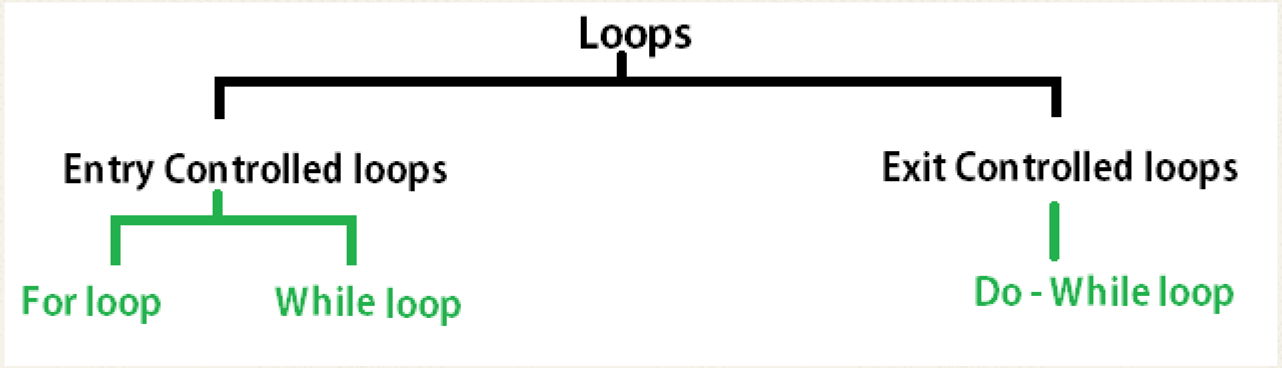
Day 3

Topics for Today

- Iteration statements
- Jump statements
- Example programs

Iteration statement

Loops are used when we need to repeatedly execute a block of statements.



Types of Loops

- Entry Controlled loops:

In this type of loops the test condition is tested before entering the loop body. For Loop and While Loop are entry controlled loops.

- Exit Controlled Loops:

In this type of loops the test condition is tested or evaluated at the end of loop body. do – while loop is exit controlled loop.

Elements of loop

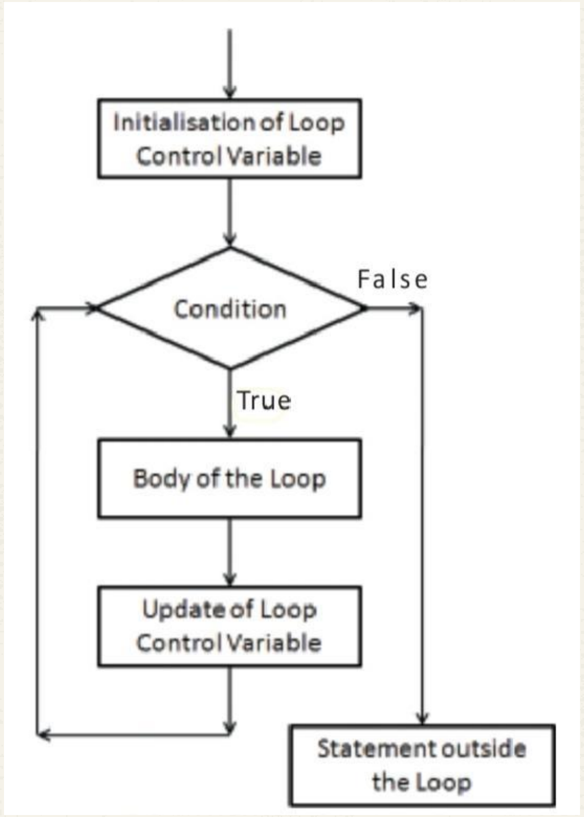
1. Initialisation: Before entering a loop, its control variable must be initialized During initialisation, the loop control variable gets its first value. initialisation statement is executed only once, at the beginning of the loop.
1. Test expression: It is a relational or logical expression whose value is either True or False. It decides whether the loop-body will be executed or not. If the test expression evaluates to True, the loop-body gets executed, otherwise it will not be executed.

