Multiplication table

```
//in: an integer
//out: mult table
int n;
cin >> n;
int j = 0;
while (j <= 10) {
    cout << n << '*' << j << " = " << n*j << endl;
    j = j + 1 // also j += 1,  also: ++j
}
```

Number of digits

```
//in: a non negative integer
//out: its number of digits
int n;
cin >> n;
int ndigits = 0;
if (n == 0) ndigits = 1;
while (n > 0) {
    ++ndigits;
    n /= 10;
}
```

Number of letters in input chanel

```
//in: a sequence of chars in input chanel finished with a dot
//out: number of letters of the sequence
char ch;
cin >> ch;
int count_letters = 0;
while (ch != '.') {
    if (('A' <= ch and ch <= 'Z') or ('a' <= ch and ch <= 'z') ++count_letters;
    cin >> ch;
}
cout << count_letters << endl;
```

Calculate x^y

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

// Input:  read two integer numbers, x and y,
//         such that y >= 0
// Output: write x^y

int main() {
    int x, y;
    cin >> x >> y;

    int i = 0;
    int p = 1;
    while (i < y) { // Repeat several times (y)
        i = i + 1;
        p = p*x;    // p = x^i
    }
    cout << p << endl;
}
```

Prime factors

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

// Input:  read a natural number n > 0
// Output: write the decomposition in prime factors

int main() {
    int n;
    cin >> n;
    int d = 2;  // Variable to store divisors

    // Divide n by divisors from 2 in ascending order
    while (n != 1) {
        if (n%d == 0) { // Check if divisible
            cout << d << endl;
            n = n/d;
        }
        else d = d + 1;
    }
}
```

Program in C++

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

// This program reads a natural number that represents an amount
// of time in seconds and writes the decomposition in hours,
// minutes and seconds

int main() {
    int N;
    cin >> N;
    int h = N / 3600;
    int m = (N % 3600) / 60;
    int s = N % 60;
    cout << h << " hours, " << m << " minutes and "
         << s << " seconds" << endl;
}
```

Example: Draw a right triangle of size n

```
//in: integer greater than zero
//out right triangle of size n
int n;
cin >> n;
int nblanks = n - 1;
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= nblanks; ++j)
        cout << ' ';
    for (int j = nblanks + 1; j <= n; ++j)
        cout << '*';
    cout << endl;
    --nblanks;
}
```

Example: Num of digits

```
//in: sequence of integers greater than zero
//out: for each input number its number of digits
int n;
while (cin >> n) {
    int ndigits = 0;
    while (n > 0) {
        ++ndigits;
        cin >> n;
    }
    cout << ndigits << endl;
}
```

Example: Draw a piramid of size n

```
//in: integer greater than zero
//out: piramid of size n
int n;
cin >> n;
int nblanks = n - 1;
int nstars = 1;
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= nblanks; ++j)
        cout << ' ';
    for (int j = 1; j <= nstars; ++j)
        cout << '*';
    cout << endl;
    --nblanks;
    nstars += 2;
}
```

Revisiting time decomposition

```
// Input:  reads an integer N >= 0 that represents
//         a certain time in seconds
// Output: writes the decomposition of N in
//         hours (h), minutes (m) and seconds (s)
//         such that 0 <= m < 60 and 0 <= s < 60.

int main() {
    int N;
    cin >> N;
    int s = N%60;
    N = N/60;
    cout << N/60 << " " << N%60 << " " << s << endl;
}
```

Max of three numbers (III)

```
int a, b, c, m;

// Pre:  a=A, b=B, c=C
// Post: a=A, b=B, c=C, m=max(A,B,C)

if (a >= b) m = a;
else m = b;          // m=max(a,b)
if (c > m) m = c;
```

Given a sequence of strings, count the number of times "anna" appears.

for instance:

in: anna lluis olga anna pere
out: 2

TREAT ALL

```
int counter = 0;
string name;
while (cin >> name)
    if (name == "anna") ++counter
cout << counter << endl;
```

Given a sequence of strings, check **if** "pere" appears

for instance:

in: pau roger pere marta pere dani pere neus
out: **true** (1 on the cout)

SEARCH

```
bool found = false;
string name;
while (not found and cin >> name)
    found = name == "pere"
    //alternatively: if (name == "pere") found = true;
cout << found << endl;
```

Given a sequence of non negative integers, compute the maximum

for instance:

in: 3 1 5 8 0
out: 8

TREAT ALL

```
int max = -1; // "fake", but suitable initialization
int n;
while (cin >> n)
    if (n > max) max = n;
cout << max << endl;
```

Counting a's

```
// Input:  sequence of characters that ends with '.'
// Output: number of times 'a' appears in the
//         sequence

int main() {
    char c;
    cin >> c;
    int count = 0;
    // Inv: count is the number of a's in the visited
    //      prefix of the sequence. c contains the next
    //      non-visited character
    while (c != '.') {
        if (c == 'a') count = count + 1;
        cin >> c;
    }

