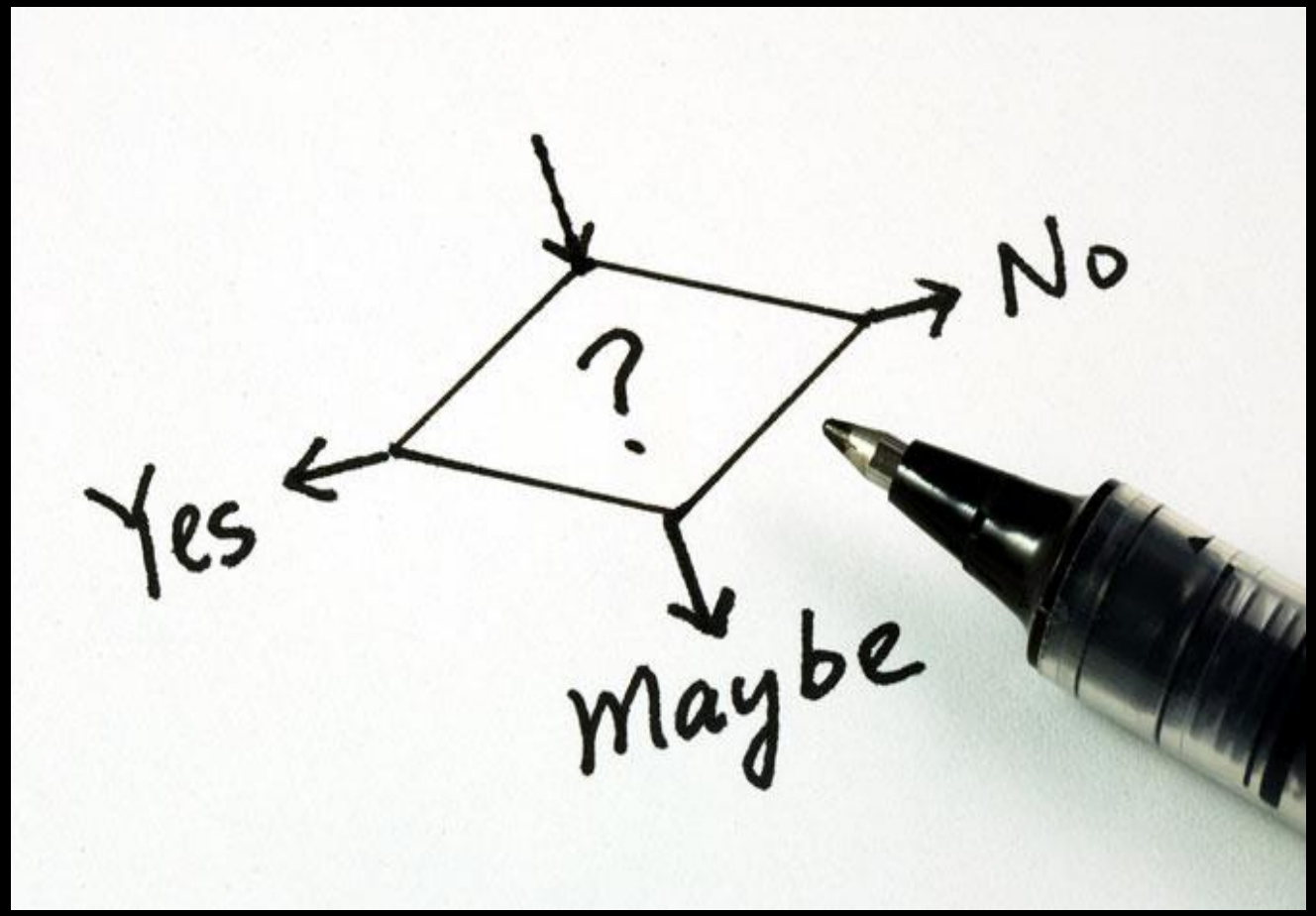
Decision making in C

Need of decision making




```
if number is odd
{
    /* code */
}
```

```
else number is even
{
    /* code */
}
```

Decision Making or Conditional Statement

- C program statements are executed sequentially.
- Decision Making statements are used to control the flow of program.
- It allows us to control whether a program segment is executed or not.
- It evaluates condition or logical expression first and based on its result (either true or false), the control is transferred to particular statement.
- If result is true then it takes one path else it takes another path.

Decision Making Statements in C

Decision Making Statements are

One way Decision:	`if`	(Also known as simple if)
Two way Decision:	`if…else`	
Multi way Decision:	`if…else if…else if…else`	
Two way Decision:	`?:` (Conditional Operator)	
n-way Decision:	`switch…case`	

Relational Operators

- Relational Operator is used to compare two expressions.
- It gives result either true or false based on relationship of two expressions.

Math	C	Meaning	Example	Result
>	>	is greater than	`5 > 4`	`true`
≥	>=	is greater than or equal to	`5 >= 4`	`true`
<	<	is less than	`5 < 4`	`false`
≤	<=	is less than or equal to	`5 <= 4`	`false`
≠	!=	is not equal to	`5 != 4`	`true`
=	==	is equal to	`5 == 4`	`false`

If statement

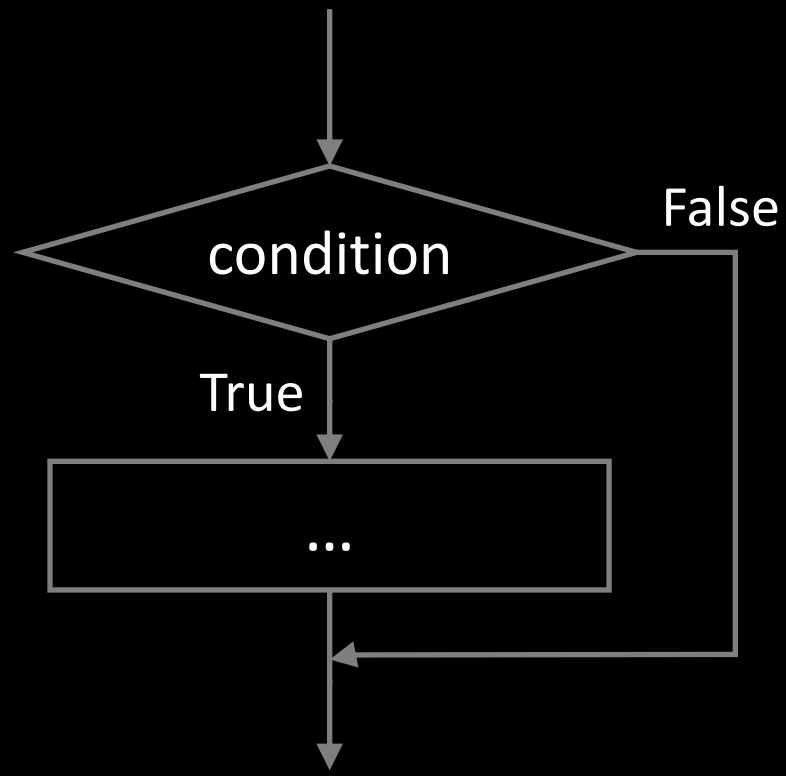
if

- `if` is single branch decision making statement.
- If condition is `true` then only body will be executed.
- `if` is a keyword.

