

IQRA NATIONAL UNIVERSITY PESHAWAR PAKISTAN

NAME: SYED JUNAID ALI SHAH

ID NO: 16373

DEPARTMENT: BS SOFTWARE ENGINEERING

PAPER: Programming Fundamentals

TEACHER NAME: Dr. FAZAL-E-MALIK

EXAM: FINAL TERM EXAM

QNo1: What is the purpose of **if statement**? Discuss its two different forms with examples?

Answer: **if statement:**

Definition:

An if statement is a programming conditional statement that, if proved true, performs a function or displays information.

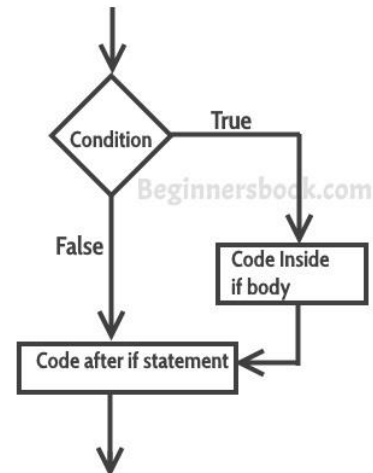
SYNTAX:

The statements inside the body of “if” only execute if the given condition returns true. If the condition returns false then the statements inside “if” are skipped.

IF {condition}
statement(s) to execute.

Flow Diagram of if statement

Flow Diagram of if statement:



Example of if statement:

```
#include <stdio.h>
int main()
{
    int x = 20;
    int y = 22;
    if (x<y)
    {
        printf("Variable x is less than y");
    }
    return 0;
}
```

Output:

Variable x is less than y

Explanation:

The condition ($x < y$) specified in the “if” returns true for the value of x and y, so the statement inside the body of if is executed.

Example of multiple if statements:

We can use multiple if statements to check more than one conditions.

```
#include <stdio.h>
int main()
{
    int x, y;
    printf("enter the value of x:");
    scanf("%d", &x);
    printf("enter the value of y:");
    scanf("%d", &y);
    if (x>y)
    {
        printf("x is greater than y\n");
    }
    if (x<y)
    {
        printf("x is less than y\n");
    }
}
```

```

}
if (x==y)
{
    printf("x is equal to y\n");
}
printf("End of Program");
return 0;
}

```

In the above example the output depends on the user input.

Output:

```

enter the value of x:20
enter the value of y:20
x is equal to y

```

Two different forms with examples:

If-else statement:

The if-else statement in C language is used to execute the code if condition is true or false. It is also called two-way selection statement.

Syntax

```

if(expression)
{
    //Statements
}
else
{
    //Statements
}

```

How "if..else" statement works..

If the expression is evaluated to nonzero (true) then if block statement(s) are executed.

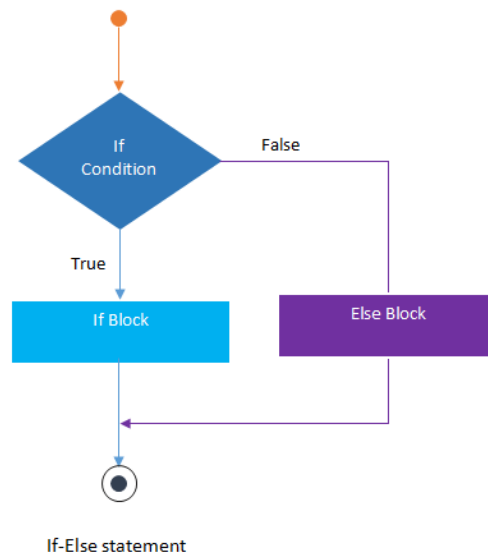
If the expression is evaluated to zero (false) then else block statement(s) are executed.

if..else Statement Example

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int num=0;
    printf("enter the number");
    scanf("%d",&num);
    if(n%2==0)
    {

```



```

printf("%d number in even", num);
}
else
{
printf("%d number in odd",num);
}
getch();
}

```

Nested If-else statement

The nested if...else statement is used when a program requires more than one test expression. It is also called a multi-way selection statement. When a series of the decision are involved in a statement, we use if else statement in nested form.

Syntax

```

if( expression )
{
    if( expression1 )
    {
        statement-block1;
    }
    else
    {
        statement-block 2;
    }
}
else
{
    statement-block 3;
}

```

Example

```

#include<stdio.h>
#include<conio.h>
void main( )
{
    int a,b,c;
    clrscr();
    printf("Please Enter 3 number");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
    {
        if(a>c)
        {
            printf("a is greatest");
        }
    }
    else
    {
        printf("c is greatest");
    }
}

```

```

else
{
if(b>c)
{
printf("b is greatest");
}
else
{
printf("c is greatest");
}
}
}
getch();
}

```

If..else If ladder

The if-else-if statement is used to execute one code from multiple conditions. It is also called multipath decision statement. It is a chain of if..else statements in which each if statement is associated with else if statement and last would be an else statement.

Syntax

```

if(condition1)
{
//statements
}
else if(condition2)
{
//statements
}
else if(condition3)
{
//statements
}
else
{
//statements
}

```

Nest if-else:

It is always legal to **nest** if-else statements, which means you can use one if or else if statement inside another if or else if statement(s).

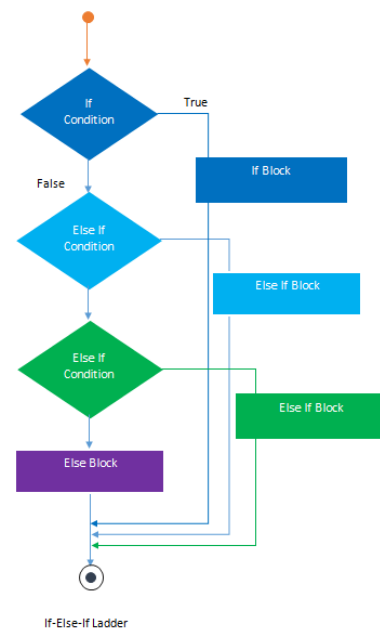
Syntax

The syntax for a **nested if** statement is as follows –

