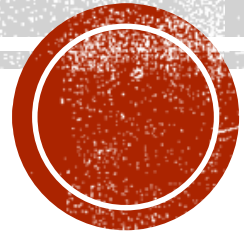
While, do…while, break, continue

Data Science 2020-2021



INTRODUCTION

1. While Loop
2. Do…. While Loop
3. Break statement
4. Continue statement
5. Exercises

1-WHILE

- ***expression*** can be :
  1. Boolean value: result of comparison operators: >, <, >=, <=, ==, !=
  2. Boolean expression: result of operators &&, ||, and !
  3. Numerical value (int, float, …): Different than zero considered as true, equal to Zero considered as false
  4. Numerical expression

- Syntax

while (expression)

instruction a;

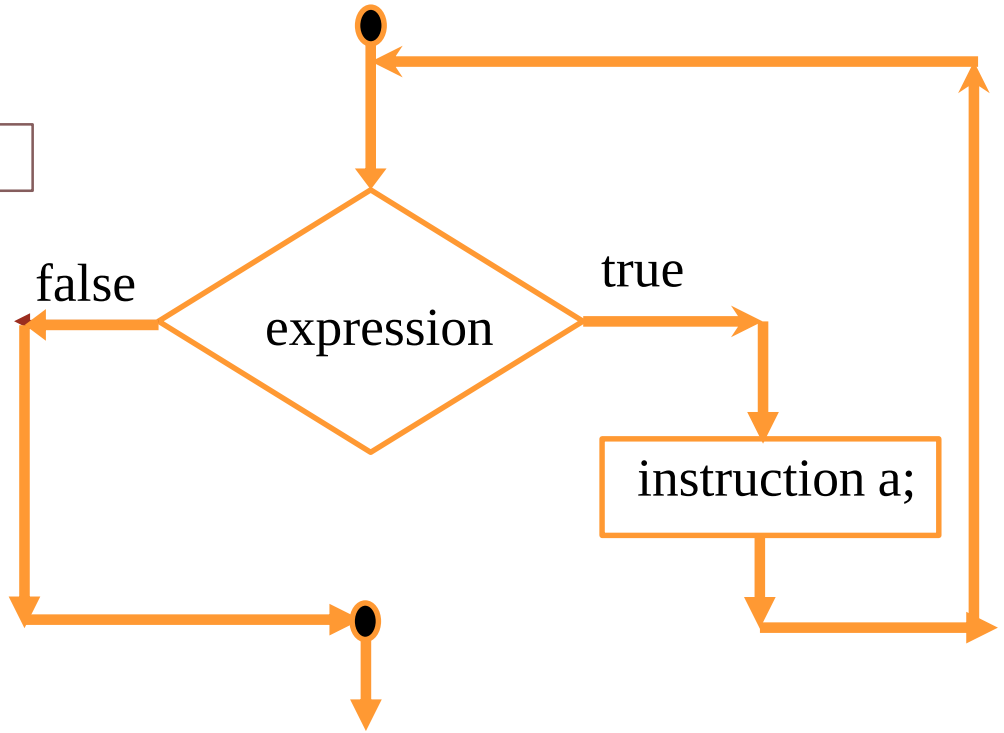
1-WHILE

- Syntax

while (expression)

instruction a;

Instruction controlled by while



- ***expression*** can be :
  1. Boolean value: result of comparison operators: >, <, >=, <=, ==, !=
  2. Boolean expression: result of operators &&, ||, and !
  3. Numerical value: (int, float, …): Different than zero considered as true, equal to Zero is considered as false
  4. Numerical expression

