C++ FINNAL TERM ASSIGMENT

NAME: **FAIZULLAH KHAN**

ID: **14840**

Section: **B**

DEPARTMENT: **BS (SE)**

SUMITTED: **DR FAZLI MALIK**

DATED: 29/09/2020.

Question No # 01

What is the purpose of if statement ? Discuss its two different forms with examples.

Answer:

If STATEMENT:

If statement is useful when you need to check multiple conditions within the program, nesting of if blocks can be avoided using if statement. All forms of the if statement are executable branching statements.

Statement Purpose:

The various forms of if statement are Fortran's main branching tools. They give Fortran an ability to make decisions in a program. The different forms of if statements that can be used include the simple logical if, then if-then else structure, and the arithmetic if.

Syntax of if statement:

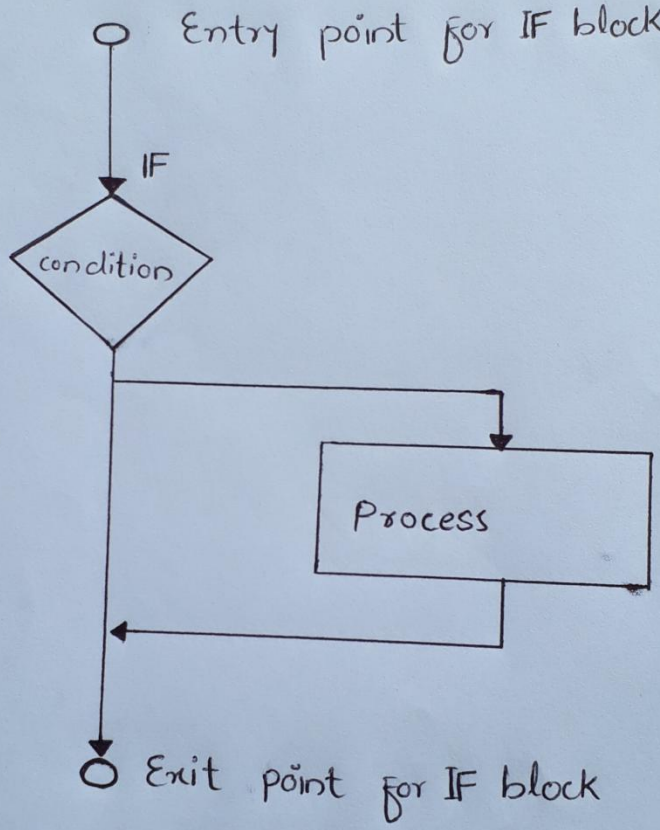
The else--- if statement

```
if (condition 1)
{
    //These statements would execute if the
      condition 1 is true
}
else if (condition 2)
{
    // These statements would execute if
       the condition 2 is true
}
 else if (condition 3)
{
    // These statements would execute if the
       condition is true.
}
.
.
.
else
{
  // These statements would execute if the
```

