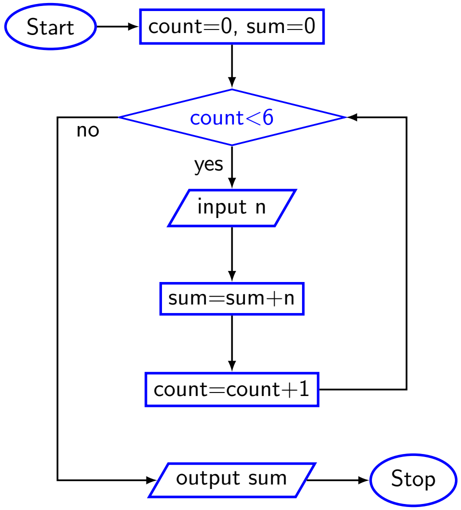
Looping

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
`arijit@iitp.ac.in`

Loops

- Group of statements that are executed repeatedly while some condition remains true
- Each execution of the group of statements is called an **iteration** of the loop

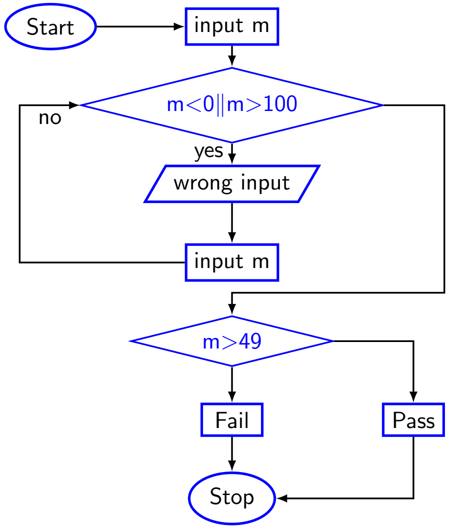
Example: Sum


Start
count=0, sum=0
count<6
no
yes
input n
sum=sum+n
count=count+1
output sum
Stop

Example: Pass/Fail

- Given an exam marks as input, display the appropriate message based on the rules below:
  - If marks is greater than 49, display "PASS", otherwise display "FAIL"
  - However, for input outside the 0-100 range, display WRONG INPUT and prompt the user to input again until a valid input is entered

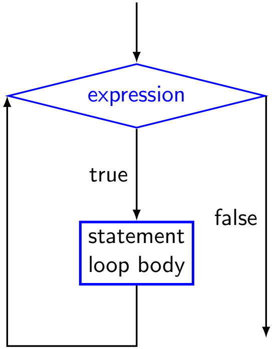
Example: Pass/Fail



Looping: `while` statement

```
while(expression)
  statement;

while(expression){
  statement;
}
```



- The condition to be tested is any expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed. Then the expression is evaluated again and the same thing repeats. The loop **terminates** when the expression evaluates to 0.

Example

```c
int i = 1, n;
scanf("%d", &n);
while(i <= n) {
  printf ("Line no :  %d\n",i);
  i = i + 1;
}
```

Example

```
int weight;
scanf("%d", &weight);
while(weight > 65) {
  printf ("Go, excercise, then come back\n");
  printf ("Enter your weight:  ");
  scanf("%d", &weight);
}
```

Sum of first N natural numbers

Sum of first N natural numbers

```
int N,count=1, sum=0;
scanf("%d", &N);
```

Sum of first N natural numbers

```
int N,count=1, sum=0;
scanf("%d", &N);
while(count <= N) {
```

Sum of first N natural numbers

```
int N,count=1, sum=0;
scanf("%d", &N);
while(count <= N) {
  sum = sum + count;
```

Sum of first N natural numbers

```
int N,count=1, sum=0;
scanf("%d", &N);
while(count <= N) {
  sum = sum + count;
  count = count + 1;
}
printf("Sum=%d\n", &sum);
```

Compute $\sum_{i=1}^{N} i^2$

```
void main(){
  int N,count=1, sum=0;
  scanf("%d", &N);
```

Compute $\sum_{i=1}^{N} i^2$

```
void main(){
  int N,count=1, sum=0;
  scanf("%d", &N);
  while(count <= N) {
```

Compute $\sum_{i=1}^{N} i^2$

```
void main(){
  int N,count=1, sum=0;
  scanf("%d", &N);
  while(count <= N) {
    sum = sum + count*count;
```

Compute $\sum_{i=1}^{N} i^2$

```
void main(){
  int N,count=1, sum=0;
  scanf("%d", &N);
  while(count <= N) {
    sum = sum + count*count;
    count = count + 1;
  }
  printf("Sum=%d\n", sum);
}
```

Compute GCD

```
void main(){
  int A,B,temp;
  scanf("%d%d", &A,&B);
  if(A>B){
    temp=A; A=B; B=temp;
  }
```

