Loops
Nested loops
Arrays

Loops

- A loop allows execution of a statement or group of statements multiple times.

- C provides the following types of loops:
  - **for** loop.
  - **while** loop: Allows testing the loop condition before executing the loop body.
  - **do while** loop: Allows testing the loop condition at the end of the loop body.

- Loop Control Statements:
  - **break** statement: Allows terminating the loop.
  - **continue** statement: Allows skipping the remainder of the loop body and restart a new iteration.
  - **goto*** statement: Allows unconditional jumps to a specific place in the code.

*Remember: Using jumps, considered a bad programming practice and should be avoided.

for and while loops

```
for ( statement1; condition; statement3)
      statement2;
```

One statement inside the loop body

```
while (condition)
statement2;
```

```
for (statement1; condition; statement4)
{
   statement2;
   statement3;
   :
}
```

Set of statements inside the loop body

```
while (condition)
{
   statement2;
   statement3
   :
}
```

while and do while loops

```
do
   statement;
while (condition);
```

≡

```
statement;
while (condition)
    statement;
```

the body of do while loop is guaranteed to be executed at least once.

```
do{
   statement1;
   statement2;
}
while (condition);
```

≡

```
statement1;
statement2;
while (condition){
    statement1;
    statement2;
}
```

for and while loop

ניתן לפתור את אותה הבעיה עם שני סוגי הלולאות. בדרך כלל לולאת while שימושית כאשר אנחנו לא יודעים מתי התנאי של הלולאה יהפוך מtrue ל false ואין לדעת כמה פעמים תרוץ הלולאה . למשל יש לכתוב תוכנית הקולטת מספרים מהמשתמש עד שהמשתמש מזין את הערך אפס. לעומת זאת בלולאת for משתמשים בדרך כלל כאשר רוצים לבצע את הלולאה מספר פעמים ידוע מראש למשל לרוץ מ 1 עד 100.

```
for(statement1;condition;statement2){
   statement3;
}
```

≡

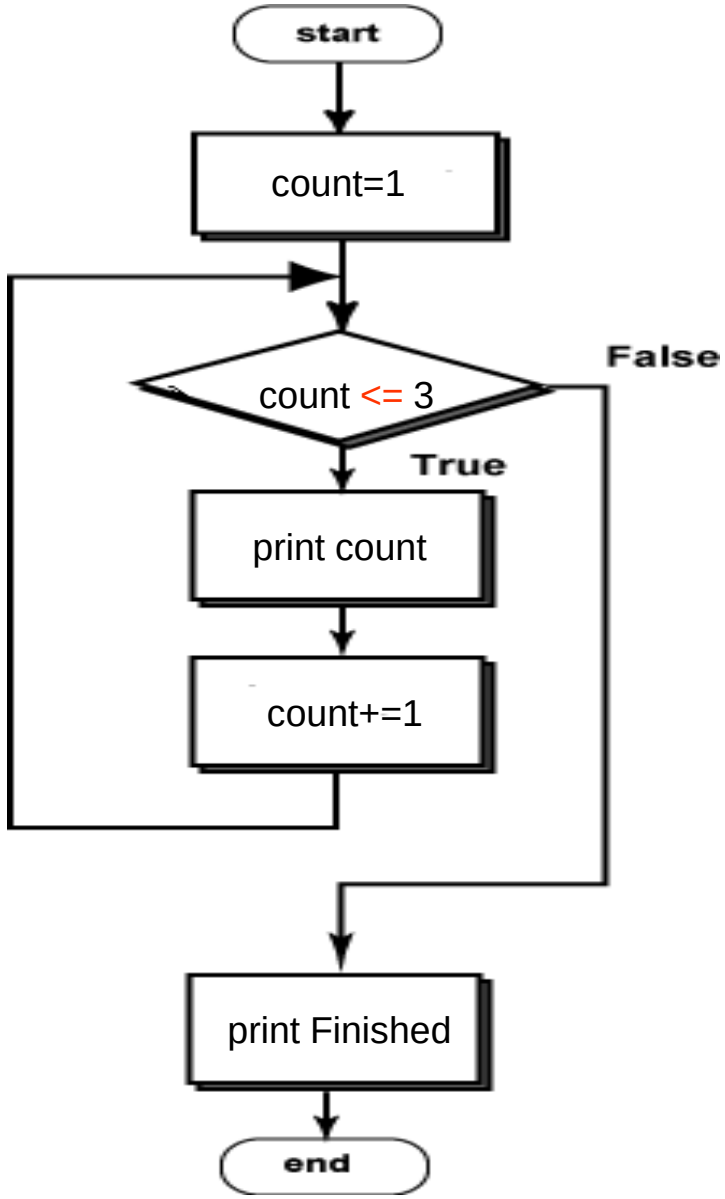
```
statement1;
while (condition){
    statement3;
    statement2;
}
```

