**SUMMER final**

**TERM paper**

**NAME #### MUHAMMAD USAMA**

**ID NO #### 16742**

**SUBJECT ####**

**(SUMMER) PROGRAMMING FUNDAMINTALS (C++)**

**TEACHER NAME ####**

**PROF. DR. FAZAL-E-MALIK**

**DEPT #### (BCS) 2ND SEMESTER**

**DATE #### 29/09/2020**

**IQRA NATIONAL UNIVERSITY(PESHAWAR)**

**HAYATABAD PHASE || | .**

**Q NO 1 :**

**Part a):**

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

**ANSWERE**

# If statements in C++::

The ability to control the flow of your program, letting it make decisions on what code to execute, is valuable to the programmer. The if statement allows you to control if a program enters a section of code or not based on whether a given condition is true or false. One of the important functions of the if statement is that it allows the program to select an action based upon the user's input. For example, by using an if statement to check a user entered password, your program can decide whether a user is allowed access to the program.  
Without a conditional statement such as the if statement, programs would run almost the exact same way every time. If statements allow the flow of the program to be changed, and so they allow algorithms and more interesting code.

Before discussing the actual structure of the if statement, let us examine the meaning of TRUE and FALSE in computer terminology. A true statement is one that evaluates to a nonzero number. A false statement evaluates to zero. When you perform comparison with the relational operators, the operator will return 1 if the comparison is true, or 0 if the comparison is false. For example, the check 0 == 2 evaluates to 0. The check 2 == 2 evaluates to a 1. If this confuses you, try to use a cout statement to output the result of those various comparisons (for example cout<< ( 2 == 1 );)

If statements is used to control the program flow based on some condition, it's used to execute some statement code block if the expression is evaluated to true. Otherwise, it will get skipped. This is the simplest way to modify the control flow of the program.

The if statement is used to check a condition and if the condition is true, we run a block of statements (called the if-block), else we process another block of statements (called the else-block). The else clause is optional.

**Statement Purpose:**

The various forms of if statements are Fortran's main branching tool. 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, the if-then-else structure, and the arithmetic if.

## **If statement:**

**Syntax of if statement:**

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.

**Two forms of if statement:**

## **Basic If Statement Syntax:**

The structure of an if statement is as follows:

Here is a simple example that shows the syntax:

|  |
| --- |
| **if** ( 5 < 10 )    cout<<"Five is now less than ten,"; |

Here, we're just evaluating the statement, "is five less than ten", to see if it is true or not; with any luck, it is! If you want, you can write your own full program including iostream and put this in the main function and run it to test.  
  
To have more than one statement execute after an if statement that evaluates to true, use braces, like we did with the body of a function. Anything inside braces is called a compound statement, or a block.  
  
For example:

|  |  |
| --- | --- |
| 1  2  3 | **if** ( TRUE ) {    Execute all statements inside the braces  } |

I recommend always putting braces following if statements. If you do this, you never have to remember to put them in when you want more than one statement to be executed, and you make the body of the if statement more visually clear.

## **Else statement:**

|  |
| --- |
| if ( TRUE ) {    // Execute these statements if TRUE  }  else {    // Execute these statements if FALSE  } |

Sometimes when the condition in an if statement evaluates to false, it would be nice to execute some code instead of the code executed when the

statement evaluates to true. The "else" statement effectively says that whatever code after it (whether a single line or code between brackets) is executed if the if statement is FALSE.  
  
It can look like this:

**If else example::**

#include <iostream>

using namespace std;

int main() {

int marks;

cout<<"Enter marks of the student:";

cin>>marks;

if (marks >= 50)

{

cout<<"congrats! you are pass:";

}else

{

cout<<"sorry! you are fail:";

}

return 0;

}

**Output::**

Enter marks of the student:80

congrats! you are pass:

--------------------------------

Process exited after 4.512 seconds with return value 0

Press any key to continue . . .