Compute GCD

```
void main(){
  int A,B,temp;
  scanf("%d%d", &A,&B);
  if(A>B){
    temp=A; A=B; B=temp;
  }
  while(B%A!=0) {
    temp = B% A;
    B = A;
    A = temp;
  }
  printf("GCD=%d\n", A);
}
```

Double your money

- Suppose your Rs 10000 is earning interest at 1% per month. How many months until you double your money ?

```
void main(){
  double money=10000.0;
  int n=0;
```

Double your money

- Suppose your Rs 10000 is earning interest at 1% per month. How many months until you double your money ?

```
void main(){
  double money=10000.0;
  int n=0;
  while(money<20000) {
    money = money * 1.01;
    n++;
  }
  printf("Months=%d\n", n);
}
```

Maximum of positive numbers

```
void main(){
  double max=0.0,n;
  printf("Enter +ve numbers, end with a negative number\n");
  scanf("%lf",&n);
```

Maximum of positive numbers

```
void main(){
  double max=0.0,n;
  printf("Enter +ve numbers, end with a negative number\n");
  scanf("%lf",&n);
  while(n>0) {
    if(n>max) max = n;
    scanf("%lf",&n);
  }
  printf("Maximum=%d\n", max);
}
```

Find the sum of digits of a number

```
void main(){
  int sum=0,n;
  scanf("%d",&n);
```

Find the sum of digits of a number

```
void main(){
  int sum=0,n;
  scanf("%d",&n);
  while(n!=0) {
    sum = sum + (n%10);
    n = n / 10;
  }
  printf("Sum=%d\n", sum);
}
```

Looping: `for` statement

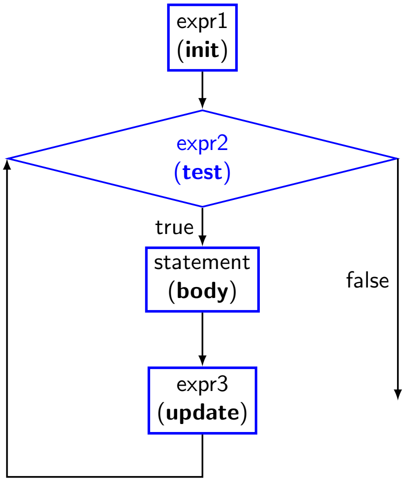
- Most commonly used looping structure in C

```
for(expr1;expr2;expr3)
  statement;

for(expr1;expr2;expr3){
  statement;
}
```

- `expr1` (init): initialize parameters
- `expr2` (test): test condition, loop continues if expression is non-0
- `expr3` (update): used to alter the value of the parameters after each iteration
- `statement` (body): body of loop

Looping: `for` statement



Computing factorial

```
void main(){
  int n,count,prod=1;
  scanf("%d",&n);
  for(count=1;count<=n;++count) {
    prod = prod * count;
  }
  printf("Factorial=%d\n", prod);
}
```

Computing e^x series upto n terms

```
void main(){
  int n,count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  scanf("%d",&n);
```

Computing e^x series upto n terms

```
void main(){
  int n,count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  scanf("%d",&n);
  for(count=1;count<=n;++count) {
```

Computing e^x series upto n terms

```
void main(){
  int n,count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  scanf("%d",&n);
  for(count=1;count<=n;++count) {
    sum += term;
```

Computing e^x series upto n terms

```
void main(){
  int n,count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  scanf("%d",&n);
  for(count=1;count<=n;++count) {
    sum += term;
    term *= x/count;
```

Computing e^x series upto n terms

```
void main(){
  int n,count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  scanf("%d",&n);
  for(count=1;count<=n;++count) {
    sum += term;
    term *= x/count;
  }
  printf("Exp(x,n)=%d\n",sum);
}
```

Computing e^x series upto 4 decimal places

```
void main(){
  int count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
```

Computing e^x series upto 4 decimal places

```
void main(){
  int count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  for(count=1;term>=0.0001;++count) {
```

Computing e^x series upto 4 decimal places

```
void main(){
  int count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  for(count=1;term>=0.0001;++count) {
    sum += term;
```

Computing e^x series upto 4 decimal places

```
void main(){
  int count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  for(count=1;term>=0.0001;++count) {
    sum += term;
    term *= x/count;
```

Computing e^x series upto 4 decimal places

```
void main(){
  int count;
  float x,term=1.0,sum=0.0;
  scanf("%f",&x);
  for(count=1;term>=0.0001;++count) {
    sum += term;
    term *= x/count;
  }
  printf("Exp(x)=%d\n",sum);
}
```

Equivalence of `for` and `while`

```
for(expr1;expr2;expr3) {
  statement;
}
```