condition return false

}

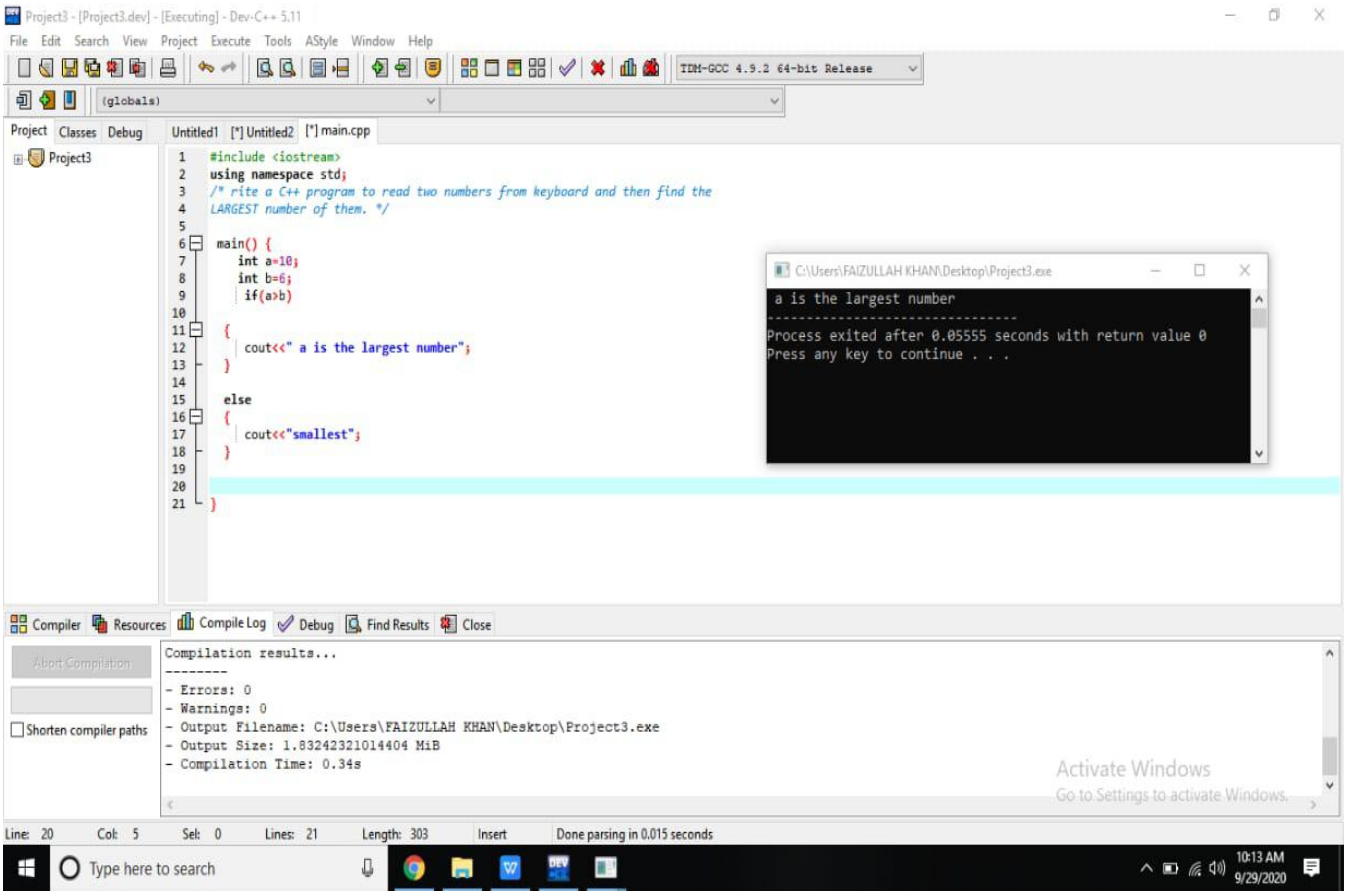
Flow Chart of if Statement:



Example : 2 If we enter the number greater or equale to zero then the condition evalute true --- Hence, the statement inside the body of if is executed. output 2 enter an integer : -4 you entered a negative integer : -4,

★★ ★★

Q1 part b


Project3 - [Project3.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Project3
Untitled1 [*] Untitled2 [*] main.cpp
#include <iostream>
using namespace std;
/* rite a C++ program to read two numbers from keyboard and then find the
LARGEST number of them. */
main() {
int a=10;
int b=6;
if(a>b)
{
cout<<" a is the largest number";
}
else
{
cout<<"smallest";
}
}
C:\Users\FAIZULLAH KHAN\Desktop\Project3.exe
a is the largest number
Process exited after 0.05555 seconds with return value 0
Press any key to continue . . .
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\FAIZULLAH KHAN\Desktop\Project3.exe
- Output Size: 1.83242321014404 MiB
- Compilation Time: 0.34s
Activate Windows
Go to Settings to activate Windows.
Line: 20 Col: 5 Sel: 0 Lines: 21 Length: 303 Insert Done parsing in 0.015 seconds
Type here to search
10:13 AM
9/29/2020

Question No # 02

Part "A" What are the logical operator? Explain them.?

