

# Loops (1)

# Loops

- is used to repeat a statement or a block of code several times
- C supports the iteration by different ways to determine the terminating time of the loop.
- Types of loop in C:
  - for
  - while
  - do...while

# while statement

```
while ( expression )  
    statement
```

- while implements the repetition in an algorithm
- Repeatedly executes a block of statements
- Tests a condition (boolean expression) at the start of each iteration
- Terminates when condition becomes false (zero)

# Example

- read in integer numbers and print out their sum

sum = 0  
count = 0

input totalNumbers

```
while (count < totalNumbers) do
{
    input next number
    add next number to sum
    add 1 to count
}
```

output sum

```
#include <stdio.h>
int main(){
    int aNum, sum = 0;
    int count = 0, totalNumbers;
    scanf("%d", &totalNumbers);
    while (count < totalNumbers)
    {
        scanf("%d", &aNum);
        sum += aNum;
        count++;
    }
    printf("Sum is %d\n",sum); return 0;
}
```



There is  
no **do**  
here

# Example (con't)

```
#include <stdio.h>
int main()
{
    int sum=0, count=0, totalNumbers,
    nextnum;

    printf("Enter the total number of
    the array:");
    scanf("%d", &totalNumbers);

    while (count < totalNumbers)
    {
        scanf("%d", &nextnum);
        sum += nextnum;
        count++;
    }
    printf("The sum is %d\n",sum);
    return 0;
}
```

total number	nextnum	sum	count
?	?	0	0
3			
	4	4	1
	-1	3	2
	2	5	3

# Common mistakes

```
while (count < totalNumbers)
    scanf("%d", &nextnum);
    sum+= nextnum;
    count++;
```

only scanf is  
repeated many times

```
while (count < totalNumbers);
{
    scanf("%d", &nextnum);
    sum+= nextnum;
    count++;
}
```

the loop is empty  
(statements are only  
;)

```
while (count < totalNumbers);
{
    scanf("%d", &nextnum);
    sum+= nextnum;
    count++;
    printf("The sum is %d\n",sum);
}
```

print command is  
repeated many times

# End-of-Input : EOF

## Checking for End-of-Input:

- In the example of calculating the sum of a given array, in order to determine the end of the array, we have to enter the total numbers of the array before enter the array.
- Instead of entering the total of numbers for inputting we can mark the end of the integer number sequence by pressing Ctrl+D in Unix or Ctrl+Z in DOS.
- The return value of scanf is the number inputted values. scanf returns EOF if the end of input is detected.

# Example

- read in integer numbers and print out their sum (ver 2)

Algorithm: (version 2)

```
sum = 0
while (not end of input)
{
    input aNum
    add aNum to sum
}
output sum
```

```
#include <stdio.h>
int main()
{
    int aNum, sum = 0;
    while
        (scanf("%d",&aNum)!=EOF)
    {
        sum += aNum;
    }
    printf("Sum is %d\n", sum);
    return 0;
}
```



# for statement

for ( *initialization*; *condition*; *update* )  
    *statement*

- Form of loop which allows for *initialization* and *iteration* control
- parts of **for** statement is optional. When the loop condition is not mentioned explicitly, it takes the default value (true)
- Update is always done after statement of the loop.

# Example

- read in integer numbers and print out their sum

sum = 0  
count = 0

input totalNumbers

```
while (count < totalNumbers) do
{
    input next number
    add next number to sum
    add 1 to count
}
```

output sum

```
#include <stdio.h>
int main()
{
    int aNum, sum = 0;
    int count, totalNumbers;
    scanf("%d", &totalNumbers);
    for (count=0; count<totalNumbers;
        count++)
    {
        scanf("%d", &aNum);
        sum += aNum;
    }
    printf("Sum is %d\n",sum);
    return 0;
}
```

# Compare while and for

```
#include <stdio.h>
int main()
{
    int sum=0, count=0,
        totalNumbers, nextnum;

    printf("Enter the total number
        of the array:");
    scanf("%d", &totalNumbers);

    while (count < totalNumbers)
    {
        scanf("%d", &nextnum);
        sum += nextnum;
        count++;
    }
    printf("The sum is %d\n", sum);
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int aNum, sum = 0;
    int count, totalNumbers;
    scanf("%d", &totalNumbers);
    for (count=0;
        count<totalNumbers;
        count++)
    {
        scanf("%d", &aNum);
        sum += aNum;
    }
    printf("Sum is %d\n", sum);
    return 0;
}
```

# Common mistakes

```
for (count=0; count<totalNumbers;)
{
    scanf("%d", &aNum);
    sum += aNum;
}
```

count variable is not updated  
after each iteration

```
for (count=0;
    count<totalNumbers;count++);
{ scanf("%d", &aNum);
  sum += aNum;
}
```

; must not be here

```
for (count=0,
    count<totalNumbers,count++)
{ scanf("%d", &aNum);
  sum += aNum;
}
```

; not , here

# Comma

- In the for statement *initialization*; *condition*; *update* are optional. If no condition is given, we have an infinitive loop.
  - for (;;) and while(1) are infinitive loops
- Some statements can be given in *initialization* and *update*. These statements must be separated by a comma.
- Example:

```
for (i=0, j=100; i<=j; i++, j--)  
    printf("(%d, %d\n", i, j);
```

Output:

(0, 100)

(1, 99)

...

(49, 51)

(50, 50)

# Exercises

- (i) Write a program that prints all 2-digits numbers where their sum = 10, for instance 19, 28,...
- (ii) Write a program that prints 100 first numbers in the following sequence: 1 2 3 5 8 13 21...
- (iii) Write a program that receives as input a positive integer  $n$  ( $n \leq 9$ ), and prints out a triangular as following if  $n = 5$

```
1
12
123
1234
12345
```

# Solution (Exercise 1)

```
for (x=1; x<=9; x++)  
{  
    printf(“%d%d\n”, x, 10-x);  
}
```

## Solution (Exercise 2)

```
first = 1; second = 2;
for (count=1; count<=100; count++)
{
    printf("%5d", first);
    tmp = first + second;
    first = second;
    second = tmp;
}
```



# Solution (Exercise 3)

```
for (i=1; i<=n; i++)  
{  
    for (j=1; j<=i; j++)  
        printf("%d", j);  
    printf("\n");  
}
```