

| **TITLE:** Write a program to accept 3 numbers from the user and find the largest of the 3 numbers using                    If - else if-else                    Ternary operator |
| --- |

**AIM:** Write a program to accept 3 numbers from the user and find the largest of the 3 numbers using

                  If - else if-else

                  Ternary operator

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in ANSI C, E. Balagurusamy, 7 th Edition, 2016, McGraw-Hill Education, India.
2. Structured Programming Approach, Pradeep Dey and Manas Ghosh, 1 st Edition, 2016, Oxford University Press, India.
3. Let Us C, Yashwant Kanetkar, 15th Edition, 2016, BPB Publications, India.

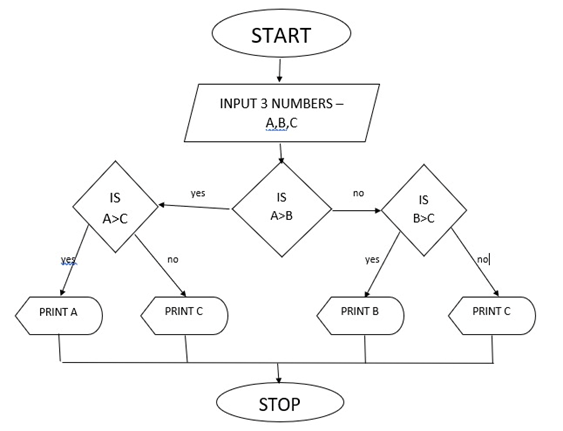
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

Ask user to input three numbers. Compare three numbers to find the largest of them using

1. Nested if else statement
2. Using ternary operator

**Algorithm:**



1. Start
2. Declare variable a for number 1, b for number 2, c for number 3
3. Print statement user to input 3 numbers
4. Take in 3 numbers from user as values for a, b, c
5. Check if a > b if true go to step 6 else goto step 9
6. Check if b > c if true go to step 7 else goto step 8
7. Print “A is the greatest number”
8. Print “B is the greatest number”
9. Check if b > c if true go step 10 else go to step 11
10. Print “B is the greatest number”
11. Print “C is the greatest number”
12. Stop

**Implementation details:**

#include <stdio.h>

int main(void)

{

int a, b, c;

printf("Enter three numbers\n");

scanf("%d", &a);

scanf("%d", &b);

scanf("%d", &c);

if (a > b)

{

if (a > c)

{

printf("%d is the greatest number\n", a);

}

else

{

printf("%d is the greatest number", c);

}

}

else

{

if (b > c)

{

printf("%d is the greatest number", b);

}

else

{

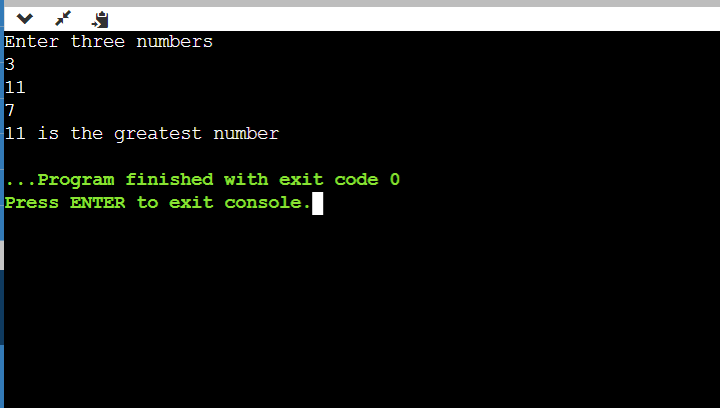
printf("%d is the greatest number", c);

}

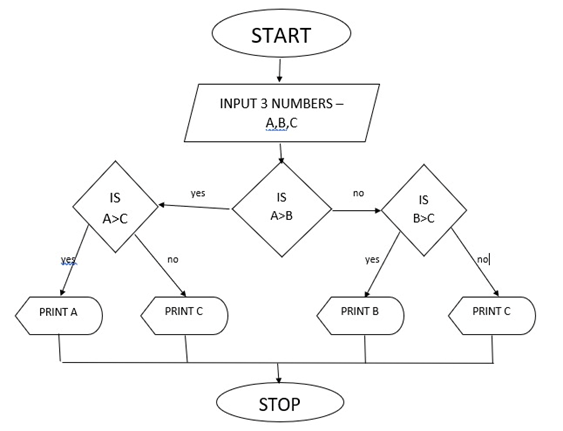
}

}

**Output(s):**



2-Using ternary operator



1. Start
2. Declare variable a for number 1, b for number 2, c for number 3
3. Print statement user to input 3 numbers
4. Take in 3 numbers from user as values for a, b, c
5. Check if a > b if true go to step 6 else goto step 9
6. Check if b > c if true go to step 7 else goto step 8
7. Print “A is the greatest number”
8. Print “B is the greatest number”
9. Check if b > c if true go step 10 else go to step 11
10. Print “B is the greatest number”
11. Print “C is the greatest number”
12. Stop