```

if( boolean_expression 1) {
// Executes when the boolean expression
1 is true
if(boolean_expression 2) {
// Executes when the boolean expression 2 is true

```



```
}  
}
```

You can nest **else if...else** in the similar way as you have nested *if* statement.

Example

```
#include <iostream>  
using namespace std;  
  
int main () {  
    // local variable declaration:  
    int a = 100;  
    int b = 200;  
  
    // check the boolean condition  
    if( a == 100 ) {  
        // if condition is true then check the following  
        if( b == 200 ) {  
            // if condition is true then print the following  
            cout << "Value of a is 100 and b is 200" << endl;  
        }  
    }  
    cout << "Exact value of a is : " << a << endl;  
    cout << "Exact value of b is : " << b << endl;  
  
    return 0;  
}
```

When the above code is compiled and executed, it produces the following result –

```
Value of a is 100 and b is 200  
Exact value of a is : 100  
Exact value of b is : 200
```

QNo1 Part B: Write a C++ program to read two numbers from keyboard and then find the LARGEST number of them?

Answer: Code:

```
main.cpp X main.cpp X main.cpp X main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int num1,num2;
8      cout << "Please Enter First Number : |";
9      cin>>num1;
10     cout<<"Please Enter 2nd Number : ";
11     cin>>num2;
12
13     if(num1 > num2)
14     {
15         cout<<endl<<"First Number is largest...."<<endl;
16     }
17     else
18     {
19         cout<<endl<<"2nd Number is largest...."<<endl;
20     }
21 }
22
```

Run:

```
"C:\Users\Sa Qib\Desktop\UnaidPaper\bin\Debug\UnaidPaper.exe"
Please Enter First Number : 32
Please Enter 2nd Number : 352

2nd Number is largest....

Process returned 0 (0x0)   execution time : 11.795 s
Press any key to continue.
```

QNo2 (a): What are the Logical Operators? Explain them?

Answer: **Logical Operators:**

Operators:

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

C++ is rich in built-in operators and provides the following types of operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Logical Operators:

Logical Operators are used with binary variables. They are mainly used in conditional statements and loops for evaluating a condition.

Logical operators in C++ are: `&&`, `||`, `!`

Let's say we have two boolean variables `b1` and `b2`.

`b1&&b2` will return true if both `b1` and `b2` are true else it would return false.

`b1||b2` will return false if both `b1` and `b2` are false else it would return true.

`!b1` would return the opposite of `b1`, that means it would be true if `b1` is false and it would return false if `b1` is true.

Example of Logical Operators

```
#include <iostream>
using namespace std;
int main(){
    bool b1 = true;
    bool b2 = false;
    cout<<"b1 && b2: "<<(b1&&b2)<<endl;
```



```
cout<<"b1 || b2: "<<(b1||b2)<<endl;
cout<<"!(b1 && b2): "<<!(b1&&b2);
return 0;
}
```

Output:

b1 && b2: 0

b1 || b2: 1

!(b1 && b2): 1

QNo2 Part (B): Write a C++ program to get Temperature in Fahrenheit **F** and then find the Atmosphere according to the below rules:

- If temperature **F** is above 40 degree Fahrenheit then display.....Very Hot.
- If temperature **F** is between 35 & 40 degree Fahrenheit then display.....Tolerable.
- If temperature **F** is between 30 & 35 degree Fahrenheit then display.....Warm.
- If temperature **F** is less than 30 degree Fahrenheit then display.....Cool.

Answer: Code:

```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int temp;
8      cout << "Please Enter Temperature in Fahrenheit : " ;
9      cin>>temp;
10
11     if(temp >= 40)
12     {
13
14         cout<<endl<<"Very Hot Temperature ...."<<endl;
15     }
16     else if(temp >= 35 && temp <= 40)
17     {
18         cout<<endl<<"Tolerable Temperature ...."<<endl;
19     }
20     else if(temp >= 30 && temp <= 35)
21     {
22
23         cout<<endl<<"Warm Temperature ...."<<endl;
24     }
25     else if(temp < 30)
26     {
27
28         cout<<endl<<"Cold Temperature ...."<<endl;
29     }
30
31 }
32
```

Run:

```
"C:\Users\Sa QiB\Desktop\FahrenheitProgram\bin\Debug\FahrenheitProgram.exe"
Please Enter Temperature in Fahrenheit : 30
Warm Temperature ....
Process returned 0 (0x0) execution time : 3.263 s
Press any key to continue.
```

QNo3: What does Looping mean? Explain different loops in C++?

Answer: **Loops:**

In computer programming, loops are used to repeat a block of code.

Explanation:

For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.

That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.

There are 3 types of loops in C++.

For loop

while loop

do...while loop

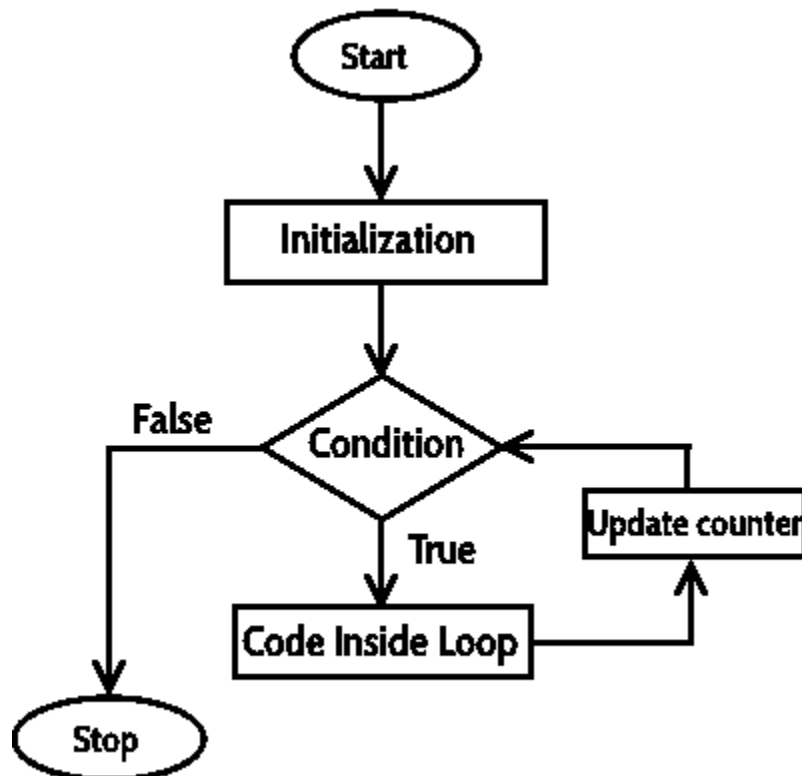
For loop:

This is one of the most frequently used loop in C programming.

Syntax of for loop:

```
for (initialization; condition test; increment or decrement)
{
    //Statements to be executed repeatedly
}
```

Flow Diagram of For loop



Step 1: First initialization happens and the counter variable gets initialized.

Step 2: In the second step the condition is checked, where the counter variable is tested for the given condition, if the condition returns true then the C statements inside the body of for loop gets executed, if the condition returns false then the for loop gets terminated and the control comes out of the loop.

Step 3: After successful execution of statements inside the body of loop, the counter variable is incremented or decremented, depending on the operation (++ or -).

Example of For loop

```
#include <stdio.h>
int main()
{
    int i;
    for (i=1; i<=3; i++)
    {
        printf("%d\n", i);
    }
}
```

```
    }  
    return 0;  
}  
Output:
```

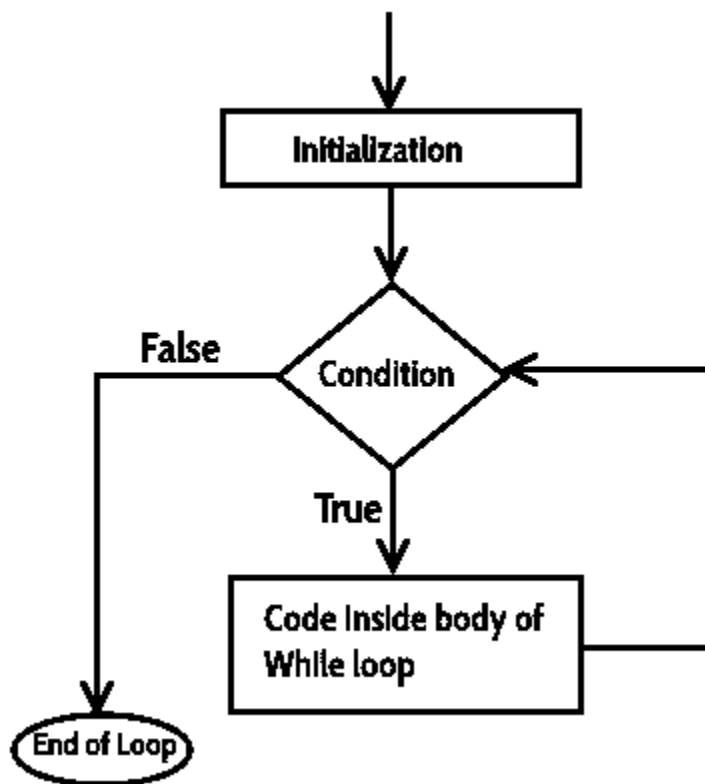
1
2
3

while loop:

Syntax of while loop:

```
while (condition test)  
{  
    //Statements to be executed repeatedly  
    // Increment (++) or Decrement (--) Operation  
}
```

Flow Diagram of while loop



Example of while loop

```
#include <stdio.h>
```

```

int main()
{
    int count=1;
    while (count <= 4)
    {
        printf("%d ", count);
        count++;
    }
    return 0;
}

```

Output:

1 2 3 4

step1: The variable count is initialized with value 1 and then it has been tested for the condition.

step2: If the condition returns true then the statements inside the body of while loop are executed else control comes out of the loop.

step3: The value of count is incremented using ++ operator then it has been tested again for the loop condition.

do..while loop:

A do while loop is similar to while loop with one exception that it executes the statements inside the body of do-while before checking the condition. On the other hand in the while loop, first the condition is checked and then the statements in while loop are executed. So you can say that if a condition is false at the first place then the do while would run once, however the while loop would not run at all.

do..while loop:

Syntax of do-while loop

