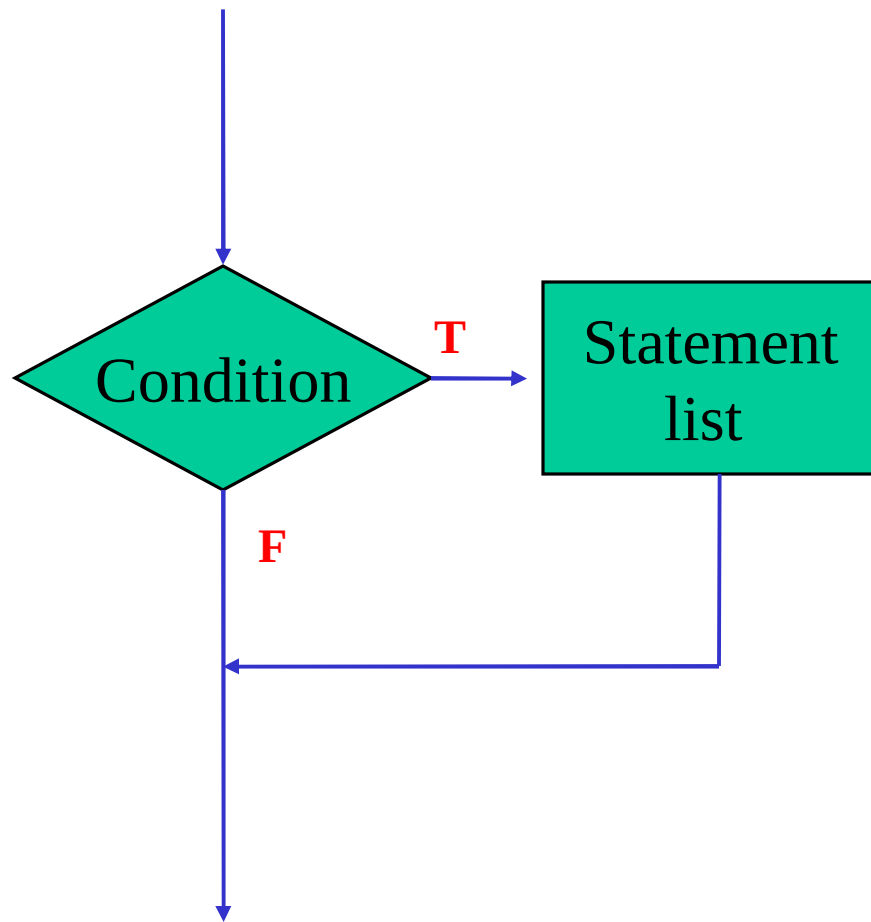
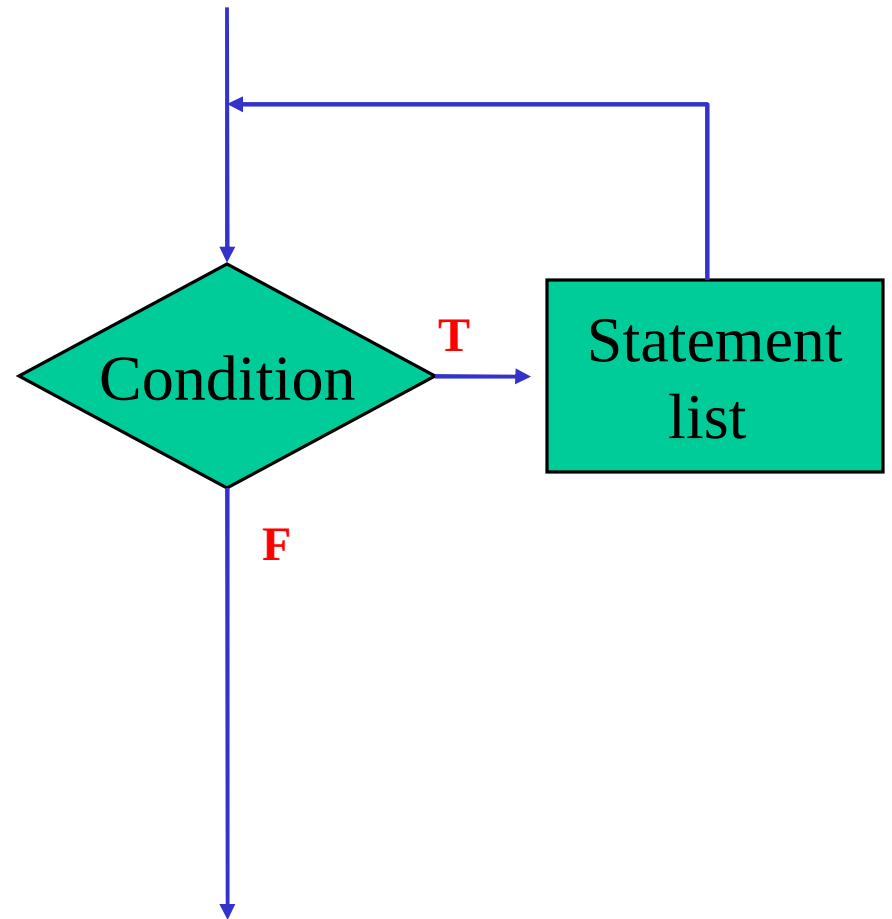