3. Update statement: The update statement modifies the loop control variable by
changing its value. The update statement is executed before the next iteration.
4. Body of the loop: The statements that need to be executed repeatedly constitute
the body of the loop. It may be a simple statement or a compound statement

While loop

Syntax

```
initialisation of loop control
variable;
while(test expression)
{
body of the loop;
updation of loop control
variable;
}
```


Initialisation of Loop Control Variable
Condition
False
True
Body of the Loop
Update of Loop Control Variable
Statement outside the Loop

While loop example

```
//To print the first 10 natural numbers
#include<iostream>
using namespace std;
int main()
{
int n = 1;
while(n <= 10)
 {
 cout<< n << " ";
 ++n;
 }
return 0;
}
```

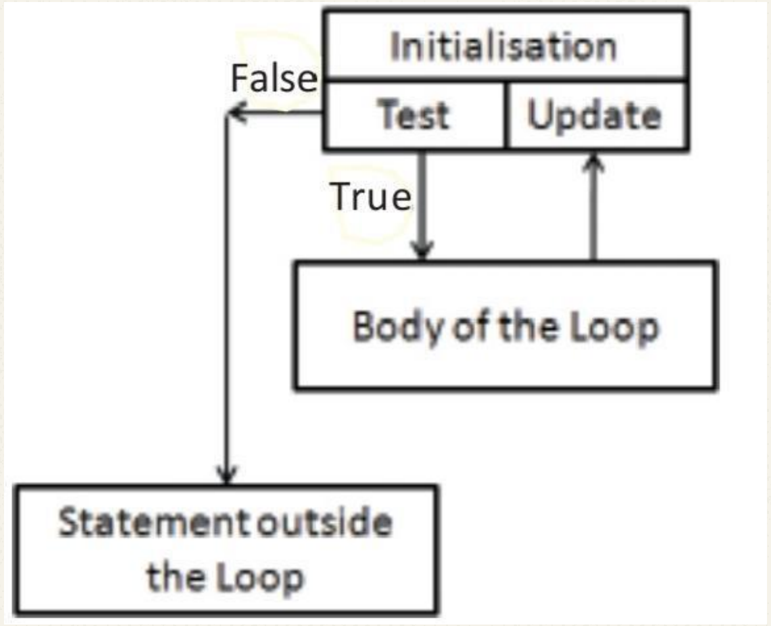
While loop example

```
// To find the sum of even numbers upto 20
#include<iostream>
using namespace std;
int main()
{
int i, sum = 0;
i = 2;
while( i<= 20)
{
 sum = sum + i;
 i = i + 2;
}
cout<<"\nThe sum of even numbers up to 20 is:
"<<sum;
return 0;
}
```

For loop

Syntax

```
for (initialization expr; test
expr; update expr)
{
    // body of the loop
    // statements we want
to execute
}
```



For loop(contd.)

- Initialization Expression: In this expression we have to initialize the loop counter to some value. for example: int i=1;
- Test Expression: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: i <= 10;
- Update Expression: After executing loop body this expression increments/decrements the loop variable by some value. for example: i++;

For loop example

```
//To find the factorial of a number using for loop
#include <iostream>
using namespace std;
int main()
{ int n, i;
long fact=1;
cout<<"Enter the number: ";
cin>>n;
for (i=1; i<=n; ++i)
 fact = fact * i;
cout << "Factorial of " << n << " is " << fact;
return 0;
}
```