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

Counting digits

```
// Input:  a non-negative number N
// Output: number of digits in N (0 has 1 digit)

int main() {
    int N;
    cin >> N;
    int ndigits = 0;

    // Inv: ndigits contains the number of digits in the
    //      tail of the number, N contains the remaining
    //      part (head) of the number
    while (N > 9) {
        ndigits = ndigits + 1;
        N = N/10;  // extracts one digit
    }

    cout << ndigits + 1 << endl;
}
```

Euclid's algorithm for gcd

```
// Input:  read two positive numbers (a and b)
// Output: write gcd(a,b)

int main() {
    int a, b;
    cin >> a >> b; // Let a=A, b=B
    // gcd(A,B) = gcd(a,b)
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    cout << a << endl;
}
```

Prime number

```
// Input:  read a natural number N>0
// Output: write "is prime" or "is not prime" depending on
//         the primality of the number

int main() {
    int N;
    cin >> N;

    int divisor = 2;
    bool is_prime = (N != 1);
    // 1 is not prime, 2 is prime, the rest enter the loop (assume prime)

    // is_prime is true while a divisor is not found
    // and becomes false as soon as the first divisor is found
    while (divisor < N) {
        if (N%divisor == 0) is_prime = false;
        divisor = divisor + 1;
    }

    if (is_prime) cout << "is prime" << endl;
    else cout << "is not prime" << endl;
}
```

Prime number: doing it faster

```
// Input:  read a natural number N>0
// Output: write "is prime" or "is not prime" depending on
//         the primality of the number

int main() {
    int N;
    cin >> N;

    int divisor = 2;
    bool is_prime = (N != 1);

    while (is_prime and divisor*divisor <= N) {
        is_prime = N%divisor != 0;
        divisor = divisor + 1;
    }

    if (is_prime) cout << "is prime" << endl;
    else cout << "is not prime" << endl;
}
```

Drawing a triangle

```
// Input:  read a number n > 0
// Output: write a triangle of size n

int main() {
    int n;
    cin >> n;
    // Inv: the rows 1..i-1 have been written
    for (int i = 1; i <= n; ++i) {
        // Inv: '*' written j-1 times in row i
        for (int j = 1; j <= i; ++j) cout << '*';
        cout << endl;
    }
}
```

Perfect numbers

```
// Input:  read a number n > 0
// Output: write a message indicating
//         whether it is perfect or not

int main() {
    int n;
    cin >> n;

    int sum = 0, d = 1;
    // Inv: sum is the sum of all divisors until d-1
    while (d <= n/2 and sum <= n) {
        if (n%d == 0) sum += d;
        d = d + 1;
    }

    if (sum == n) cout << "is perfect" << endl;
    else cout << "is not perfect" << endl;
}
```

Maximum of a sequence

```
int main() {
    int m;

    int elem;
    cin >> m;
    // Inv: m is the largest element read
    //      from the sequence
    while (cin >> elem) {
        if (elem > m) m = elem;
    }
    cout << m << endl;
}
```

Why is this necessary?

Checks for end-of-sequence and reads a new element.

Finding a number greater than n

```
int main() {
    int n, num;
    cin >> n;
    bool found = false;

    // Inv: found indicates that a number
    //      greater than N has been found
    while (not found and cin >> num) {
        found = num > n;
    }
    cout << found << endl;
}
```

Increasing number

```
// Pre: n >= 0
// Post: It writes YES if the sequence of digits representing n (in base 10)
// is increasing, and it writes NO otherwise

int main() {
    int n;
    cin >> n;
    // The algorithm visits the digits from LSB to MSB.
    bool incr = true;
    int previous = 9;  // Stores a previous "fake" digit

    // Inv: n contains the digits no yet treated, previous contains the
    //      last treated digit (that can never be greater than 9),
    //      incr implies all the treated digits form an increasing sequence
    while (incr and n > 0) {
        int next = n%10;
        incr = next <= previous;
        previous = next;
        n /= 10;
    }

    if (incr) cout << "YES" << endl;
    else cout << "NO" << endl;
}
```

Insert a number in an ordered sequence

```
int first;
cin >> first;

bool found = false;      // controls the search of the location
int next;                // the next element in the sequence

// Inv: All the read elements that are smaller than the first have been written
//      not found => no number greater than or equal to the first has been
//      found yet
while (not found and cin >> next) {
    if (next >= first) found = true;
    else cout << next << " ";
}

cout << first;

if (found) {
    cout << " " << next;
    // Inv: all the previous numbers have been written
    while (cin >> next) cout << " " << next;
}
cout << endl;
```

Calculating n!

```
// Pre: n ≥ 0
// It writes n!
int main() {
    int n;
    cin >> n;
    int i = 0;
    int f = 1;
    // Invariant: f = i! and i ≤ n
    while (i < n) {
        // f = i! and i < n
        i = i + 1;
        f = f*i;
        // f = i! and i ≤ n
    }
    // f = i! and i ≤ n and i ≥ n
    // f = n!
    cout << f << endl;
}
```

Reversing digits

```
// Pre: n ≥ 0
// Post: It writes n with reversed digits (base 10)

int main() {
    int n;
    cin >> n;
    int r;

    r = 0;
    // Invariant (graphical): →
    while (n > 0) {
        r = 10*r + n%10;
        n = n/10;
    }

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