1-WHILE

- Syntax

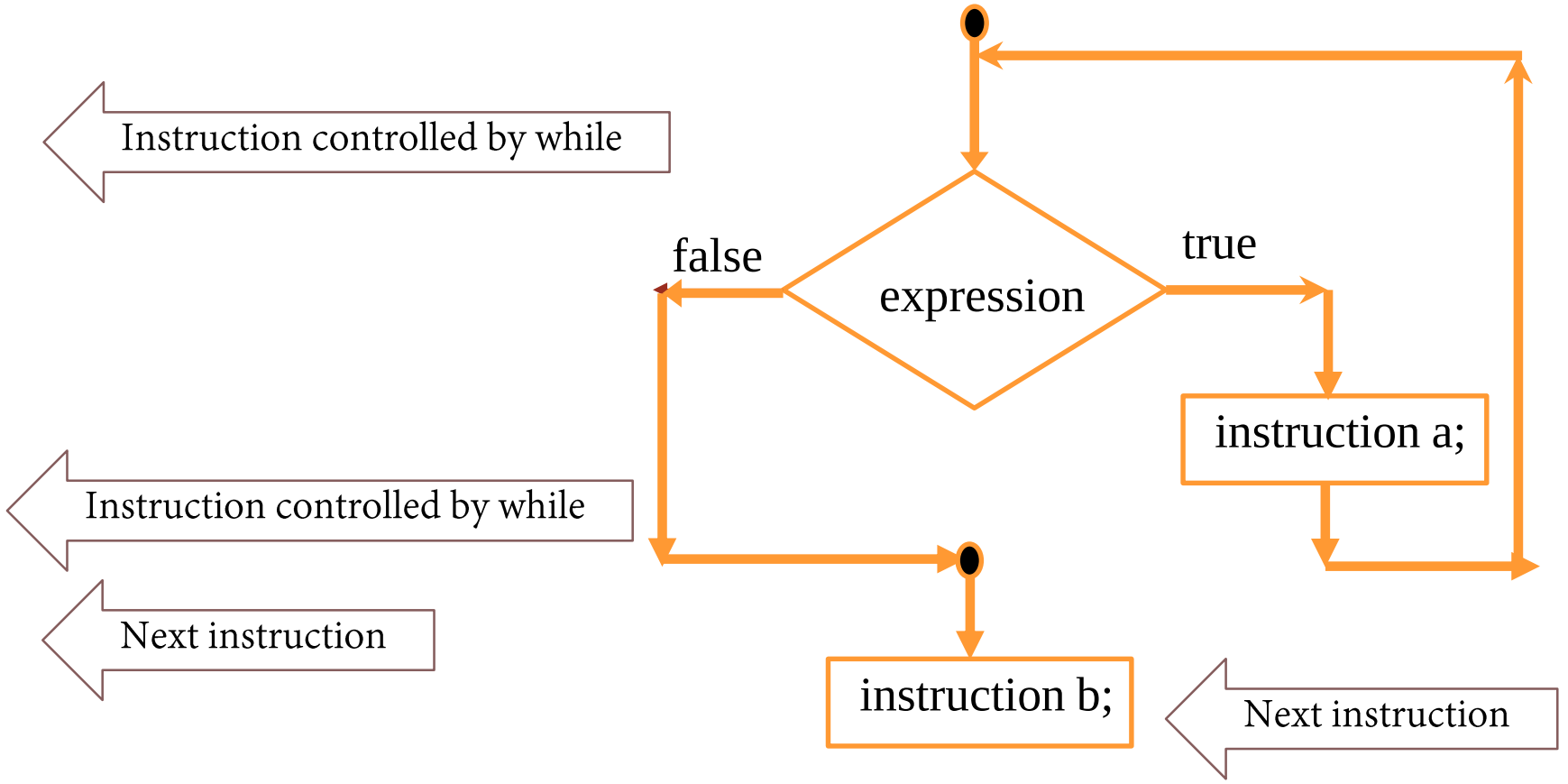
while (expression)

instruction a;

while (expression)

instruction a;

instruction b;