Syntax

```
if(condition)
{
    // Body of the if
    // true part
}
```

Flowchart of `if`



WAP to print Zero if given number is 0

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a == 0)
    {
        printf("Zero");
    }
}
```

Output

```
Enter Number:0
Zero
```

WAP to print Positive or Negative Number

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a >= 0)
    {
        printf("Positive Number");
    }
    if(a < 0)
    {
        printf("Negative Number");
    }
}
```

Output

```
Enter Number:5
Positive Number
```

Output

```
Enter Number:-5
Negative Number
```

Modulus Operator

- `%` is modulus operator in C
- It divides the value of one expression (number) by the value of another expression (number), and returns the remainder.
- Syntax: `express1 % express2`
- E.g.
  - `7%2` Answer: 1
  - `6%2` Answer: 0
  - `25%10` Answer: 5
  - `37%28` Answer: 9

WAP to print Odd or Even Number

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a%2 == 0)
    {
        printf("Even Number");
    }
    if(a%2 != 0)
    {
        printf("Odd Number");
    }
}
```

Output

```
Enter Number:12
Even Number
```

Output

```
Enter Number:11
Odd Number
```

If..else statement

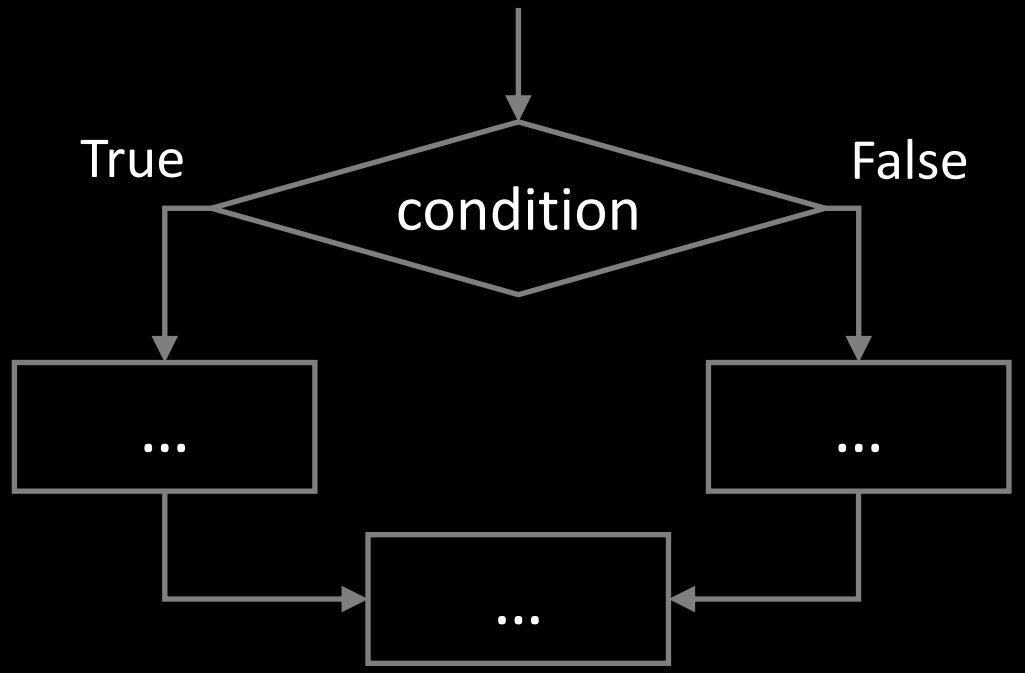
if...else

- `if…else` is two branch decision making statement
- If condition is true then true part will be executed else false part will be executed
- `else` is keyword

Syntax

```
if(condition)
{
    // true part
}
else
{
    // false part
}
```

Flowchart of `if…else`



WAP to print Positive or Negative Number using **if…else**

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a >= 0)
    {
        printf("Positive Number");
    }
    else
    {
        printf("Negative Number");
    }
}
```

Output

```
Enter Number:5
Positive Number
```

Output

```
Enter Number:-5
Negative Number
```

WAP to print Odd or Even Number using **if…else**

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a%2 == 0)
    {
        printf("Even Number");
    }
    else
    {
        printf("Odd Number");
    }
}
```

Output

```
Enter Number:12
Even Number
```

Output

```
Enter Number:11
Odd Number
```

WAP to find largest number from given 2 numbers using if

Program

```c
#include<stdio.h>
void main()
{
    int a, b;
    printf("Enter Two Numbers:");
    scanf("%d%d",&a,&b);
    if(a > b)
    {
        printf("%d is largest", a);
    }
    if(a < b)
    {
         printf("%d is largest", b);
    }
}
```

Output

```
Enter Two Numbers:4
5
5 is largest
```

WAP to find largest number from given 2 numbers using **if…else**

Program

```c
#include<stdio.h>
void main()
{
    int a, b;
    printf("Enter Two Numbers:");
    scanf("%d%d",&a,&b);
    if(a > b)
    {
        printf("%d is largest", a);
    }
    else
    {
         printf("%d is largest", b);
    }
}
```

Output

```
Enter Two Numbers:4
5
5 is largest
```

{ }

- If body of `if` contains only one statement then `{ }` are not compulsory
- But if body of `if` contains more than one statements then `{ }` are compulsory

