Loops,

and code repetition.
Often some activity needs to be repeated many times. Be it a recursive formula, we wish to calculate, or periodic checking of some variable state.
One of the fundamental tasks is to write code statements that allow for consecutive, repetitive execution of code. In **all** programming languages this is achieved with **loops**. Today we will get familiar with the syntax of loop in **C**:

- We will start with **goto** (but we hope to forever forget this one)
- *while* and *do while*
- and my personal favorite, the *for* loop

Example:

Write a code that prints 100 consecutive integers starting with 0.:

In [1]:
```c
#include <stdio.h>

int main()
{
    printf("%d\n", 0);
    printf("%d\n", 1);
    printf("%d\n", 2);
    printf("%d\n", 3);
}
```

```
0
1
2
3
```

goto label

- Provides a jump to from goto to a labeled statment
- **highly discouraged**
- and is a mark of poor programming skills

In [8]:
```c
#include <stdio.h>

int main()
{
    int i = 0;
    start: // a label start here
    printf ("%d " ,i );
    ++i;
    if (i <=5)
        goto start;
}
```

```
0 1 2 3 4 5
```

Mind that if you use *goto* bad things happen!

The image has been borrowed from the XKCD comic web page: https://xkcd.com/292/

while (condition) {instructions}

first check the condition, than wxecute

In [9]:
```
#include <stdio.h>

int main()
{
    int i=0;
    while(i < 10)
    {
        printf ("%d " ,i );
        ++i;
    }
    printf("\n After a While \n");
}
```

```
0 1 2 3 4 5 6 7 8 9
 After a While
```

In [11]:
```
#include <stdio.h>

int main()
{
    int i=0;
    while(i <= 10)
    {
        printf ("%d " ,i );
        ++i;
    }
    printf("\n After a While \n");
}
```

```
0 1 2 3 4 5 6 7 8 9 10
 After a While
```

In [15]:
```
#include <stdio.h>

int main()
{
    int i=0;
    while(i < 10)
    {
        i++;
        printf ("%d " ,i );
    }
    printf("\n After a While \n");
}
```

```
1 2 3 4 5 6 7 8 9 10
 After a While
```

do {instructions} while (condition)

Execute at least once, even if condition is not meet

In [16]:
```c
#include <stdio.h>

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

```
0 1 2 3 4 5 6 7 8 9
```

In [17]:
```c
#include <stdio.h>

int main()
{
    int i=100;
    do
    {
        printf ("%d " ,i );
        ++i;
    }
    while(i < 10);
}
```

```
100
```

Example:

Write a program that write even (or odd) numbers from 0 up to 50. Use while() and do..while() loops.

In [2]:
```c
#include <stdio.h>

int main()
{
    int i = 0;
    printf("I see even numbers!\n");
    while(i <= 50)
    {
        if(i%2 == 0)
        {
            printf("%d ", i);
        }
        ++i;
    }
}
```

```
I see even numbers!
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
```

Do the same for odd numbers, do..while()

In [4]:
```c
#include <stdio.h>

int main()
{
    int i = 0;
    printf("I see odd numbers!\n");
```

```
    do
    {
        if(i%2)
        {
            printf("%d ", i);
        }
        ++i;
    }
    while(i<50);
}
```

```
I see odd numbers!
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

Yet another way to skin the cat:

In [5]:

```
#include <stdio.h>

int main()
{
    int i = 0;
    printf("I see even numbers!\n");
    while(i <= 50)
    {
        printf("%d ", i);
        i+=2;
    }
}
```

```
I see even numbers!
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
```

for(initialization; condition; incrementation) {instructions}

- The initialization step is executed first, and only once
- Condition is checked before execution. Will not execute if initially condition is false!
- Incrementation is performed after the instructions are executed, as a last step
- initialization; condition; incrementation can be left out, as long as semicolons ; are in. I.e: for(;;) will work and result in an infinite loop

In [6]:

```
#include <stdio.h>

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

```
0 1 2 3 4 5 6 7 8 9
```

In [15]:

```
#include <stdio.h>

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

```
        printf ("%d " ,i );
        ++i;
    }
}
```

Mind where variable is declared!

