Programming Languages

CENTRAL UNIVERSITY

FAITH • INTEGRITY • EXCELLENCE

PRESENTED BY:
DR. K. KISSI MIREKU

Computer Programming – Keywords

CENTRAL UNIVERSITY

FAITH • INTEGRITY • EXCELLENCE

Computer Programming - Keywords

- **int**, **long**, and **float**, are some of the many other keywords supported by C programming language which are used for different purpose.
- Different programming languages provide different set of reserved keywords, but there is one important & common rule in all the programming languages that we cannot use a reserved keyword to name our variables.
    - *which means we cannot name our variable like **int** or **float** rather these keywords can only be used to specify a variable data type.*
- Keywords are **predefined, reserved words used in programming that have special meanings to the compiler**. Keywords are part of the syntax and they cannot be used as an identifier.

Computer Programming - Keywords

For example, if you will try to use any reserved keyword for the purpose of variable name, then you will get a syntax error.

```c
#include <stdio.h>

int main() {
   int float;
   float = 10;

   printf( "Value of float = %d\n", float);
}
```

When you compile the above program, it produces the following error −

```
main.c: In function 'main':
main.c:5:8: error: two or more data types in declaration specifiers
   int float;
......
```

Let's now give a proper name to our integer variable, then the above program should compile and execute successfully −

```c
#include <stdio.h>

int main() {
   int count;
   count = 10;

   printf( "Value of count = %d\n", count);
```

Computer Programming - keywords

C Programming Reserved Keywords

- A table with keywords supported by C Programming language –

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_Packed
double			

Computer Programming - keywords

Java Programming Reserved Keywords

- A table with keywords supported by Java Programming language

abstract	assert	boolean	break	volatile
byte	case	catch	char	while
class	const	continue	default	void
do	double	else	enum	try
extends	final	finally	float	transient
for	goto	if	implements	throws
import	instanceof	int	interface	switch
long	native	new	package	synchronized
private	protected	public	return	this
short	static	strictfp	super	throw

Computer Programming - keywords

Python Programming Reserved Keywords

- A table with keywords supported by Python Programming

and	exec	not	else	is
assert	finally	or	except	lambda
break	for	pass	with	yield
class	from	print	del	import
continue	global	raise	elif	in
def	if	return	Try	while

We know you cannot memorize all these keywords, but we have listed them down for your reference purpose and to explain the concept of **reserved keywords**. So just be careful while giving a name to your variable, you should not use any reserved keyword for that programming language.

Computer Programming – Operators

CENTRAL UNIVERSITY

FAITH • INTEGRITY • EXCELLENCE

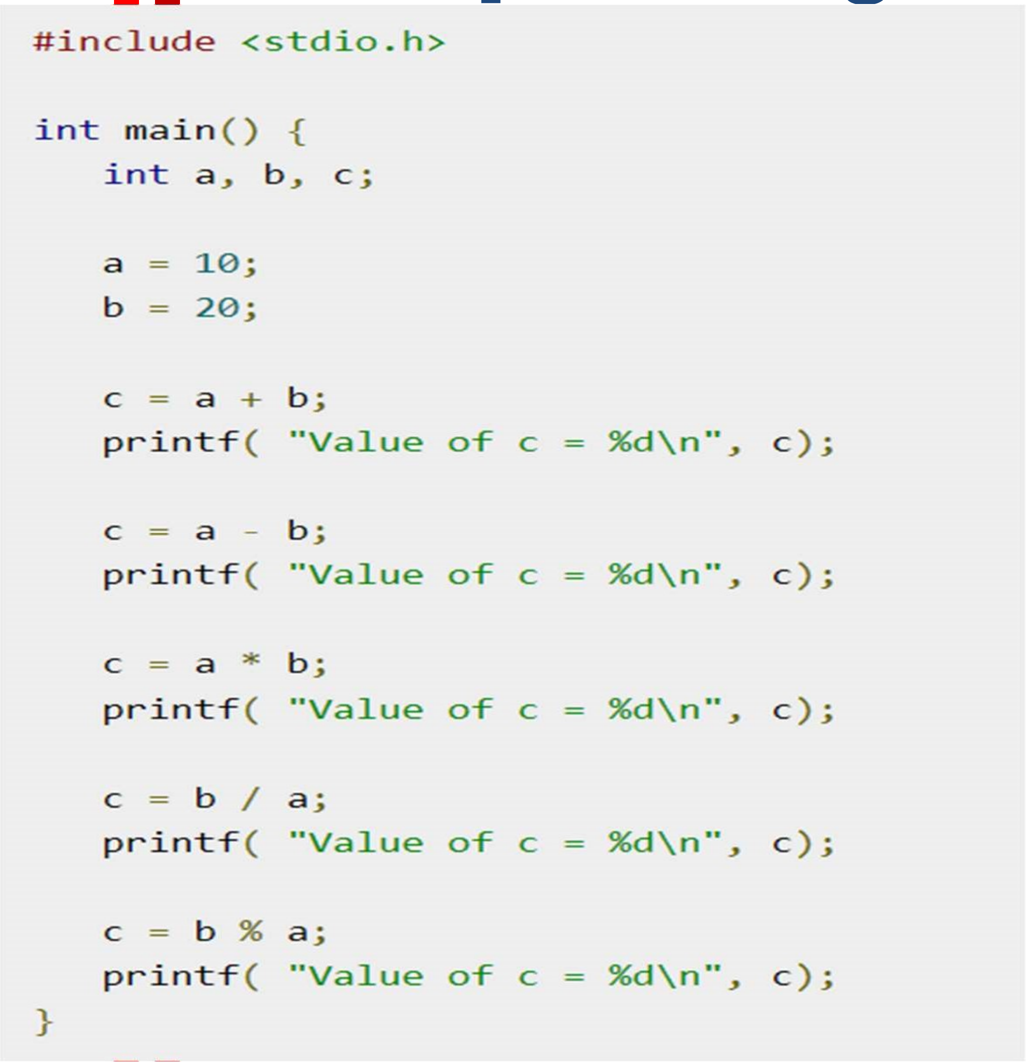
Computer Programming - Operators

An operator in a programming language is a symbol that tells the compiler or interpreter to perform specific mathematical, relational or logical operation and produce final result.

