Programming Fundamentals Lab

Lab # 07

Relational Operators, Logical Operators, If, I-else, Multiple If, If-else-if

Instructor: Fariba Laiq

Email: fariba.laiq@nu.edu.pk

Course Code: CL1002

Semester Fall 2023

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar Campus

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;
    printf("a: %d b: %d c: %d\n", a, b, c);
    printf("%d == %d is %d \n", a, b, a == b);
printf("%d == %d is %d \n", a, c, a == c);
printf("%d > %d is %d \n", a, b, a > b);
printf("%d > %d is %d \n", a, c, a > c);
printf("%d < %d is %d \n", a, b, a < b);
printf("%d < %d is %d \n", a, c, a < c);
printf("%d != %d is %d \n", a, b, a != b);
printf("%d != %d is %d \n", a, c, a != c);
printf("%d >= %d is %d \n", a, b, a >= b);
printf("%d >= %d is %d \n", a, c, a >= c);
printf("%d <= %d is %d \n", a, b, a <= b);
printf("%d <= %d is %d \n", a, c, a <= c);
return 0;
}
```

Output:

```
a: 5 b: 5 c: 10
5 == 5 is 1
5 == 10 is 0
5 > 5 is 0
5 > 10 is 0
5 < 5 is 0
5 < 10 is 1
5 != 5 is 0
5 != 10 is 1
5 >= 5 is 1
```

5 >= 10 is 0

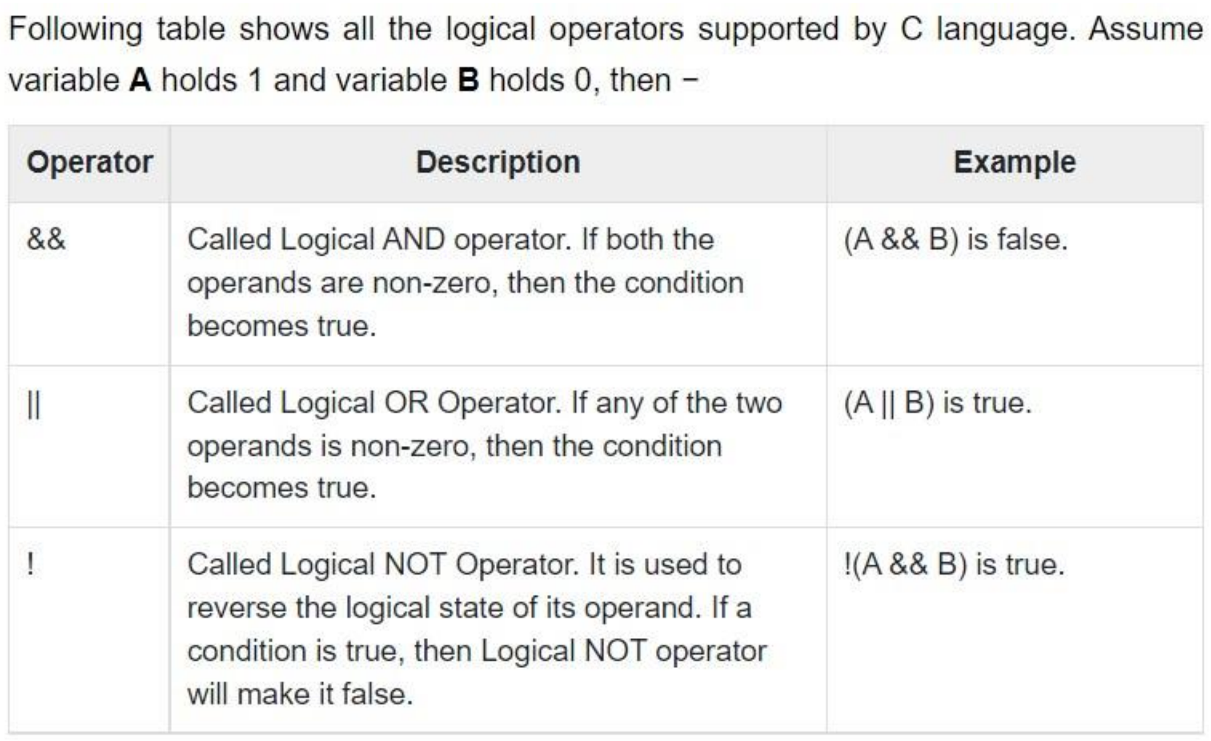
5 <= 5 is 1

5 <= 10 is 1

Types of Logical Operators in C

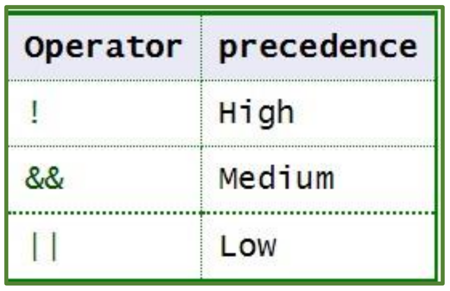
We have three major logical operators in the C language:

- Logical NOT (!)
- Logical OR (||)
- Logical AND (&&)



Following table shows all the logical operators supported by C language. Assume variable **A** holds 1 and variable **B** holds 0, then −

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.



Operator	precedence
!	High
&&	Medium
||	Low

Example Code:

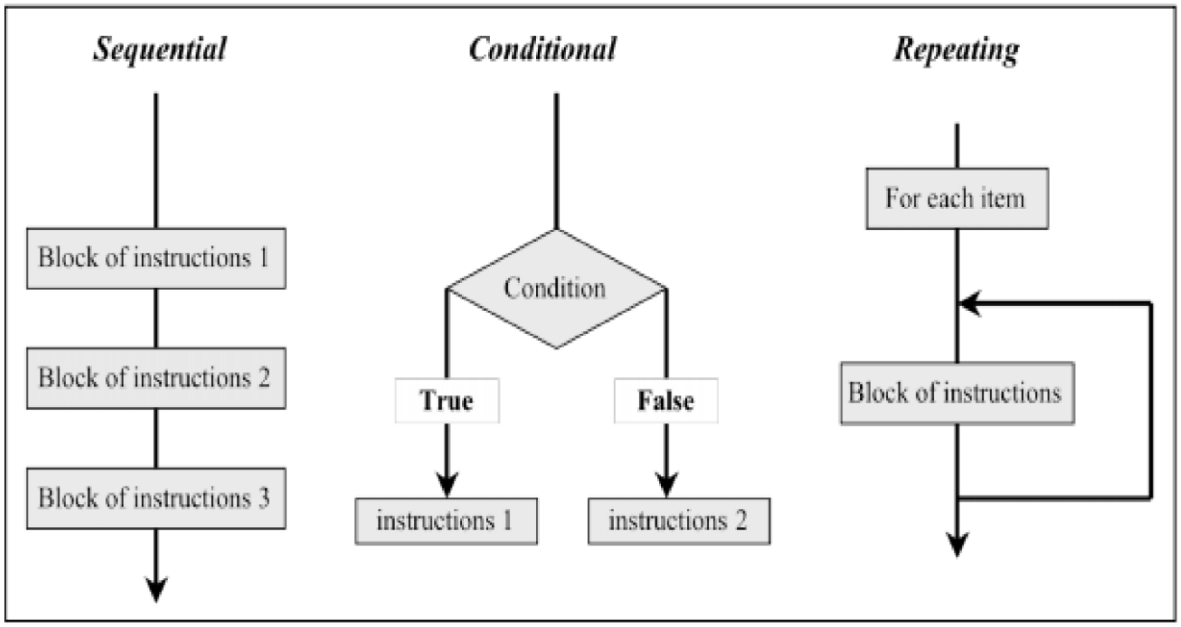
```
#include <stdio.h>
int main()
{
        int a = 5, b = 5, c = 10, result;
        printf("a: %d b: %d c: %d\n", a, b, c);
        result = (a == b) && (c > b);
        printf("(a == b) && (c > b) is %d \n", result);
        result = (a == b) && (c < b);
        printf("(a == b) && (c < b) is %d \n", result);
        result = (a == b) || (c < b);
        printf("(a == b) || (c < b) is %d \n", result);
        result = (a != b) || (c < b);
        printf("(a != b) || (c < b) is %d \n", result);
        result = !(a != b);
        printf("!(a != b) is %d \n", result);
        result = !(a == b);
        printf("!(a == b) is %d \n", result);
        return 0;
}
```

Output:

```
a: 5 b: 5 c: 10
(a == b) && (c > b) is 1
(a == b) && (c < b) is 0
(a == b) || (c < b) is 1
(a != b) || (c < b) is 0
!(a != b) is 1
!(a == b) is 0
```

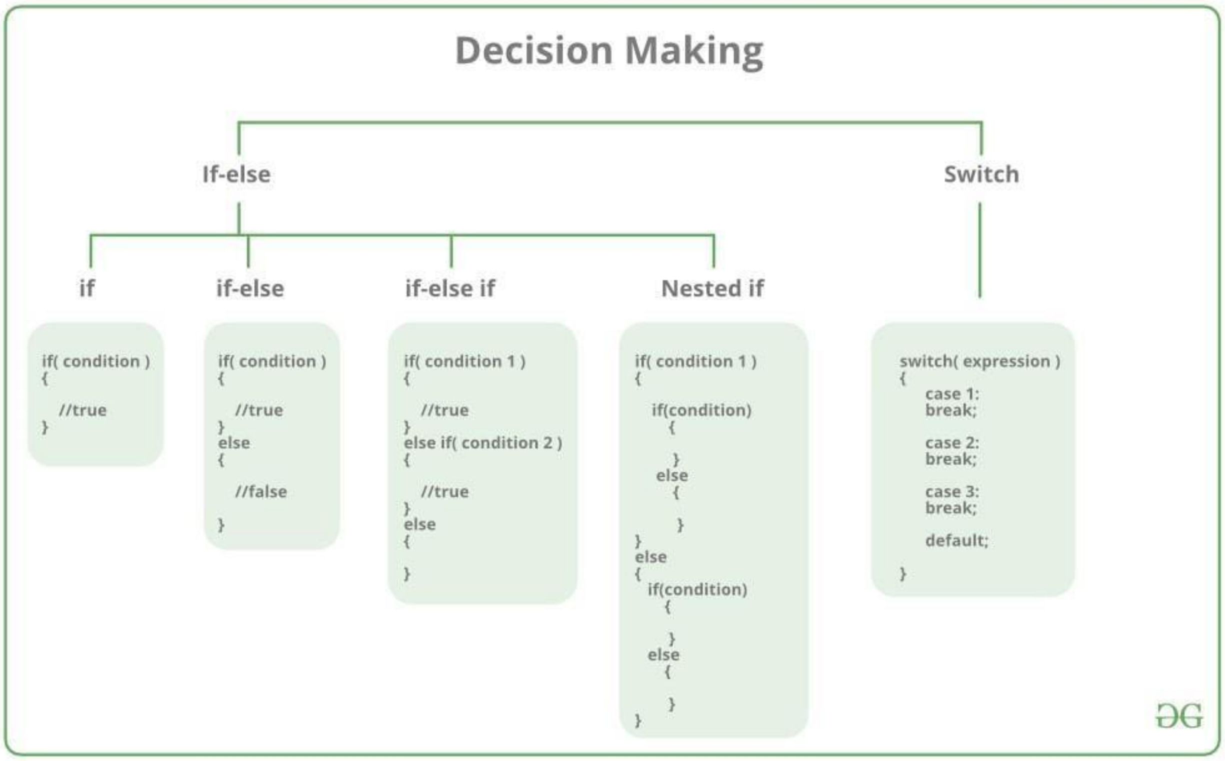
Control Structures in C

Algorithms require two important control structures: iteration(repeating) and selection(conditional). Both are supported by C in various forms. The programmer can choose the statement that is most useful for the given circumstance.



Selection Statements / Decision Making

- There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions, we will execute the next block of code.
- For example, in C if x occurs then execute y else execute z. There can also be multiple conditions like in C if x occurs then execute p, else if condition y occurs execute q, else execute r. This condition of C else-if is one of the many ways of importing multiple conditions.
- Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.
- Following is the general form of a typical decision-making structure found in most of the programming languages.



```
if( condition )
{
   //true
}
```

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

```
if( condition 1 )
{
   //true
}
else if( condition 2 )
{
   //true
}
else
{
}
```

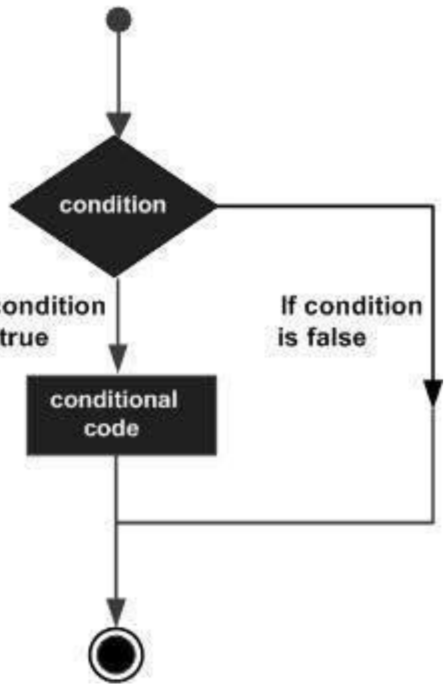
```
if( condition 1 )
{
   if(condition)
     {
     }
    else
     {
     }
}
else
{
  if(condition)
    {
    }
  else
    {
    }
}
```

```
switch( expression )
{
     case 1:
     break;

     case 2:
     break;

     case 3:
     break;

     default;
}
```



1. If Statement in C (One-Way Decision)

if statement is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

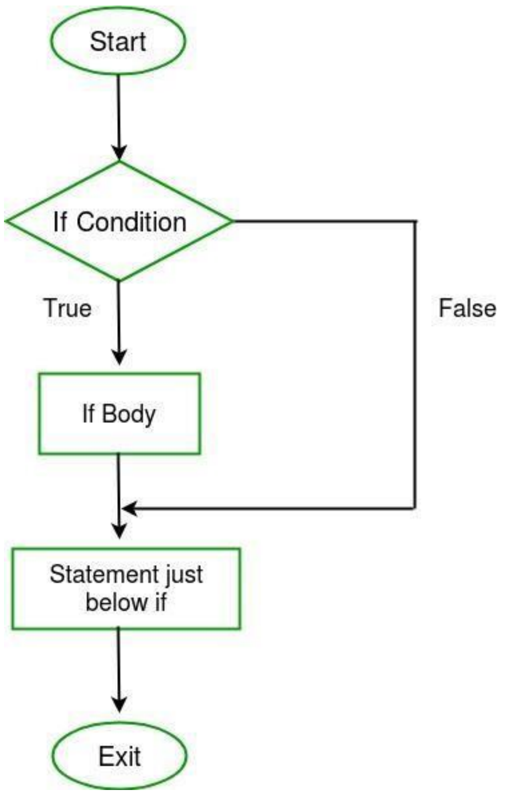
Syntax:

```
if(condition

)


{
   // Statements to execute if
   // condition is true
}
```

Flowchart of if statement



Example 1

```
// C program to illustrate If statement

#include <stdio.h>

int main()

{

      int i;

       printf("Enter an integer: ");

       scanf("%d", &i);

      if (i > 0) {

        printf("\ni is positive ");

      }
}
```

Output

```
Enter an integer: 4

i is positive
```

If the condition present in the if statement is false. So, the block below the if statement is not executed.

2. if-else in C (Two-Way Decision)

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the C else statement. We can use the else statement with the if statement to execute a block of code when the condition is false.

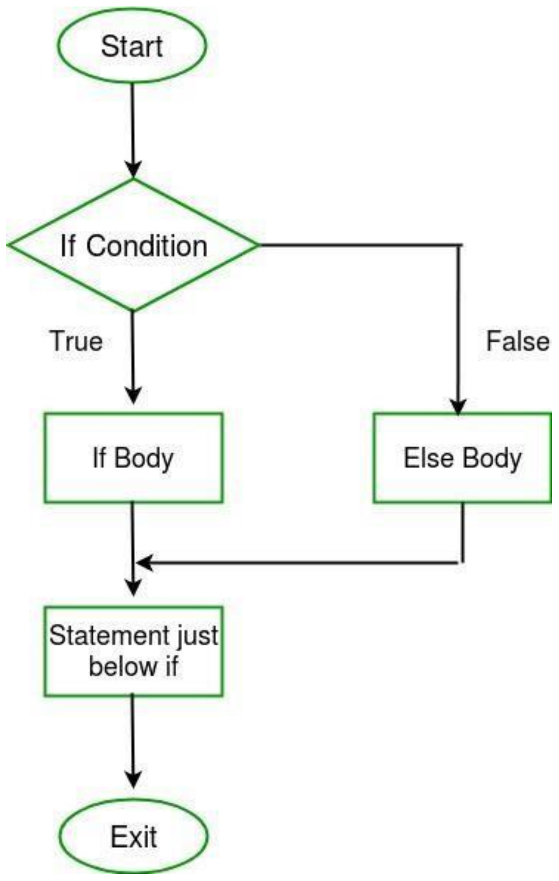
Syntax:

```
if (condition)

{

    // Executes this block if
    // condition is true
else
{
    // Executes this block if
    // condition is false
}
```

Flowchart of if-else



Example 2

```
#include <stdio.h>
int main() {
        int i = 20;
        if (i >= 0) {
                printf("i is positive or zero");
        } else {
                printf("i is negative");
        }
        return 0;
}
```

Output

```
i is positive or zero
```

The block of code following the else statement is executed as the condition present in the if statement is false.

```
#include <stdio.h>
int main()
{
        int age = 11;
        if(age>=13 && age<=19) {
                printf("You are teenager!\n");
        }  else {
                printf("you are not a teenager\n");
        }
        return 0;
}
```

Output

```
You are not a teenager!
```

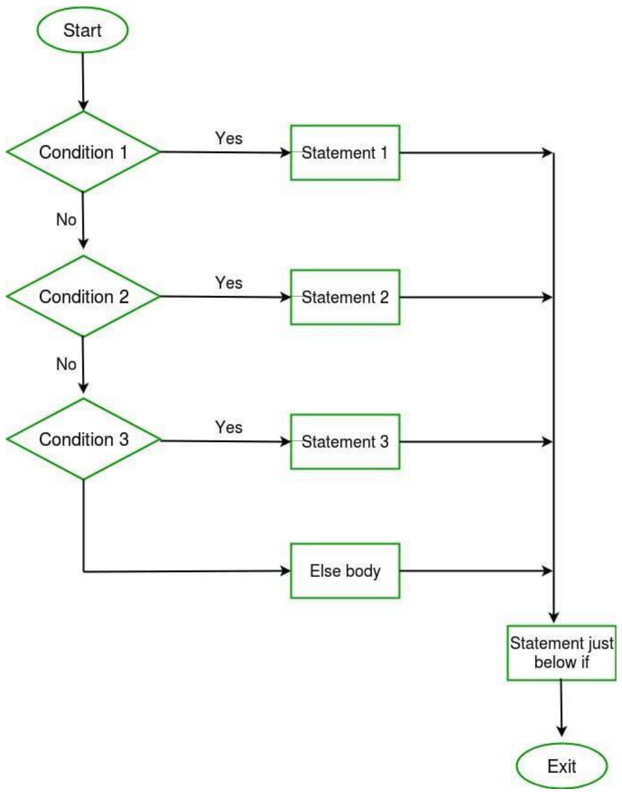
3. if-else-if ladder in C (Multi Way Decisions)

Here, a user can decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:

```
if (condition)
{
     Statements;
}
else if (condition)
{
     Statements;
}
else
{
      Statements to execute if   none of the above
conditions are true; }
```

Flowchart of if-else-if



Example 3:

```
#include <stdio.h>
int main()
{
        int i = -10;
        if (i > 10) {
                printf("i is Positive");
        } else if (i == 0) {
                printf("i is zero");
        } else {
                printf("i is negative");
        }
}
```

Output:

```
i is negative
```

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Example 4:

Example 5

```
#include <stdio.h>
int main()
{
        char c;
        printf("Enter a char:");
        scanf("%c", &c);
        if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u') {
                printf("\nYou entered a vowel");
        } else {
                printf("\nYou entered a consonant");
        }
        return 0;
}
```

Output

```
Enter a char: a

You entered a vowel
```

4. Multiple If's:

Where we have a lot of choices, and more than one condition can be true so it's better to use multiple if's. So that all the conditions should be checked.

```
#include <stdio.h>
int main()
{
	int age=18;
	if(age>=13 && age<=19)
	{
		printf("You are a teenager");
	}
	if(age>=18)
	{
		printf("\nYou are also eligible for voting");
	}
}
```

Output:

```
You are a teenager
You are also eligible for voting
```