```

do
{
    //Statements

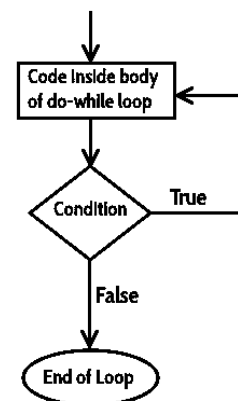
```

```

}while(condition test);

```

Flow diagram of do while loop:



Example of do while loop

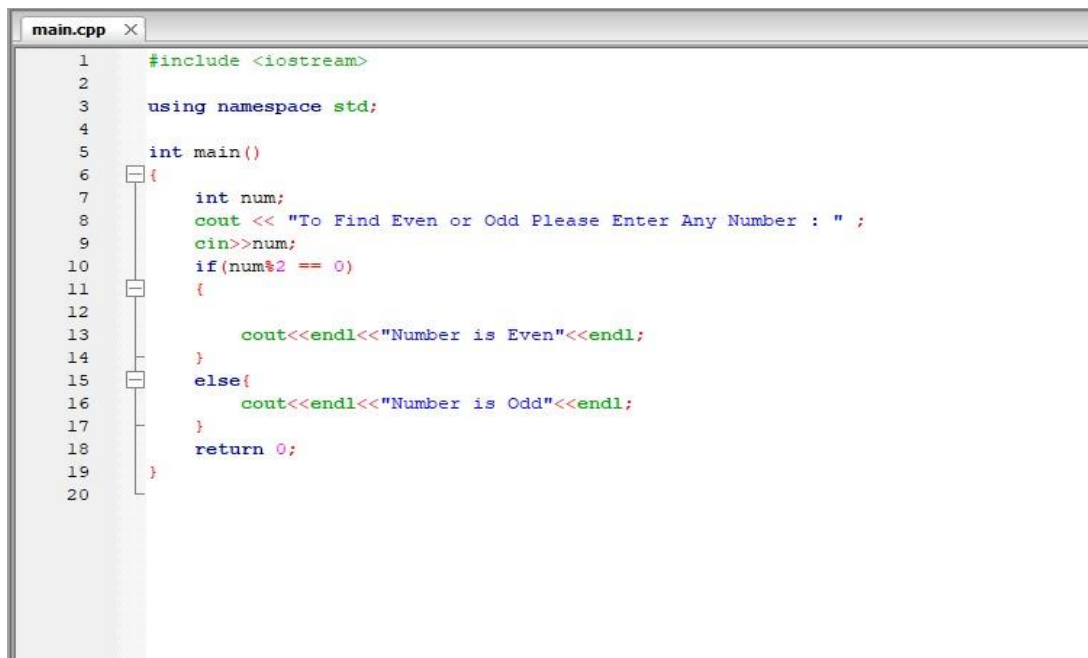
```
#include <stdio.h>
int main()
{
    int j=0;
    do
    {
        printf("Value of variable j is: %d\n", j);
        j++;
    }while (j<=3);
    return 0;
}
```

Output:

```
Value of variable j is: 0
Value of variable j is: 1
Value of variable j is: 2
Value of variable j is: 3
```

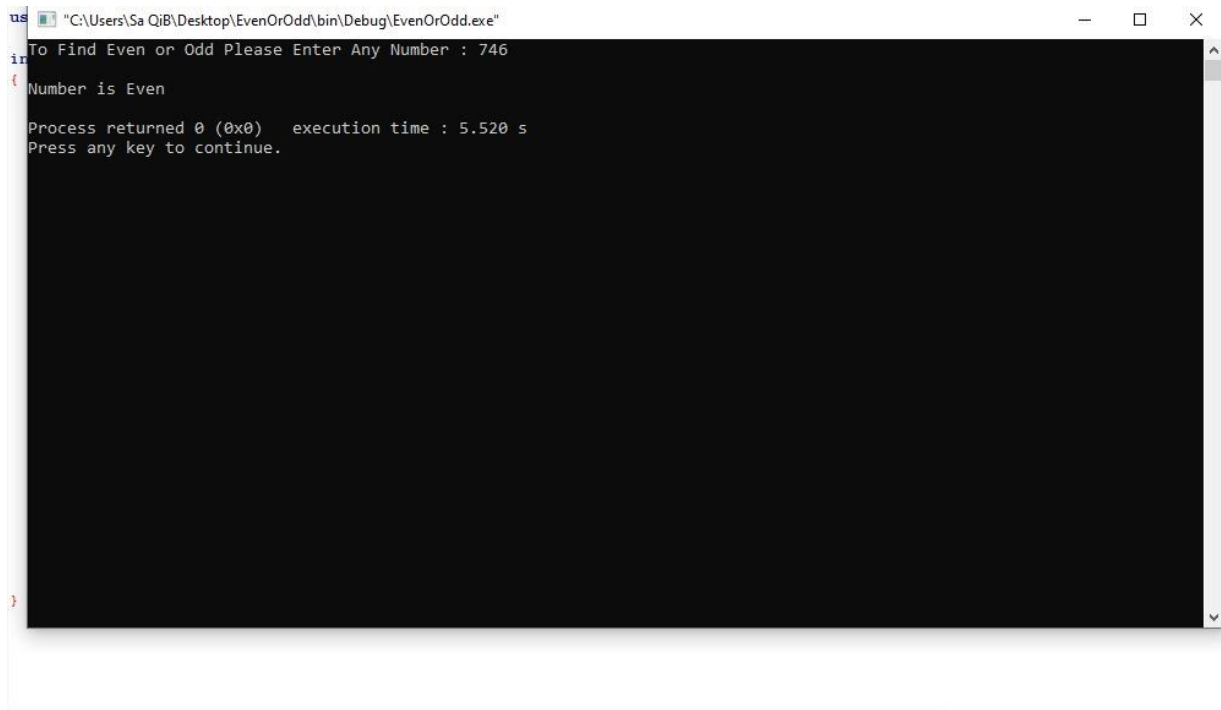
QNo3 Part (b): Write a C++ program to read a number from keyboard and then determine whether it is Even or Odd number?

Answer: **Code:**



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int num;
8      cout << "To Find Even or Odd Please Enter Any Number : " ;
9      cin>>num;
10     if(num%2 == 0)
11     {
12
13         cout<<endl<<"Number is Even"<<endl;
14     }
15     else{
16         cout<<endl<<"Number is Odd"<<endl;
17     }
18     return 0;
19 }
20
```

Run:



```
"C:\Users\Sa QiB\Desktop\EvenOrOdd\bin\Debug\EvenOrOdd.exe"
To Find Even or Odd Please Enter Any Number : 746
Number is Even
Process returned 0 (0x0)   execution time : 5.520 s
Press any key to continue.
```

QNo4: What is the purpose of using break and continue statements?

Answer: Break Statements:

The break statement terminates the smallest enclosing loop (i. e., while, do-while, for or switch statement)

Break statement terminates the loop. To use it in a for loop, you can get input from user every time and display the output when user enters a negative number. The output gets displayed then and exited using the break statement.