```c
if(a >= b)
{
    printf("%d is largest", a);
}
```

Both are same

```c
if(a >= b)
    printf("%d is largest", a);
```

If…else if…else if…else Ladder if

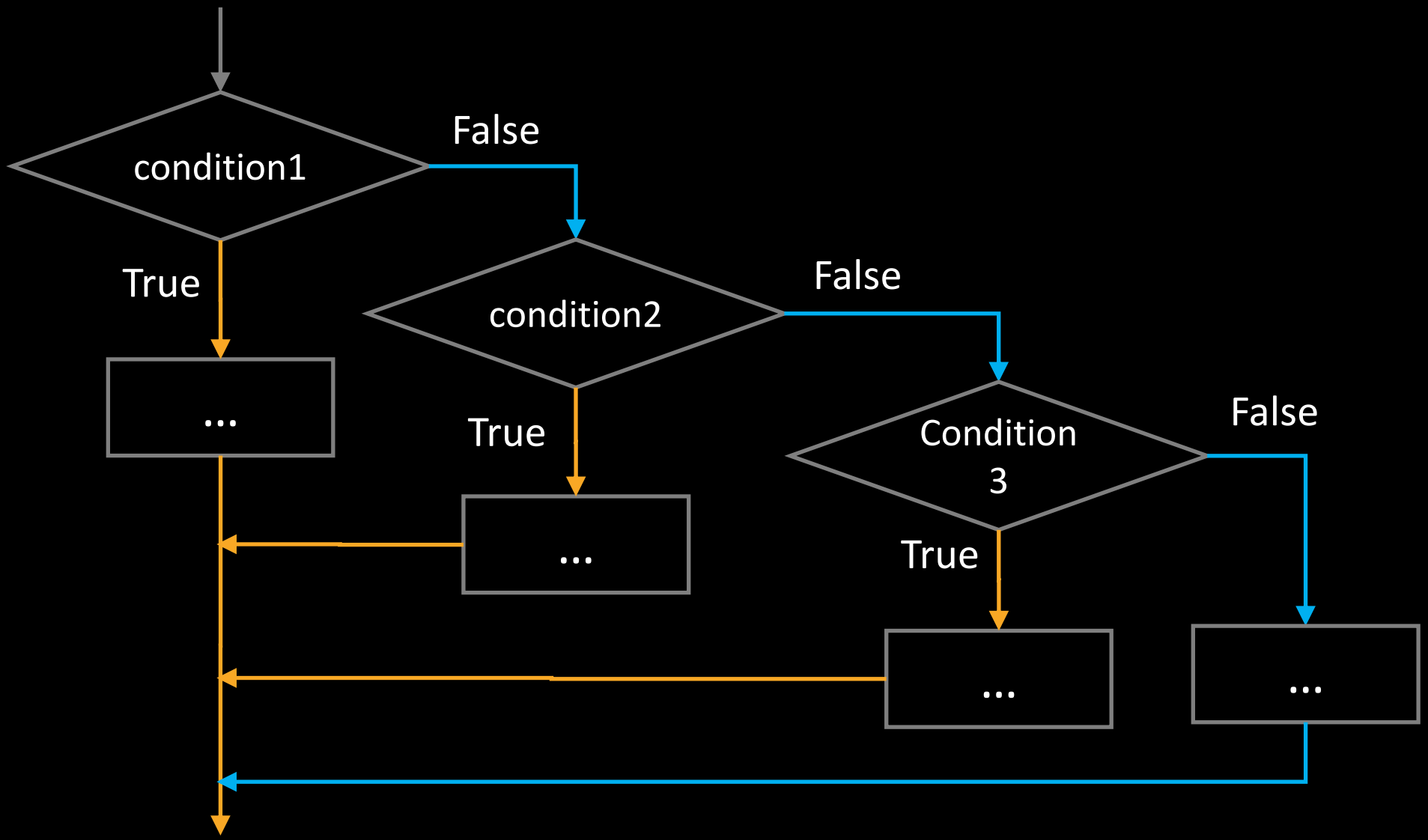
If…else if…else if…else

- **if…else if…else if…else** is multi branch decision making statement.
- If first **if** condition is true then remaining **if** conditions will not be evaluated.
- If first **if** condition is false then second **if** condition will be evaluated and if it is true then remaining **if** conditions will not be evaluated.
- **if…else if…else if…else** is also known as if…else if ladder

Syntax

```
if(condition-1)
  statement-1;
else if(condition-2)
  statement-2;
else
  statement-3;
```

`if…else if…else` ladder flowchart



WAP to print Zero, Positive or Negative Number

Program

```c
#include<stdio.h>
void main()
{
    int a;
    printf("Enter Number:");
    scanf("%d",&a);
    if(a > 0)
        printf("Positive Number");
    else if(a==0)
        printf("Zero");
    else
        printf("Negative Number");
}
```

Output

```
Enter Number:5
Positive Number
```

Output

```
Enter Number:-5
Negative Number
```

Nested if

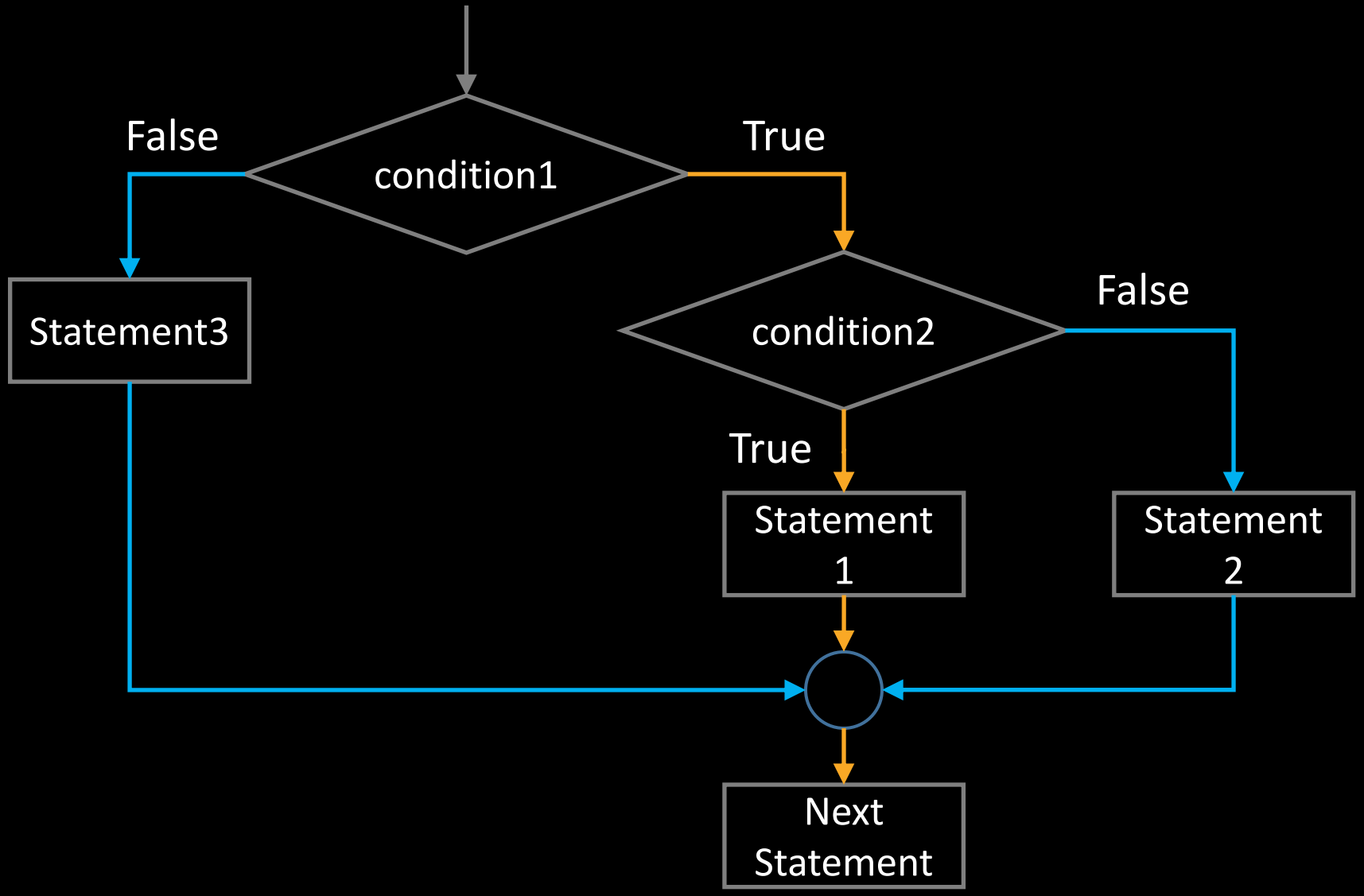
Nested if

- If condition-1 is true then condition-2 is evaluated. If it is true then statement-1 will be executed.
- If condition-1 is false then statement-3 will be executed.

Syntax