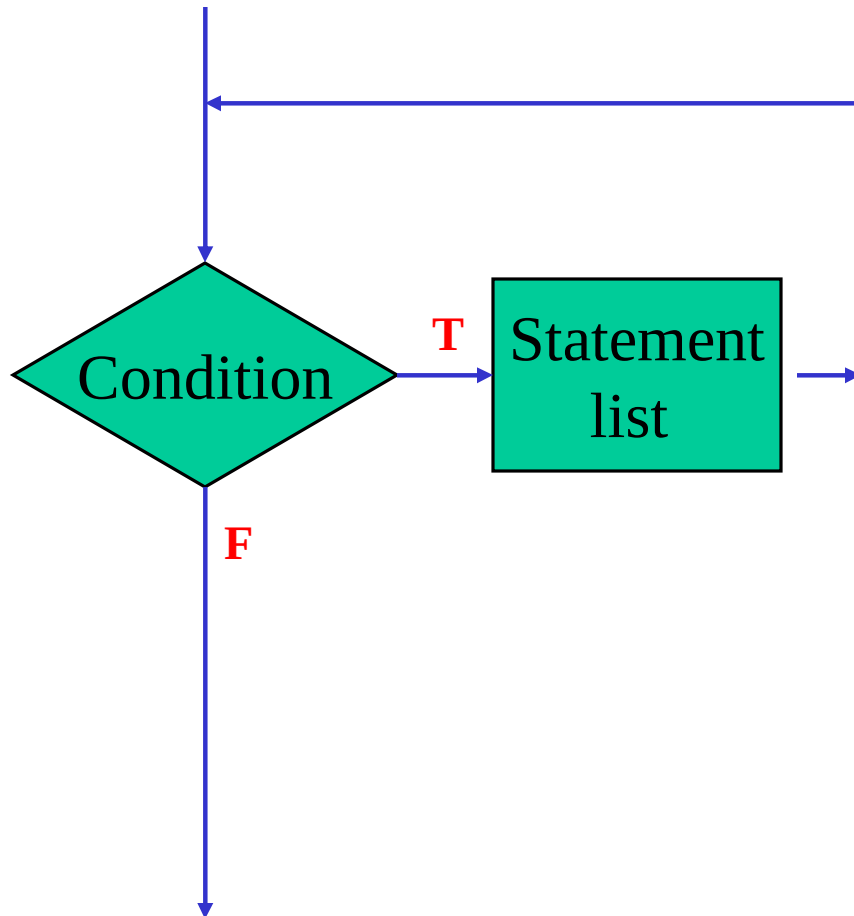
Branching



Loops



while



```
while (Condition)
{
    Statement list
}
```

Example 1: **while**

```
string ans = "n";
```

(ans != "Y" || ans != "y")

```
while (ans != "Y" && ans != "y")
```

```
{  
    cout << "Would you marry me?";  
    cin >> ans;  
}
```

```
cout << "Great!!";
```

**Can I put ;
here?**

No!!

Should I put ; here?

Up to you!!

Example 2: **while**

Will there be any problem?

What if one inputs -1?

```
int no_times;
```

```
cout << "How many times do you want to say?";  
cin  >> no_times;
```

```
while (no_times != 0)  
{  
    cout << "Hello!" << endl;  
    no_times--;  
}
```

```
while (no_times > 0)
```

```
cout << "End!!" << endl;
```

Example 3: **while**

```
int a, b, sum;
```

```
cout << "This program will return the ";  
cout << "summation of integers from a to b.\n\n";  
cout << "Input two integers a and b:";  
cin  >> a >> b;
```

```
sum = 0;
```

```
while (a <= b)  
{  
    sum += a;  
    a++;  
}
```

Don't forget to set
sum = 0;

sum = sum + a;