It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

Example:

```
#include <iostream>
using namespace std;
main()
{
    int i;
    cout << "The loop with break produces output as: \n";

    for (i = 1; i <= 5; i++) {

        // Program comes out of loop when
        // i becomes multiple of 3.
        if ((i % 3) == 0)
            break;
        else
            cout << i << " ";
    }
}
```

**The loop with break produces output as:
1 2**

Explanation:

1. Now when the loop iterates for the first time, the value of $i=1$, the if statement evaluates to be false, so the else condition/ statement is executed.
2. Again loop iterates now the value of $i= 2$, else statement is implemented as if statement evaluates to be false.
3. Loop iterates again now $i=3$; if the condition evaluates to be true and the loop breaks.

Continue Statement:

The continue statement skips the rest of the loop statement and causes the next iteration of the loop to take place.

In the same way, continue statement in a for loop works, but it will not display the negative numbers. The continue statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Example:

```
#include <iostream>
using namespace std;
main()
{
    int i;
    cout << "\nThe loop with continue produces output as: \n";
    for (i = 1; i <= 5; i++) {

        // The loop prints all values except
        // those that are multiple of 3.
        if ((i % 3) == 0)
            continue;
        cout << i << " ";
    }
}
```

Output:

The loop with continue produces output as:
1 2 4 5

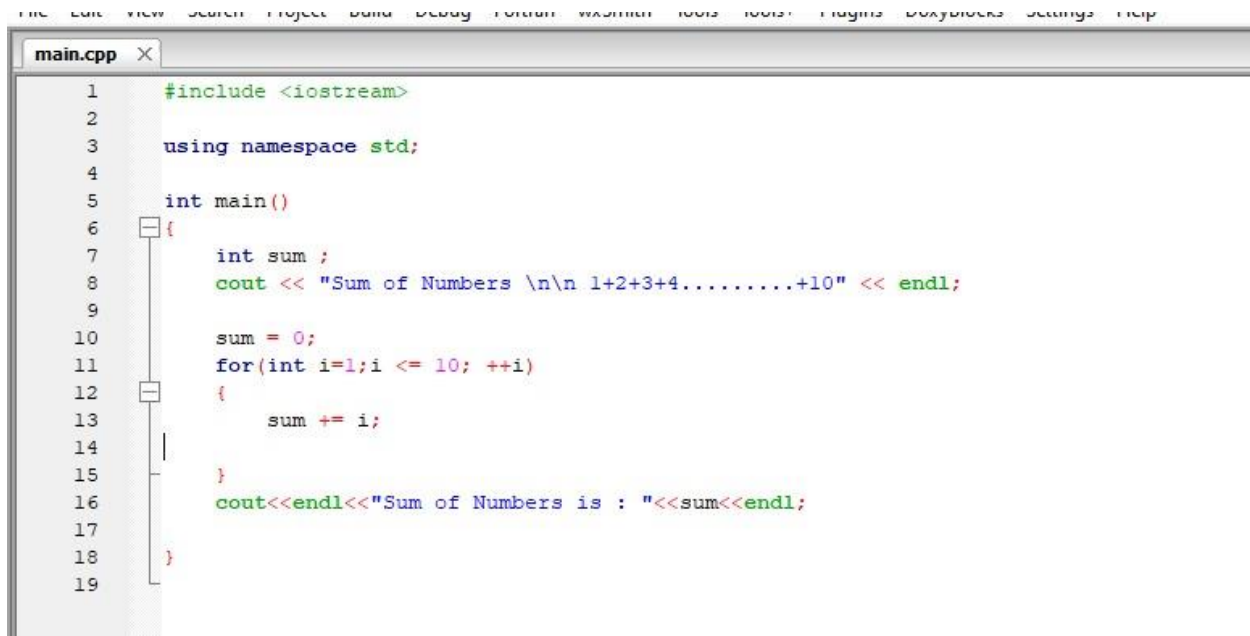
Explanation:

1. Now when the loop iterate for the first time then the value of $i=1$, the if statement evaluates to be false, so the else condition/ statement 2 is implemented.
2. Again loop iterates now the value of $i= 2$, else statement is implemented as if statement evaluates to be false.
3. Loop iterates again now $i=3$; if the condition evaluates to be true, here the code stop in between and start new iterate until the end condition met.

QNo4 Part (b): Write a C++ program to find the sum of the following numbers:

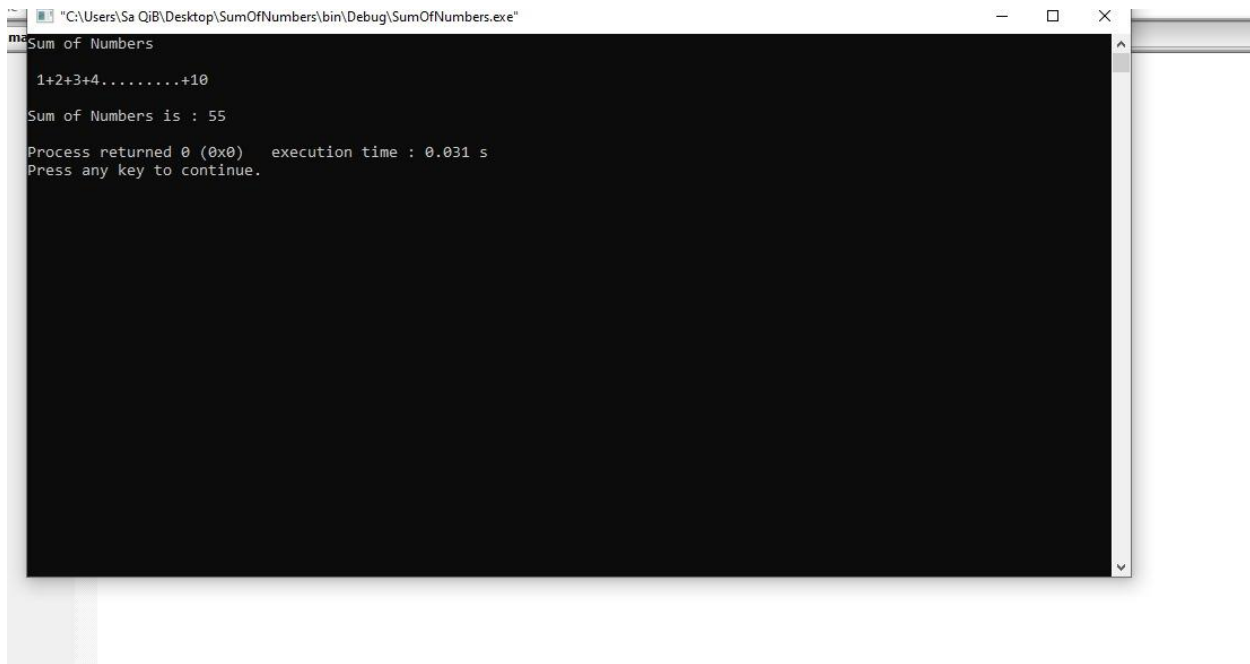
1+2+3+.....+10?