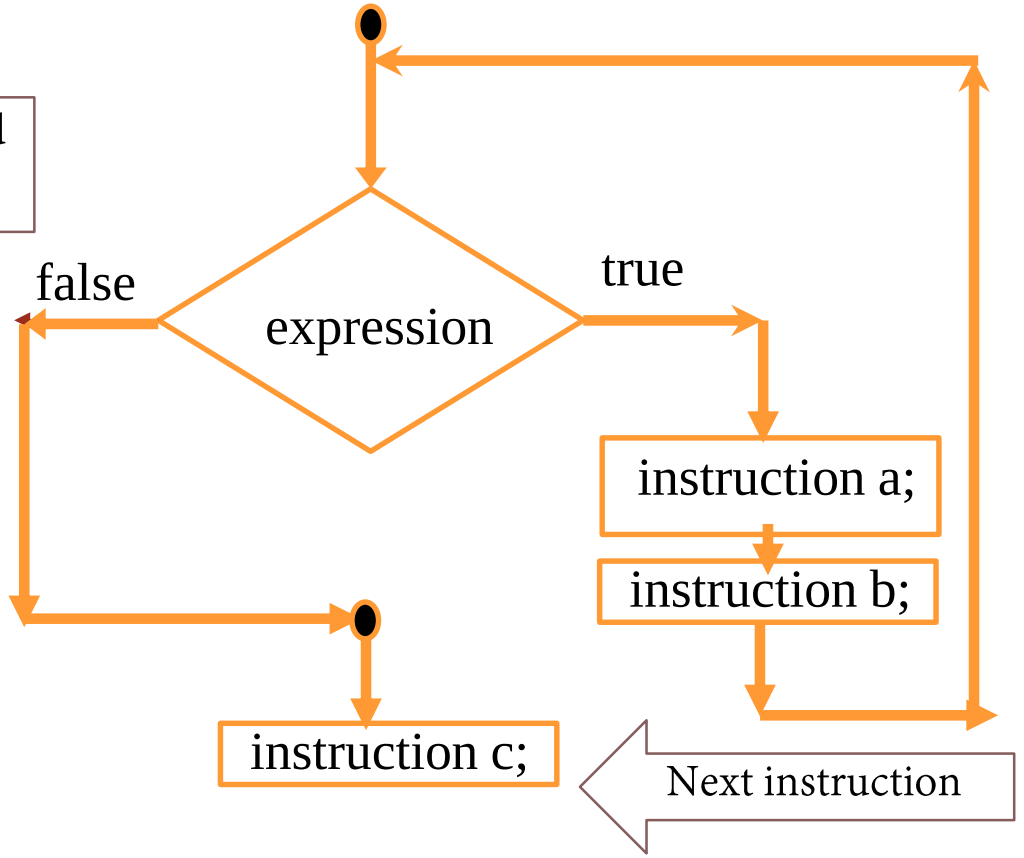
1-WHILE

- Syntax

```
while (expression)
    { instruction a;
      instruction b;
    }
Instruction c;
```

Block of Instructions controlled by while

Next instruction



1-WHILE

Example 1:

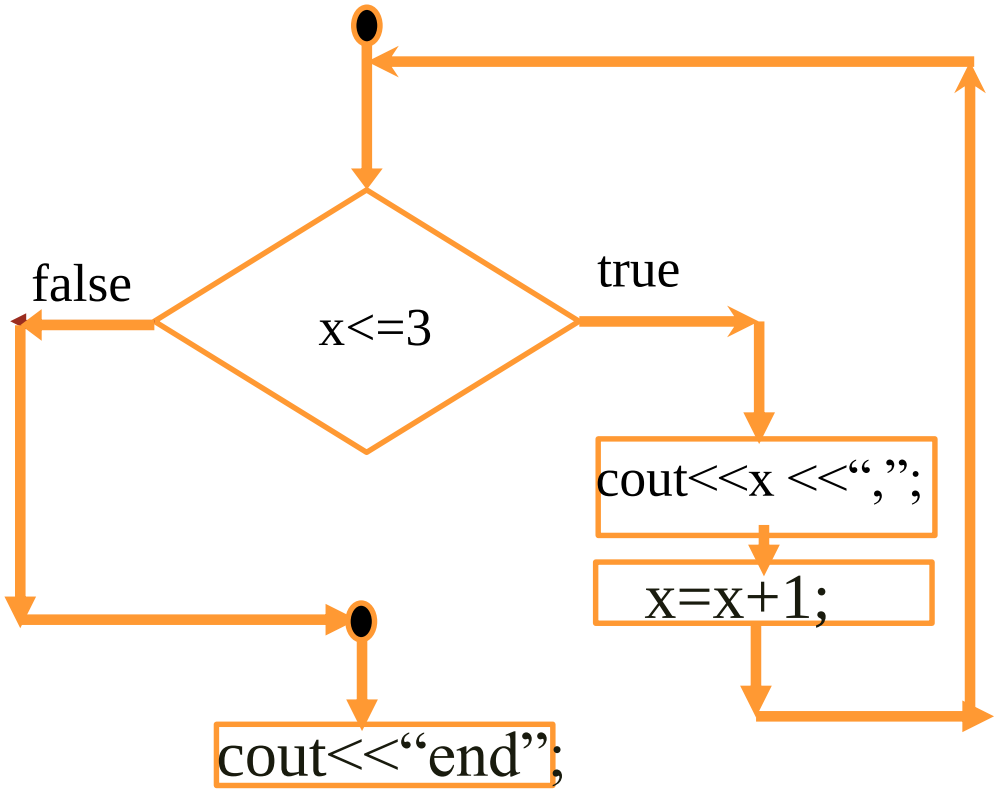
```
while (x<=3)
    {
   cout<<x <<“,”;
    x=x+1;
   }
cout<<“end”;
```

x =4;

end

x =2;