Example 1

count < 4 <equal to> count <= 3

```
#include <stdio.h>
int main() {
 int count;
 count = 1;
 while ( count <= 3 ) {
  printf("%d\n", count);
  count += 1;
 }
 printf("Finished\n");
 return 0;
}
```

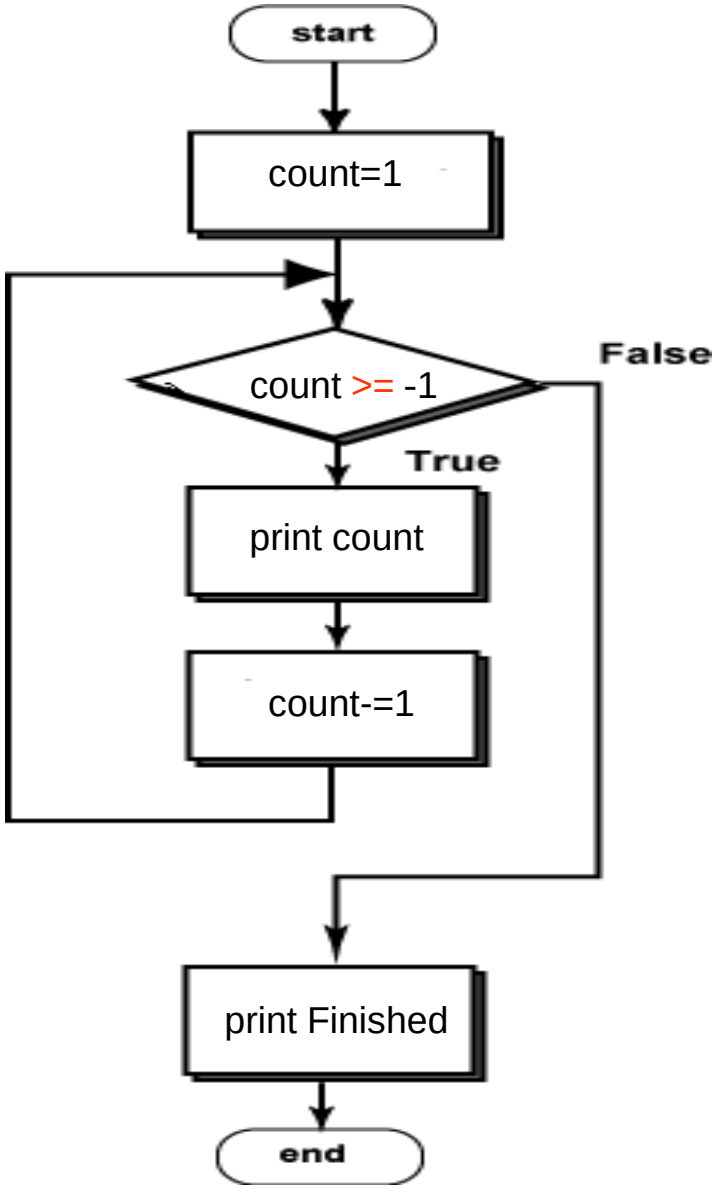
Output:
1
2
3
Finished



Example 2

```
#include <stdio.h>
int main() {
 int count;
 count = 1;
 while ( count >= -1 ) {
  printf("%d\n", count);
  count -= 1;
 }
 printf("Finished\n");
 return 0;
}
```

```
Output:
1
0
-1
Finished
```



Example 3

```
#include <stdio.h>
int main() {
 int count;
 count = 0;
 while ( count <= 3) {
  printf("%d\t", count);
  count += 1;
 }
 return 0;
}//→0 1 2 3
```

```
#include <stdio.h>
int main() {
 int count;
 count = 0;
 while ((count+=1) <= 3) {
  printf("%d\t", count);
 }
 return 0;
}//→1 2 3
```

```
#include <stdio.h>
int main() {
 int count;
 count = 0;
 while (++count <= 3) {
  printf("%d\t", count);
 }
 return 0;
}//→1 2 3
```

```
#include <stdio.h>
int main() {
 int count;
 count = 0;
 while (count++ <= 3) {
  printf("%d\t", count);
 }
 return 0;
}//→1 2 3 4
```

Be careful these are infinite loops

Boolean data type dose not exist in C. Instead, in a logical statement, zero value means false and others are true.

```
#include <stdio.h>
int main() {
 while(1) {
  printf("Never stop");
 }
 return 0;
}
```

```
#include <stdio.h>
int main() {
 for(;;) {
  printf("Never stop");
 }
 return 0;
}
```

```
#include <stdio.h>
int main() {
 while(-1) {
  printf("Never stop");
 }
 return 0;
}
```

break, continue and goto

```
#include <stdio.h>
int main() {
  char c;
  while(1)  {
    printf("Enter 'b' or 'c' or 'q':");
    scanf("%c",&c);
    switch(c)  {
       case 'b': printf("\nbreak Belongs to the switch!\n");
                 break;
       case 'c': printf("\nContinue belongs to the loop\n");
                 continue;
       case 'q': printf("\nquit loop using goto\n");
                 goto L2;
       case '\n':printf("\nWarning \\n – use CTRL-D!\n");
                 break;
       default : printf("\nnot valid value\n");
    }
  }

  L2:
       printf("\nEND\n");
  return 1;
}
```

A program which receives 2 integers from STDIN and print all the numbers in between them.

```
#include <stdio.h>
int main() {
        int start,end;
        printf("Insert start value");
        scanf("%d",&start);
        printf("Insert end value");
        scanf("%d",&end);
        while( start <= end ) {
             printf("%d ",start);
            start++;
      }

 return 0;
}
```

Show all even integers in the range [1,100]

```
#include <stdio.h>
int main() {
        int i=1;
         while( i <= 100 ) {
             if( (i % 2) == 0 )
                printf("%d ",i);
              i++;
         }
 return 0;
}
```

version 1

Even number is an Integer number that if divided by 2 it yields no remainder (remainder is zero).

```
#include <stdio.h>
int main() {
        int i;
        for(i=1; i<=100 ; i++)
        {
             if( (i % 2) == 0 )
                printf("%d ",i);
        }
 return 0;
}
```

version 2

Show the count of digits in a given integer number

```c
#include <stdio.h>
int main() {
    int digitsNum = 0;
    int num = 123;
    int temp = num;

    while( temp != 0 ) {
        temp = temp / 10;
        digitsNum++;
    }

    printf("%d\n",digitsNum);
 return 0;
}
```

version 1

vesion1: this is amlost correct!!!
If num=0 then digitsNum=0.
How can it be fixed? Lets look at version2.

```c
#include <stdio.h>
int main() {
    int digitsNum = 0;
    int num = 0;
    int temp = num;

    temp = temp / 10;
    digitsNum++;
    while( temp != 0 ) {
        temp = temp / 10;
        digitsNum++;
    }

    printf("%d\n",digitsNum);
 return 0;
}
```

version 2

vesion2: this is ok. But we have similar multiple code written before and inside the body of the loop.
Can it be improved? Lets look at version3

```c
#include <stdio.h>
int main() {
    int digitsNum = 0;
    int num = 123;
    int temp = num;

    do {
      temp = temp / 10;
      digitsNum++;
    }
    while( temp != 0 );

    printf("%d\n",digitsNum);
 return 0;
}
```

version 3

Given a positive integer number write a program which checks whether the number is a *palindrome or not

```
#include <stdio.h>
int main() {
    unsigned int num=1;
    unsigned int temp=num;
    unsigned int newNum=0;
    while( temp != 0 ){
        newNum *= 10;
        newNum += temp % 10;
        temp /= 10;
    }
    if( newNum == num)
      printf("yes");
    else
      printf("no");
 return 0;
}
```

*A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward e.g. aba 121 are palindroms but 123 abc are not.

Given a positive integer number of type char - write a program that prints the binary show of the number.

```
#include <stdio.h>
int main() {
 unsigned char num=255;
 unsigned int temp=num , bin=1;

 while( temp != 0 ) {
   bin *= 10;
   bin += temp % 2;
   temp /= 2;
 }

 temp = bin;
 bin = 0;
 while( temp != 1 ) {
   bin *= 10;
   bin += temp % 10;
   temp /= 10;
 }

 printf("%d\n",bin);
 return 0;
}
```

הייצוג הבינארי לאחר חלוקה ב 2 בסדר הפוך (העמודה הכחולה) שימו לב : הספרה השמאלית ביותר הינה הספרה 1 והיא לא נכללת בחישוב שלנו. למה?

הפיכת הסדר של הספרות שנשמרו במשתנה bin מהשלב הקודם

שארית	תוצאת חילוק	חילוק ב2 עם שארית
0	2	**4**
0	1	2
1	0	1

האלגוריתם עוצר כאשר תוצאת החילוק 0

write a program which calculates the factorial of a given positive integer number

```
#include <stdio.h>
int main() {
 unsigned short int n=5,i;
 unsigned long int factorial=1;

  for (i = n ; i > 0 ; i-- )
      factorial *= i;

  if( n < 0 )
     printf("Invalid input");
  else
     printf("%lu",factorial);

 return 0;
}
```

GCD of a given 2 numbers.

```c
#include <stdio.h>
int main() {
 int numA = 616;
 int numB = 165;
 int a = numA , b = numB , c;
 while( (a % b) != 0) {
     c = a%b;
     a = b;
     b = c;
 }
 printf("%d\n",b);
 return 0;
}
```

version1

```c
#include <stdio.h>
int main() {
  int numA = 18;
  int numB = 12;
  int a = numA , b = numB , c;
  do{
      c=a%b;
      a = b;
      b = c;
  }
  while(c!=0) ;
  printf("%d\n",a);
 return 0;
}
```

version2

Euclid's Algorithm for GCD

a%b	a/b	b	a
121	3	165	616
44	1	121	165
33	2	44	121
11	1	33	44
0	3	11	33
-	-	0	11

Gcd (18 , 12) = 6
Gcd (616 , 165) = 11
Gcd (1071 , 1029) = 21

calculates the sum of the below set
excluding all the numbers divided by 3 with
no remainder (3,6,9..).
s = 1+2+4+5+7….20

```
#include <stdio.h>
int main() {
  int s=0,i;
  for (i=1 ; i <= 20 ; i++ ) {
      if ( (i%3) != 0 )
           s += i;
  }
  printf("%d\n",s);
  return 0;
}
```

Calculate the sum of the below set

$$s = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \ldots\ldots\ldots \frac{1}{10}$$