Answer: Code:



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int sum ;
8      cout << "Sum of Numbers \n\n 1+2+3+4.....+10" << endl;
9
10     sum = 0;
11     for(int i=1;i <= 10; ++i)
12     {
13         sum += i;
14     }
15
16     cout<<endl<<"Sum of Numbers is : "<<sum<<endl;
17
18 }
19
```

Out Put:



```
"C:\Users\Sa Qi\Desktop\SumOfNumbers\bin\Debug\SumOfNumbers.exe"
Sum of Numbers
1+2+3+4.....+10
Sum of Numbers is : 55
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

QNO5: Explain the the following with proper examples

- a) C++ Character set
- b) Constants
- c) Variables
- d) Keywords
- e) Relational Operators

Answer: 1. **C++ Character set:**

Character set is the combination of English language (Alphabets and White spaces) and math's symbols (Digits and Special symbols). Character Set means that the characters and symbols that a C++ Program can understand and accept. These are grouped to form the commands, expressions, words, c-statements and other tokens for C++ Language.

Character Set is the combination of alphabets or characters, digits, special symbols and white spaces same as learning English is to first learns the alphabets, then learn to combine these alphabets to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. More about a C++ program we can say that it is a sequence of characters. These character from the character set plays the different role in different way in the C++ compiler. The special characters are listed in Table

Special Characters				
+	>	/	[\
!	;	"]	{
<	*	.	%	}
:	^	,	~	#
-	(=	-	
?)	,	&	

1. Alphabets:

Alphabets are represented by A-Z or a-z. C- Language is case sensitive so it takes different meaning for small and upper case letters. By using this character set C statements and character constants can be written very easily. There are total 26 letters used in C-programming.

2. Digits:

Digits are represented by 0-9 or by combination of these digits. By using the digits numeric constant can be written easily. Also numeric data can be assigned to the C-tokens. There are total 10 digits used in the C-programming.

3. Special Symbols:

All the keyboard keys except alphabet, digits and white spaces are the special symbols. These are some punctuation marks and some special symbols used for special purpose.

There are total 30 special symbols used in the C-programming. Special symbols are used for C-statements like to create an arithmetic statement +, -, * etc. , to create relational statement <, >, <=, >=, == etc. , to create assignment statement =, to create logical statement &&, || etc. are required.

4. White Spaces:

White spaces has blank space, new line return, Horizontal tab space, carriage ctrl, Form feed etc. are all used for special purpose. Also note that Turbo-C Compiler always ignore these white space characters in both high level and low level programming.

2. Constants:

A constant value is the one which does not change during the execution of a program.”

Constants refer to fixed values that the program may not alter. Constants can be of any of the basic data types. The way each constant is represented depends upon its type. Constants are also called literals.

Defining Constants

There are two simple ways in C++ to define constants –

- Using **#define** preprocessor.
- Using **const** keyword.

Numeric Constant:

These have numeric value having combination of sequence of digits i.e. from 0-9 as alone digit or combination of 0-9 with or without decimal point (precision value) having positive or negative sign. These are further sub-divided into two categories as:

- (i) Integer Numeric Constant
- (ii) Float or Real Numeric Constant

(i) Integer Numeric Constant

An Integer Numeric Constant is a sequence of digits (combination of 0-9 digits without any decimal point or without precision), optionally preceded by a plus or minus sign.

There are 3 types of integer numeric constant namely decimal integer, octal integers and hexadecimal integer.

(a) **Decimal Integer Numeric Constant:** These have no decimal point in it and are either be alone or be the combination of 0-9 digits. These have either +ve or -ve sign. For example: 1214, -1321, 10,254, -78, +99 etc.

(b) **Octal Integer Numeric Constant:** These consist of combination of digits from 0-7 with positive or negative sign. It has leading with 0 or 0 (upper or lower case) means Octal or octal. For example: 0317,003, -045 etc.

(c) **Hexadecimal Integer Numeric Constant:** These have hexadecimal data. It has leading ox, OX, Ox or x, X. These have combination of 0-9 and A-F (a-f) or alone. The letters a-f or A-F represents the numbers 10-15. For example: 0x232, 0x92, 0xACD, 0xAEF etc.

(ii) Float or Real Numeric Constant

Float Numeric Constants consists of a fractional part in their representation; Integer constants are inadequate to represent quantities that vary continuously. These quantities are represented by numbers containing fractional parts like 26.082.

Examples of real constants are: 0.0026, -0.97, 435.29, +487.0, 3.4E-2, 4.5E5

(a) **Mantissa part:** The part without E and having a decimal point is called Mantissa Real part e.g. 45.5, -22.43, 0.5 etc. it is also called without exponent part.

(b) **Exponent part:** The exponent part has an E within it. It is also called a scientific notation. Here E has base value 10. it computes the power. For example: 4.2x10² can be written as 4.2E2, 4.2x10⁻⁵ can be written as 4.2E- 5.-Similarly some more valid real numeric constant are as: 54.73 E -4,51.9 E +11 etc.