```
if(condition-1)
{
    if(condition-2)
    {
      statement-1;
    }
    else
    {
      statement-2;
    }
}
else
{
  statement-3;
}
```

Nested `if` flowchart



WAP to print maximum from given three numbers

Program

```c
void main(){
    int a, b, c;
    printf("Enter Three Numbers:");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
    {
      if(a>c)
        printf("%d is max",a);
      else
        printf("%d is max",c);
   }
   else
   {
     if(b>c)
       printf("%d is max",b);
     else
       printf("%d is max",c);
   }
}
```

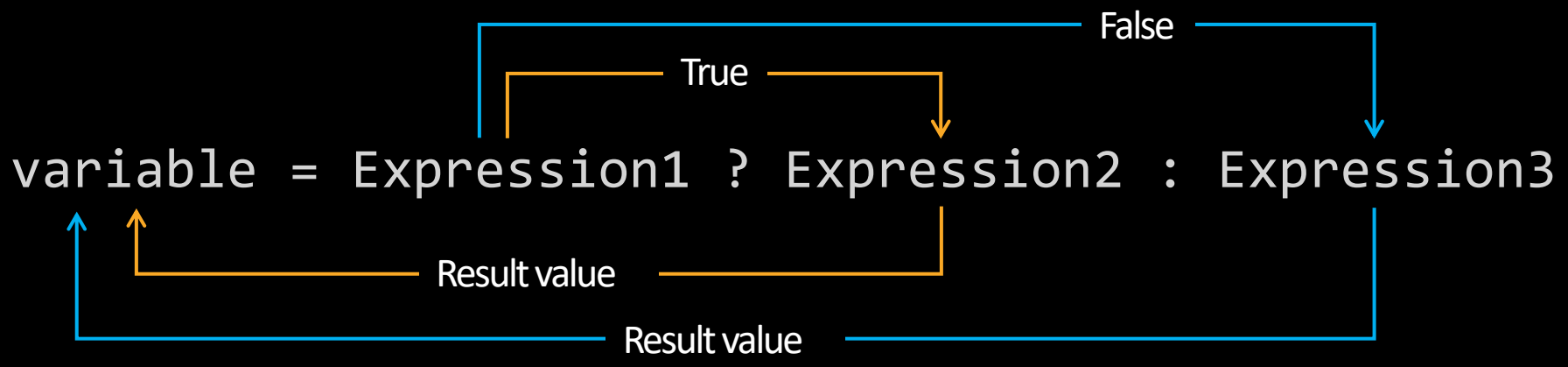
Output

```
Enter Three Numbers:7
5
9
9 is max
```

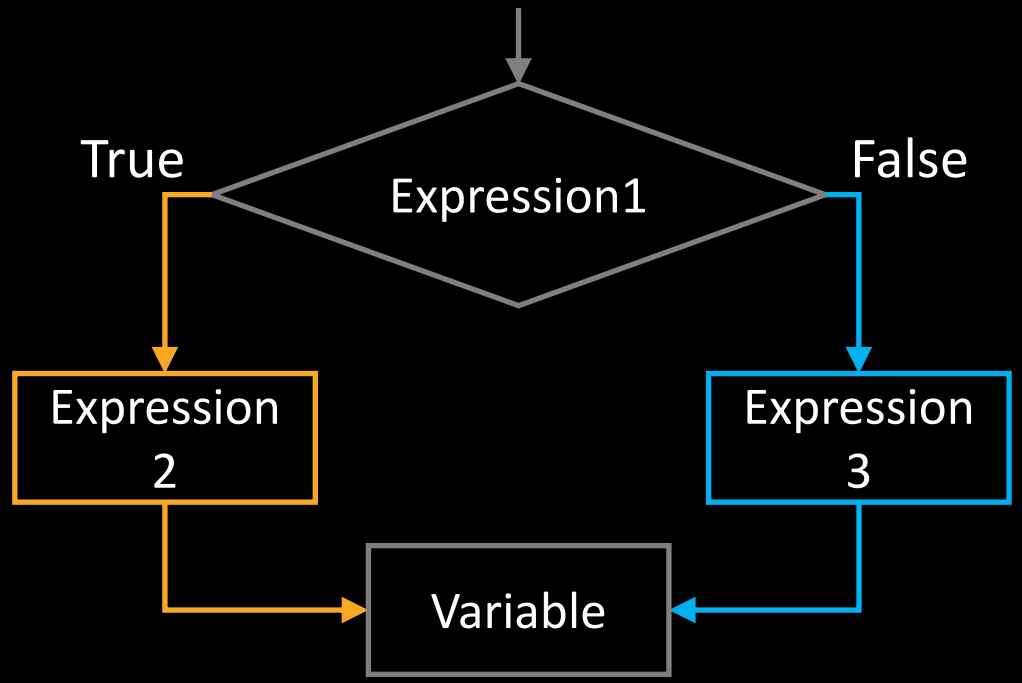
Conditional Operator

? : (Conditional Operator)

- The conditional works operator is similar to the if-else.
- It is also known as a ternary operator.
- It returns first value of expression (before colon(:)) if expression is true and second value of expression if expression is false.



Conditional operator flowchart



- Here, Expression1 is the condition to be evaluated.
- If the condition(Expression1) is True then Expression2 will be executed and the result will be returned.
- Otherwise, if condition(Expression1) is false then Expression3 will be executed and the result will be returned.

WAP to find largest number from given 2 numbers using ? :

Program

```c
#include<stdio.h>
void main()
{
    int a, b, max;
    printf("Enter Two Numbers:");
    scanf("%d%d",&a,&b);
    max = a>b?a:b;
    printf("%d is largest",max);
}
```

Output

```
Enter Two Numbers:4
5
5 is largest
```

switch…case

switch...case

- The switch statement allows to execute one code block among many alternatives.
- It works similar to if...else..if ladder.

Syntax

```
switch (expression)
{
    case constant1:
      // statements
      break;
    case constant2:
      // statements
      break;
    .
    .
    .
    default:
      // default statements
}
```

- The expression is evaluated once and compared with the values of each `case`.
- If there is a match, the corresponding statements after the matching `case` are executed.
- If there is no match, the `default` statements are executed.
- If we do not use `break`, all statements after the matching label are executed.
- The `default` clause inside the `switch` statement is optional.

WAP that asks day number and prints day name using **switch…case**

```c
void main(){
    int day;
    printf("Enter day number(1-7):");
    scanf("%d",&day);
    switch(day)
    {
        case 1:
                printf("Sunday");
                break;
        case 2:
                printf("Monday");
                break;
        case 3:
                printf("Tuesday");
                break;
        case 4:
                printf("Wednesday");
                break;
        case 5:
                printf("Thursday");
                break;
        case 6:
                printf("Friday");
                break;
        case 7:
                printf("Saturday");
                break;
        default:
                printf("Wrong input");
                break;
    }
}
```

Output