❑Arithmetic Operators

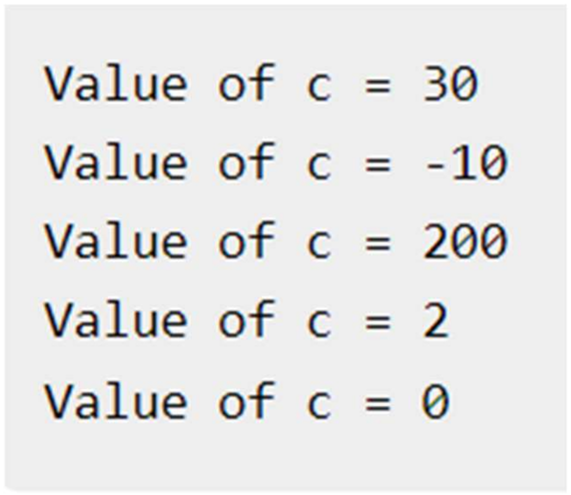
Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	This gives remainder of an integer division	B % A will give 0

Computer Programming - Operators



```c
#include <stdio.h>

int main() {
   int a, b, c;

   a = 10;
   b = 20;

   c = a + b;
   printf( "Value of c = %d\n", c);

   c = a - b;
   printf( "Value of c = %d\n", c);

   c = a * b;
   printf( "Value of c = %d\n", c);

   c = b / a;
   printf( "Value of c = %d\n", c);

   c = b % a;
   printf( "Value of c = %d\n", c);
}
```

- An example of C Programming to understand the mathematical operators −
- When the above program is executed, it produces the following result −



```
Value of c = 30
Value of c = -10
Value of c = 200
Value of c = 2
Value of c = 0
```

Computer Programming - Operators

- **Relational Operators**

Consider a situation where we create two variables and assign them some values as follows −

- A = 10
- B = 20

Here, it is obvious that variable A is greater than B in values. So, we need the help of some symbols to write such expressions which are called relational expressions. If we use C programming language, then it will be written as follows −

- A > B

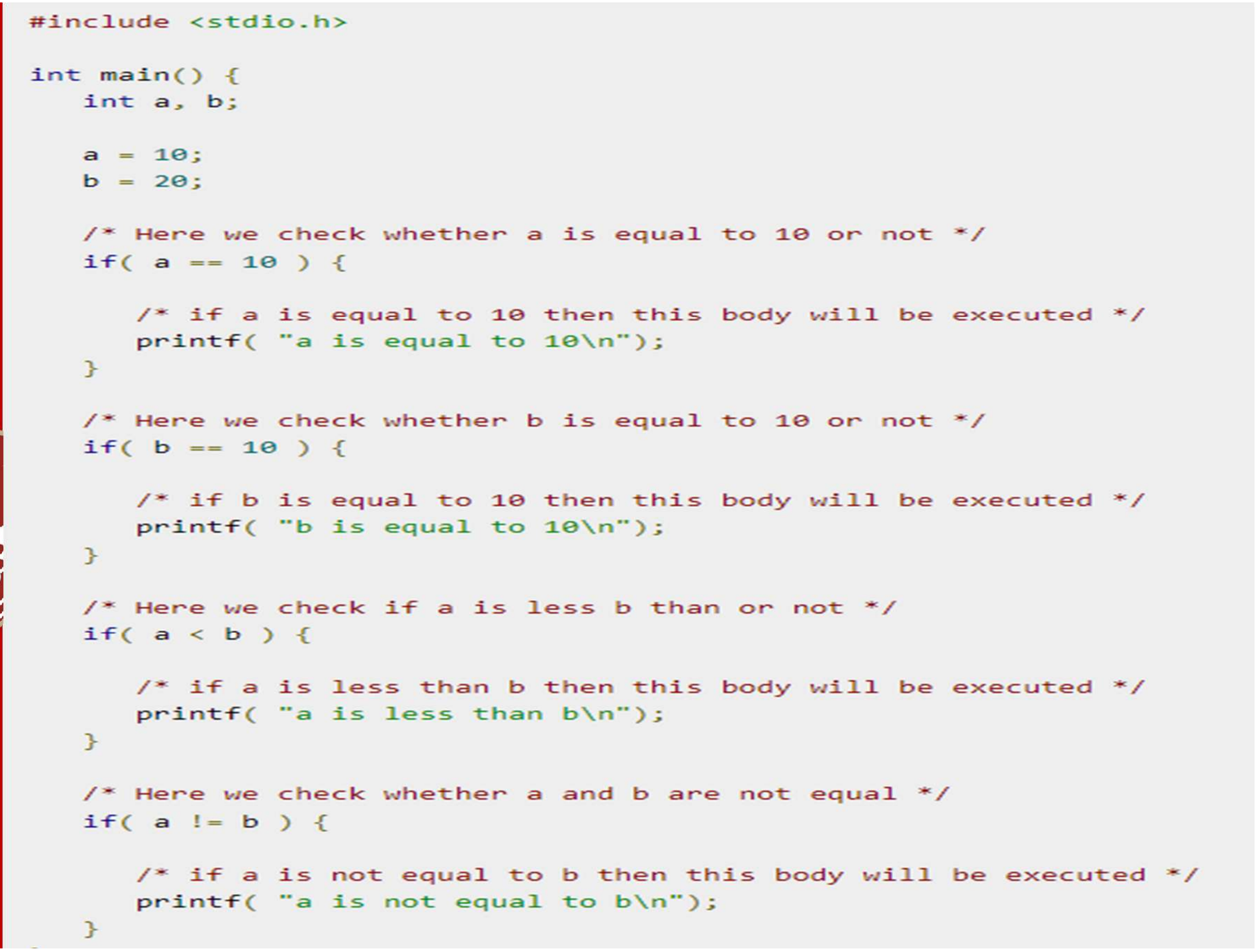
Here, we used a symbol > and it is called a relational operator and in their simplest form, they produce Boolean results which means the result will be either true or false.

Computer Programming - Operators

❑Relational Operators

The following table lists down a few of the important relational operators available in C programming language.

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.



```c
#include <stdio.h>

int main() {
   int a, b;

   a = 10;
   b = 20;

   /* Here we check whether a is equal to 10 or not */
   if( a == 10 ) {

      /* if a is equal to 10 then this body will be executed */
      printf( "a is equal to 10\n");
   }

   /* Here we check whether b is equal to 10 or not */
   if( b == 10 ) {

      /* if b is equal to 10 then this body will be executed */
      printf( "b is equal to 10\n");
   }

   /* Here we check if a is less b than or not */
   if( a < b ) {

      /* if a is less than b then this body will be executed */
      printf( "a is less than b\n");
   }

   /* Here we check whether a and b are not equal */
   if( a != b ) {

      /* if a is not equal to b then this body will be executed */
      printf( "a is not equal to b\n");
   }
```