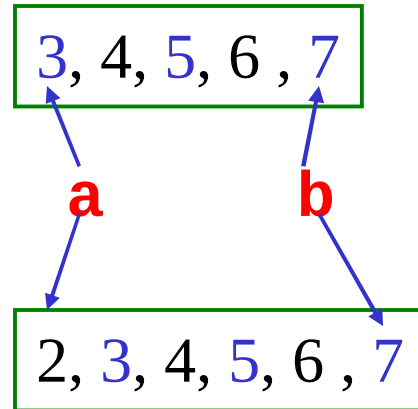
```
cout << "The sum is " << sum << endl;
```

Example 4: **while**

```
int a,b,sum=0;
```

```
cout << "This program will return the sum ";  
cout << "of odd numbers between a and b.\n\n";  
cout << "Input two integers a and b:";  
cin  >> a >> b;
```

```
while (a <= b)  
{  
    if (a % 2)  
        sum += a;  
    a++;  
}
```



```
if (a % 2 == 0)  
    a++;  
while (a <= b)  
{  
    sum += a;  
    a += 2;  
}
```

```
cout << "The answer is " << sum << endl;
```

Example 5: **while** $3N+1$ problem

```
long n,i=0;

cout << "Input an integer:";
cin  >> n;

while (n > 1)
{
    if (n % 2)
        n = 3*n+1;
    else
        n /= 2;
    cout << ++i << ":" << n << endl;
}

cout << "Done!!" << endl;
```

Input an integer:7

1:22
2:11
3:34
4:17
5:52
6:26
7:13
8:40
9:20
10:10
11:5
12:16
13:8
14:4
15:2
16:1
Done!!
Press a

Input an integer:11

1:34
2:17
3:52
4:26
5:13
6:40
7:20
8:10
9:5
10:16
11:8
12:4
13:2
14:1
Done!!
Press a