```
Enter day number(1-7):5
Thursday
```

Practice programs

1) Write a program to check whether entered character is vowel or not?
2) Write a program to perform Addition, Subtraction, Multiplication and Division of 2 numbers as per user's choice (using if…else/Nested if/Ladder if).
3) Write a program to read marks of five subjects. Calculate percentage and print class accordingly. Fail below 35, Pass Class between 35 to 45, Second Class between 45 to 60, First Class between 60 to 70, Distinction if more than 70.
4) Write a program to find out largest number from given 3 numbers (Conditional operator).
5) Write a program to print number of days in the given month.

Looping

Life is all about Repetition.

We do same thing everyday

What is loop?

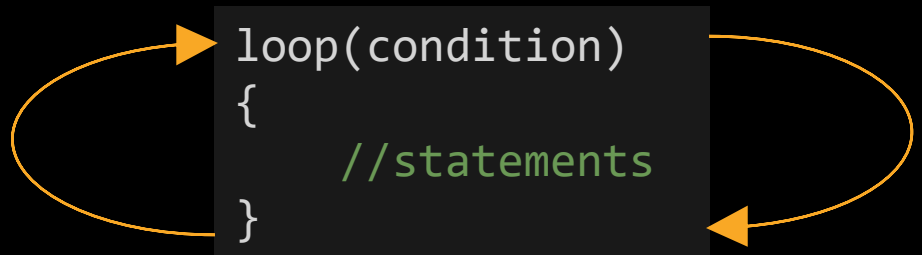
- Loop is used to execute the block of code several times according to the condition given in the loop. It means it executes the same code multiple times.

"Hello" 5

```
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
printf("Hello\n");
```

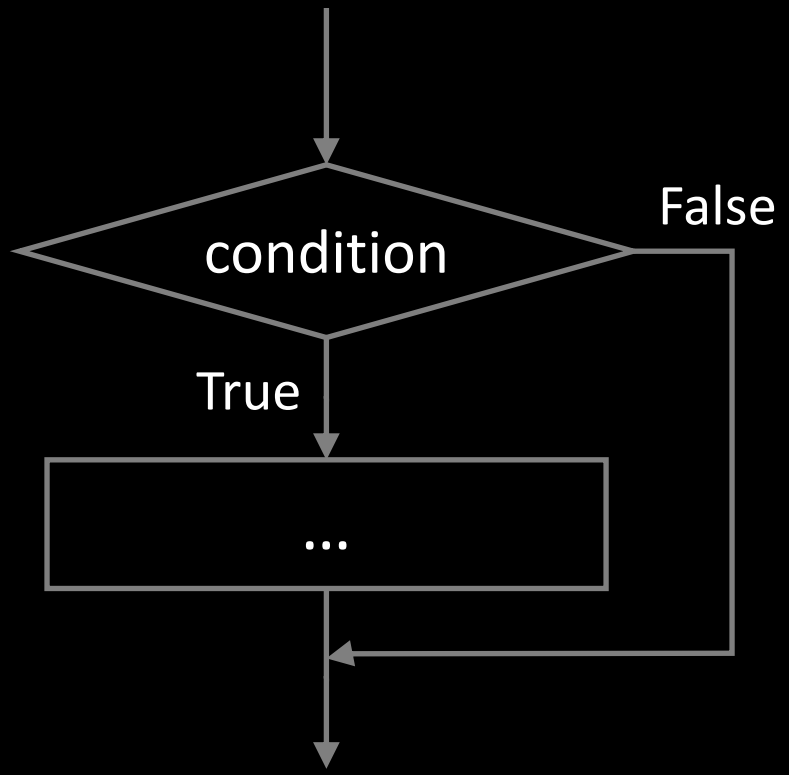
Output

```
Hello
Hello
Hello
Hello
Hello
```



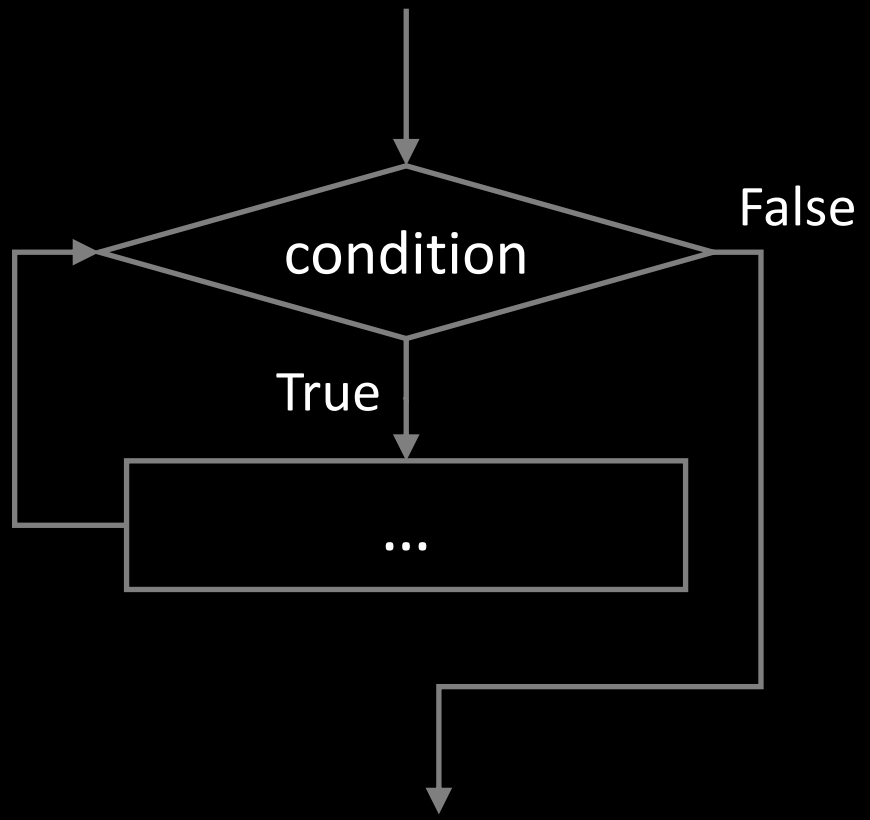
`if` v/s `while`

Flowchart of `if`



v/s

Flowchart of `while`



Looping or Iterative Statements in C

Looping Statements are

Entry Controlled Loop:	`while`, `for`
Exit Controlled Loop:	`do…while`
Virtual Loop:	`goto`

While loop

`While` Loop

- `while` is an entry controlled loop
- Statements inside the body of `while` are repeatedly executed till the condition is true
- `while` is keyword

Syntax

```
while(condition)
{
    // Body of the while
    // true part
}
```

WAP to print 1 to n(while loop)

Program

```c
#include <stdio.h>
void main()
{
    int i,n;
    i=1;
    printf("Enter n:");
    scanf("%d",&n);
    while(i<=n)
    {
        printf("%d\n",i);
        i=i+1;
    }
}
```

Output

```
Enter n:10
1
2
3
4
5
6
7
8
9
10
```

WAP to print multiplication table(while loop)

Program

```c
#include<stdio.h>
void main()
{
    int i=1,n;
    printf("Enter n for multiplication table:");
    scanf("%d",&n);
    while(i<=10)
    {
        printf("%d * %d = %d\n",n,i,n*i);
        i=i+1;
    }
}
```

Output

```
Enter n for multiplication table:5
5 * 1  = 5
5 * 2  = 10
5 * 3  = 15
5 * 4  = 20
5 * 5  = 25
5 * 6  = 30
5 * 7  = 35
5 * 8  = 40
5 * 9  = 45
5 * 10 = 50
```

WAP to Sum of 5 numbers entered by user(while loop)

Program

```c
#include<stdio.h>
void main()
{
    int sum=0, i=1,n;
    while(i<=5)
    {
        printf("Enter a number=");
        scanf("%d",&n);
        sum=sum+n;
        i=i+1;
    }
    printf("Sum is=%d",sum);
}
```

Output

```
Enter a number=10
Enter a number=20
Enter a number=30
Enter a number=40
Enter a number=50
Sum is=150
```

Syntax and Logic

Swimming Rules

1. Breath control
2. Kicking legs
3. Back stroke with arms
4. Front stroke with arms
5. Crawling in water

To Swim

Syntax

```
while(condition)
{
    // Body of the while
    // true part
}
```

Logic

```
int i = 1;
while (i <= 5)
{
    printf("%d\n", i);
    i=i+1;
}
```

How to build logic? Step-1

Step 1: Understand the problem statement

- e.g. Write a program to find factors of a number.
- Run following questions through mind

- What is the factor of a number?
  - Factor is a number that divides another number evenly with no remainder.
  - For example, 1,2,3,4,6,12 are factors of 12.
- How many variables needed? What should be their data types?(Inputs/Outputs)
  - To get number from user we need variable **`n`**.
  - Now we need to divide **`n`** with 1,2,3,...,n. For this we will declare a loop variable **`i`** initialized as 1.
  - Both variables should be of `integer` data type.
- What control structure you require?
  - First we need `a loop` to divide **`n`** by 1,2,3,…,n, loop will start from 1 and ends at **`n`**.
  - Inside loop we need `if structure` to check **`n%i==0`** (Number n is evenly divisible by **`i`** or not).

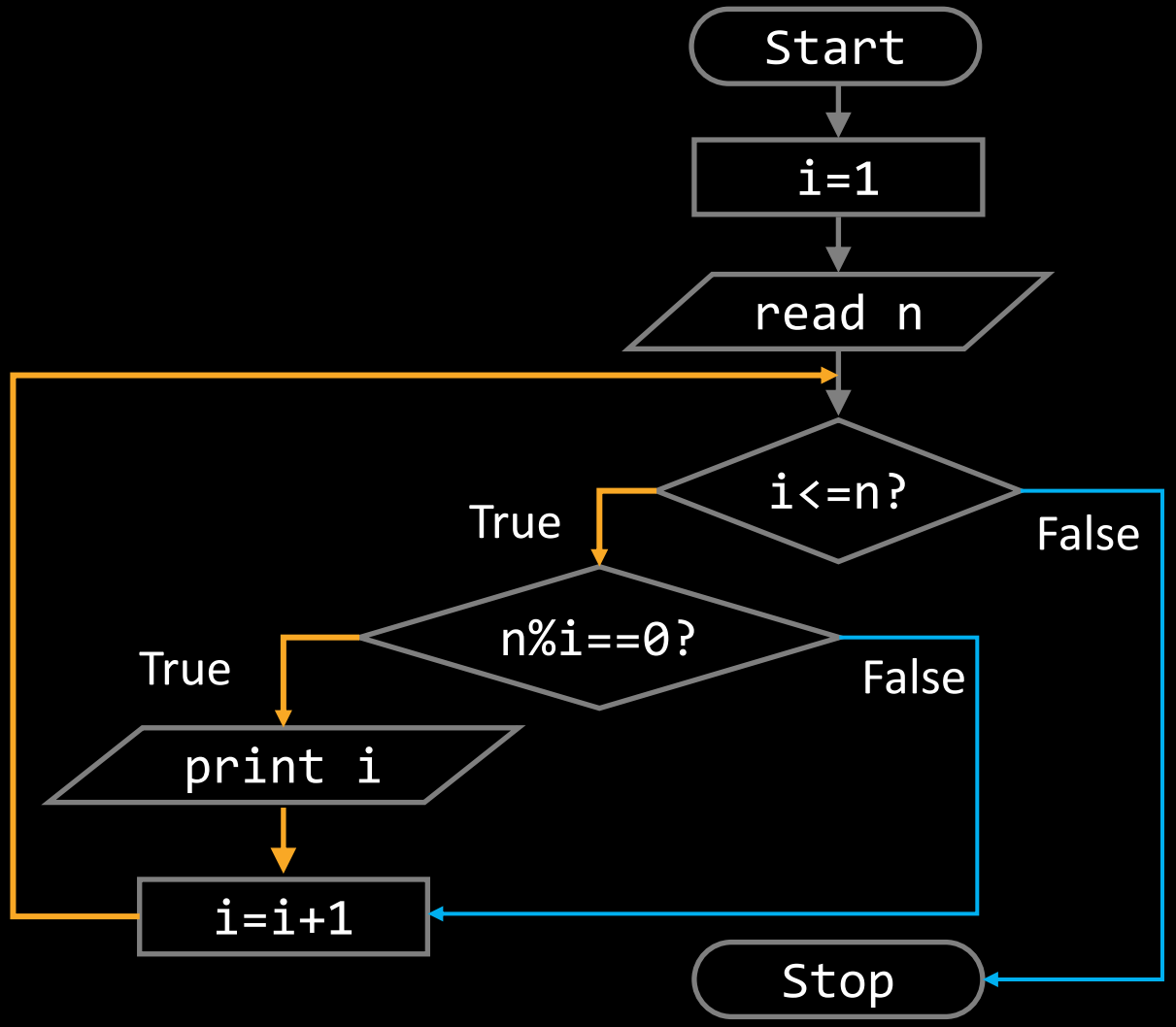
How to build logic? Step-2

Step 2: Think for 1 or 2 examples

- Consider `n=6`, now take `i=1`
  - `6%1==0`, TRUE; So, 1 is factor of 6
  - `6%2==0`, TRUE; So, 2 is factor of 6
  - `6%3==0`, TRUE; So, 3 is factor of 6
  - `6%4==2`, FALSE; S0, 4 is not factor of 6
  - `6%5==1`, FALSE; S0, 5 is not factor of 6
  - `6%6==0`, TRUE; S0, 6 is factor of 6
- From this we can infer that loop variable **`i`** starts with `1` and incremented by one for next iteration then ends at value **`n`**.
- Consider `n=10`, factors are `1,2,5,10`
- Consider `n=11`, factor is `1,11`
- From this we can infer that 1 and number itself are always factors of any number **`n`**.

How to build logic? Step-3

Step 3: Draw flowchart/steps on paper or in mind



Steps

```
Step 1: Start
Step 2: Declare variables n,i
Step 3: Initialize variable
        i ← 1
Step 4: Read value of n
Step 5: Repeat the steps until i = n
     5.1: if n%i == 0
             Display i
     5.2: i=i+1
Step 7: Stop
```

How to build logic? Step-4

Step 4: Writing Pseudo-code

- Pseudo-code is an informal way to express the design of a computer program or an algorithm.
- It does not require any strict programming language syntax.

Pseudo-code

```
Initialize i=1 integer
Declare n as integer
Input n
while i<n
    if n%i
        print i
    end if
    increment i=i+1
end while
```

WAP to find factors of a number(while loop)

Program

```c
#include <stdio.h>
void main()
{
    int i=1,n;
    printf("Enter n to find factors=");
    scanf("%d",&n);
    while(i<=n)
    {
        if(n%i==0)
            printf("%d,",i);
        i=i+1;
    }
}
```

Output

```
Enter n to find factors=12
1,2,3,4,6,12,
```

WAP to print reverse a number(while loop)

Program

```c
#include <stdio.h>
void main()
{
    int n;
    printf("Enter a number=");
    scanf("%d",&n);
    while(n!=0)
    {
        printf("%d",n%10);
        n=n/10;
    }
}
```

Output

```
Enter a number=1234
4321
```

WAP to check given number is perfect or not(while loop)

```c
void main(){
    int i=1,n,sum=0;
    printf("Enter a number:");
    scanf("%d",&n);
    while(i<n)
    {
        if(n%i==0)
        {
            printf("%d+",i);
            sum=sum+i;
        }
        i=i+1;
    }
    printf("=%d",sum);
    if(sum==n)
        printf("\n%d is a perfect number",n);
    else
        printf("\n%d is not a perfect number",n);
}
```

Output

```
Enter a number:6
1+2+3=6
6 is a perfect number
```

Output

```
Enter a number:8
1+2+4+=7
8 is not a perfect number
```

Output

```
Enter a number:496
1+2+4+8+16+31+62+124+248+=496
496 is a perfect number
```

WAP to check given number is prime or not(while loop)

```c
void main()
{
    int n, i=2,flag=0;
    printf("Enter a number:");
    scanf("%d",&n);
    while(i<=n/2)
    {
        if(n%i==0)
        {
            flag=1;
            break;
        }
        i++;
    }
    if (flag==0)
        printf("%d is a prime number",n);
    else
        printf("%d is not a prime number",n);
}
```

Output

```
Enter a number:7
7 is a prime number
```

Output

```
Enter a number:9
9 is not a prime number
```

for loop

for Loop

- for is an entry controlled loop
- Statements inside the body of for are repeatedly executed till the condition is true
- for is keyword

Syntax

```
for (initialization; condition; updateStatement)
{
    // statements
}
```

- The initialization statement is executed only once.
- Then, the condition is evaluated. If the condition is false, the for loop is terminated.
- If the condition is true, statements inside the body of for loop are executed, and the update statement is updated.
- Again the condition is evaluated.

WAP to print numbers 1 to n (for loop)

Program

```c
#include<stdio.h>
void main()
{
    int i,n;
    printf("Enter a number:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("%d\n",i);
    }
}
```

Output

```
Enter a number:5
1
2
3
4
5
```

WAP to find factors of a number (for loop)

Program

```c
#include <stdio.h>
void main()
{
    int i,n;
    printf("Enter n to find factors=");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        if(n%i==0)
            printf("%d,",i);
    }
}
```

Output

```
Enter n to find factors=12
1,2,3,4,6,12,
```

WAP to check given number is perfect or not(for loop)

```c
void main(){
    int i,n,sum=0;
    printf("Enter a number:");
    scanf("%d",&n);
    for(i=1;i<n;i++)
    {
        if(n%i==0)
        {
            printf("%d+",i);
            sum=sum+i;
        }
    }
    printf("=%d",sum);
    if(sum==n)
        printf("\n%d is a perfect number",n);
    else
        printf("\n%d is not a perfect number",n);
}
```

Output

```
Enter a number:6
1+2+3=6
6 is a perfect number
```

Output

```
Enter a number:8
1+2+4+=7
8 is not a perfect number
```

Output

```
Enter a number:496
1+2+4+8+16+31+62+124+248+=496
496 is a perfect number
```

do while loop

`do while` Loop

- `do while` is an exit controlled loop.
- Statements inside the body of `do while` are repeatedly executed till the condition is true.
- `Do` and `while` are keywords.

Syntax

```
do
{
    // statement
}
while (condition);
```

- Loop body will be executed first, and then condition is checked.
- If the condition is true, the body of the loop is executed again and the condition is evaluated.
- This process goes on until the condition becomes false.
- If the condition is false, the loop ends.

WAP to print Odd numbers between 1 to n(do while loop)

Program

```c
void main()
{
   int i=1,n;
   printf("Enter a number:");
   scanf("%d",&n);
   do
   {
      if(i%2!=0)
      {
         printf("%d,",i);
      }
      i=i+1;
   }
   while(i<=n);
}
```

Output

```
Enter a number:5
1,3,5
```

WAP to find factors of a number(do while loop)

Program

```c
void main()
{
   int i=1,n;
   printf("Enter a number:");
   scanf("%d",&n);
   do
   {
      if(n%i==0)
      {
         printf("%d,",i);
      }
      i=i+1;
   }
   while(i<=n);
}
```

Output

```
Enter a number:6
1,2,3,6,
```

WAP to print reverse a number(do while loop)

Program

```c
void main()
{
    int n;
    printf("Enter a number:");
    scanf("%d",&n);
    do
    {
        printf("%d",n%10);
        n=n/10;
    }
    while(n!=0);
}
```

Output

```
Enter a number=1234
4321
```

goto statement

`goto` Statement

- `goto` is an virtual loop
- The `goto` statement allows us to transfer control of the program to the specified `label`.
- `goto` is keyword

Syntax

```
goto label;
.
.
.
label:
```

Syntax

```
label:
.
.
.
goto label;
```

- The `label` is an identifier. When the `goto` statement is encountered, the control of the program jumps to `label:` and starts executing the code.

WAP to print Odd numbers between 1 to n(goto)

Program

```c
void main()
{
    int i=1,n;
    printf("Enter a number:");
    scanf("%d",&n);
    odd:
    if(i%2!=0)
    {
        printf("%d,",i);
    }
    i=i+1;
    if(i<=n)
    {
        goto odd;
    }
}
```

Output