2,3,end



2- DO-WHILE

- ***expression*** can be :
  1. Boolean value: result of comparison operators: >, <, >=, <=, ==, !=
  2. Boolean expression: result of operators &&, ||, and !
  3. Numerical value: (int, float, …): Different than zero considered as true, equal to Zero is considered as false
  4. Numerical expression

- Syntax

do

instruction a;

while (expression);

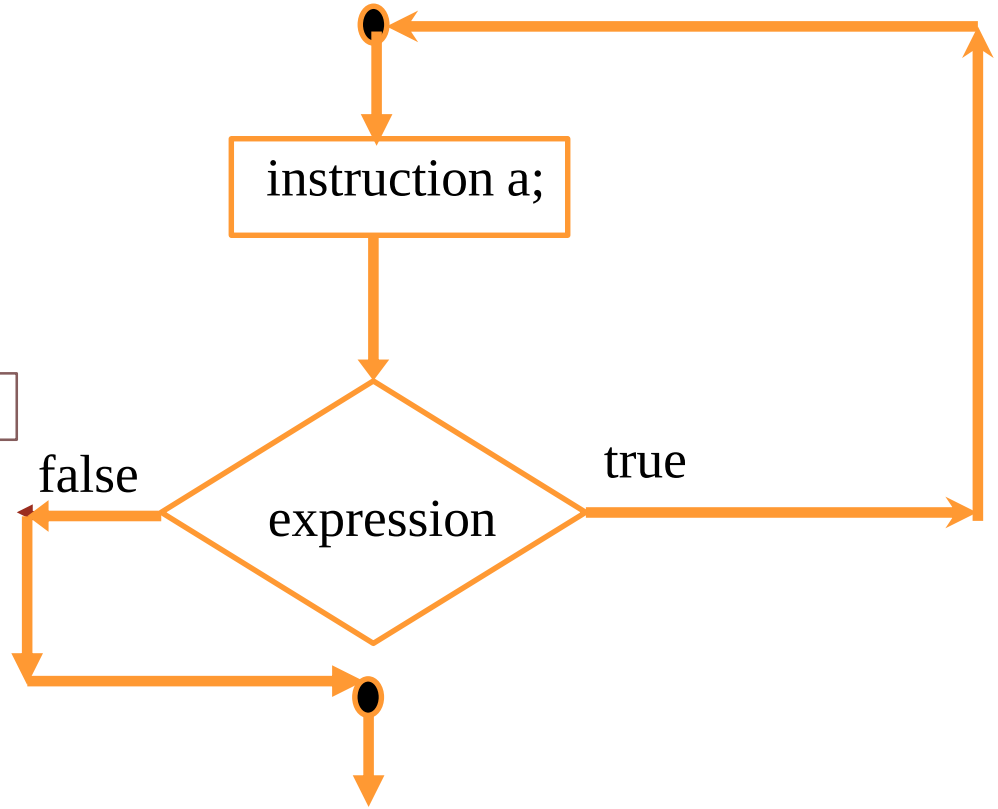
2- DO-WHILE

- Syntax

do

instruction a;

while (expression);