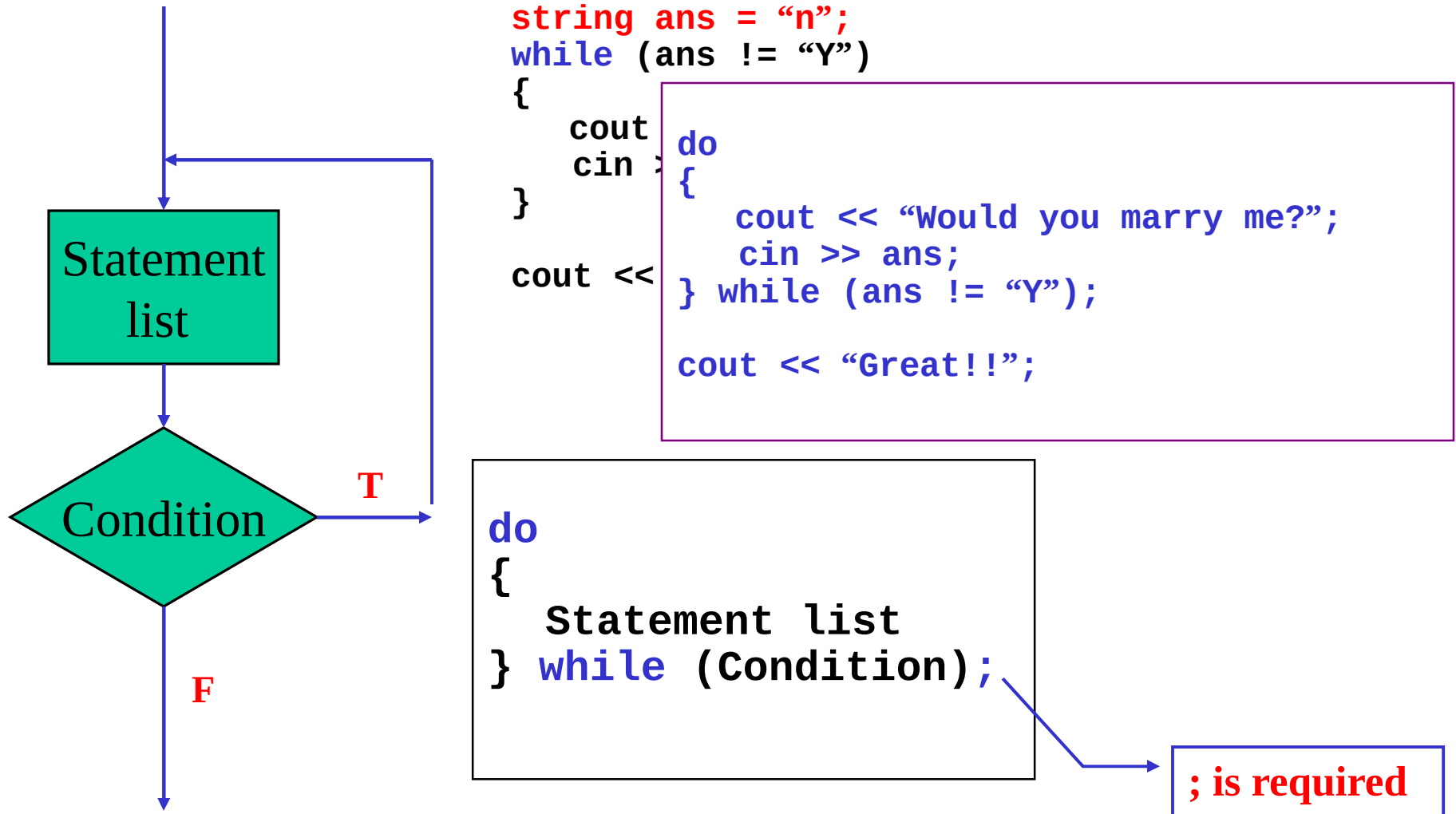
Input an integer:3759

1:11278
2:5639
3:16918
4:8459
5:25378
6:12689
7:38068
.....
.....
83:16
84:8
85:4
86:2
87:1
Done!!
Press any key to continue

Will we always get out of a loop?

That is the question!! No one knows!

do-while



Example 1: **do-while**

```
int i;
```

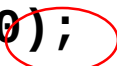
```
....
```

```
do
```

```
{
```

```
    cout << "Please input a number between "   
        << "10 and 20:";
```

```
    cin >> i;
```

```
} while (i < 10 || i > 20); 
```

```
.....
```

Primality Test

A **prime** number is a positive integer that cannot be **factorized**, i.e., no numbers other than 1 and itself can divide it.

is 893 a prime?

We can write a loop
to test these.

Is $(893 \% 2 == 0)$?

Is $(893 \% 3 == 0)$?

Is $(893 \% 4 == 0)$?

Is $(893 \% 5 == 0)$?

Is $(893 \% 6 == 0)$?

.

.

.

Is $(893 \% 892 == 0)$?

Example: Silly Primality Test

```
long int p,r,i=2;

cout << "Input an positive integer:";
cin  >> p;

cout << p << " is ";

do
{
    r = p % i;
    i++;
} while (i < p && r != 0);

if (i == p)
    cout << "a prime number.";
else
{
    cout << "not a prime number.";
    cout << "The least factor is " << --i;
}
```

Break in a loop

The **break** statement in a loop will force the program to **jump out** of the loop immediately.

```
do
{
    r = p % i;
    if (r == 0) break;
    i++;
} while (i < p);
```

```
do {
    cout << "Would you marry me?";
    cin >> ans;

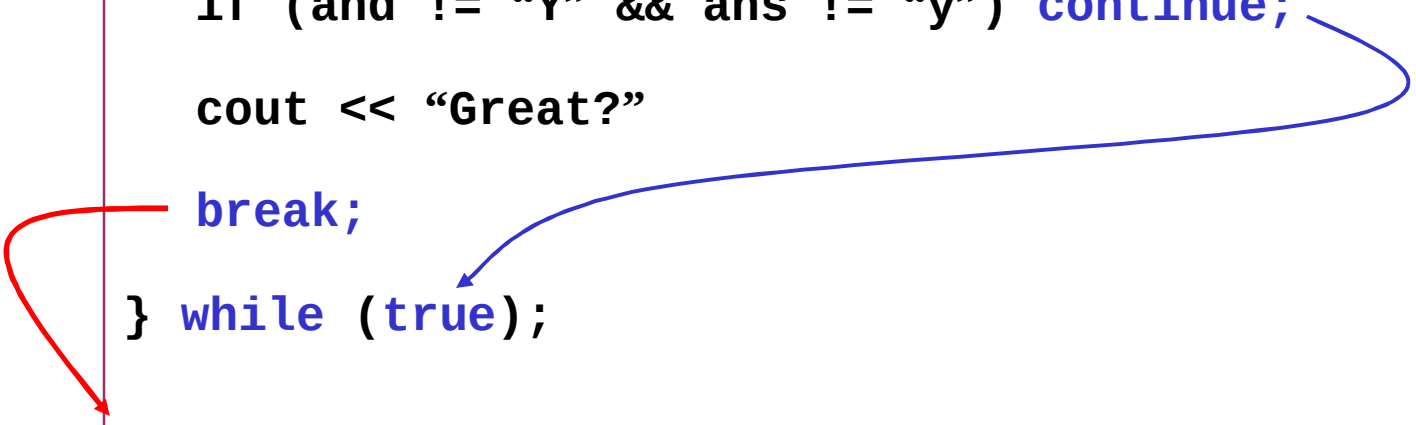
    if (ans == "F" || ans == "f") break;

    cout << "Really?"
    cin >> ans;
} while (ans != "Y" && ans != "y");
cout << "Great!!";
```