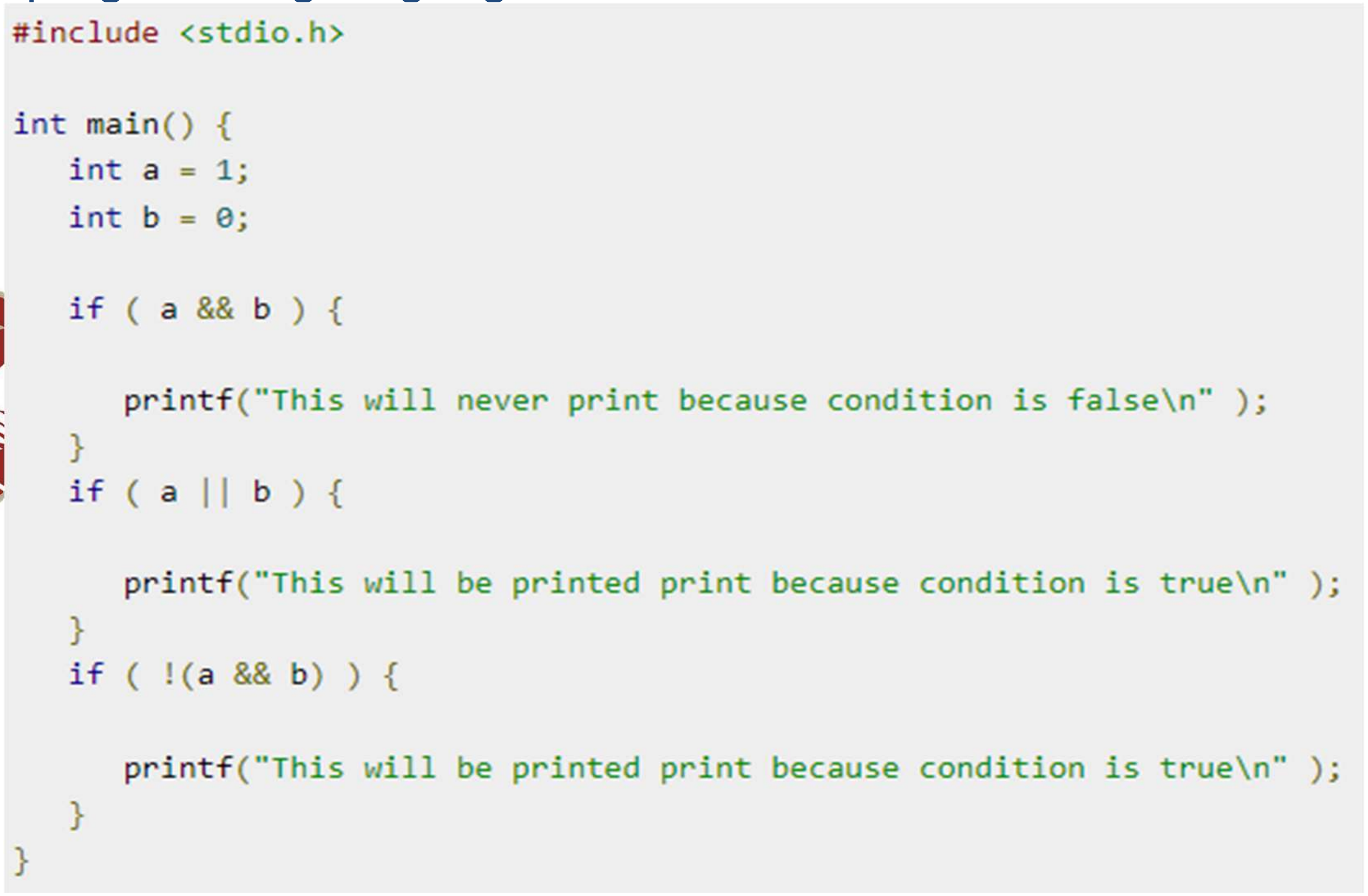
Computer Programming - Operators

- **Logical Operators**
- Logical operators are very important in any programming language and they help us take decisions based on certain conditions.
- Suppose we want to combine the result of two conditions, then logical **AND** and **OR** logical operators help us in producing the final result.
- The following table shows all the logical operators supported by the 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 condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

Computer Programming - Operators

- Try the following example to understand all the logical operators available in C programming language −

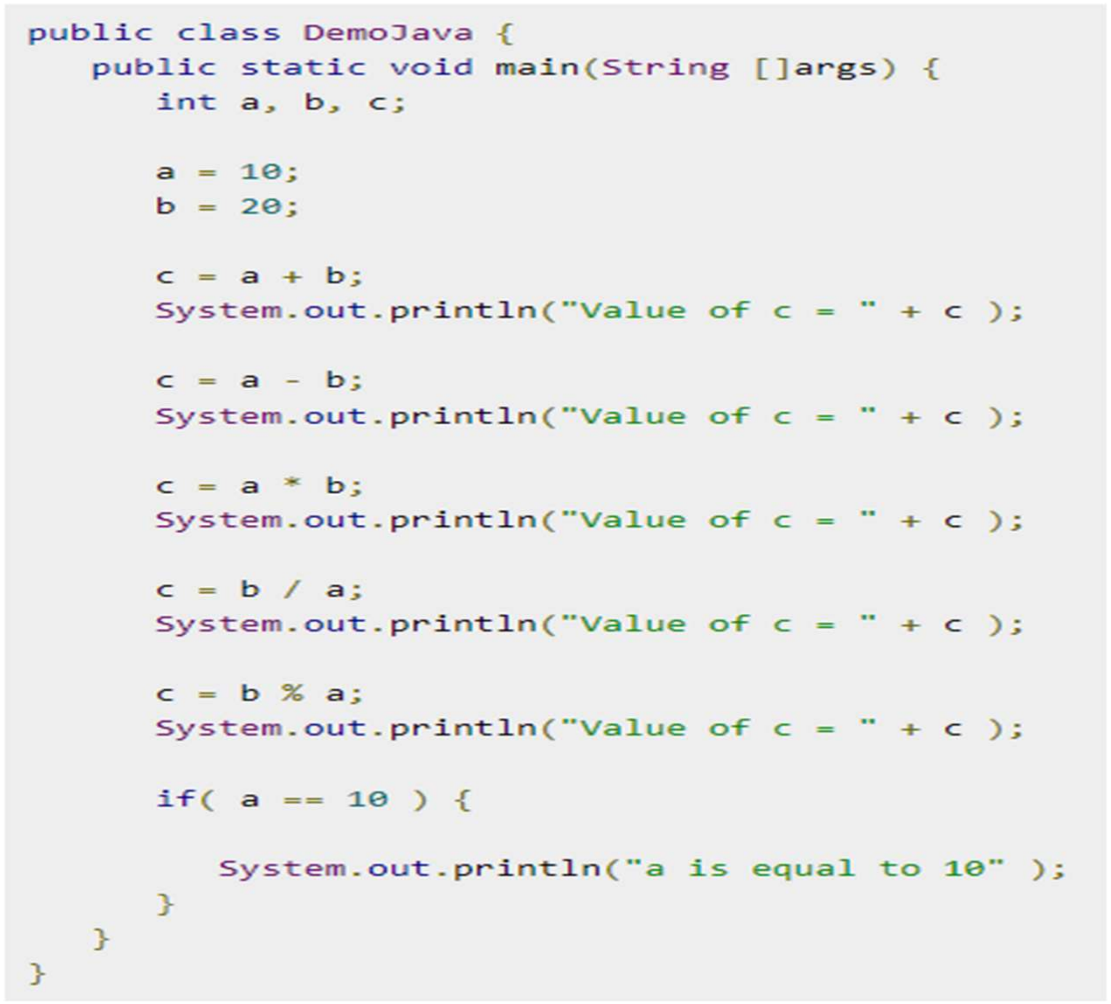


```
#include <stdio.h>

int main() {
   int a = 1;
   int b = 0;

   if ( a && b ) {

      printf("This will never print because condition is false\n" );
   }
   if ( a || b ) {

      printf("This will be printed print because condition is true\n" );
   }
   if ( !(a && b) ) {

      printf("This will be printed print because condition is true\n" );
   }
}
```