2- DO-WHILE

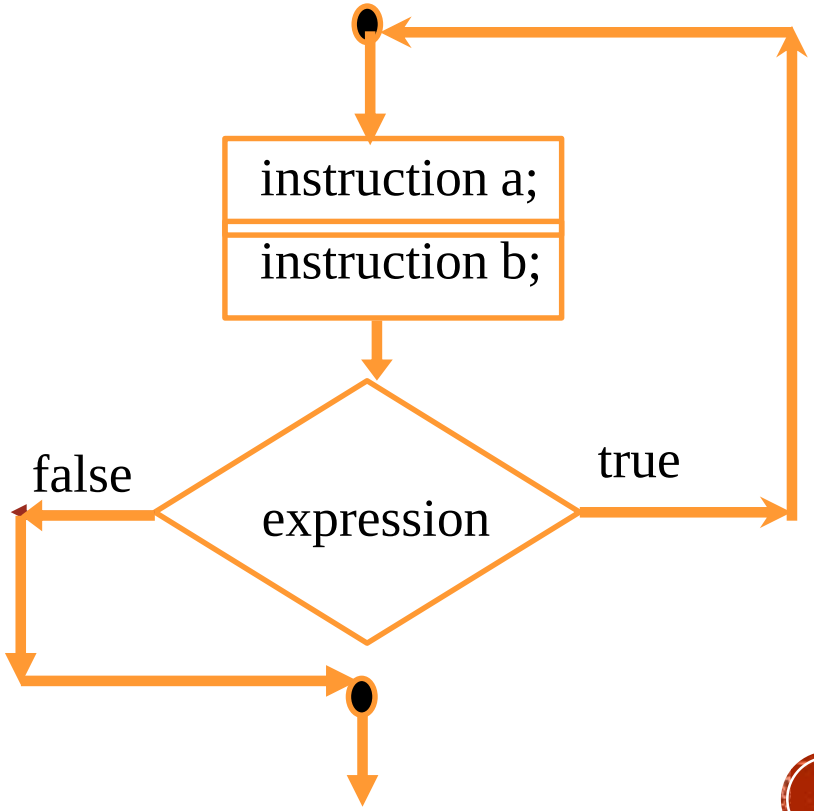
- Syntax

```
do
    instruction a;
while (expression);
```

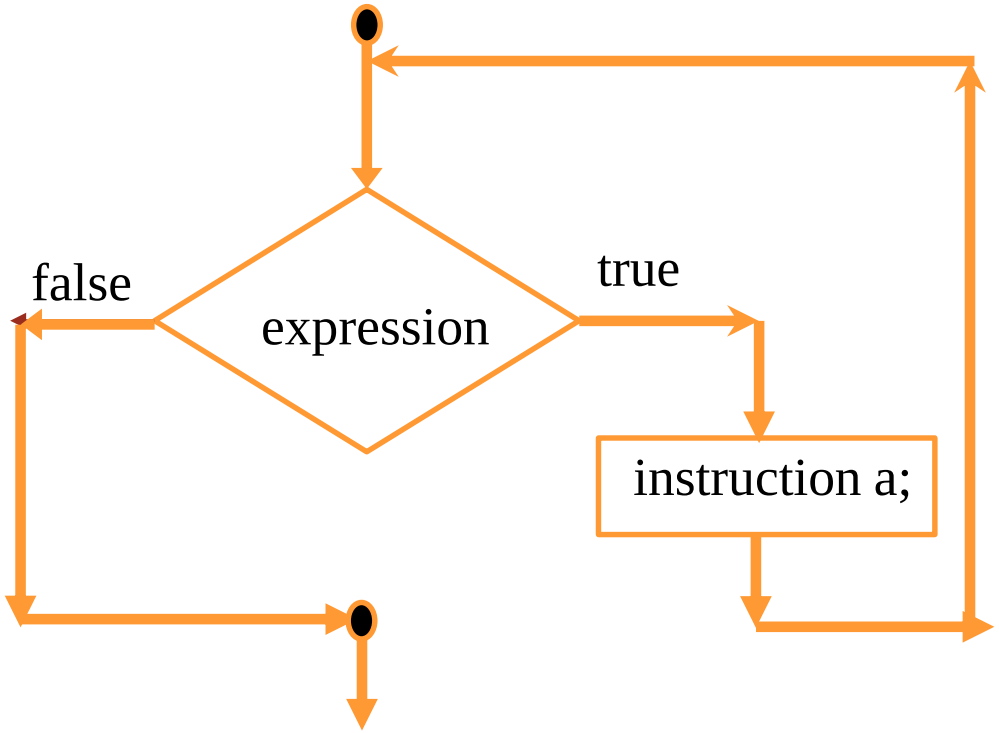
Instruction controlled by do-while

```
do
{
    instruction a;
    instruction b;
}
while (expression);
```