Continue in a loop

The **continue** statement in a loop will force the program to **check** the loop condition immediately.

```
do {  
    cout << "Would you marry me?";  
    cin >> ans;  
  
    if (and != "Y" && ans != "y") continue;  
  
    cout << "Great?"  
  
    break;  
} while (true);  
....
```



Euclid Algorithm

```
#include <iostream>
using namespace std;

void main()
{
    int a,b;

    cout << "Input two positive integers:";
    cin >> a >> b;

    int r = a % b;

    while (r)
    {
        a = b;
        b = r;
        r = a % b;
    }
    cout << "The GCD is " << b << ".\n";
}
```

Primality Test with while loop

```
#include <iostream>
using namespace std;
void main()
{
    bool is_prime=true;
    int d=2,p;
    cout << "Input a positive integers:";
    cin >> p;

    while (d <= p/2)
    {
        if (p % d == 0)
        {
            is_prime=false;
            break;
        }
        d++;
    }

    if (is_prime) ...
}
```

Can we change to
`while (d < p/2) ?`

Primality Test with do-while loop

```
#include <iostream>
using namespace std;
void main()
{
    bool is_prime=true;
    int d=2,p;
    cout << "Input a positive integers:";
    cin >> p;

    do {
        if (p % d == 0)
        {
            is_prime=false;
            break;
        }
        d++;
    } while (d <= p/2);

    if (is_prime) ...
}
```



Can we change to
while (d < p/2) ?

Primality Test with do-while loop (a bit better)

```
void main()
{
    bool is_prime=true;
    int d=3,p;
    cout << "Input a positive integers:";
    cin >> p;

    do {
        if ((p % d == 0) || (p % 2 == 0))
        {
            is_prime=false;
            break;
        }
        d += 2;
    } while (d < p/2);

    if (is_prime) ...
}
```



Primality Test with do-while loop (yet another improvement)

```
void main()
{
    bool is_prime=true;
    int d=3,p;
    cout << "Input a positive integers:";
    cin >> p;
    if (p % 2 == 0)
        is_prime=false;
    else do
    {
        if (p % d == 0)
        {
            is_prime=false;
            break;
        }
        d += 2;
    } while (d < p/2);

    if (is_prime) ...
}
```

What if I forget
else here?

Definite Loop

- In programming a **definite** loop is more welcome.
I.e., **number of iterations is known** before the loop begins, at least the upper bound is known.

I.e., **repeat the loop 100 times.**

Precisely speaking, there is no definite loop in C++

The general format for a **for** loop

```
      1           2           3  
for (Initialization_action; Condition; Condition_update)  
{  
    statement_list;  
}
```

Factorial of n is $n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$

```
int n, f=1;  
cin >> n;  
for (i=2; i<=n; i++)  
{  
    f *= i;  
}  
cout << "The factorial of " << n << " is " << f << ".";
```

Compare: **for** and **while**

1	2	3
<pre>for (<i>Initialization_action</i>; <i>Condition</i>; <i>Condition_update</i>)</pre>		
<pre>{</pre>		
<pre> statement_list;</pre>		
<pre>}</pre>		

```
int n, f=1;
```

```
cin >> n;
```

```
for (i=2; i<=n; i++)  
{  
    f *= i;  
}
```

```
i=2;  
while (i<=n)  
{  
    f *= i;  
    i++;  
}
```

```
cout << "The factorial of " << n << " is " << f << ".";
```

For Loop is not really a **definite** loop

```
int n,i;  
n = 100;  
  
for (i=1; i <= n; i++)  
{  
    statement_list;  
}
```

V.S.

```
int n,i;  
n = 100;  
i = 1;  
  
while (i <= n)  
{  
    statement_list;  
    i++;  
}
```

Break in a for loop

The **break** statement in a **for loop** will force the program to **jump out** of the **for loop** immediately.