**Part b) :**

**Write a C++ program to read two numbers from keyboard and then find the   
LARGEST number of them.**

**PROGRAM CODE::**

#include <iostream>

using namespace std;

int main() {

int a, b, large;

cout<<"please enter any two integer numbers "<<endl;

cin>>a>>b;

if(a>b)

{

large = a;

}else

{

large = b;

}

cout<<" The largest number between this two values is \n "<<large;

return 0;

}

**Output::**

please enter any two integer numbers

34

67

The largest number between this two values is

67

--------------------------------

Process exited after 3.622 seconds with return value 0

Press any key to continue . . .

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Q NO 2:**

**PART A) :**

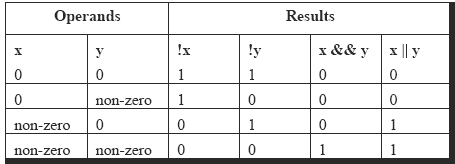
**What are the Logical Operators? Explain them.**

**Answere::**

A logical operator is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depends only on that of the original expressions and on the meaning of the operator. Common logical operators include AND, OR, and NOT.

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](https://www.programiz.com/c-programming/c-if-else-statement" \o "C if else)

| Operator | Meaning | Example |
| --- | --- | --- |
| && | Logical AND. True only if all operands are true | If c = 5 and d = 2 then, expression ((c==5) && (d>5)) equals to 0. |
| || | Logical OR. True only if either one operand is true | If c = 5 and d = 2 then, expression ((c==5) || (d>5)) equals to 1. |
| ! | Logical NOT. True only if the operand is 0 | If c = 5 then, expression !(c==5) equals to 0. |



**Example program of logical opraters::**

// Working of logical operators

#include <iostream>

using namespace std;

int main()

{

int a = 5, b = 5, c = 10, result;

result = (a == b) && (c > b);

cout<<"(a == b) && (c > b) is \n" << result<<"\n\t"<<endl;

result = (a == b) && (c < b);

cout<<"(a == b) && (c < b) is \n"<< result<<"\n\t"<<endl;

result = (a == b) || (c < b);

cout<<"(a == b) || (c < b) is \n"<< result<<"\n\t"<<endl;

result = (a != b) || (c < b);

cout<<"(a != b) || (c < b) is \n"<< result<<"\n\t"<<endl;

result = !(a != b);

cout<<"!(a != b) is \n"<< result<<"\n\t"<<endl;

result = !(a == b);

cout<<"!(a == b) is \n"<< result<<"\n\t"<<endl;

return 0;

}

**Output::**

(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

--------------------------------

Process exited after 0.01722 seconds with return value 0

Press any key to continue . . .

**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.

**ANSWERE::**

**CODE::**

#include <iostream>

using namespace std;

int main() {

int fahrenheit;

cout<<"please enter the tempreture in fahrenheit: "<<endl;

cin>>fahrenheit;

if(fahrenheit >= 40)

{

cout<<"Very Hot";

}

if(fahrenheit >= 35 && fahrenheit < 40)

{

cout<<"Tolerable";

}

if(fahrenheit >= 30 && fahrenheit < 35)

{

cout<<"Warm";

}

if(fahrenheit < 30)

{

cout<<"Cool";

}

return 0;

}

**Outputs::**

please enter the tempreture in fahrenheit:

45

Very Hot

--------------------------------

Process exited after 5.185 seconds with return value 0

Press any key to continue . . .

//ANOTHER EXECUTION OF CODE;

please enter the tempreture in fahrenheit:

38

Tolerable

--------------------------------

Process exited after 6.02 seconds with return value 0

Press any key to continue . . .

//ANOTHER EXECUTION OF CODE;

please enter the tempreture in fahrenheit:

33

Warm

--------------------------------

Process exited after 3.134 seconds with return value 0

Press any key to continue . . .

//ANOTHER EXECUTION OF CODE;

please enter the tempreture in fahrenheit:

23

Cool

--------------------------------

Process exited after 3.651 seconds with return value 0

Press any key to continue . . .

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Q NO 3:**

**PART A) :**

**What does Looping mean? Explain different loops in C++.**

**Answere::**

**Loops::**

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. .

Loops are control structures used to repeat a given section of code a certain number of times or until a particular condition is met.