Equivalence of `for` and `while`

```
for(expr1;expr2;expr3) {
  statement;
}
```

```
expr1;
while(expr2){
  statement;
  expr3;
}
```

Sum of first N natural numbers

```
int N,count=1, sum=0;
scanf("%d", &N);
while(count <= N) {
  sum = sum + count;
  count = count + 1;
}
printf("Sum=%d\n", &sum);
```

```
int N,count=1, sum=0;
scanf("%d", &N);
for(;count<=N;count++) {
  sum = sum + count;
}
printf("Sum=%d\n", &sum);
```

Some observation on `for`

- Initialization, loop-continuation test, and update can contain arithmetic expressions

```
for ( k = x; k <= 4 * x * y; k += y / x )
```

- Update may be negative (decrement)

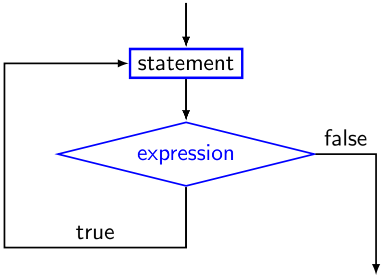
```
for (digit = 9; digit >= 0; --digit)
```

- If loop continuation test is initially 0 (false)
  - Body of for structure not performed
  - No statement executed
  - Program proceeds with statement after for structure

Looping: `do-while` statement

```
do
  statement
while(expression);

do{
  statement
} while(expression);
```



Example

- Prompt user to input ”month” value, keep prompting until a correct value of month is given as input

Example

- Prompt user to input ”month” value, keep prompting until a correct value of month is given as input

```
do{
  printf("Please input month[1-12]");
  scanf("%d",&month);
} while((month<1)||(month>12));
```

Decimal to (reverse) binary conversion

```
void main(){
  int dec;
  scanf("%d",&dec);
```

Decimal to (reverse) binary conversion

```
void main(){
  int dec;
  scanf("%d",&dec);
  do{
    printf("%2d",(dec%2));
```

Decimal to (reverse) binary conversion

```
void main(){
  int dec;
  scanf("%d",&dec);
  do{
    printf("%2d",(dec%2));
    dec /= 2;
```

Decimal to (reverse) binary conversion

```
void main(){
  int dec;
  scanf("%d",&dec);
  do{
    printf("%2d",(dec%2));
    dec /= 2;
  } while(dec!=0)
  printf("\n");
}
```

Echo characters typed on screen until end of line

```
void main(){
  char echo;
  do{
```

Echo characters typed on screen until end of line

```
void main(){
  char echo;
  do{
    scanf("%c",&echo);
    printf("%c",echo);
```

Echo characters typed on screen until end of line

```
void main(){
  char echo;
  do{
    scanf("%c",&echo);
    printf("%c",echo);
  }while(echo!='\n');
}
```

Infinite loop

```
while(1){
   statements
}
```

Infinite loop

```
while(1){
   statements
}
```

```
for(;;){
   statements
}
```

Infinite loop

```
while(1){
   statements
}
```

```
for(;;){
   statements
}
```

```
do{
   statements
}while(1);
```

The `break` statement

- Break out of the loop body
  - Can use with while, do while, for, switch
  - Does not work with if, else
- Causes immediate exit from a while, do/while, for or switch structure
- Program execution continues with the first statement after the structure

Example

```
void main(){
  int fact=1, i=1;
  while(i<10){
    fact=fact*i;
    if(fact>100){
      printf("Factorial of %d above 100",i);
      break;
    }
    ++i;
  }
}
```

Example: Prime number

```
void main(){
  int n, i=2;
  scanf("%d",&n);
```

Example: Prime number

```
void main(){
  int n, i=2;
  scanf("%d",&n);
  while(i<n){
```

Example: Prime number

```
void main(){
  int n, i=2;
  scanf("%d",&n);
  while(i<n){
    if(n%i==0){
```

Example: Prime number

```
void main(){
  int n, i=2;
  scanf("%d",&n);
  while(i<n){
    if(n%i==0){
      printf("%d is not a prime",n);
```

Example: Prime number

```
void main(){
  int n, i=2;
  scanf("%d",&n);
  while(i<n){
    if(n%i==0){
      printf("%d is not a prime",n);
      break;
    }
```

Example: Prime number

```c
void main(){
  int n, i=2;
  scanf("%d",&n);
  while(i<n){
    if(n%i==0){
      printf("%d is not a prime",n);
      break;
    }
    ++i;
  }
  if(i==n) printf("%d is a prime",n);
}
```

Example: Prime number - efficient version

```
#include <math.h>
void main(){
  int n, i=2, flag=0;
  double limit;
  scanf("%d",&n);
  limit=sqrt(n);
  while(i<=limit){
    if(n%i==0){
      printf("%d is not a prime",n);
      flag=1; break;
    }
    ++i;
  }
  if(flag==0) printf("%d is a prime",n);
}
```