Instructions controlled by do-while

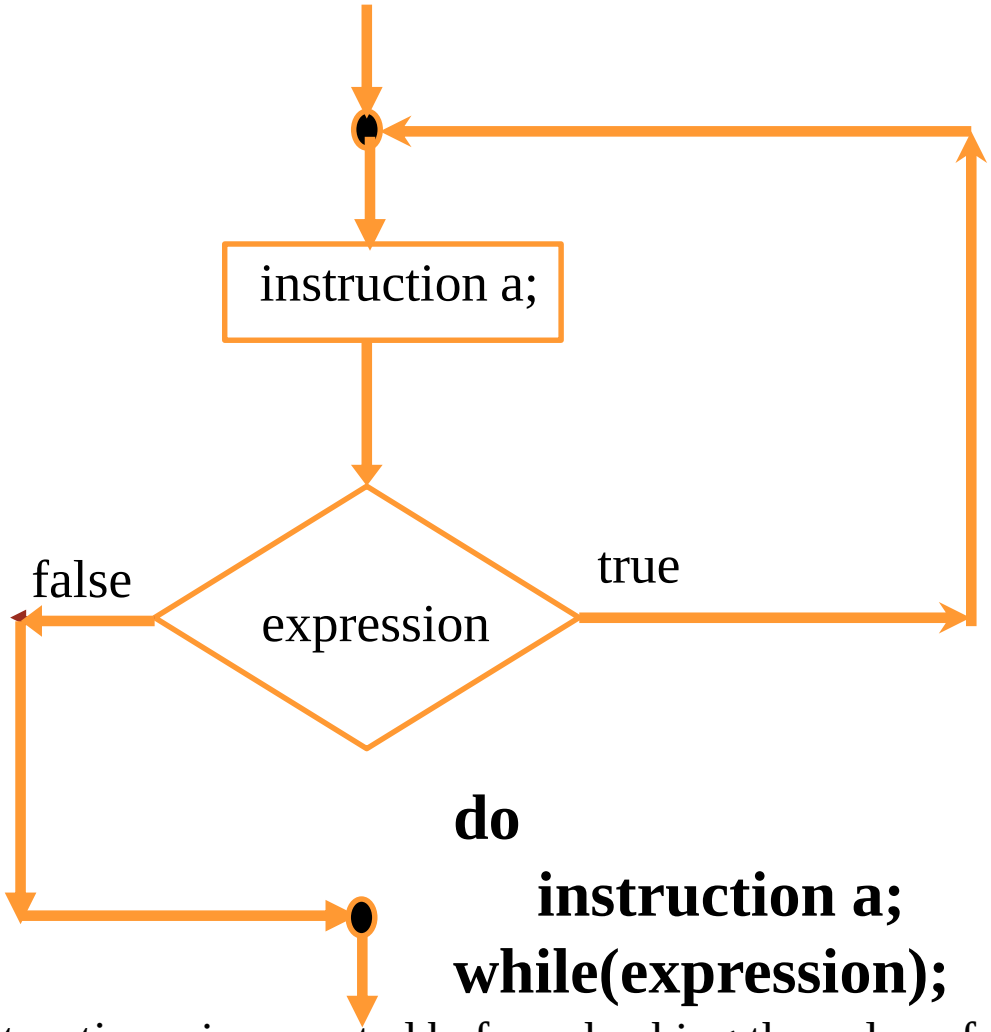


WHILE



while(expression)
instruction a;

- *expression* is checked at the beginning, therefore *expression* shall have an initial value
- if at the beginning the value of *expression* is false, then no execution of *instruction a* of while.



do
instruction a;
while(expression);

- *Instruction a* is executed before checking the value of *expression*
- if at the beginning the value of *expression* is true, then we have execution one time of *instruction a* of do-while.