```
Enter a number:5
1,3,5
```

WAP to find factors of a number(goto)

Program

```c
void main()
{
    int i=1,n;
    printf("Enter a number:");
    scanf("%d",&n);
    odd:
    if(n%i==0)
    {
        printf("%d,",i);
    }
    i=i+1;
    if(i<=n)
    {
        goto odd;
    }
}
```

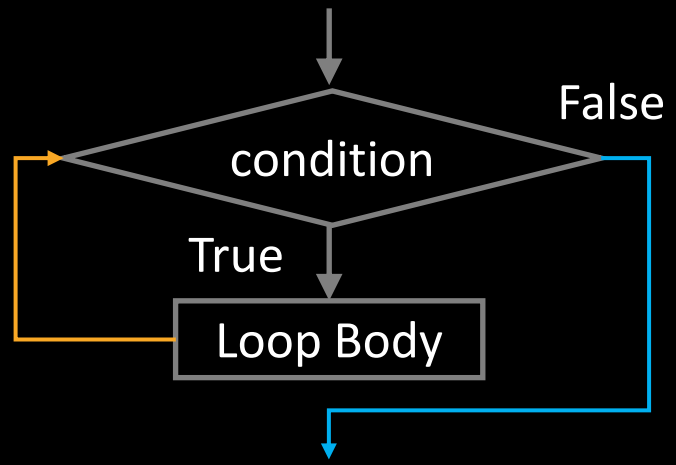
Output

```
Enter a number:6
1,2,3,6,
```

Types of loops

Entry Control Loop

```
int i=1;
while(i<=10)
{
    printf("%d",i++);
}
```

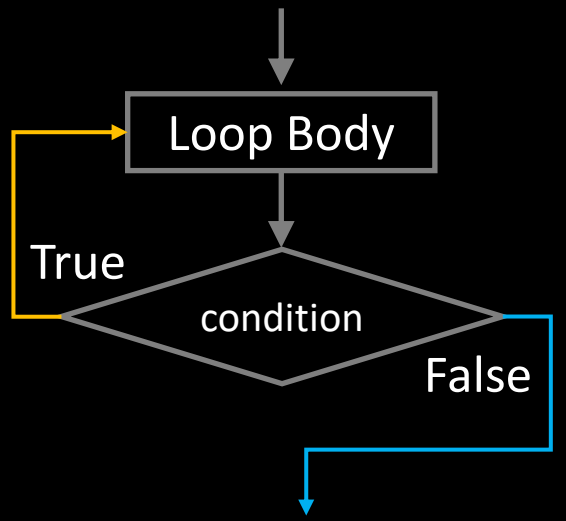


Entry Control Loop

```
int i;
for(i=1;i<=10;i++)
{
    printf("%d",i);
}
```

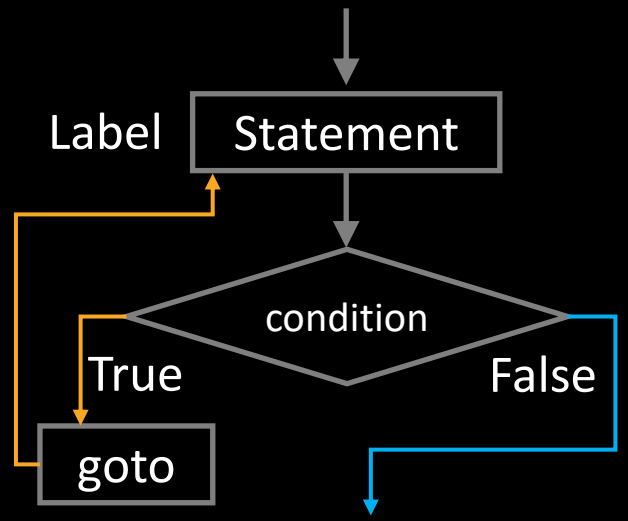
Exit Control Loop

```
int i=1;
do
{
    printf("%d",i++);
}
while(i<=10);
```



Virtual Loop

```
int i=1;
   labelprint:
    printf("%d",i++);
   if(i<=10)
     goto labelprint;
```



Pattern

Always detect pattern in pattern

Pattern

There are important points to note in pattern

1. Determine, how many rows?
2. Determine, how many numbers/characters/columns in a row?
3. Determine, Increment/Decrement among the number of rows.
4. Determine, starting in each row

```
1
11
111
1111
11111
```

```
1
12
123
1234
12345
```

```
1
23
456
78910
```

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

WAP to print given pattern (nested loop)

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

No. of rows: 5

No. of characters
Row-1: *
Row-2: **
Row-3: ***
Row-4: ****
Row-5: *****

Inner loop: Increment
Outer loop: Increment

Starting: *

Program

```c
void main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1; j<=i; j++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

WAP to print given pattern (nested loop)

```
1
12
123
1234
12345
```

No. of rows: 5

No. of values
Row-1: 1
Row-2: 12
Row-3: 123
Row-4: 1234
Row-5: 12345

Inner loop: Increment
Outer loop: Increment

Starting: 1

Program

```c
void main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1; j<=i; j++)
        {
            printf("%d",j);
        }
        printf("\n");
    }
}
```

WAP to print given pattern (nested loop)

```
5
54
543
5432
54321
```

No. of rows: 5

No. of values
Row-1: 5
Row-2: 54
Row-3: 543
Row-4: 5432
Row-5: 54321

Inner loop: Decrement
Outer loop: Decrement/Increment

Starting: 5

Program

```c
void main()
{
    int i,j;
    for(i=5;i>0;i--)
    {
        for(j=5; j>=i ; j--)
        {
            printf("%d",j);
        }
        printf("\n");
    }
}
```

WAP to print given pattern (nested loop)

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

No. of rows: 5

No. of values
Row-1: ----*
Row-2: ---**
Row-3: --***
Row-4: -****
Row-5: *****

Inner loop: Decrement
Outer loop: Decrement/Increment

Starting: -(space)
Ending: *

Program

```
1  void main()
2  {
3      int i,j,k;
4      for(i=1;i<=5;i++)
5      {
6          for(k=5;k>i;k--)
7          {
8              printf(" ");
9          }
10         for(j=1;j<=i;j++)
11         {
12             printf("*");
13         }
14         printf("\n");
15     }
16 }
```

First we need to print 4 spaces before printing *

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

After printing spaces this inner loop prints *

Practice programs

1) Write a program to find sum of first N odd numbers. Ex. 1+3+5+7+……..+N
2) Write a program to find 1+1/2+1/3+1/4+....+1/n.
3) Write a program to print all Armstrong numbers in a given range. For example 153 = 1^3 + 5^3 + 3^3. So, 153 is Armstrong number.
4) Write a program to print given number in reverse order
5) Write a program to check whether a given string is palindrome or not.
6) Write a program to print Multiplication Table up to n.

1	2	3	4	5	6	7	.
2	4	6	8	10	12	14	.
3	6	9	12	15	18	21	.
4	8	12	16	20	24	28	.
5	10	15	20	25	30	35	.
.	.	.	.	.	.	.	.

7) Construct C programs to print the following patterns using loop statement.

```
1
22
333
4444
55555
```

```
*
# #
* * *
# # # #
* * * * *
```

```
1
0 1
1 0 1
0 1 0 1
```

```
    1
  2   2
3   3   3
4   4   4   4
```

```
    1
  A   B
2   3   4
C   D   E   F
```

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

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