Answer: Logical operator are mainly used to control program flow. Usually, you will find them as part of an if, a while, or some other control statement. The logical operators are:

: && (logical / AND):

- Used to combine two conditions
- true if both conditions are true

```
if (gender == 1 && age >= 65)
    senior ++;
```

: || (logical / OR):

- true if either of condition is true

```
if (semester Avg >= 90 || final exam
>= 90)
Count << ("student grade is A");
```

• ! (Logical NOT, logical negation):

- Returns true when its condition is false, & vice versa.

```
if (!(grade == 20))
    cout << "hello world");
```

Alternative

```
if (grade != 20)
    cout << "hello world");
```

Using Logical Operators:

- A company insures its drivers in the following cases.
  - * If the driver is married.
  - * If the driver is unmarried, male and above 30 years of age.
  - * If the driver is unmarried, female and above 25 years of age.

Example of Using Logical Operators:

```
if (ms == 'M')
    cout << "Driver is insured";
else
```

```
{
 If (sex == 'M')
  {
   If (age > 30)
    cout << "Driver is insured";
  else
    cout << "Driver is not insured";
  }
  else
    {
        If (age > 25)
        cout << " Driver is insured;"
      else
        cout << "Driver is not insured;"
      }
}
  If ((ms == 'M') || (ms=='U' && sex=='M' && age
        >30) ||
     (ms == 'U' && sex == 'F' && age > 25))
    else
        cout << " Driver is not insured)";
```

Question No# 2

Part # "B" Write a C++ program to get Temperature in Fahrenheit and then find the Atmosphere ----------?

Answer:

```
# include <iostream.h>
 using namespace std;
  int main()
{
    float fahr, cel;
    char option;
    Cout<<" choose from following options:"<< endl;
    Cout << "1 celcious to faranheit."<< endl;
    Cout << "2. faraheit to celcious."<< endl;
    cin >> option;
// option for covering celcious into
   faranheit.
  if (option == 1)
  {
        Cout<<" enter the temp in celcious :";
        cin >> option;
       Faranhite = (1.8* celcious) +32.0;
// temp conversion formula
      Cout << "\n temp in degree fanheit :";<<
          hanhight <<"f" << endl;
}
```

```
// option for covering faranheit into celcious
    else if (option == 2)
{
    Cout << "enter the temp into faranheit:";
     cin >> faraheit;
    celcious = (fanheight - 32) / 1.8;
        // Temperature Conversion formula
    Cout << "\n temperature in degree celcious:"
          << celcious << "C" << endl;
        }
          else
        Cout << "Error wrong input." << endl;
         return 0;
}
```

Question No #03:

Part "A" What does looping mean? Explain different loops in C++?

Answer:

LOOPING:

Definition: A loop executes the sequence of statements many times until the stated condition becomes false. A loop consists of two parts, a body of a loop and a control statement. The control statement is a combination of some condition that direct the body of loop to execute until the specified condition become false. The purpose of the loop is to repeat the same code a number of times.

Types of Loop:

i ⇒ The while loop.

ii ⇒ The do-while loop.

iii ⇒ The for loop.