2- DO-WHILE

Example 1:

```
int x=4;
do {
    cout<<x <<“,”;
     x=x+1;
    } while (x<=3);
cout<<“end”;
```

```
4,end
```

2- DO-WHILE

Example 2: reads a positive number

2- DO-WHILE

Example 2: reads a positive number

```
int x;
do {
cout<<“enter a positive number ”;
cin>>x;
} while (x<=0);
cout<<“x=”<<x;
```

```
Enter a positive number 10
x=10
```

```
Enter a positive number -3
Enter a positive number -20
Enter a positive number 0
Enter a positive number 9
x=9
```

2- DO-WHILE

Example 2: reads a positive number

```
int x;
do {
cout<<“enter a positive number ”;
cin>>x;
} while (x<=0);
cout<<“x=”<<x;
```

In loop while: x shall have an initial value

```
 int x;
while (x<=0){
cout<<“enter a positive number ”;
cin>>x;
}
cout<<“x=”<<x;
```

2- DO-WHILE

Example 2: reads a positive number

```
int x;
do {
cout<<“enter a positive number ”;
cin>>x;
} while (x<=0);
cout<<“x=”<<x;
```

In loop while: x shall have an initial value

First method:

```
 int x;
cout<<“enter a positive number ”;
cin>>x;
while (x<=0){
cout<<“enter a positive number ”;
cin>>x;
}
cout<<“x=”<<x;
```

2- DO-WHILE

Example 2: reads a positive number

```
int x;
do {
cout<<“enter a positive number ”;
cin>>x;
} while (x<=0);
cout<<“x=”<<x;
```

In loop while: x shall have an initial value

Second method:

```
int x=-2;
while (x<=0){
cout<<“enter a positive
number ”;
cin>>x;
}
cout<<“x=”<<x;
```

INTRODUCTION

1. While Loop
2. Do…. While Loop
3. **Break statement**
4. **Continue statement**
5. **Exercises**

3- Break statement

- The break statement is used to **terminate loops**. It can be used within a for, while, do -while, or switch statement.
- Syntax:

  break;

Example 1:

```
int i;
for(i=1;i<=10;i++)
{
cout<<i<<“,”;
break;
}
cout<<“end”;
```

```
1,end
```

3- Break statement

- The break statement is used to **terminate loops**. It can be used within a for, while, do -while, or switch statement.
- Syntax:

  break;

Example 1:

```
int i;
for(i=1;i<=10;i++)
{
cout<<i<<",";
break;
}
cout<<"end";
```

```
1,end
```

Example 2:

```
int i;
for(i=1;i<=10;i++)
{
break;
cout<<i<<",";
}
cout<<"end";
```

```
end
```

3- Break statement

- The break statement is used to **terminate loops**. It can be used within a for, while, do -while, or switch statement.
- Syntax:

  break;

Example 1:

```
int i;
for(i=1;i<=10;i++)
{
cout<<i<<“,”;
break;
}
cout<<“end”;
```

```
1,end
```

Remark:

Break shall be controlled by a structure of selection (if, if-else), inside the loop.

3- BREAK STATEMENT

- The break statement is used to terminate loops. It can be used within a for, while, do -while, or switch statement.
- Syntax:

```
break;
```

Example 3:

```
int i=1;
while(1)
{
cout<<i<<",";
if(i==3)
   break;
i=i+1;
}
```

```
1,2,3,end
```

Remark:

Break shall be controlled by a structure of selection (if, if-else), inside the loop.

4- CONTINUE STATEMENT

- Syntax :

  continue;

- The *continue* statement is used inside loops: while, a do - while and for statement.
- The *continue* statement is used to bypass the remainder of the current iteration of a loop. The loop does not terminate when a *continue* statement is encountered. Rather, the remaining loop statements/iterations are skipped and the computation proceeds directly to the next iteration of the loop.
- The statement *continue* shall be controlled using the structure of selection (if, if-else).

4- CONTINUE STATEMENT

Example 1:

```
int i,S=0;
for(i=1;i<=3;i++)
{
cout<<“S=”<<S<<“,”;
continue;
S=S+i;
}
cout<<“end”;
```

S=0,S=0,S=0,end

4- CONTINUE STATEMENT

Example 1:

```
int i,S;
for(i=1,S=0;i<=3;i++)
{
cout<<“S=”<<S<<“,”;
continue;
S=S+i;
}
cout<<“end”;
```

S=0,S=0,S=0,end

The statement *continue* shall be controlled using the structure of selection (if, if-else).

4- CONTINUE STATEMENT

Example 2: Program that prints the sum all odd integer less than 10

```
int i,S=0;
for(i=1;i<=10;i++)
{
if(i%2==0)
   continue;

cout<<i<<“+”;
S=S+i;
}
cout<<“\nS=”<<S;
```

```
1+3+5+7+9+
S=25
```

EXERCISE 1.

Write a program which reads a positive integer value N, calculates and shows the result of the expression: 1 + 4 + 7 + 10 +... + N

Write a program which reads a positive integer value N, calculates and shows the result of the expression: 1 + 4 + 7 + 10 +... + N

EXERCISE 1-SOLUTION

```
#include<iostream>
using namespace std;
int main()
{
    int N,i,S;
    do
    {
        cout<<"enter N >0: ";
        cin>>N;
    }while(N<=0);
    for(i=1,S=0;i<=N;i=i+3)
        S=S+i;
    cout<<"S= "<<S;
return 0;
}
```

```
#include<iostream>
using namespace std;
int main()
{
    int N,i,S;
    do
    {
        cout<<"enter N >0: ";
        cin>>N;
    }while(N<=0);
    i=1;
    S=0;
    while(i<=N)
    {
        S=S+i;
        i=i+3;
    }
    cout<<"S= "<<S;
return 0;
}
```

```
enter N >0: -6
enter N >0: -15
enter N >0: 4
S= 5
```

EXERCISE 2

Write a program which reads a sequence of real numbers. The program stops when the user enters a negative number. the program shows the sum of these numbers.

Write a program which reads a sequence of real numbers. The program stops when the user enters a negative number. The program shows the sum of these numbers.

EXERCISE 2-SOLUTION

```
#include<iostream>
using namespace std;
int main()
{
    float X,S=0;
    while(true)
    {
        cout<<"enter X ";
        cin>>X;
        if (X<0)
            break;
        S=S+X;
    }
    cout<<"S= "<<S;
return 0;
}
```

```
enter X 5
enter X 3
enter X 4
enter X -1
S= 12
--------------------------------
Process exited after 16.86 seconds with return value 0
Press any key to continue . . .
```

EXERCISE 2- REMARK

Write a program which reads a sequence of real numbers. The program stops when the user enters a negative number. the program shows the sum of these numbers.

```
#include<iostream>
using namespace std;
int main()
{
    float X,S=0;
    do
    {
        cout<<"enter X ";
        cin>>X;
        S=S+X;
    }while(X>=0);
    cout<<"S= "<<S;
return 0;
}
```

Wrong Answer

```
enter X 5
enter X 2
enter X -3
S= 4
----------
```

```
#include<iostream>
using namespace std;
int main()
{
    float X,S=0;
    do
    {
        cout<<"enter X ";
        cin>>X;
        if(X>0)
            S=S+X;
    }while(X>=0);
    cout<<"S= "<<S;
return 0;
}
```

Correct Answer

```
enter X 5
enter X 2
enter X -3
S= 7
```

EXERCISE 5

Write a program which indicates if a positive number N, filled by the user, is prime or not (N is a prime number if its only divisors are 1 and N).

EXERCISE 3 - SOLUTION

Write a program which indicates if a positive number N, filled by the user, is prime or not (N is a prime number if its only divisors are 1 and N).

```
#include<iostream>
using namespace std;
int main()
{
int N,i,nb;
do
{
    cout<<"enter a positive number";
    cin>>N;
}while(N<=0);
for (i=1,nb=0;i<=N;i++)
    if(N%i==0)
        nb=nb+1;
if(nb>2)
    cout<<N<<" is not a prime number";
else
    cout<<N<<" is a prime number";
return 0;

}
```

```
enter a positive number-6
enter a positive number8
8 is not a prime number
--------------------------------
Process exited after 11.84 seconds with return value 0
Press any key to continue . . .
```

EXERCISE 3 -SOLUTION

Write a program which indicates if a positive number N, filled by the user, is prime or not (N is a prime number if its only divisors are 1 and N).

```
int main()
{
int N,i,p=0;
do
{
    cout<<"enter a positive number";
    cin>>N;
}while(N<=0);
for (i=2;i<N;i++)
    if(N%i==0)
    {
        p=1;
        break;
    }
if(p==1)
    cout<<N<<" is not a prime number";
else
    cout<<N<<" is a prime number";
return 0;
}
```

```
int main()
{
int N,i;
do
{
    cout<<"enter a positive number";
    cin>>N;
}while(N<=0);
for (i=2;i<N;i++)
    if(N%i==0)
        break;
if(i<N)
    cout<<N<<" is not a prime number";
else
    cout<<N<<" is a prime number";
return 0;
}
```