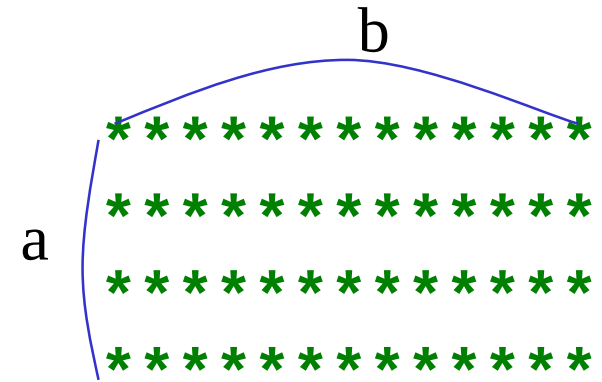
The **continue** statement in a **for loop** will force the program to **update** the loop condition and then **check** the condition immediately.

```
for (Initialization_action; Condition; Condition_update)  
{  
    statement_list;  
}
```

Nested loops (loop in loop)

```
cin >> a >> b;
```

```
for (int i = 0; i < a; i++)  
{  
    for (int j=0; j<b; j++)  
    {  
        cout << "*";  
    }  
    cout << endl;  
}
```

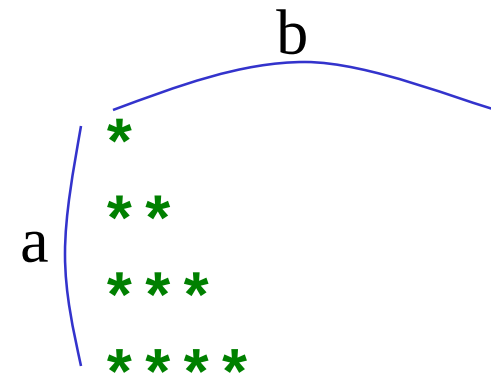


Nested loops (2)

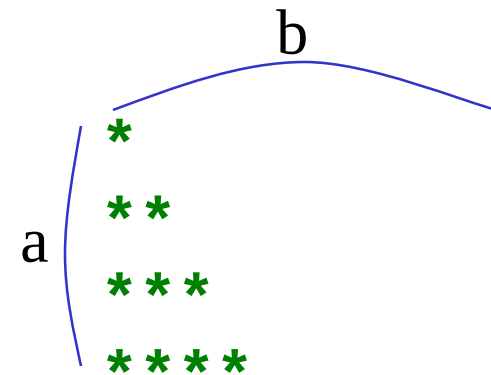
```
int a,b;

cin >> a >> b;

for (int i = 0; i < a; i++)
{
    for (int j=0; j<b; j++)
    {
        if (j > i)
            break;
        cout << "*";
    }
    cout << endl;
}
```



Nested loops (3)



```
int a,b;
```

```
cin >> a >> b;
```

```
for (int i = 0; i < a; i++)
```

```
{    for (int j=0; j<b && j <= i; j++)
```

```
    {
```

```
        cout << "*";
```

```
    }
```

```
    cout << endl;
```

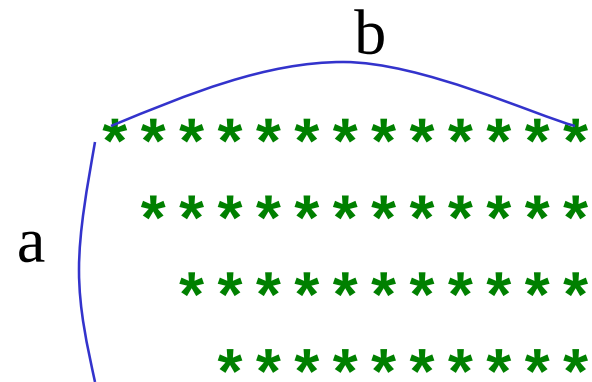
```
}
```

```
if (j > i) break;
```

Nested loops (4)

```
int a,b;  
  
cin >> a >> b;  
  
for (int i = 0; i < a; i++)  
{  
    for (int j=0; j<b; j++)  
    {  
        if (j < i)  
            cout << " ";  
        else  
            cout << "*";  
    }  
    cout << endl;  
}
```

=



Nested loops (5)

```
int a,i,j;  
cin >> a;
```

```
for (i = 0; i < a; i++) {  
    for (j=0; j<a; j++) {  
        if (j < a-i)  
            cout << " ";  
        else cout << "*";  
    }  
}
```

```
for (j=0; j<a; j++) {  
    if (j > i) break;  
    cout << "*";  
}
```

```
    cout << endl;  
}
```

```
*  
 **  
 ***  
 ****  
 *****  
 *****
```