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

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**

# C++ For Loop

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

**Syntax**

for (statement 1; statement 2; statement 3) {  
  // code block to be executed  
}

Statement 1  is executed (one time) before the execution of the code block.

Statement 2  defines the condition for executing the code block.

Statement 3  is executed (every time) after the code block has been executed.

**Example**

#include <iostream>

using namespace std;

int main() {

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

{

cout<<i<<"\n";

}

}

#### **Example explained**

Statement 1 sets a variable before the loop starts (int i = 1).

Statement 2 defines the condition for the loop to run (i must be less than 5).

If the condition is true, the loop will start over again, if it is false, the loop will end.

Statement 3 increases a value (i++) each time the code block in the loop has been executed.

## **Flowchart of for Loop in C++**

**Flowchart of for loop**



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **While loop C++::**

The syntax of a while loop in C++ is − while(condition) { statement(s); } Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

**SYNTAX::**

while (condition) {  
  // code block to be executed  
}

**EXAMPLE PROGRAM::**

//while loop in c++.

#include <iostream>

using namespace std;

int main() {

int i = 0;

while (i < 5)

{

cout << i << "\n";

i++;

}

return 0;

}

**Output::**

0

1

2

3

4

--------------------------------

Process exited after 0.02707 seconds with return value 0

Press any key to continue . . .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **DO\_WHILE LOOP:**

Post test repetition structure

The statement in the loop block are executed at least once, whether the loops condition is true or false

**EXAMPLE PROGRAM::**

//do\_while loop in c++.

#include <iostream>

using namespace std;

int main() {

int i = 1;

do

{

cout<<"the value of i is= \t"<<i<<endl;

i++;

}

while(i <= 5);

return 0;

}

**Output::**

the value of i is= 1

the value of i is= 2

the value of i is= 3

the value of i is= 4

the value of i is= 5

--------------------------------

Process exited after 0.0168 seconds with return value 0

Press any key to continue . . .

**PART B) :**

**Write a C++ program to read a number from keyboard and then determine whether it is Even or Odd number?**

#include <iostream>

using namespace std;

int main() {

int num;

cout<<"please enter an integer number "<<endl;

cin>>num;

if(num % 2==0)

{

cout<<"your entered number is="<<num<<"\nSo this number is EVEN.";

}else

{

cout<<"your entered number is= "<<num<<"\nSo this number is ODD.";

}

return 0;

}

Output::

please enter an integer number

5

your entered number is= 5

So this number is ODD.

--------------------------------

Process exited after 8.171 seconds with return value 0

Press any key to continue . . .

**Q NO 4:**

**PART A) :**

**What is the purpose of using break and continue statements?**

**Answere::**

**EXAMPLE OF CONTINUE KEYWORD::**

continue statement allows to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed

A continue is usually associated with an if.

**Code::**

//coninue keyword in c++.

#include <iostream>

using namespace std;

int main() {

int a = 10;

while(a < 20)

{

if( a == 15)

{

//skip the iteration.

a = a + 1;

continue;

}

cout<<"the value of a is = "<<a<<endl;

a++;

}

return 0;

}

**Output::**

the value of a is = 10

the value of a is = 11

the value of a is = 12

the value of a is = 13

the value of a is = 14

the value of a is = 16

the value of a is = 17

the value of a is = 18

the value of a is = 19

--------------------------------

Process exited after 0.01596 seconds with return value 0

Press any key to continue . . .

**The break statement::**

There are situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.

The keyword break allows us to do this

When break is encountered inside any loop, control automatically passes to the first statement after the loop.

A break is usually associated with an if.

The keyword break, breaks the control only from the loop in which it is placed

Example: Write a program to determine whether a number is prime or not. A prime number is one, which is divisible only by 1 or itself.

EXAMPLE:

#include <iostream>

using namespace std;

int main(){

int num =10;

while(num<=200) {

cout<<"Value of num is: "<<num<<endl;

if (num==12) {

break;

}

num++;

}

cout<<"Hey, I'm out of the loop";

return 0;

}

Output::

Value of num is: 10

Value of num is: 11

Value of num is: 12

Hey, I'm out of the loop

--------------------------------

Process exited after 0.03764 seconds with return value 0

Press any key to continue . . .