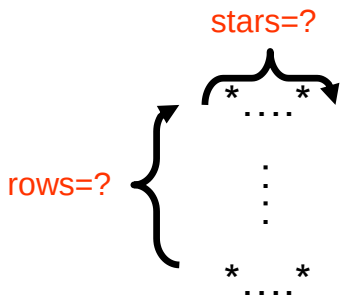
```
#include <stdio.h>
int main() {
 double s=0;
 int i;
 for(i=2; i <= 10 ; i++ ){
    s += 1.0/i;
 }
 printf("%f\n",s);
 return 0;
}
```

Nested Loops

Rows of Stars

```c
#include <stdio.h>
int main() {
    int stars, rows, i, j;
    printf("Insert number of stars:");
    scanf("%d", &stars);
    printf("Insert number of rows:");
    scanf("%d", &rows);
    for(i=0; i < rows ; i++ ) {
        for (j=0; j < stars ; j++) {
            putchar('*');
        }
        putchar('\n');
    }
 return 0;
}
```

version1




```c
#include <stdio.h>
int main() {
  int rows, stars;
  int i, j;
  printf("Insert number of stars:");
  scanf("%d",&stars);
  printf("Insert number of rows:");
  scanf("%d",&rows);
  i=1;
  while ( i <= rows ) {
    j=1;
    while ( j <= stars ) {
      putchar('*');
      j = j + 1;
    }
    putchar('\n');
    i = i + 1;
  }
  return 0;
}
```

version2

Multiplication Table

```
#include <stdio.h>
int main() {
 short i, j;
 for(i=1 ; i<10 ; i++) {
     for(j=1 ; j<10 ; j++) {
          printf("%d\t", i*j);
     }
     putchar('\n');
 }
 return 0;
}
```

i/j	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5					i* j				
6									
7									
8									
9									

Calculate the sum of the below set
S=1!+2!+…+n!

```
#include <stdio.h>
int main() {
   int n=5,i,j;
   int factorial , s=0;
   for(i=1 ; i<=n ; i++) {
     factorial = 1;
     for(j=i ; j>0 ; j-- )
        factorial *= j;
     s += factorial;
   }
  printf("%d\n",s);
  return 0;
}
```

version1

Time complexity: O(n^2)

```
#include <stdio.h>
int main() {
   int n=5,i;
   int factorial=1 , s=0;
   for(i=1 ; i<=n ;i++ ) {
      factorial *= i;
      s += factorial;
   }
   printf("%d\n",s);
  return 0;
}
```

version2

Time complexity: O(n)

Print all N-digits palindrome numbers
e.g. N=2 → 11,22,33,..99

```
#include <stdio.h>
#include <math.h>
int main() {
  int n=3;
  int temp , newNum, num;
  for(num=(int)pow(10,n-1); num < pow(10,n) ; num++) {
     temp=num;
     newNum=0;
     while( temp !=0 ) {
          newNum *= 10;
          newNum += temp%10;
          temp /= 10;
     }
     if( newNum == num )
        printf("%d\t",num);
  }
  return 0;
}
```

* Once using math.h you need to compile with -lm flag:
 gcc -ansi -Wall -pedantic test.c -lm

Arrays

Array declaration

```
int main() {
   int arr[5];
   arr[0]=1;
   arr[1]=2;
   arr[2]=3;
   arr[3]=4;
   arr[4]=5;
   :
}
```

ok

```
#define N 5
int main() {
   int arr[N];
   arr[0]=1;
   arr[1]=2;
   arr[2]=3;
   arr[3]=4;
   arr[4]=5;
   :
}
```

ok

```
#define N 5
int main() {
  int arr[N] = {1 , 2 , 3 , 4, 5};
  :
}
```

ok

```
int arr[5];
int main() {
 :
}
```

ok

global array with default value zero

```
int main() {
   int arr[] = {1 , 2 , 3 , 4, 5};
   :
}
```