Where is my penny?

```
double s,t,r;
int i;

cout << "Input two real numbers for paid and cost: ";
cin >> s >> t;
cout << "s = " << s << ", t = " << t << endl;
r = s-t;
cout << "r = s-t = " << r << endl;
cout << "r*100 = " << r*100 << endl << endl;
i = (s-t)*100;
cout << "i = (s-t)*100 = " << i << endl;
i = r*100;
cout << "i = r*100 = " << i << endl;
i = (s*100)-(t*100);
cout << "i = (s*100)-(t*100) = " << i << endl;
```

Input two real numbers for paid and cost: 20 3.99

s = 20, t = 3.99

r = s-t = 16.01

r*100 = 1601

i = (s-t)*100 = 1600

i = r*100 = 1600

i = (s*100)-(t*100) = 1601

Input two real numbers for paid and cost: 200 3.99

s = 200, t = 3.99

r = s-t = 196.01

r*100 = 19601

i = (s-t)*100 = 19601

i = r*100 = 19601

i = (s*100)-(t*100) = 19601

Scopes of Variables

```
int main()  
{
```

```
.....  
.....
```

```
int i=0;  
  
cout << i << endl;  
for (i = 1; i < 5; i++)  
{  
    cout << "loop in :: " << i << endl;  
    i+=2;  
    cout << "loop end :: " << i << endl;  
}  
cout << i << endl;  
return 0;  
}
```

```
0  
loop in :: 1  
loop end :: 3  
loop in :: 4  
loop end :: 6  
7  
Press any key to  
continue
```

Nested Scopes

```
int main()
```

```
{
```

```
    int i=0;
```

```
    cout << i << endl;
```

```
    for (i = 1; i < 5; i++)
```

```
    {
```

```
        cout << "loop in :: " << i << endl;
```

```
        int i=3;
```

```
        i+=2;
```

```
        cout << "loop end :: " << i << endl;
```

```
    }
```

```
    cout << i << endl;
```

```
    return 0;
```

```
}
```

```
0
```

```
loop in :: 1
```

```
loop end :: 5
```

```
loop in :: 2
```

```
loop end :: 5
```

```
loop in :: 3
```

```
loop end :: 5
```

```
loop in :: 4
```

```
loop end :: 5
```

```
5
```

```
Press any key to
```

Nested Scopes

```
int main()
{
    int i=0;

    cout << i << endl;
    for (i = 1; i < 5; i++)
    {
        cout << "loop in :: " << i << endl;
        {
            int i=3;
            i+=2;
        }
        cout << "loop end :: " << i << endl;
    }
    cout << i << endl;
    return 0;
}
```

```
0
loop in :: 1
loop end :: 1
loop in :: 2
loop end :: 2
loop in :: 3
loop end :: 3
loop in :: 4
loop end :: 4
5
Press any key to
```

Loops and Scopes

```
int main()
{
    int i=0,j;

    cout << i << endl;
    for (i = 1; i < 5; i++)
    {
        cout << "loop in :: " << i <<

        for (j = i; j < 3; j++)
        {
            cout << "\t inner for loop j :: " << j
                << endl;
        }
        cout << "loop end :: " << i << endl;
    }
    cout << i << endl;
    return 0;
}
```

```
0
loop in :: 1
    inner for loop j :: 1
    inner for loop j :: 2
loop end :: 1
loop in :: 2
    inner for loop j :: 2
loop end :: 2
loop in :: 3
loop end :: 3
loop in :: 4
loop end :: 4
5
```

Loops and Scopes

```
int main()
{
    int i=0;

    cout << i << endl;
    for (i = 1; i < 5; i++)
    {
        cout << "loop in :: " << i << endl;

        for (i = i; i < 3; i++)
        {
            cout << "        inner for loop i :: " << i
                << endl;
        }
        cout << "loop end :: " << i << endl;
    }
    cout << i << endl;
    return 0;
}
```

```
0
loop in :: 1
        inner for loop i :: 1
        inner for loop i :: 2
loop end :: 3
loop in :: 4
loop end :: 4
5
```