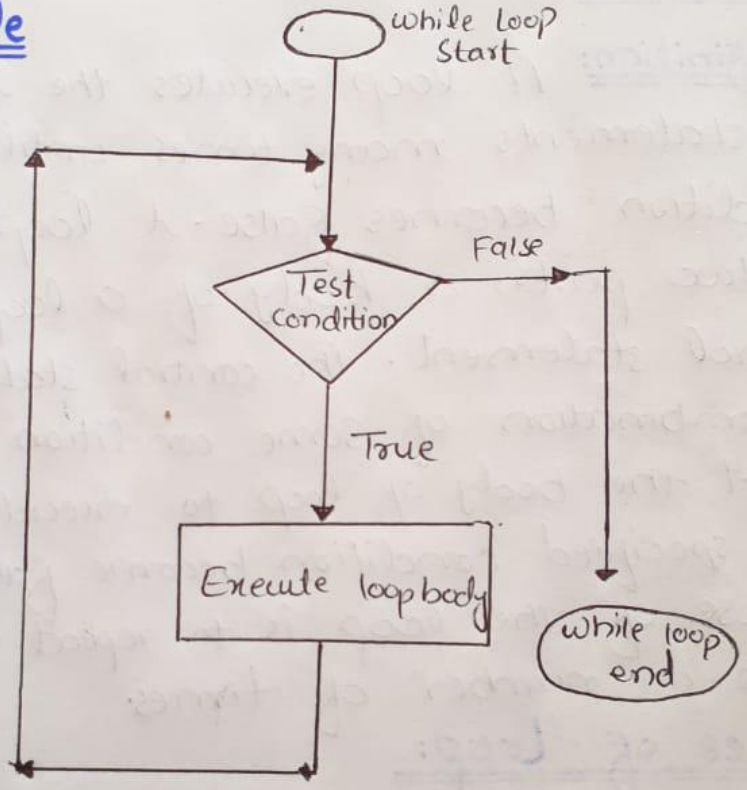
iv ⇒ Nested loop

P.T.O

(i) **While Loop:**

While loop condition is evaulated first and if it return true then the statement inside while loop execute, this happens repeatedly until the condition false.

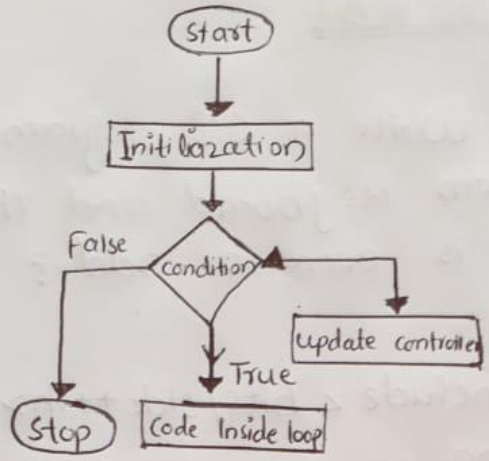
Example



For Loop:

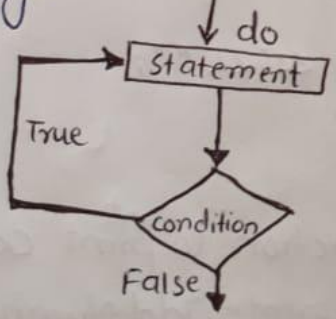
Execute sequence of statements multiple times and abbreviates the code that manages the loop variable.

Example: When your are displaying numbers from 1 to 100 times.



Do-while Loop: Do-while loop checks the condition at the end of the loop. This means that the statements inside the loop body will be executed at least once even if the condition is never true.

Example: An entry controlled loop, as the test condition is checked before entering the loop body.



Nested Loop: Nested loop exist inside the body of another loop it is called nested loop.

★★ ★★

Question No # 03

Part "B"

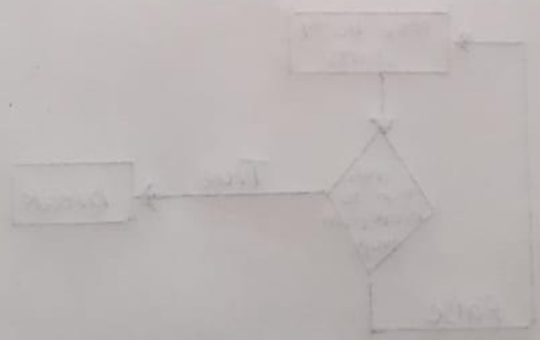
Write a C++ program to read a number from keyboard and then determine whether it is even or odd ?

Answer:

```
# include < bits/stdc++.h>
using name space std;
// Function to print even numbers
void print Even Numbers (int N)
{
    cout << "Even:";
for (int i= 1;i<=2 * N;i++) {
    // Numbers that are divisible by 2
    if (i% 2 ==0)
        cout <<i<< " ";
    }
}
    // Function to print odd numbers
    void print Odd Numbers (int N)
    {
    cout << "\n Odd: ";
    for (int i =1; i <=2 * N;++) {
```

```
// Numbers that are not divisible by 2
if (i % 2 != 0)
    cout << i << " ";
  }
}

  // Driver code
 int main()
 {
      int N = 5;
   Print Even Numbers (N);
   Print Odd Numbers (N);
    return 0;
 }
```