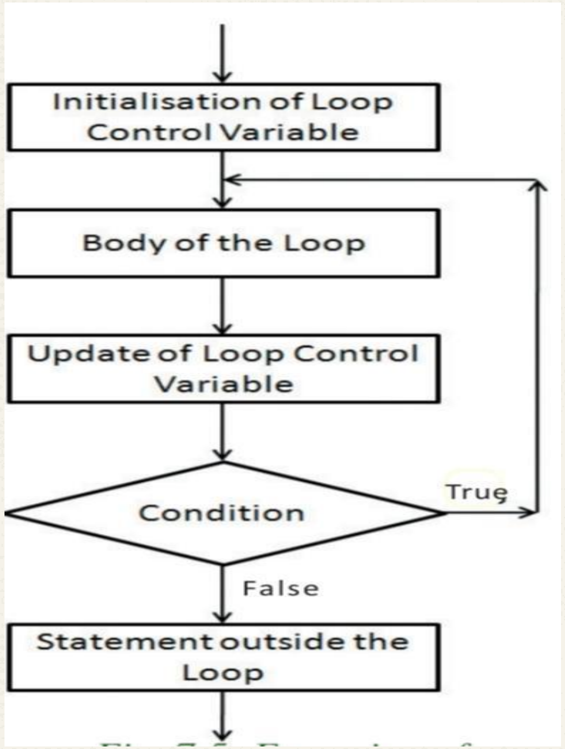
For loop example

```
//To find the average score of n students
#include<iostream>
using namespace std;
int main()
{
int i, sum, score, n;
float avg;
cout << "How many students? ";
cin >> n ;
for( i=1, sum=0; i<=n; ++i)
{
cout << "Enter the score of
student " << i << ": ";
cin >> score;
sum = sum + score;
}
avg = (float)sum / n;
cout << "Class Average: " <<
avg;
return 0;
}
```

Do-while loop

Syntax

```
initialization expression;
do
{
   // statements

   update_expression;
} while (test_expression);
```


Initialisation of Loop Control Variable
Body of the Loop
Update of Loop Control Variable
Condition
True
False
Statement outside the Loop

Do-while example

```
//To find the area of rectangles
#include <iostream>
using namespace std;
int main()
{
float length, breadth, area;
char ch;
do
{
cout << "Enter length and breadth: ";
cin >> length >> breadth;
area = length * breadth;
cout << "Area = " << area;
cout << "Any more rectangle (Y/N)? ";
cin >> ch;
} while (ch == 'Y' || ch == 'y');
return 0;
}
```

Nested loop

Nested loop means a loop statement inside another loop statement.

Nested for loop

Syntax for Nested For loop:

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

   for ( initialization; condition; increment ) {

      // statement of inside loop
   }

   // statement of outer loop
}
```

Nested while loop

Syntax for Nested While loop:

```
while(condition) {

  while(condition) {

    // statement of inside loop
  }

  // statement of outer loop
}
```

Nested Do-while loop

Syntax for Nested Do-While loop:

```
do{

   do{

      // statement of inside loop
   }while(condition);

   // statement of outer loop
}while(condition);
```

Nested loop

Syntax:

```
do{

   while(condition) {

      for ( initialization; condition; increment ) {

         // statement of inside for loop
      }

      // statement of inside while loop
   }

   // statement of outer do-while loop
}while(condition);
```

Nested loop example

```
// To display a triangle of stars
#include<iostream>
using namespace std;
int main()
{ int i, j;
char ch = '*';
for(i=1; i<=5; ++i) //outer loop
{
cout<< "\n" ;
for(j=1; j<=i; ++j) // inner loop
cout<<ch;
}
return 0;
}
```

Program 1

```
// To input two numbers and perform an arithmetic operation based on user’s choice
#include<iostream>
using namespace std;
int main()
 {
char ch;
float n1, n2;
cout<<"Enter two numbers: ";
cin>>n1>>n2;
do
{
cout<<"\nNumber 1: “<<n1<<“\tNumber 2: "<<n2;
cout<<"\n\t\tOperator  Menu";
cout<<"\n\t1. Addition (+)";
cout<<"\n\t2. Subtraction (–)";
cout<<"\n\t3. Multiplication (*)";
cout<<"\n\t4. Division (/)";
cout<<"\n\t5. Exit (E)";
cout<<"\nEnter Option number or operator: ";
cin>>ch;
```

Program 1(contd.)