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PART B) :**

**Write a C++ program to find the sum of the following numbers:**

**1+2+3+………+10.**

#include <iostream>

using namespace std;

int main() {

int times, lnp, sum, i;

times = lnp = sum = i = 0;

cout<<"please enter the number of integers you want to add: "<<endl;

cin>>times;

for(i = times; i > 0; i--)

{

cout<<"Enter an integer number: \t ";

cin>>lnp;

sum = sum + lnp;

}

cout<<" \n ";

cout<<"The sum of your "<<times <<" numbers is \n"<<sum;

cout<<" . "<<endl;

return 0;

}

**Output::**

please enter the number of integers you want to add:

10

Enter an integer number: 1

Enter an integer number: 2

Enter an integer number: 3

Enter an integer number: 4

Enter an integer number: 5

Enter an integer number: 6

Enter an integer number: 7

Enter an integer number: 8

Enter an integer number: 9

Enter an integer number: 10

The sum of your 10 numbers is

55 .

--------------------------------

Process exited after 14.44 seconds with return value 0

Press any key to continue . . .

**Q NO 5:**

**Explain the the following with proper examples.**

1. **C++ Character set.**
2. **Constants.**
3. **Variables.**
4. **Keywords.**
5. **Relational Operators.**

**ANSWERE:**

1. **Relational Operators.**

**C++ Relational Operators**

In computer science, a relational operator is a programming language construct or operator that tests or defines some kind of relation between two entities. These include numerical equality (e.g., 5 = 5) and inequalities (e.g., 4 ≥ 3). ... Relational operators can be seen as special cases of logical predicates.

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.

n C++ Programming, the values stored in two variables can be compared using following operators and relation between them can be determined.

Various C++ relational operators available are-

**Operator, Meaning:**

> Greater than  
>= Greater than or equal to  
== Is equal to  
!= Is not equal to   
<= Less than or equal to

< Less than.

**CODE::**

//SIMPLE RELATIONAL OPRATERS

#include<iostream>

using namespace std;

int main ()

{

int a = 25, b = 5;

cout << "Simple Relational Operator Example Program \n";

if( a>b )

cout<<"A is Big";

else if( a== b)

cout<<"A and B are Equal";

else

cout<<"B is Big";

return 0;

}

**OUTPUT::**

Simple Relational Operator Example Program

A is Big

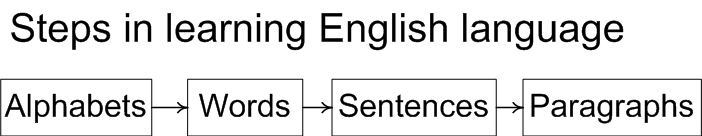
--------------------------------

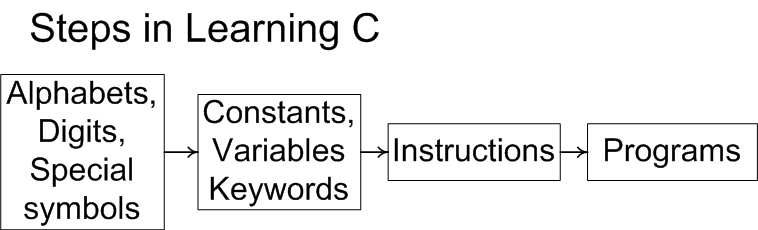
Process exited after 0.01774 seconds with return value 0

Press any key to continue . . .

1. **C++ Character set.**

In C++, character set is a set of all valid characters that can be used in a C++ Program. Characters set is used to specify the characters or symbols recognized by the language. Character set is a set of all valid characters that can be used to form words, numbers and expression's in source programs.