Question No #04

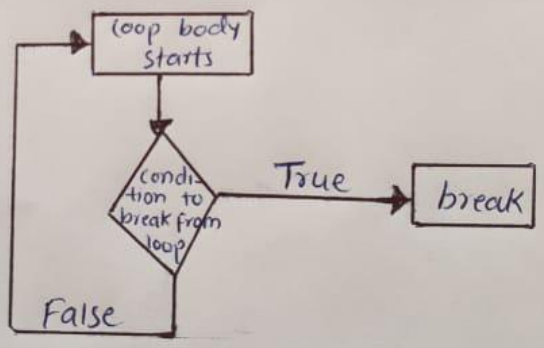
Part "A" What is the purpose of using break and continue statements ?

Answer:

BREAK STATEMENT: The break C++ is a loop control statement which is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

Syntax:

```
for (int i=0; i<10; i++)
    if (i==4) {
      break;
    } cout<<i<<"\n";
}
```

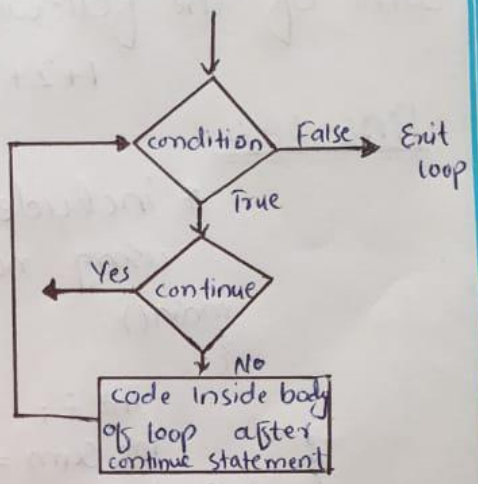


Continue Statement: The continue statement is used inside the loops. when a continue statement encountered inside a loop, controller jump into the begining of the loop for next enteraction skiping the execution

of statement inside the body of loop for current interaction.

Example:

```
for (int i=0, i < 10; i++) {
  if (i==4) {
    continue;
  }
  cout << i << "\n";
}
```



Question No#04:

Part "B" Write a C++ program to find sum of the following numbers:

1+2+3------ +10

Answer:

```
# include <studio.h>
  using name space std;
main()
{
    int f;
    int sum = 0;
  Cout << " Sum of number";
for (f=1; f <= 10; f++)
  {
    Cout << f;
  Sum = Sum + f;
  }
}
  Cout << I;
  }
}
```

Question No #05

Part A Explain the following with proper examples.

(A) C++ character Set.

Answer:

C++ CHARACTER SET:

Letters :- A-Z , a-z.

Digits :- 0-9

Special Symbols :- Space + - * / ^ \ ( ) [ ] { } = != < > . ' " $, ; : % ! ....

White Space :- Blank Space, Horizontal tab (→), carriage return (↵), Newline, Form feed.

Other characters :- C++ can process any of the 256 ASCII characters as data or as literals

Example

Out put :-

(B) CONSTANT:

Definition:

A constant, like a variable is a memory location where a value can be stored. Unlike variable, constants, never change in value. You must initialize a constant when it is created C++ has two types of constants. literal and symbolic.

A literal constant is a value typed directly into your program where it is needed.

Example.

```
long width = 5;
```

(C) VARIABLE:

Definition: Variable are containers for storing data values. In C++ there are different types of variable (defined with different keyword),

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,

---

(D): KEY WORDS:

Key words are the reserved keywords that are defined by the Compiler to perform the internal option, written in lower case. Keywords have some meaning which is defined by the Compiler to accomplish a task in code they cannot be used as a variable in programing C++ provide 64 keywords - for break, continue, switch, int float, double, char, try, catch,

while etc.

Example:

```
# include < iostream.h>
 using namespace std;
 int main()
 {
int n;
Cout << "Enter number :" << endl;
 cin >> n;
 if (n > 0)
{

 Cout << " You have entered positive
       number ";
}

 return 0;
}
```

(E) Relational Operators:

Definition: A rational operators is used to check the relationship between two operands.

Example:

```
// checks if a is greater than b
a > b; Here, > is a relational operators.
```

END OF PAPER