**Implementation details:**

#include <stdio.h>

int main(void)

{

int a, b, c;

printf("Enter three numbers\n");

scanf("%d", &a);

scanf("%d", &b);

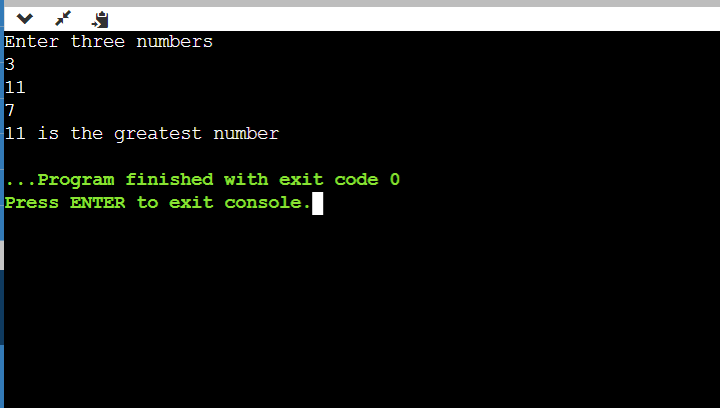
scanf("%d", &c);

(a > b)? ((a > c)? printf("%d is the greatest number\n", a) : printf("%d is the greatest number\n", c)):

((b > c)? printf("%d is the greatest number\n", b) : printf("%d is the greatest number\n", c));

}

**Output(s):**



**Conclusion:**

Thus, we learnt how to use if-else-if and ternary operator.

**Post Lab Descriptive Questions**

1. **Explain bitwise operators with examples**
2. **Write associative rules and precedence table of various operators.**

1]Bitwise operators:

The Bitwise Operator in C performs its operation on the individual bits of its operand, where operands are values or expressions on which an operator operates. We can further subcategorize bitwise operators into three subtypes based on their working principles, logical (Bitwise AND, OR, and XOR), Shift (Right Shift and left shift), and Complement (Bitwise NOT).

The result is also always a bit .

Eg. &, |, ~, ^ , <<, >>

| **X** | **y** | **x & y** | **x | y** | **x ^ y** |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

2] **Associative rules and precedence table of various operators :**

| **Operator** | **Description of Operator** | **Associativity** |
| --- | --- | --- |
| . | Direct member selection | Left to right |
| -> | Indirect member selection | Left to right |
| [] | Array element reference | Left to right |
| () | Functional call | Left to right |
| ~ | Bitwise(1’s) complement | Right to left |
| ! | Logical negation | Right to left |
| – | Unary minus | Right to left |
| + | Unary plus | Right to left |
| — | Decrement | Right to left |
| ++ | Increment | Right to left |
| \* | Pointer reference | Right to left |
| & | Dereference (Address) | Right to left |
| (type) | Typecast (conversion) | Right to left |
| sizeof | Returns the size of an object | Right to left |
| % | Remainder | Left to right |
| / | Divide | Left to right |
| \* | Multiply | Left to right |
| – | Binary minus (subtraction) | Left to right |
| + | Binary plus (Addition) | Left to right |
| >> | Right shift | Left to right |
| << | Left shift | Left to right |
| > | Greater than | Left to right |
| < | Less than | Left to right |
| >= | Greater than or equal | Left to right |
| <= | Less than or equal | Left to right |
| == | Equal to | Left to right |
| != | Not equal to | Left to right |
| ^ | Bitwise exclusive OR | Left to right |
| & | Bitwise AND | Left to right |
| || | Logical OR | Left to right |
| | | Bitwise OR | Left to right |
| ?: | Conditional Operator | Right to left |
| && | Logical AND | Left to right |
| , | Separator of expressions | Left to right |
| = | Simple assignment | Right to left |
| /= | Assign quotient | Right to left |
| \*= | Assign product | Right to left |
| %= | Assign remainder | Right to left |
| -= | Assign difference | Right to left |
| += | Assign sum | Right to left |
| |= | Assign bitwise OR | Right to left |
| ^= | Assign bitwise XOR | Right to left |
| &= | Assign bitwise AND | Right to left |
| >>= | Assign right shift | Right to left |
| <<= | Assign left shift | Right to left |

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**