ok

```
int main() {
  int N=5;
  int arr[N] = {1 , 2 , 3 , 4, 5};
  :
}
```

error

```
int main() {
  int N=5;
  int arr[N];
  :
}
```

warning

```
int main() {
  const int N=5;
  int arr[N];
  :
}
```

warning

```
#define N 3
int main() {
   int arr[N];
   arr[0]=1;
   arr[1]=2;
   arr[2]=3;
   arr[3]=4;
   arr[4]=5;
   :
}
```

dangerous

```
#define N 3
int main() {
   int arr[N] = {1 , 2 , 3 , 4, 5};
   :
}
```

dangerous

Print all elements of array

```c
#include <stdio.h>
int main() {
   int arr[] = {1 , 2 , 5 , 2 , 3},i;
   for(i=0 ; i<5 ; i++)
      printf("%d ", arr[ i ]);
  return 0;
}
```

version1

```c
#include <stdio.h>
int main() {
   int arr[] = {1 , 2 , 5 , 2 , 3},i;
   for(i=0 ; i<5 ; i++)
      printf("%d ", *(arr+i));
  return 0;
}
```

version2

```c
#include <stdio.h>
void print(int a[ ], int n) {
  int i;
  for(i=0; i<n; i++) {
    printf("%d\t", a[i]);
  }
  putchar('\n');
}
int main() {
 int arr[ ] = {1,2,3,4,5};
 print(arr,5);
 return 1;
}
```

version3

Size of array using sizeof

```
#include <stdio.h>
int main() {
  int i;
   int arr[ ] = {1,2,3};
    for(i=0; i<sizeof(arr)/sizeof(int); i++)
          printf("%d ", arr[i]);
   return 1;
}
```

Be Careful!!!

```
#include <stdio.h>
void check(int a[])
{
  printf("%d\n",sizeof(a)/sizeof(int));
}
int main() {
   int arr[ ] = {1,2,3};
   printf("%d\n",sizeof(arr)/sizeof(int));
   check(arr);
   return 1;
}
```

Output: 1

Output: 3

read elements into array

```c
#include <stdio.h>
int main() {
  int arr[5],i;
  printf("Insert 5 numbers:");
  for(i=0 ; i<5 ; i++) {
     scanf("%d", &arr[i]);
  }
  return 0;
}
```

version1

```c
#include <stdio.h>
int main() {
  int arr[5],i;
  printf("Insert 5 numbers:");
  for(i=0 ; i<5 ; i++) {
     scanf("%d", arr+i);
  }
  return 0;
}
```

version2

Find the maximum number in a given array

```
#include <stdio.h>
int main() {
  int arr[] = {3 , 9 , 2 , 7 , 27 , 0 , 9 , 1};
  int max = arr[0], i;
  for(i=0 ; i < 8 ; i++) {
     if( arr[i] > max )
         max = arr[i];
  }
  printf("%d\n",max);
  return 0;
}
```

Linear Search

```c
#include <stdio.h>
#define N 5
int main() {
  int arr[]={1 , 9 , 3 , 77 , 5};
  int num = 77;
  int i;

  for(i=0 ; i < N ; i++) {
     if( arr[i] == num)
        break;
  }

  if(i<N)
    printf("The Number %d is at place %d\n",num,i);
  else
    printf("Number doesn't exist");
  return 0;
}
```

version1

```c
#include <stdio.h>
int main() {
    const int N=5;
    int arr[] = {1 , 9 , 3 , 77 , 5};
    int num = 77;
    int i=0;

    while(i<N && (arr[i]!=num))
            i++;

    if( i < N)
      printf("The Number %d is at place %d\n",num,i);
    else
      printf("Number doesn't exist");
  return 0;
}
```