Loops and Scopes

```
int main()
{
    int i=0;

    cout << i << endl;
    for (i = 1; i < 5; i++)
    {
        cout << "loop in :: "
            << i << endl;

        int i=0;
        for (i = i; i < 3; i++)
        {
            cout << "        inner for loop i :: "
                << i << endl;
        }
        cout << "loop end :: " << i << endl;
    }
    cout << i << endl;
    return 0;
}
```

```
0
loop in :: 1
    inner for loop i :: 0
    inner for loop i :: 1
    inner for loop i :: 2
loop end :: 3
loop in :: 2
    inner for loop i :: 0
    inner for loop i :: 1
    inner for loop i :: 2
loop end :: 3
loop in :: 3
    inner for loop i :: 0
    inner for loop i :: 1
    inner for loop i :: 2
loop end :: 3
loop in :: 4
    inner for loop i :: 0
    inner for loop i :: 1
    inner for loop i :: 2
loop end :: 3
5
```

Homogeneous aggregate

name:	Tom,	John,	student-3,	student_4,	,	student-20
--------------	------	-------	------------	------------	--------	------------

mid1:	70,	67,	86,	59,	,	80
--------------	-----	-----	-----	-----	--------	----

final:	69,	77,	79,	64,	,	90
---------------	-----	-----	-----	-----	--------	----

GPA:	3.02,	2.89,	3.21,	2.78,	,	3.67
-------------	-------	-------	-------	-------	--------	------

```
// Using array
```

```
string name[20];  
int     mid1[20];  
int     final[20];  
double GPA[20];
```

```
name[0] = "Tom";  
GPA[2] = 3.21;
```

```
// Using tvector class
```

```
#include "tvector.h"  
.....  
tvector<string> name(20);  
tvector<int>    mid1(20);  
tvector<int>    final(20);  
tvector<double> GPA(20);
```

```
name[1] = "John";  
GPA[19] = 3.67;
```

Search in a tvector

```
// Using tvector class

#include "tvector.h"
.....
tvector<string> name(20);
tvector<int>     mid1(20);
tvector<int>     final(20);
tvector<double> GPA(20);
.....
.....

// What is Susan's GAP?

for (int i=0; i < name.length(); i++)
{
    if (name[i] == "Susan")
        cout << GPA[i];
}
```

What can be in an Array

```
// Using tvector class
.....
.....
struct student
{
    string name;
    int     mid1;
    int     final;
    double GPA;
};

.....
struct student class101[20];
.....

// What is Susan's GAP?

for (int i=0; i < class101.length(); i++)
{
    if ( (class101[i]).name == "Susan")
        cout << (class101[i]).GPA;
}
```

Enumerated Types

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

Monday Tuesday Wednesday Thursday Friday Saturday Sunday

```
enum day {Monday, Tuesday, Wednesday, Thursday,  
          Friday, Saturday, Sunday};
```

```
tvector<int> MyclassHr(7);
```

```
MyclassHr[Monday] = 2;
```

```
MyclassHr[Tuesday] = MyclassHr[Thursday] = 0;
```

```
MyclassHr[Wednesday] = 4;
```

```
day ThreeDays[3];
```

```
day ADay;
```

```
ThreeDay[0]=Saturday;
```

```
.....
```

```
if (ADay == Saturday || ADay == Sunday)  
    cout << "It's weekend";
```

```
day FirstDay = day(0);
```

Two dimensional array, **Matrix**

$$\begin{vmatrix} 1 & 3 & 2 \\ 1 & 1 & 3 \\ 0 & 1 & 2 \end{vmatrix} + \begin{vmatrix} 2 & 4 & 0 \\ 2 & 1 & 1 \\ 0 & 2 & 1 \end{vmatrix} = \begin{vmatrix} 3 & 7 & 2 \\ 3 & 2 & 4 \\ 0 & 3 & 3 \end{vmatrix}$$

```
// Using array
```

```
int A[3][3];  
int B[3][3];  
int C[3][3];
```

```
A[1][2] = 3;  
A[1][1] = 1;
```

```
C[2][1] = A[2][1]+B[2][1];
```

```
// Using apmatrix class
```

```
#include "apmatrix.h"
```

```
.....  
apmatrix<int> A(3,3);  
apmatrix<int> B(3,3);  
apmatrix<int> C(3,3);
```

```
A[1][2] = 3;  
A[1][1] = 1;
```

```
C[2][1] = A[2][1]+B[2][1];
```

Operation on **Matrix**

```
// Using apmatrix class

#include "apmatrix.h"
.....
apmatrix<int> A(3,3);
apmatrix<int> B(3,3);
apmatrix<int> C(3,3);
int i,i;
.....
.....

for (i=0;i<A.numrows();i++)
    for (j=0;j<A.numcols();j++)
        C[i][j] = A[i][j] + B[i][j];
```

Easy problem: How to printout a matrix?

Challenging problem: How to do multiplication?