In [16]:

```
#include <stdio.h>

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

```
/tmp/tmp2g__d4j1.c: In function ‘main’:
/tmp/tmp2g__d4j1.c:9:25: error: ‘i’ undeclared (first use in this function)
    9 |     printf (" | i=%d " ,i );
      |                         ^
/tmp/tmp2g__d4j1.c:9:25: note: each undeclared identifier is reported only once for each f
unction it appears in
[C kernel] GCC exited with code 1, the executable will not be executed
```

In [17]:

```
#include <stdio.h>

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

```
i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9  | i=9
```

An example of declaration before the loop

In [19]:

```
#include <stdio.h>

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

```
i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9  | i=10
```

In [21]:

```
#include <stdio.h>

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

```
    for(int i=0; i<10; ++i) // but also  for( ; i<10; ++i)
    {
        printf ("i=%d " ,i );
    }
    printf (" | i=%d " ,i );
}
```

```
i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9  | i=0
```

Nested loops

a for in a for

Write a program that prints a 4x5 (4 rows and 5 collumns) array. $A_{ij} = i + j$

In [23]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<4; ++i)
    {
        for(int j=0; j<5; ++j) // no brackets since only a single instruction
            printf ("%d\t" , i+j);
        printf("\n"); // new line
    }
}
```

```
0       1       2       3       4
1       2       3       4       5
2       3       4       5       6
3       4       5       6       7
```

Infinite loops

Sometimes (intentionally or not) an infinite loop is created (see one of the examples below)

- while(1) {}
- for(;;) {}

Flow control

i.e. how to manipulate execution of a loop, stop it, or skip the remaining instructions

break

The loop is terminated and the code following the loop is executed

In [32]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<100; ++i)
    {
        printf("iteration %d started ", i);
        if(i%3 == 0 && i > 0)
        {
            printf("\n");
            break;
        }
```

```
        printf("Iteration %d ended\n", i);
    }
    printf("The loop has ended\n");
}
```

```
iteration 0 started Iteration 0 ended
iteration 1 started Iteration 1 ended
iteration 2 started Iteration 2 ended
iteration 3 started
The loop has ended
```

Here a program that prints numbers, but stops as soon as the first number dividable by 7, that is not 7, is encountered.

In [45]:

```
#include <stdio.h>

int main()
{
    for(int i=0; ; ++i)
    {
        if(i%7 == 0 && i>7)
        {
            break;
        }
        printf("%d ", i);
    }
}
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

continue

The loop execution is stooped, and started from the beginning

In [51]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<10; ++i)
    {
        if(i%3 == 0)
            continue;
        printf("%d ", i);
    }
}
```

```
1 2 4 5 7 8
```

Similar as before, print numbers from 0 up to 50, but skip numbers dividable by 7.

In [55]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<=50; ++i)
    {
        if(i%7 == 0)
        {
            printf("\n");
            continue;
        }
        printf("%d ", i);
```

```
    }
}
```

```
1 2 3 4 5 6
8 9 10 11 12 13
15 16 17 18 19 20
22 23 24 25 26 27
29 30 31 32 33 34
36 37 38 39 40 41
43 44 45 46 47 48
50
```

In [56]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<30; ++i)
    {
        if (i%3 == 0)
        {
            //printing ?
            continue;
        }
        printf("%d ", i);
    }
}
```

```
1 2 4 5 7 8 10 11 13 14 16 17 19 20 22 23 25 26 28 29
```

Examples

Print odd or even numbers

Take 1: Use **for** loop to print even numbers

In [58]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<100; i+=2)
    {
        printf("%d ", i);
    }
}
```

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

Take 2: use **for** and **continue** for even numbers

In [63]:

```
#include <stdio.h>

int main()
{
    for(int i=0; i<100; ++i)
    {
        if(i%2 == 1)
            continue;
        printf("%d ", i);
    }
}
```

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62
64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98
```

This is an example from a test

Write a complete program printing odd numbers from 0 to 50, but not those dividable by 9. The program

In [64]:

```c
#include <stdio.h>

int main()
{
    for(int i=1; i<50; i+=2)
    {
        if(i%9 != 0)
            printf("%d ", i);
    }
}
```

```
1 3 5 7 11 13 15 17 19 21 23 25 29 31 33 35 37 39 41 43 47 49
```

An alternative approach

In [65]:

```c
#include <stdio.h>

int main()
{
    for(int i=0; i<50; ++i)
    {
        if(i%2 == 0) continue;
        if(i%9 != 0)
            printf("%d ", i);
    }
}
```

```
1 3 5 7 11 13 15 17 19 21 23 25 29 31 33 35 37 39 41 43 47 49
```

Write a winning game

In [ ]:

```c
#include <stdio.h>

int main()
{
    while(1)
    {
        int uv = 0;
        printf("Give me a value that is not greater than 10\n");
        scanf("%d", &uv);
        int aiv = uv + 1;
        if(uv > 10)
        {
            printf("You cheat, I win!!\n");
            break;
        }
        printf("My is greater I win!!\n");
        printf("%d %d\n", uv, aiv);
    }
}
```

Fibonaci sequence

Write a program that prints the Fibonaci numbers, those are numbers that belong ot a Fibonaci sequence, i.e. each number in a sequence after the first two is sum of the two preceding ones

$n_0 = 0$

$n_1 = 1$

...