version2

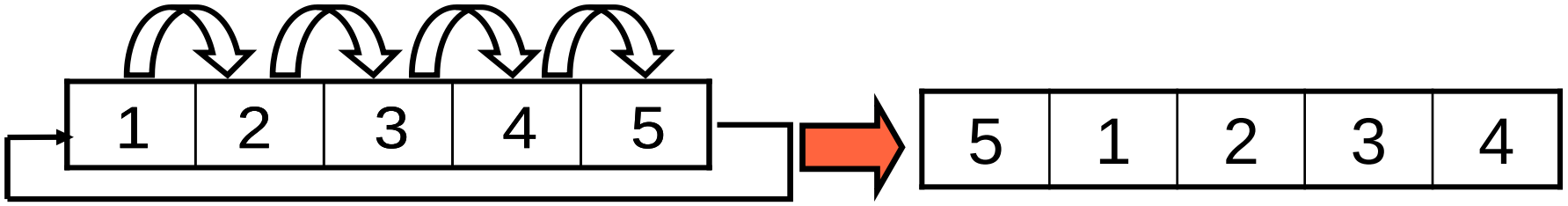
Binary Search

```
#include <stdio.h>
#define N 7
int main() {
   int arr[]= {1,2,3,4,5,6,7};
   int left=0, right=N-1,num=7, mid;

   while(left<=right) {
        mid = (left+right)/2;
        if(arr[mid]==num) {
           printf("Place:%d", mid);
                break;
        }
        else if(num<arr[mid])
           right=mid-1;
        else
           left = mid +1;
   }

   if(left>right)
     printf("Not found");
  return 0;
}
```

Sorted array

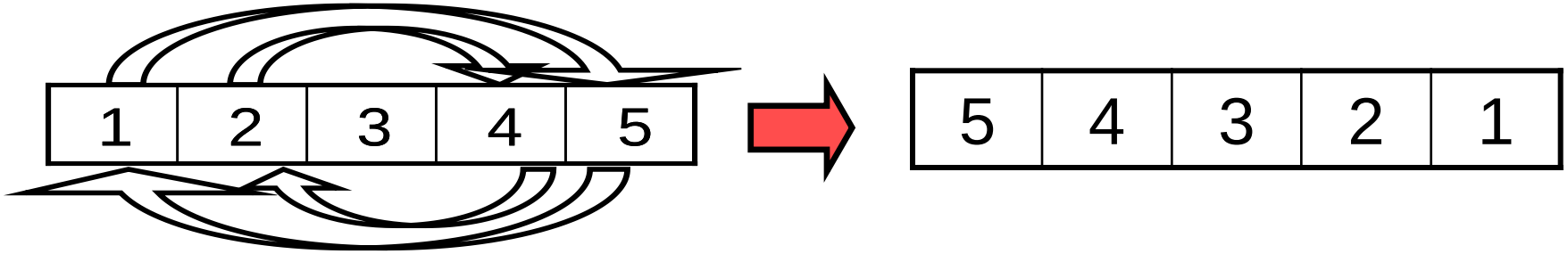
Shift array elements to the left

```
#include <stdio.h>
int main() {
  int arr[] = {1 , 2 , 3 , 4 , 5};
  int temp = arr[4],i;
  for(i=4 ; i>0 ; i--) {
      arr[i] = arr[i-1];
  }
  arr[0] = temp;
  return 0;
}
```



symmetric swap of array elements

```
#include <stdio.h>
#define N 5
int main() {
  int arr[] = {1 , 2 , 3 , 4 , 5};
  int temp,i;
  for(i=0 ; i<N/2 ; i++) {
     temp = arr[i];
     arr[i] = arr[N-1-i];
     arr[N-1-i] = temp;
  }
 return 0;
}
```



Bubble Sort

```
#include <stdio.h>
#define N 8
int main() {
 int arr[]={3,2,3,1,9,7,2,5},i,j;
 for(i=0;i<N;i++) {
    for(j=0;j<N-1-i;j++) {
        if(arr[j]>arr[j+1]){
          int tmp=arr[j];
          arr[j]=arr[j+1];
          arr[j+1]=tmp;
        }
    }
 }
 return 0;
}
```

END