```
switch(ch)
{
case '1' :
case '+' : cout<<n1<<" + "<<n2<<"
= "<<n1+n2;
 break;
case '2' :
case '-' : cout<<n1<<" - "<<n2<<"
= "<<n1-n2;
 break;
case '3' :
case '*' : cout<<n1<<" * "<<n2<<"
= "<<n1*n2;
 break;
case '4' :
case '/' : cout<<n1<<" / "<<n2<<"
= "<<n1/n2;
 break;
case '5' :
case 'E' :
case 'e' : cout<<"Thank You for
using the program";
 break;
default : cout<<"Invalid
Choice!!";
}
} while (ch!='5' && ch!='E' &&
ch!='e');
return 0;
}
```

Jump statements

- Jump statements are used to manipulate the flow of the program if some conditions are met.
- It is used to terminating or continues the loop inside a program or to stop the execution of a function.
- In C++ there is four jump statement: continue, break, return, and goto.

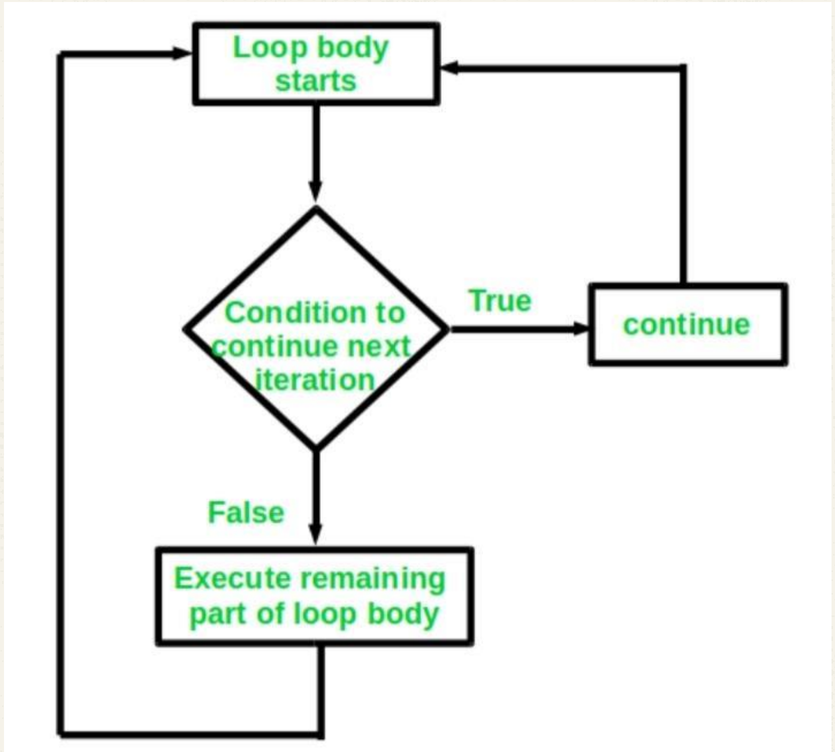
Continue statement

- Continue: It is used to execute other parts of the loop while skipping some parts declared inside the condition, rather than terminating the loop, it continues to execute the next iteration of the same loop
- This statement can be used inside for loop or while or do-while loop.

Continue statement(contd.)

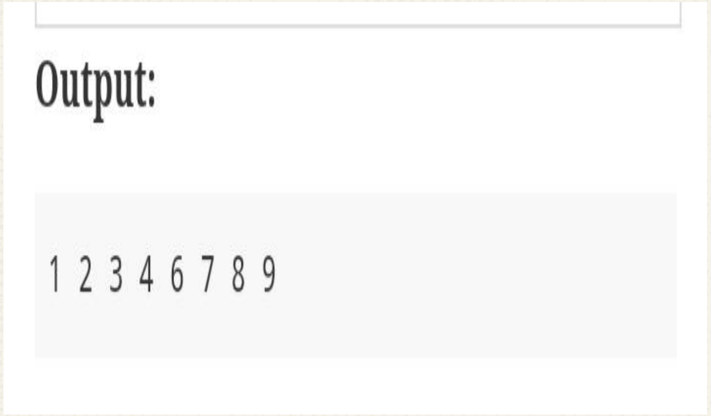
Syntax

continue;


Loop body starts
Condition to continue next iteration
True
continue
False
Execute remaining part of loop body

Continue statement example