Character Constant

Character constants have either a single character or group of characters or a character with backslash used for special purpose. These are further subdivided into three types:

- (i) Single Character Constant
- (ii) String Character Constant

(iii) Backslash Character Constant

(i) Single Character Constant

Single Character constants are enclosed between single quotes('). For example, 'a' and '%' are both character constants. So these are also called single quote character constant.

For example: 'a', 'M', '5', '+', '1' etc. are some valid single character constant.

(ii) String Character Constant

C++ supports another type of constant: the string. A string is a set of characters enclosed in double quotes ("). For example: "Punar Deep" is a string.

You must not confuse strings with characters. A single character constant is enclosed in single quotes, as in 'a'. However, "a" is a string containing only one letter.

For example: "Dinesh", "Hello", "2013", "2013-2020", "5+3", "?+!" etc. are some valid string character constant. These are used for printing purpose or display purpose in the C++ program's output statements. These can also be used for assigning the string data to the character (string) type variables.

(iii) Backslash Character Constants:

Enclosing character constants in single quotes works for most printing characters. A few, however, such as the carriage return, can't be. For this reason, C++ includes the special backslash character constants, shown below, so that you may easily enter these special characters as constants.

These are also referred to as escape sequences. You should use the backslash codes instead of their ASCII equivalents to help ensure portability.

These are used for special purpose in the C++ language. These are used in output statements

like cout etc. `#include <iostream>`

`using namespace std;`

`#define LENGTH 10`

`#define WIDTH 5`

`#define NEWLINE '\n'`

`int main() {`

`int area;`

`area = LENGTH * WIDTH;`

`cout << area;`

`cout << NEWLINE;`

`return 0;`

`}`

When the above code is compiled and executed, it produces the following result 50 .

3.Variables:

Variables are containers for storing data values.

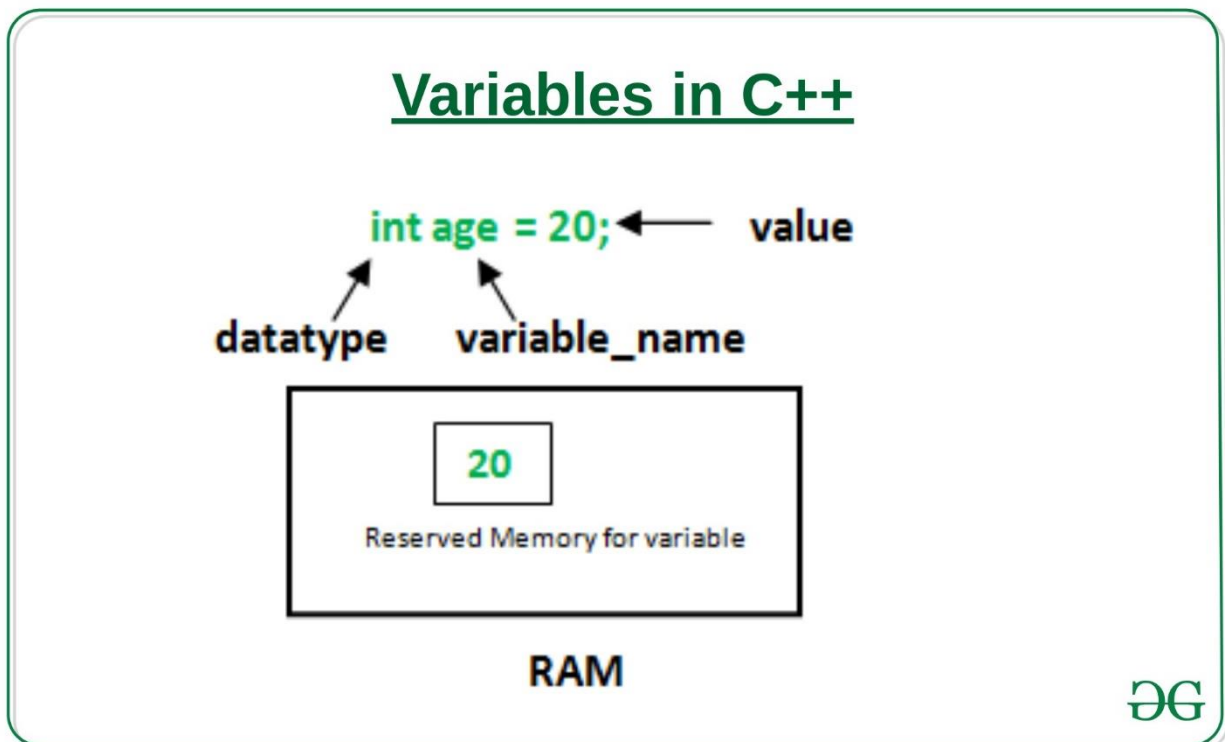
A variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.

In C++, there are different types of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- double - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- string - stores text, such as "Hello World". String values are surrounded by double quotes
- bool - stores values with two states: true or false.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In C++, all the variables must be declared before use.

A variable name can consist of alphabets (both upper and lower case), numbers and the underscore ' _ ' character. However, the name must not start with a number.



Syntax

```
type variable = value;
```

datatype: Type of data that can be stored in this variable.

variable_name: Name given to the variable.

value: It is the initial value stored in the variable.