$n_i = n_{i-1} + n_{i-2}$

In [71]:

```c
#include <stdio.h>

int main()
{
    int n0 = 0;
    int n1 = 1;
    int n2;

    printf("n0=%d\n", n0);
    printf("n1=%d\n", n1);
    for(int i=2; i<10; ++i)
    {
        n2 = n1 + n0;
        printf("n%d=%d\n", i, n2);
        n0 = n1;
        n1 = n2;
    }
}
```

```
n0=0
n1=1
n2=1
n3=2
n4=3
n5=5
n6=8
n7=13
n8=21
n9=34
```

Fourier expansion of a square wave

Write a program, calculating the Fourier expansion of a square wave.

$f(x) = \frac{4}{\pi} \sum_{n=1}^{N} \frac{1}{n} sin(\frac{n\pi x}{L})$

Print values of the formula for a set of values from 0 uo to L

In [9]:

```c
//%cflags:-lm

#include <stdio.h>
#include <math.h>
#define PI 4.0 * atan(1.0)

double L;

double trig_fun(double x, int N)
{
    double s = 0;
```

```
    for(int n=1; n<=N; ++n)
    {
        s += 1.0/n * sin((n * PI * x)/L);
    }
    return 4.0 / PI * s;
}

int main()
{
    double x = 0.0;
    double h = 0.01;
    L = 1;
    for(int i=0; i<101; ++i)
    {
        printf("x=%lf y=%lf\n", x, trig_fun(x, 10));
        x += h;
    }

    printf("My program\n");
}
```

```
x=0.000000 y=0.000000
x=0.010000 y=1.196416
x=0.020000 y=1.196537
x=0.030000 y=1.188364
x=0.040000 y=1.178112
x=0.050000 y=1.167026
x=0.060000 y=1.155524
x=0.070000 y=1.143785
x=0.080000 y=1.131896
x=0.090000 y=1.119908
x=0.100000 y=1.107851
x=0.110000 y=1.095743
x=0.120000 y=1.083598
x=0.130000 y=1.071423
x=0.140000 y=1.059226
x=0.150000 y=1.047010
x=0.160000 y=1.034779
x=0.170000 y=1.022536
x=0.180000 y=1.010283
x=0.190000 y=0.998021
x=0.200000 y=0.985752
x=0.210000 y=0.973476
x=0.220000 y=0.961196
x=0.230000 y=0.948910
x=0.240000 y=0.936621
x=0.250000 y=0.924327
x=0.260000 y=0.912031
x=0.270000 y=0.899732
x=0.280000 y=0.887430
x=0.290000 y=0.875126
x=0.300000 y=0.862820
x=0.310000 y=0.850512
x=0.320000 y=0.838202
x=0.330000 y=0.825891
x=0.340000 y=0.813578
x=0.350000 y=0.801265
x=0.360000 y=0.788950
x=0.370000 y=0.776634
x=0.380000 y=0.764317
x=0.390000 y=0.751999
x=0.400000 y=0.739680
x=0.410000 y=0.727360
x=0.420000 y=0.715040
x=0.430000 y=0.702719
```

```
x=0.440000 y=0.690398
x=0.450000 y=0.678076
x=0.460000 y=0.665753
x=0.470000 y=0.653430
x=0.480000 y=0.641106
x=0.490000 y=0.628782
x=0.500000 y=0.616458
x=0.510000 y=0.604133
x=0.520000 y=0.591807
x=0.530000 y=0.579482
x=0.540000 y=0.567156
x=0.550000 y=0.554830
x=0.560000 y=0.542503
x=0.570000 y=0.530177
x=0.580000 y=0.517850
x=0.590000 y=0.505522
x=0.600000 y=0.493195
x=0.610000 y=0.480867
x=0.620000 y=0.468539
x=0.630000 y=0.456211
x=0.640000 y=0.443883
x=0.650000 y=0.431555
x=0.660000 y=0.419226
x=0.670000 y=0.406897
x=0.680000 y=0.394568
x=0.690000 y=0.382239
x=0.700000 y=0.369910
x=0.710000 y=0.357581
x=0.720000 y=0.345251
x=0.730000 y=0.332922
x=0.740000 y=0.320592
x=0.750000 y=0.308262
x=0.760000 y=0.295933
x=0.770000 y=0.283603
x=0.780000 y=0.271273
x=0.790000 y=0.258943
x=0.800000 y=0.246613
x=0.810000 y=0.234282
x=0.820000 y=0.221952
x=0.830000 y=0.209622
x=0.840000 y=0.197291
x=0.850000 y=0.184961
x=0.860000 y=0.172630
x=0.870000 y=0.160300
x=0.880000 y=0.147969
x=0.890000 y=0.135639
x=0.900000 y=0.123308
x=0.910000 y=0.110977
x=0.920000 y=0.098646
x=0.930000 y=0.086316
x=0.940000 y=0.073985
x=0.950000 y=0.061654
x=0.960000 y=0.049323
x=0.970000 y=0.036992
x=0.980000 y=0.024662
x=0.990000 y=0.012331
x=1.000000 y=-0.000000
My program
```