```
// C++ program to demonstrate the continue statement
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i < 10; i++) {
    if (i == 5)

        continue;

      cout << i << " ";

   }

   return 0;
}
```


Output:

```
1 2 3 4 6 7 8 9
```

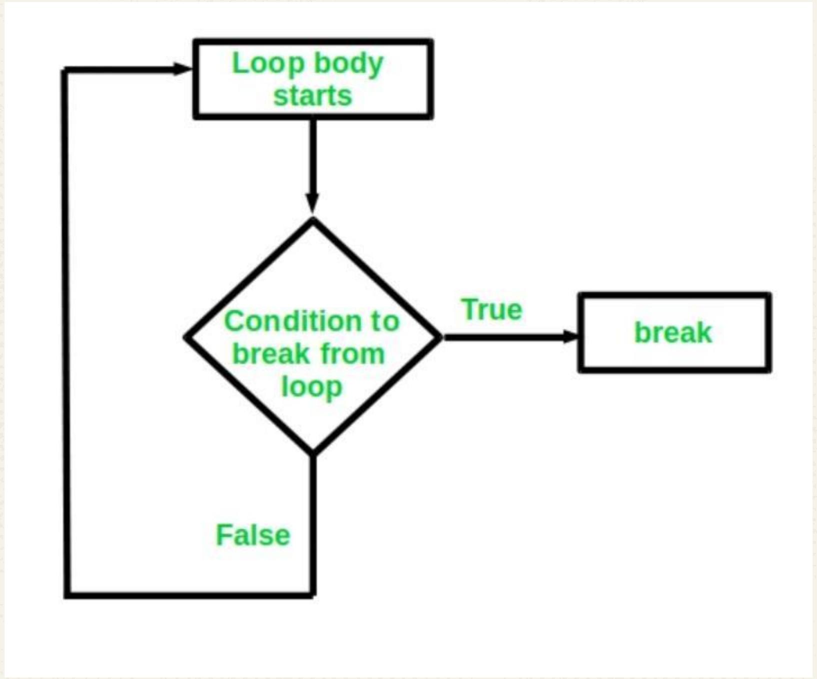
Break statement

- Break: It is used to terminate the whole loop if the condition is met.
- Break statement is used with decision-making statements such as if, if-else, or switch statement which is inside the for loop which can be for loop, while loop, or do-while loop.
- It forces the loop to stop the execution of the further iteration.

Break statement(contd.)

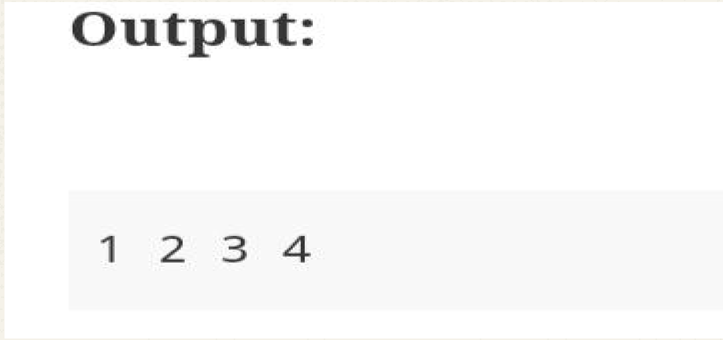
Syntax

```
break;
```



Break statement example

```
// C++ program to demonstrate the break statement
#include <iostream>
using namespace std;
int main()
{

    for (int i = 1; i < 10; i++) {
  // Breaking Condition
        if (i == 5)

            break;

        cout << i << " ";

    }

    return 0;
}
```


Output:

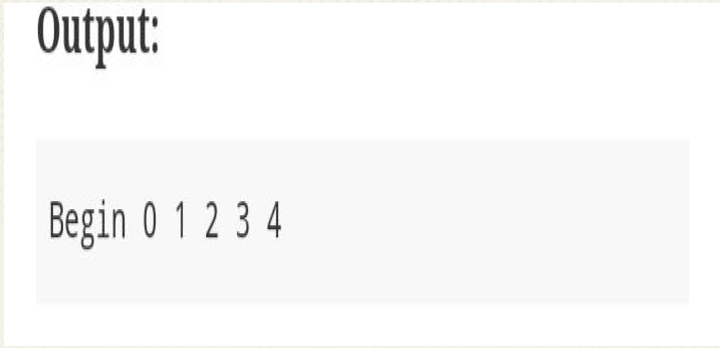
```
1  2  3  4
```

Return statement

- Return: It takes control out of the function itself.
- It is used to terminate the entire function after the execution of the function or after some condition.
- Every function has a return statement with some returning value except the void() function.
- Syntax : return expression;

Return statement example

```
// C++ program to demonstrate the  return statement
#include <iostream>
using namespace std;
int main()
{

    cout << "Begin ";
     for (int i = 0; i < 10; i++) {
    // Termination condition
          if (i == 5)

            return 0;

        cout << i << " ";

    }

    cout << "end";
    return 0;
}
```


Output:

```
Begin 0 1 2 3 4
```

Program explaination

The program starts execution by printing “Begin” then the for loop starts to print the value of, it will print the value of i from 0 to 4 but as soon as i becomes equal to 5 it will terminate the whole function i.e., it will never go to print the “end” statement of the program.

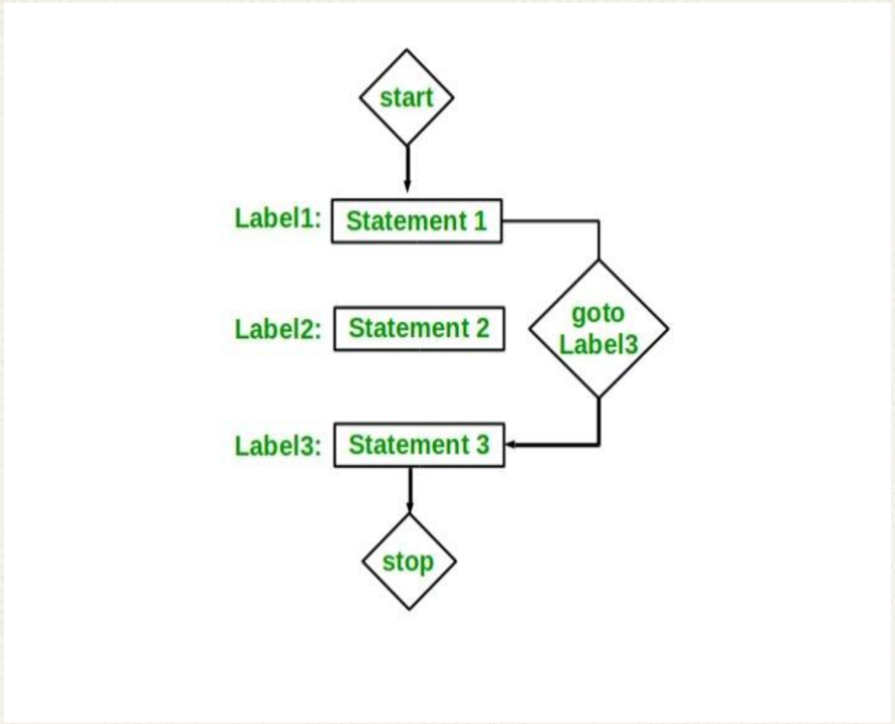
Goto statement

- Goto: This statement is used to jump directly to that part of the program to which it is being called.
- Every goto statement is associated with the label which takes them to part of the program for which they are called.
- The label statements can be written anywhere in the program it is not necessary to use before or after goto statement.

Goto statement(contd.)

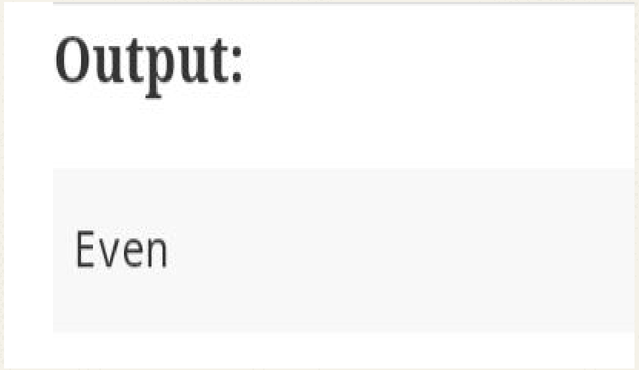
Syntax

```
goto label_name;
.
.
.
label_name:
```



Goto statement example

```
// C++ program to demonstrate the
goto statement
#include <iostream>
using namespace std;
int main()
{
  int n = 4;
  if (n % 2 == 0)
      goto label1;
    else
      goto label2;
label1:
  cout << "Even" << endl;
  return 0;
label2:
    cout << "Odd" << endl;
}
```


Output:

```
Even
```

Program explaination

The program is used to check whether the number is even or odd if the number pressed by the user says it is 4 so the condition is met by the if statement and control go to label1 and label1 prints that the number is even. Here it is not necessary to write a label statement after the goto statement we can write it before goto statement also it will work fine.

Program 3

```
//To check whether the given number is prime or not
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
int i, num;
cout<<"Enter the number: ";
cin>>num;
for(i=2; i<=num/2; ++i)
{
if(num%i == 0)
{
cout<<"Not a Prime Number";
exit(0);
}
}
cout<<"Prime Number";
return 0;
}
```

Program 4

```
// To print n terms of the Fibonacci series
#include <iostream>
using namespace std;
int main()
{
int first=0, second=1, third, n;
cout<<"\nEnter number of terms in the series: ";
cin>>n;
cout<<first<<"\t"<<second;
for(int i=3; i<=n; ++i)
{
third = first + second;
cout<<"\t"<<third;
first = second;
second = third;
}
return 0;
}
```

Program 5

```
// To check whether the given
number is palindrome or not
#include <iostream>
using namespace std;
int main()
{
int num, copy, digit, rev=0;
cout<<"Enter the number: ";
cin>>num;
copy=num;
while(num != 0)
{
digit = num % 10;
rev = (rev * 10)+ digit;
num = num/10;
}
cout<<"The reverse of the number is:
"<<rev;
if (rev == copy)
cout<<"\nThe given number is a
palindrome.";
else
cout<<"\nThe given number is not a
palindrome.";
return 0;
}
```

Program 6

```
// To accept n integers and print the largest among them
#include <iostream>
using namespace std;
int main()
{
int num, big, count;
cout<<"How many Numbers in the list? ";
cin >> count;
cout<<"\nEnter first number: ";
cin >> num;
big = num;
for(int i=2; i<=count; i++)
{
cout<<"\nEnter next number: ";
cin >> num;
if(num > big) big = num;
}
cout<<"\nThe largest number is " << big;
return 0;
}
```

Thank You