**C++ character set::**

**Code example::**

#include <iostream>

using namespace std;

int main()

{

char letter, digit, special, white;

cout<<"W3Adda - C++ Character Set Example"<<endl;

cout<<"Enter a Letter : ";

cin>>letter;

cout<<"You entered a letter '"<<letter<<"'"<<"\n";

cout<<"Enter a Digit : ";

cin>>digit;

cout<<"You entered a digit '"<<digit<<"'"<<"\n";

cout<<"Enter a special character : ";

cin>>special;

cout<<"You entered a special character '"<<special<<"'"<<"\n";

cout<<"A horizontal(\t) tab";

return 0;

}

**Output::**

W3Adda - C++ Character Set Example

Enter a Letter : e

You entered a letter 'e'

Enter a Digit : E

You entered a digit 'E'

Enter a special character : 4

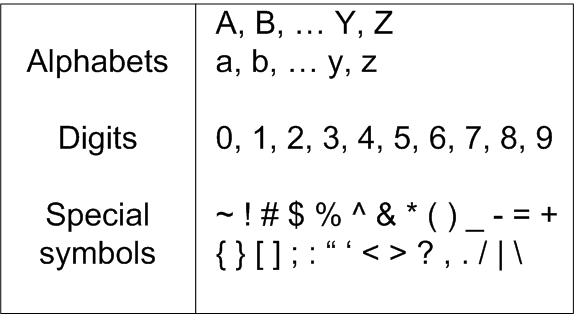
You entered a special character '4'

A horizontal( ) tab

--------------------------------

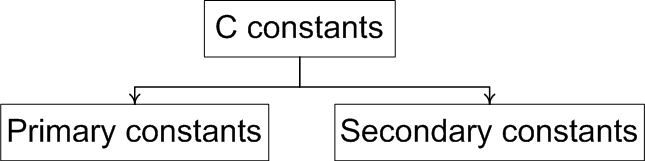
Process exited after 6.708 seconds with return value 0

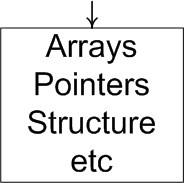
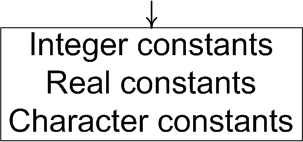
Press any key to continue . . .



1. **Constants.**

Constant is an entity that doesn’t change





**Integer constant::**

**Rules:**

Must have at least one digit

It must not have a decimal point

It can be either positive or negative

If there is no sign an integer constant is assumed to be positive

No commas or blanks are allowed within an integer constant

Examples: 135, -67, 3401, -5670

**Real constant::**

**Rules:**

Also called Floating Point constants

A real constant must have at least one digit

It must have a decimal point

It could be either positive or negative

Default sign is positive

No commas or blanks are allowed within a real constant

Examples: +325.34, 426.0, -32.76, -48.5792

**Character constant::**

A character constant is a

single alphabet

a single digit

or a single special symbol

Enclosed within single inverted commas, ‘ ’

Both the inverted commas should point to the left

The maximum length can be 1 character

Examples: ’A’, ‘I’, ‘5’, ‘=‘

1. **Variables.**

An entity that may change dSeries of characters (letters, digits, underscores)

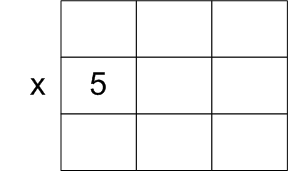
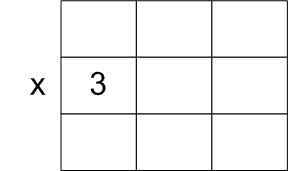
Must begin with a letter or underscore

Case sensitive

Meaningful naming scheme

during program execution

These are names given to locations in memory.



No commas or blanks are allowed within a variable name.

No Special symbol are used in name.

Examples: Interger1, Sum, \_FirstNum

Invalid variable names

#sum, 12x, first name

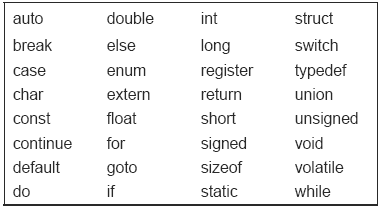
1. **Keywords.**

These are reserved words

Compiler knows their meaning

Cannot be used as variable name

Cannot be changed.



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*The END\*\*\*\*\*\*\*\***