Examples:

```
// Declaring float variable
```

```
float simpleInterest;
```

```
// Declaring integer variable
```

```
int time, speed;
```

```
// Declaring character variable
```

```
char var;
```

Difference between variable declaration and definition:

The variable declaration refers to the part where a variable is first declared or introduced before its first use. A variable definition is a part where the variable is assigned a memory location and a value. Most of the times, variable declaration and definition are done together. See the following C++ program for better clarification:

```
#include <iostream>
using namespace std;
int main()
{
    // declaration and definition
    // of variable 'a123'
    char a123 = 'a';

    // This is also both declaration and definition
    // as 'b' is allocated memory and
    // assigned some garbage value.
    float b;

    // multiple declarations and definitions
    int _c, _d45, e;
    // Let us print a variable
    cout << a123 << endl;
    return 0;
}
```

Output:

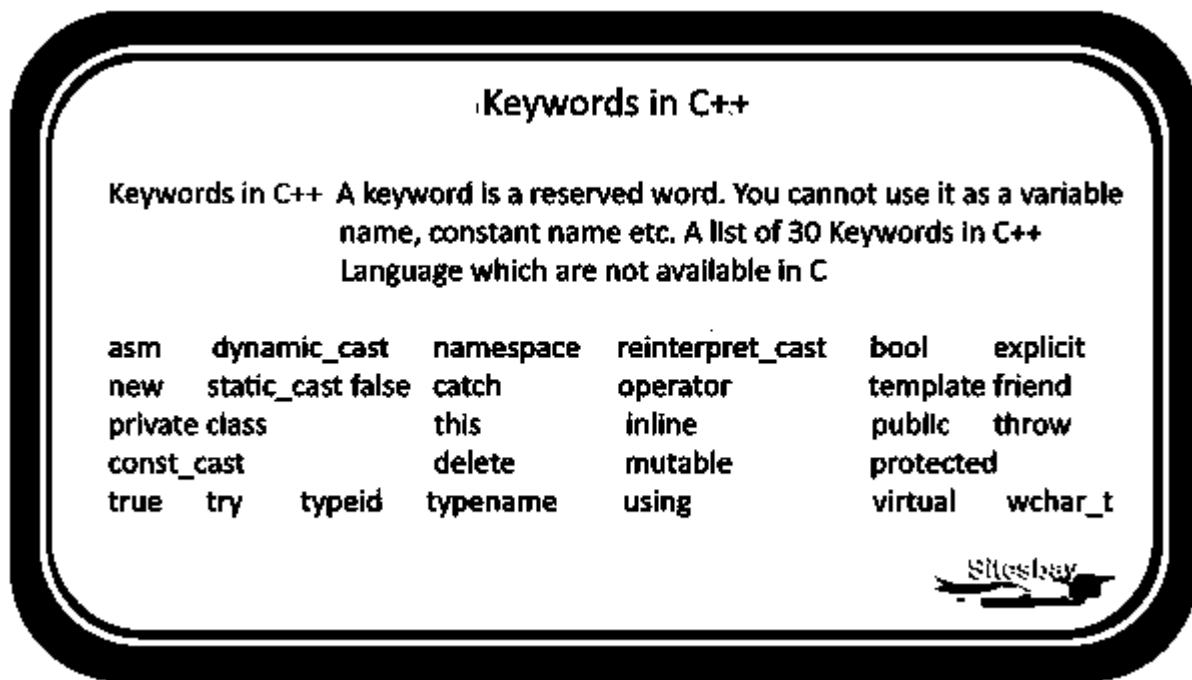
a

4. Keywords:

Keyword is a predefined or reserved word in C++ library with a fixed meaning and used to perform an internal operation. C++ Language supports more than 64 keywords.

Keywords are those words whose meaning is already defined by Compiler. These keywords cannot be used as an identifier. Note that keywords are the collection of reserved words and predefined identifiers. Predefined identifiers are identifiers that are defined by the compiler but can be changed in meaning by the user.

Every Keyword exists in lower case latter like auto, break, case, const, continue, int etc.



Another 30 reserved words that were not in C, these are new to C++

asm	dynamic_cast	namespace	reinterpret_cast
bool	explicit	new	static_cast
catch	false	operator	template
class	friend	private	this

const_cast	inline	public	throw
delete	mutable	protected	true
try	typeid	typename	using

5.Relational Operators:

A relational operator is used to check the relationship between two operands.

For example,

// checks if a is greater than b

a > b;

Here, > is a relational operator. It checks if a is greater than b or not.

If the relation is true, it returns 1 whereas if the relation is false, it returns 0.

The following table summarizes the relational operators used in C++.

Operator	Meaning	Example
==	Is Equal To	3 == 5 gives us false
!=	Not Equal To	3 != 5 gives us true
>	Greater Than	3 > 5 gives us false
<	Less Than	3 < 5 gives us true
>=	Greater Than or Equal To	3 >= 5 give us false
<=	Less Than or Equal To	3 <= 5 gives us true

== Operator

The equal to == operator returns

true - if both the operands are equal or the same

false - if the operands are unequal

For example,

int x = 10;

int y = 15;

int z = 10;

x == y // false

x == z // true

Note: The relational operator == is not the same as the assignment operator =. The assignment operator = assigns a value to a variable, constant, array, or vector. It does not compare two operands.

!= Operator

The not equal to != operator returns

true - if both operands are unequal

false - if both operands are equal.

For example,

```
int x = 10;  
int y = 15;  
int z = 10;
```

```
x != y // true  
x != z // false
```

> Operator

The greater than > operator returns

true - if the left operand is greater than the right

false - if the left operand is less than the right

For example,

```
int x = 10;  
int y = 15;
```

```
x > y // false  
y > x // true
```

< Operator

The less than operator < returns

true - if the left operand is less than the right

false - if the left operand is greater than right

For example,

```
int x = 10;  
int y = 15;
```

```
x < y // true  
y < x // false
```

>= Operator

The greater than or equal to >= operator returns

true - if the left operand is either greater than or equal to the right

false - if the left operand is less than the right

For example,

```
int x = 10;  
int y = 15;  
int z = 10;
```

```
x >= y // false  
y >= x // true  
z >= x // true
```

<= Operator

The less than or equal to operator <= returns

true - if the left operand is either less than or equal to the right

false - if the left operand is greater than right

For example,

```
int x = 10;
```

```
int y = 15;
```

```
x > y // false
```

```
y > x // true
```