Computer Programming - Operators

- **Java programming language –**

This program will create two variables **a** and **b**, very similar to C programming, then we assign 10 and 20 in these variables and finally, we will use different arithmetic and relational operators –



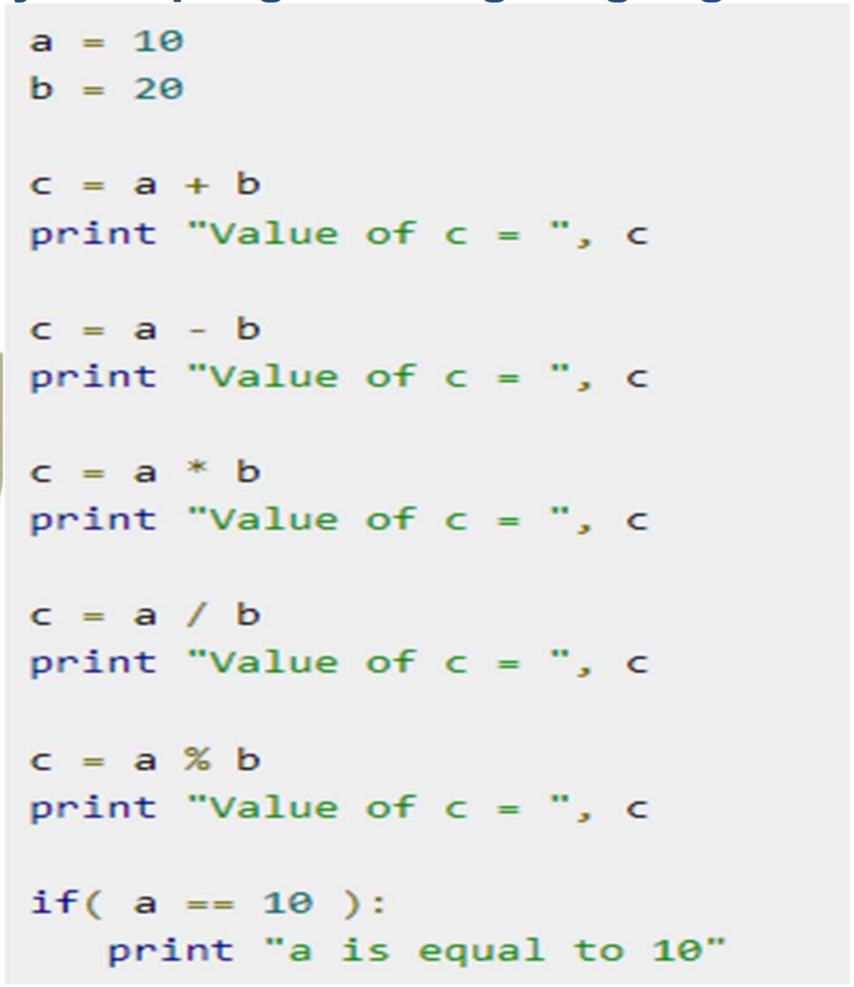
```
public class DemoJava {
   public static void main(String []args) {
      int a, b, c;

      a = 10;
      b = 20;

      c = a + b;
      System.out.println("Value of c = " + c );

      c = a - b;
      System.out.println("Value of c = " + c );

      c = a * b;
      System.out.println("Value of c = " + c );

      c = b / a;
      System.out.println("Value of c = " + c );

      c = b % a;
      System.out.println("Value of c = " + c );

      if( a == 10 ) {

         System.out.println("a is equal to 10" );
      }
   }
}
```

Output

```
Value of c = 30
Value of c = -10
Value of c = 200
Value of c = 2
Value of c = 0
a is equal to 10
```

Computer Programming - Operators

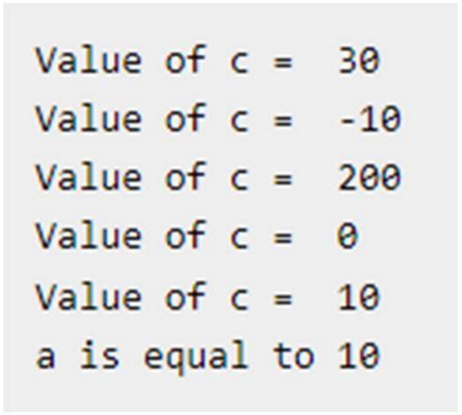
- **Python programming language −**



```
a = 10
b = 20

c = a + b
print "Value of c = ", c

c = a - b
print "Value of c = ", c

c = a * b
print "Value of c = ", c

c = a / b
print "Value of c = ", c

c = a % b
print "Value of c = ", c

if( a == 10 ):
   print "a is equal to 10"
```

This program will create two variables **a** and **b** and at the same time, assign 10 and 20 in those variables

When the program is executed, it produces the following result −



```
Value of c =  30
Value of c =  -10
Value of c =  200
Value of c =  0
Value of c =  10
a is equal to 10
```

Decision Statements

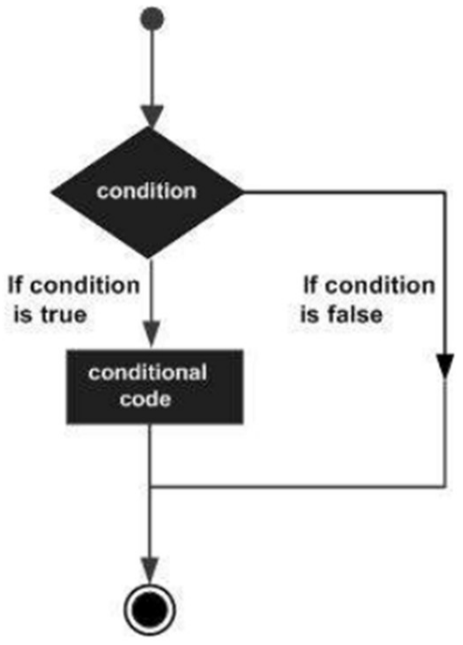
CENTRAL UNIVERSITY

FAITH • INTEGRITY • EXCELLENCE

Decision Statements

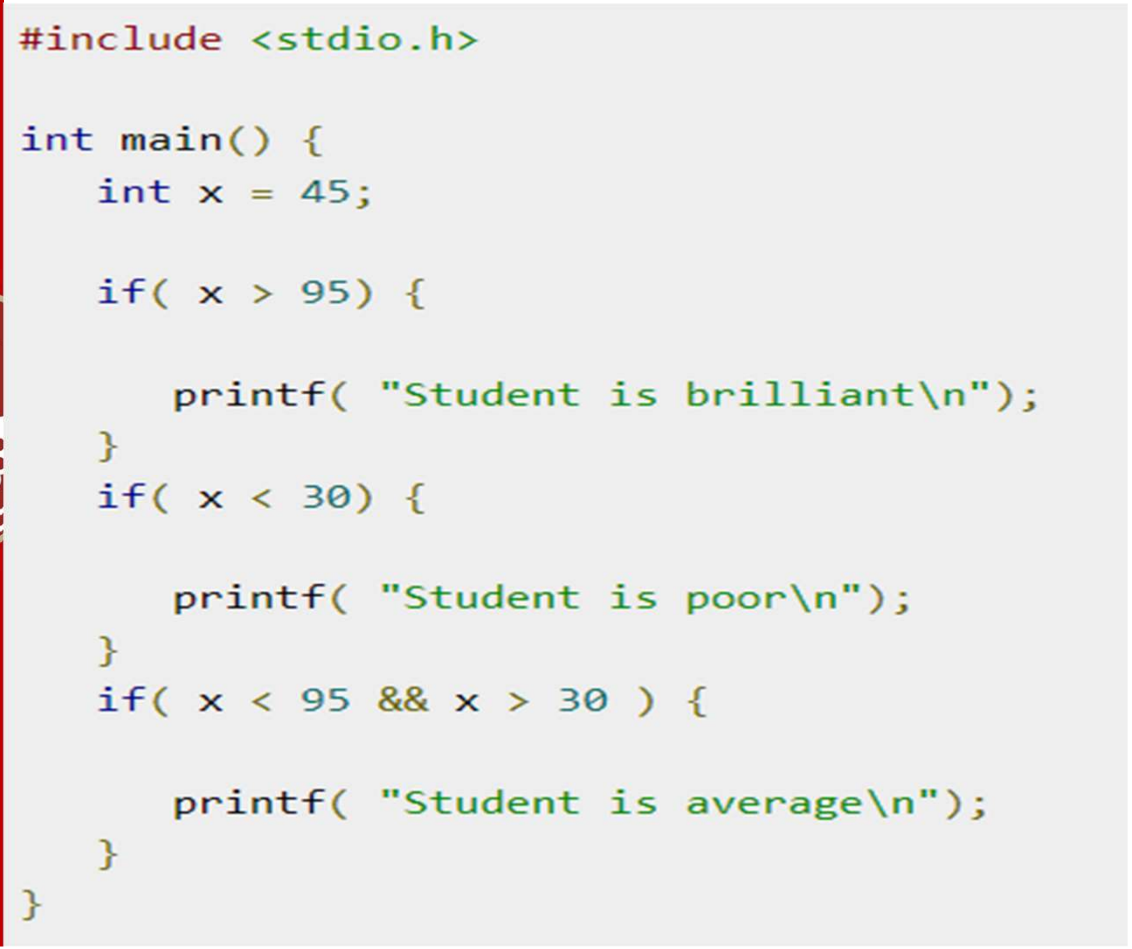
- There will be many situations when you will be given two or more options and you will have to select an option based on the given conditions. For example, we want to print a remark about a student based on his secured marks. Following is the situation −
- **Assume given marks are x for a student:**
- **If given marks are more than 95, then**
- **Student is brilliant**
- **If given marks are less than 30, then**
- **Student is poor**
- **If given marks are less than 95 and more than 30, then**
- **Student is average**

Almost all the programming languages provide conditional statements that work based on the following flow diagram −



Decision Statements

Let's write a C program with the help of **if conditional statements** to convert the above given situation into a programming code −



```c
#include <stdio.h>

int main() {
   int x = 45;

   if( x > 95) {

      printf( "Student is brilliant\n");
   }
   if( x < 30) {

      printf( "Student is poor\n");
   }
   if( x < 95 && x > 30 ) {

      printf( "Student is average\n");
   }
}
```

When the above program is executed, it produces the following result −

```
Student is average
```

Decision Statements

- The above program uses **if conditional statements**. Here, the first **if statement** checks whether the given condition i.e., variable x is greater than 95 or not and if it finds the condition is true, then the conditional body is entered to execute the given statements. Here we have only one *printf()* statement to print a remark about the student.
- Similarly, the second **if statement** works. Finally, the third **if statement** is executed, here we have the following two conditions −
- First condition is **x > 95**
- Second condition is **x < 30**
- The computer evaluates both the given conditions and then, the overall result is combined with the help of the binary operator **&&**. If the final result is true, then the conditional statement will be executed, otherwise no statement will be executed.

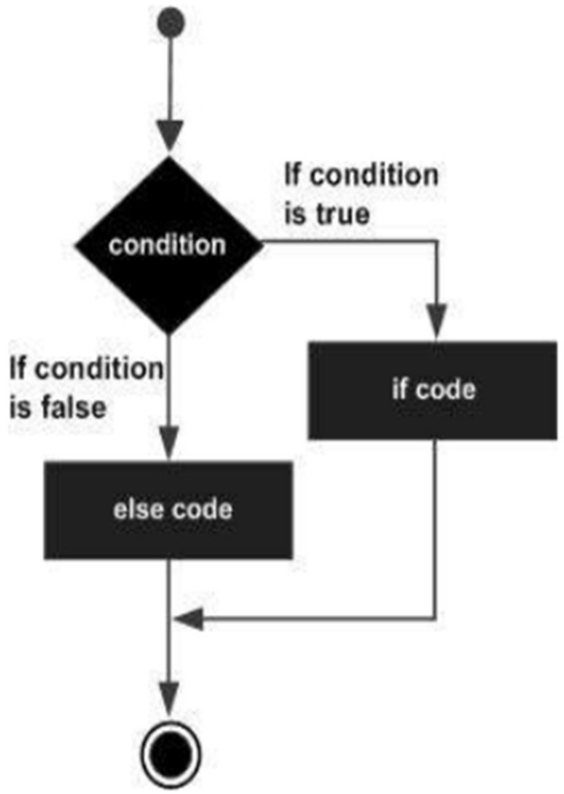
Decision Statements

❑ **if...else statement**

- An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false. The syntax of an **if...else** statement in C programming language is −

```
If (boolean_expression)
{ /* Statement(s) will execute if the boolean
expression is true */ }
else { /* Statement(s) will execute if the boolean
expression is false */ }
```

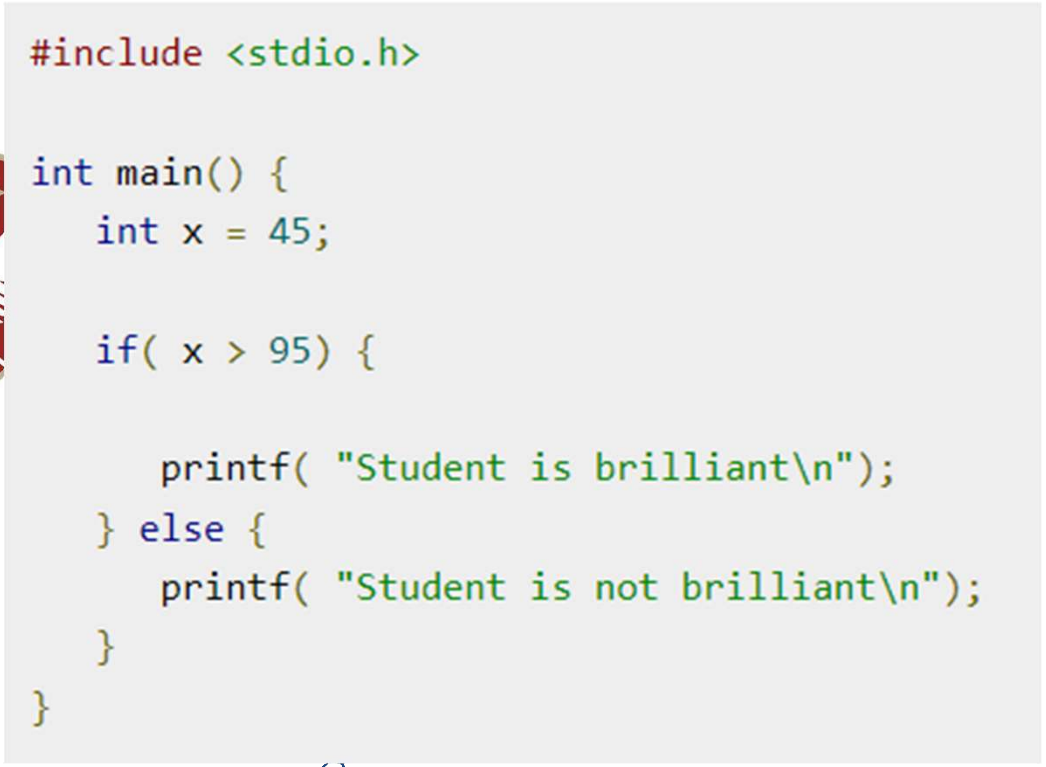
The above syntax can be represented in the form of a flow diagram as shown beside −



Decision Statements

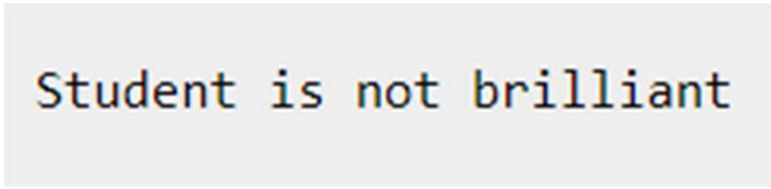
❑ if...else statement

- An **if...else** statement is useful when we have to take a decision out of two options. For example, if a student secures more marks than 95, then the student is brilliant, otherwise no such situation can be coded, as follows −



```c
#include <stdio.h>

int main() {
   int x = 45;

   if( x > 95) {

      printf( "Student is brilliant\n");
   } else {
      printf( "Student is not brilliant\n");
   }
}
```

When the above program is executed, it produces the following result −



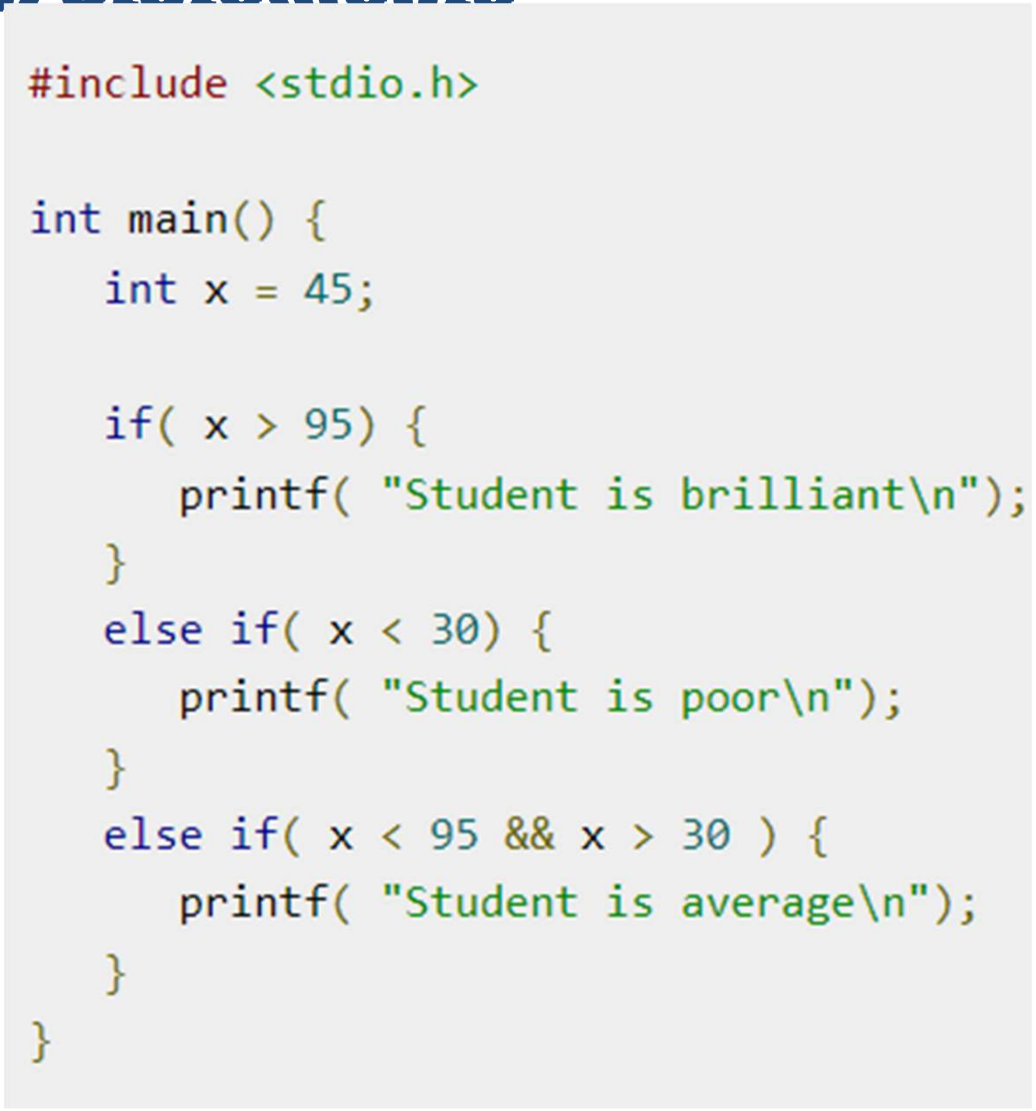
```
Student is not brilliant
```

Decision Statements

❑ **if...elseif...else statement**

- An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions.
- While using **if, else if, else** statements, there are a few points to keep in mind −
- An **if** can have zero or one **else's** and it must come after an **else if**.
- An **if** can have zero to many **else…if's** and they must come before the **else**.
- Once an **else…if** succeeds, none of the remaining **else…if's** or **else's** will be tested.
- The syntax of an **if...else if...else** statement in C programming language is −

Decision Statements



```c
#include <stdio.h>

int main() {
   int x = 45;

   if( x > 95) {
      printf( "Student is brilliant\n");
   }
   else if( x < 30) {
      printf( "Student is poor\n");
   }
   else if( x < 95 && x > 30 ) {
      printf( "Student is average\n");
   }
}
```

Decision Statements

❑ The Switch Statement

- A **switch** statement is an alternative of **if statements** which allows a variable to be tested for equality against a list of values. Each value is called a **case**, and the variable being switched on is checked for each switch case. It has the following syntax −

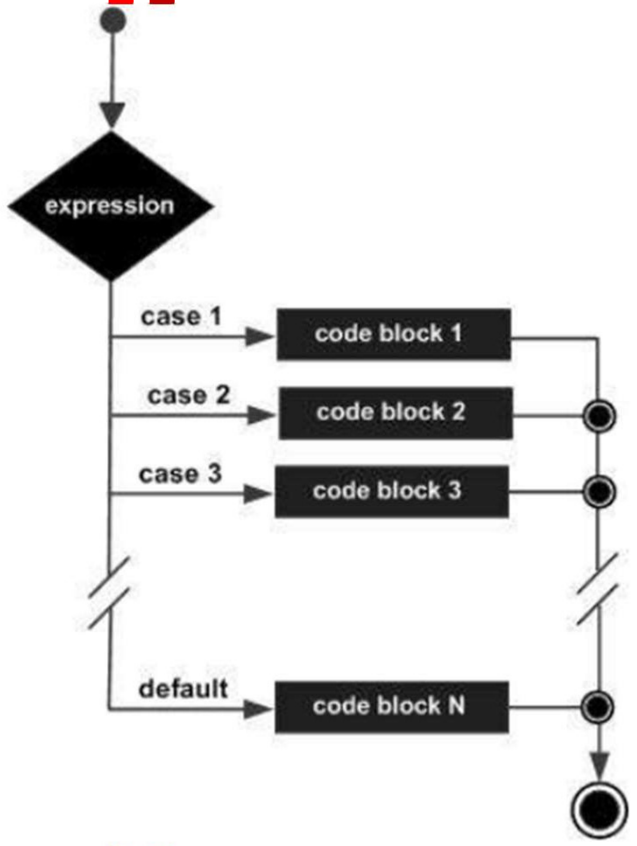
```
switch(expression){
   case ONE :
      statement(s);
      break;
   case TWO:
      statement(s);
      break;
   ......

   default :
      statement(s);
}
```

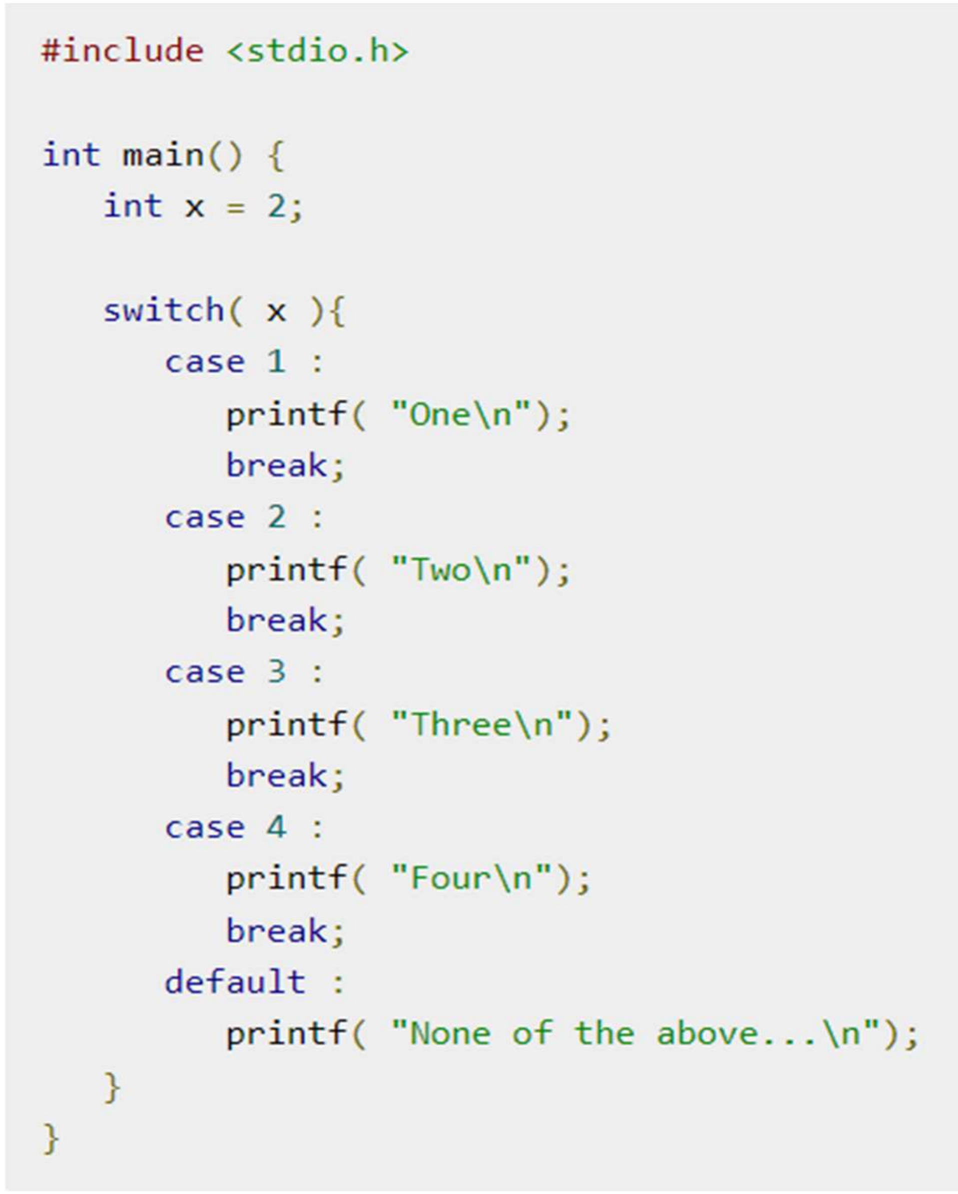
The **expression** used in a **switch** statement must give an integer value, which will be compared for equality with different cases given. Wherever an expression value matches with a case value, the body of that case will be executed and finally, the switch will be terminated using a **break** statement.
If no break statements are provided, then the computer continues executing other statements available below to the matched case. If none of the cases matches, then the default case body is executed.

Decision Statements

❑ The Switch Statement

- The syntax can be represented in the form of a flow diagram as shown below



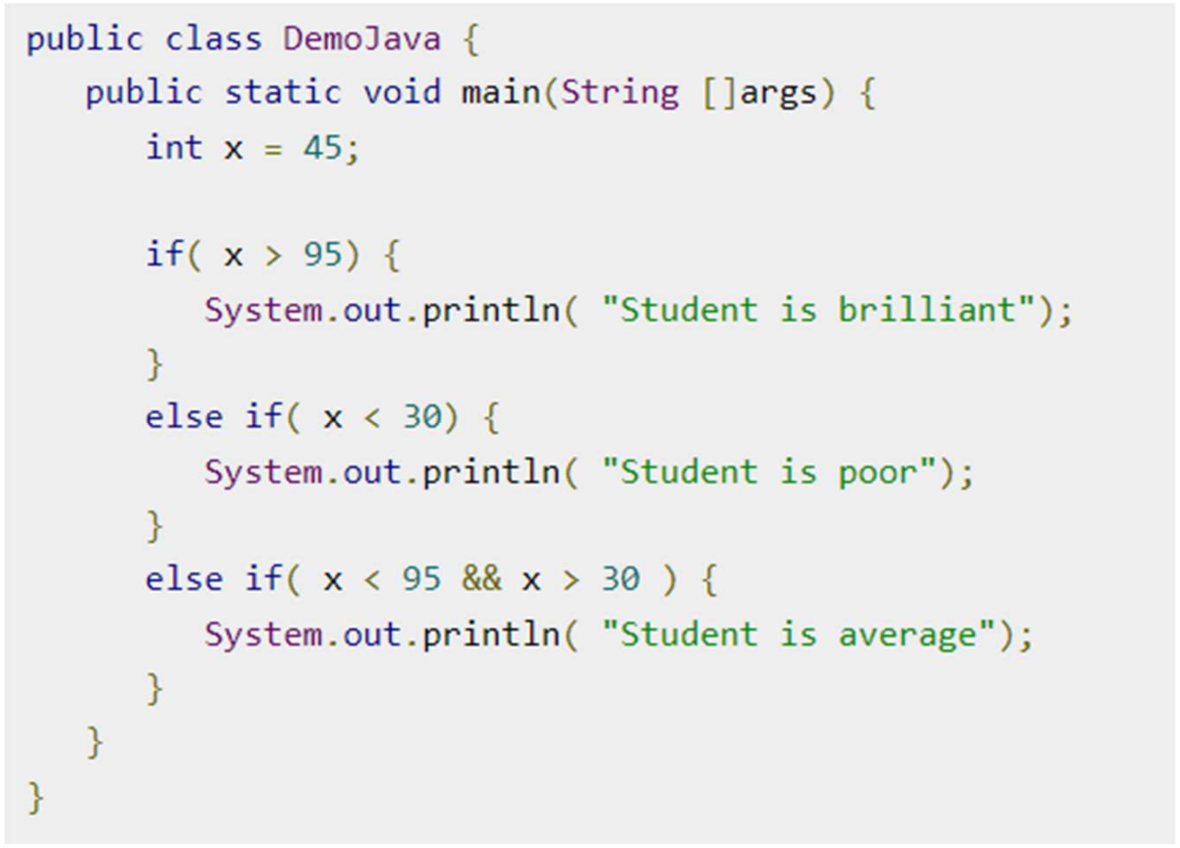


```
#include <stdio.h>

int main() {
   int x = 2;

   switch( x ){
      case 1 :
         printf( "One\n");
         break;
      case 2 :
         printf( "Two\n");
         break;
      case 3 :
         printf( "Three\n");
         break;
      case 4 :
         printf( "Four\n");
         break;
      default :
         printf( "None of the above...\n");
   }
}
```

Decision Statements

❑ Decision in Java

- You can try to execute the following program to see the output, which must be identical to the result generated by the above C example.



```java
public class DemoJava {
   public static void main(String []args) {
      int x = 45;

      if( x > 95) {
         System.out.println( "Student is brilliant");
      }
      else if( x < 30) {
         System.out.println( "Student is poor");
      }
      else if( x < 95 && x > 30 ) {
         System.out.println( "Student is average");
      }
   }
}
```

Decisions in Python

- Python provides **if**, **if...else**, **if...elif...else**, and **switch** statements. Here, you must note that Python does not make use of curly braces for conditional body, instead it simply identifies the body of the block using indentation of the statements.
- You can try to execute the following program to see the output −

```
x = 45

if x > 95:
   print "Student is brilliant"
elif x < 30:
   print "Student is poor"
elif x < 95 and x > 30:
   print "Student is average"

print "The end"
```

When the above program is executed, it produces the following result −

```
Student is average
The end
```

Assignment 2

1. Practise the examples in this presentation and especially work on the flowchart diagram