The `continue` statement

- Skips the remaining statements in the body of a while, for or do/while structure
  - Proceeds with the next iteration of the loop
- while and do/while loop
  - Loop-continuation test is evaluated immediately after the continue statement is executed
- for loop
  - `expr3` is evaluated, then `expr2` is evaluated

An example with `break` & `continue`

```
void main(){
  int fact, i=1;
  double limit;
  while(1){
    fact=fact*i;
    ++i;
    if(i<=10)
      continue;
    break;
  }
}
```

Some pitfalls

```
while(sum<=NUM);
    sum=sum+2;
```

```
for(i=0;i<=NUM;++i);
    sum=sum+i;
```

```
for(i=1;i!=10;i=i+2)
    sum=sum+i;
```

```
double x;
for(x=0;x<2.0;x=x+0.2)
    printf("%.18f\n",x);
```

Nested loops: Printing 2D figure

- How to print the following?

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

- Approach:

```
    repeat 3 times
      print a row of 5 *'s
      OR
      repeat 5 times
        print *
```

Nested loops

```
const int ROWS=3;
const int COLS=3;
...
row=1;
while(row<=ROWS){
  /* print a of 5 *'s */
  ...
  ++row;
}
```

```
row=1;
while(row<=ROWS){
  /* print a of 5 *'s */
  col=1;
  while(col<=COLS){
    printf("*");
    col++;
  }
  printf("\n");
  ++row;
}
```

2D Figure with `for` loop

- Print following:

  ******

  ******

  ******

```
const int ROWS=3;
const int COLS=3;
...
```

2D Figure with `for` loop

- Print following:

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

```
const int ROWS=3;
const int COLS=3;
...
for(row=1;row<=ROWS;++row){
```

2D Figure with `for` loop

- Print following:

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

```
const int ROWS=3;
const int COLS=3;
...
for(row=1;row<=ROWS;++row){
  for(col=1;col<=COLS;++col){
```

2D Figure with `for` loop

- Print following:

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

```
const int ROWS=3;
const int COLS=3;
...
for(row=1;row<=ROWS;++row){
  for(col=1;col<=COLS;++col){
    printf("*");
```

2D Figure with `for` loop

- Print following:

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

```
const int ROWS=3;
const int COLS=3;
...
for(row=1;row<=ROWS;++row){
  for(col=1;col<=COLS;++col){
    printf("*");
  }
```

2D Figure with `for` loop

- Print following:

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

```
const int ROWS=3;
const int COLS=3;
...
for(row=1;row<=ROWS;++row){
  for(col=1;col<=COLS;++col){
    printf("*");
  }
  printf("\n");
}
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<=row;++col){
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<=row;++col){
    printf("*");
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<=row;++col){
    printf("*");
  }
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<=row;++col){
    printf("*");
  }
  printf("\n");
}
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<row;++col)
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<row;++col)
    printf(" ");
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<row;++col)
    printf(" ");
  for(col=1;col<=ROWS-row+1;++col)
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<row;++col)
    printf(" ");
  for(col=1;col<=ROWS-row+1;++col)
    printf("*");
```

2D Figure

- Print following:

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

```
const int ROWS=5;
...
int row, col;
for(row=1;row<=ROWS;++row){
  for(col=1;col<row;++col)
    printf(" ");
  for(col=1;col<=ROWS-row+1;++col)
    printf("*");
  printf("\n");
}
```

`break` & `continue` with nested loops

- For nested loops, break and continue are matched with the nearest loops (for, while, do-while)
- Example:

```
while(i<n){
  for(k=1;k<m;++k){
    if(k%i==0) break;
  }
  i=i+1;
}
```

Example

```
void  main(){
  int low,high,desired,i,flag=0;
  scanf("%d%d%d",&low,&high,&desired);
  i=low;
  while(i<high){
    for(j=i+1;j<=high;++j){
      if(j%i==desired){
        flag=1;
        break;
      }
    }
    if(flag==1) break;
    i=i+1;
  }
}
```

The comma operator

- Separates expressions
- Syntax: `expr-1, expr-2, ...,expr-n`
    - `expr-1, expr-2,...` are all expressions
- Is itself an expression, which evaluates to the value of the last expression in the sequence
- Since all but last expression values are discarded, not of much general use
- But useful in for loops, by using side effects of the expressions

Example

- We can give several expressions separated by commas in place of expr1 and expr3 in a for loop to do multiple assignments for example

```
for(fact=1,i=1;i<=10;++i)
  fact=fact*i;

for(sum=0,i=1;i<=N;++i)
  sum=sum+i*i;
```