In [7]:

```
//%cflags:-lm
#include <stdio.h>
#include <math.h>
#define PI 4.0 * atan(1.0)

//This is a prototype of a function
```

```c
double trig_sq(double, double, int);

int main(){
    double L = 1.0;
    int n = 99;
    int N = 20;
    double h = L / (n-1);
    double x = 0;
    double y;

    for(int i=0; i<n; ++i)
    {
        y = trig_sq(x, L, N);
        printf("%lf %lf\n", x, y);
        x += h;
    }
}

double trig_sq(double x, double L, int N)
{
    double res = 0;
    for(int n=1; n<=N; ++n)
    {
        res += 1.0 / (double)n * sin( n * PI * x / L );
    }
    return 4.0 / PI * res;
}
```

```
0.000000 0.000000
0.010204 0.491334
0.020408 0.913068
0.030612 1.212942
0.040816 1.368223
0.051020 1.389286
0.061224 1.313828
0.071429 1.193940
0.081633 1.080353
0.091837 1.008797
0.102041 0.992470
0.112245 1.022311
0.122449 1.074179
0.132653 1.119842
0.142857 1.137709
0.153061 1.119735
0.163265 1.072573
0.173469 1.013244
0.183673 0.961491
0.193878 0.931970
0.204082 0.929163
0.214286 0.946663
0.224490 0.970675
0.234694 0.986050
0.244898 0.982321
0.255102 0.957431
0.265306 0.917889
0.275510 0.875566
0.285714 0.842610
0.295918 0.826610
0.306122 0.827868
0.316327 0.839746
0.326531 0.851731
0.336735 0.853872
0.346939 0.840738
0.357143 0.813351
0.367347 0.778439
```

```
0.377551 0.745438
0.387755 0.722557
0.397959 0.713521
0.408163 0.716272
0.418367 0.724054
0.428571 0.728353
0.438776 0.722418
0.448980 0.703927
0.459184 0.675765
0.469388 0.644650
0.479592 0.618263
0.489796 0.602092
0.500000 0.597264
0.510204 0.600236
0.520408 0.604388
0.530612 0.602818
0.540816 0.591172
0.551020 0.569348
0.561224 0.541416
0.571429 0.513844
0.581633 0.492779
0.591837 0.481546
0.602041 0.479351
0.612245 0.481707
0.622449 0.482357
0.632653 0.475888
0.642857 0.459927
0.653061 0.436057
0.663265 0.409084
0.673469 0.384999
0.683673 0.368493
0.693878 0.361049
0.704082 0.360390
0.714286 0.361479
0.724490 0.358638
0.734694 0.347897
0.744898 0.328607
0.755102 0.303657
0.765306 0.278233
0.775510 0.257653
0.785714 0.245165
0.795918 0.240636
0.806122 0.240640
0.816327 0.239916
0.826531 0.233554
0.836735 0.219025
0.846939 0.197205
0.857143 0.171980
0.867347 0.148618
0.877551 0.131575
0.887755 0.122659
0.897959 0.120297
0.908163 0.120207
0.918367 0.117185
0.928571 0.107272
0.938776 0.089380
0.948980 0.065742
0.959184 0.040967
0.969388 0.020127
0.979592 0.006646
0.989796 0.000886
1.000000 0.000000
```

In [ ]:

```
//%cflags:-lm
```

```c
#include <stdio.h>
#include <math.h>
#define PI 4.0*atan(1.0)

double fun(double x, int N, double L)
{
    double result = 0.0;
    for(int i=1; i<=N; ++i)
    {
        result += 1.0/i * sin((i*PI*x)/L);
    }
    return 4.0 / PI * result;
}

int main()
{
    int N = 2;
    double L = 2*PI;
    double x0 = 0, x1 = L;
    double h = (x1-x0)/99;

    for(int i=0; i<100; ++i)
    {
        double x = x0 + i*h;
        double y = fun(x, N, L);
        printf("x=%lf y=%lf\n", x, y);
    }
}
```

In [ ]: