LIST OF PROGRAMS

S.NO	NAME OF EXPERIMENTS	Date
1.	**Numbers**	**04 Sept 2021**
2.	**Array, String**	**09 Sept 2021**
3.	**Function**	**16 Sept 2021**
4	**Class and Object**	**30 Sept 2021**
5	**Constructor and Destructor**	**04 Oct 2021**
6		
7		
8		
9		
10		
11		
12		
13		
14		

Experiments – 01

AIM-Programs on Numbers, String

01. Find LCM

LCM of two integers **a** and **b** is the smallest positive integer that is divisible by both **a** and **b**.

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main()
{
    int n1, n2, max;
    cout << "Enter two numbers: ";
    cin >> n1 >> n2;
    // maximum value between n1 and n2 is stored in max
    max = (n1 > n2) ? n1 : n2;
    do
    {
        if (max % n1 == 0 && max % n2 == 0)
        {
            cout << "LCM = " << max;
            break;
        }
        else
            ++max;
    } while (true);
    return 0;
}
```

Output

```
/tmp/x7F43gC4yU.o
Enter two numbers: 12 18
LCM = 36
```

In above program, user is asked to integer two integers **n1** and **n2** and largest of those two numbers is stored in **max**.

It is checked whether **max** is divisible by **n1** and **n2**, if it's divisible by both numbers, **max** (which contains LCM) is printed and loop is terminated.
If not, value of **max** is incremented by 1 and same process goes on until **max** is divisible by both **n1** and **n2**.

02. C++ SQRT

The **sqrt()** function in C++ returns the square root of a number. This function is defined in the **cmath** header file.
Mathematically, **sqrt(x) = √x.**

sqrt() Syntax

The syntax of the **sqrt()** function is:

```
sqrt(double num);
```

sqrt() Parameters

The **sqrt()** function takes the following parameter:

- **num** - a non-negative number whose square root is to be computed

Note: If a negative argument is passed to sqrt(), domain error occurs.

sqrt() Return Value

The **sqrt()** function returns:

- the square root of the given argument

sqrt() Prototypes

The prototypes of **sqrt()** as defined in the **cmath** header file are are:

```
double sqrt(double x);

float sqrt(float x);
```

long double sqrt(long double x);

// for integral type

double sqrt(T x);

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  double num = 10.25;
  double result = sqrt(num);
  cout << "Square root of " << num << " is " << result<<"\n";

  long num2 = 464453422;
  result= sqrt(num2);
  cout << "Square root of " << num2 << " is " << result<<"\n";
  return 0;
}
```

Output

```
/tmp/x7F43gC4yU.o
Square root of 10.25 is 3.20156
Square root of 464453422 is 21551.2
```

03. C++ cbrt()

The cbrt() function in C++ returns the cube root of a number.

[**Mathematics**] $\sqrt[3]{x}$ = **cbrt(x)** [In C Programming]

This function is defined in **<cmath>** header file.

cbrt() prototype [As of C++ 11 standard]

double cbrt(double x);

float cbrt(float x);

long double cbrt(long double x);

double cbrt(T x); // For integral type

cbrt() Parameters

The cbrt() function takes a single argument whose cube root is to be calculated.

cbrt() Return value

The cbrt() function returns the cube root of the given argument.

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    long x = 964353422;
    double result = cbrt(x);
    cout << "Cube root of " << x << " is " << result << endl;
    x = -1000.0;
    result = cbrt(x);
    cout << "Cube root of " << x << " is " << result << endl;
    return 0;
}
```

Output

```
/tmp/xmvb6Bt0w4.o
Cube root of 964353422 is 987.974
Cube root of -1000 is -10
```

04. Simple Calculator

To understand this example, you should have the knowledge of the following C++ programming topics:

- C++ switch..case Statement
- C++ break Statement
- C++ continue Statement

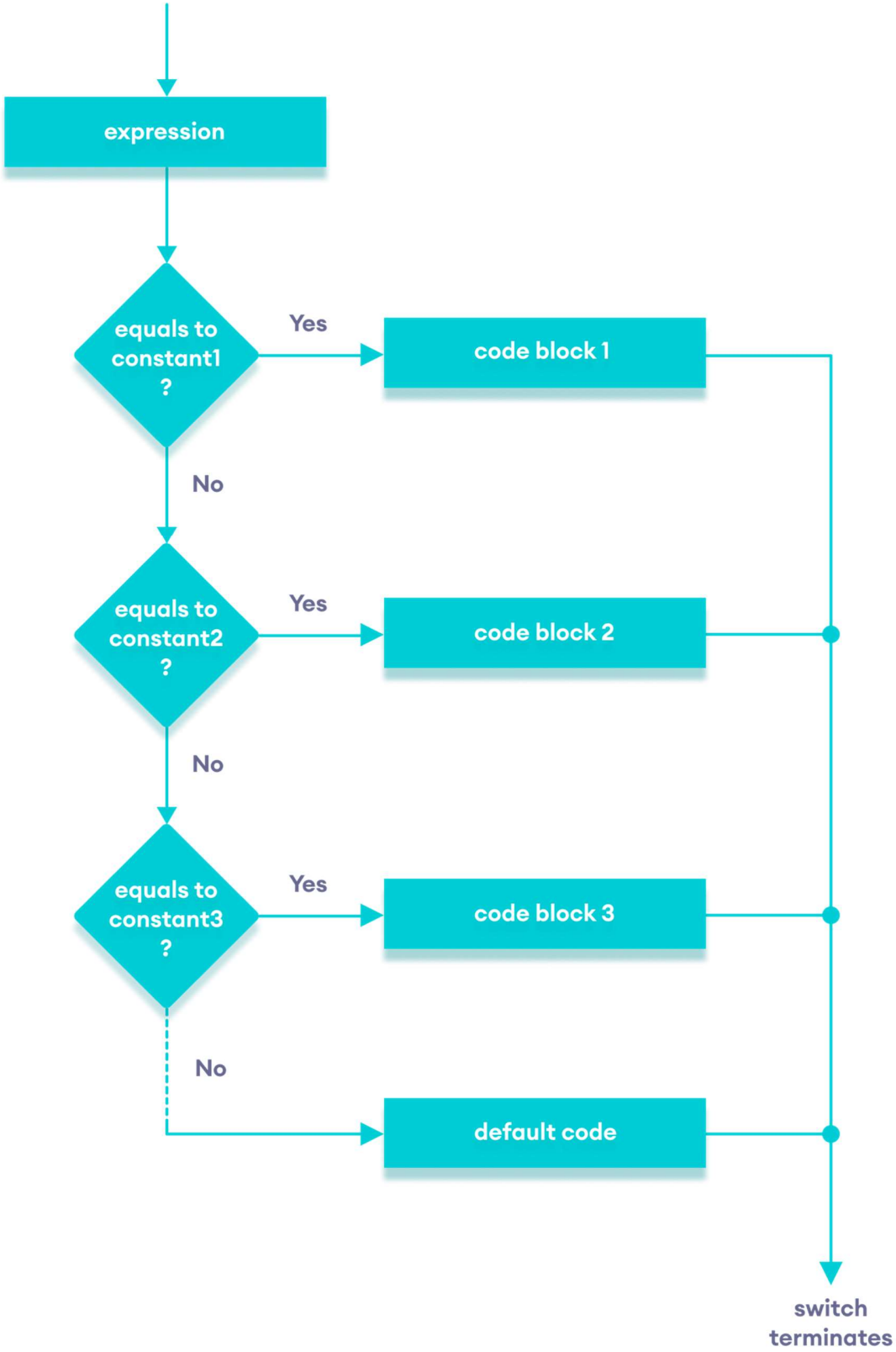
This program takes an arithmetic operator (+, -, *, /) and two operands from a user and performs the operation on those two operands depending upon the operator entered by the user.

How does the switch statement work?

The `expression` is evaluated once and compared with the values of each `case` label.

- If there is a match, the corresponding code after the matching label is executed. For example, if the value of the variable is equal to `constant2`, the code after `case constant2:` is executed until the break statement is encountered.
- If there is no match, the code after `default:` is executed.

Note: We can do the same thing with the if...else..if ladder. However, the syntax of the switch statement is cleaner and much easier to read and write.


expression
equals to constant1 ?
Yes
code block 1
No
equals to constant2 ?
Yes
code block 2
No
equals to constant3 ?
Yes
code block 3
No
default code
switch terminates

Working of C++ break Statement

In C++, the `break` statement terminates the loop when it is encountered.

The syntax of the `break` statement is:

break;

```
for (init; condition; update) {
    // code
    if (condition to break) {
        break;
    }
    // code
}
```

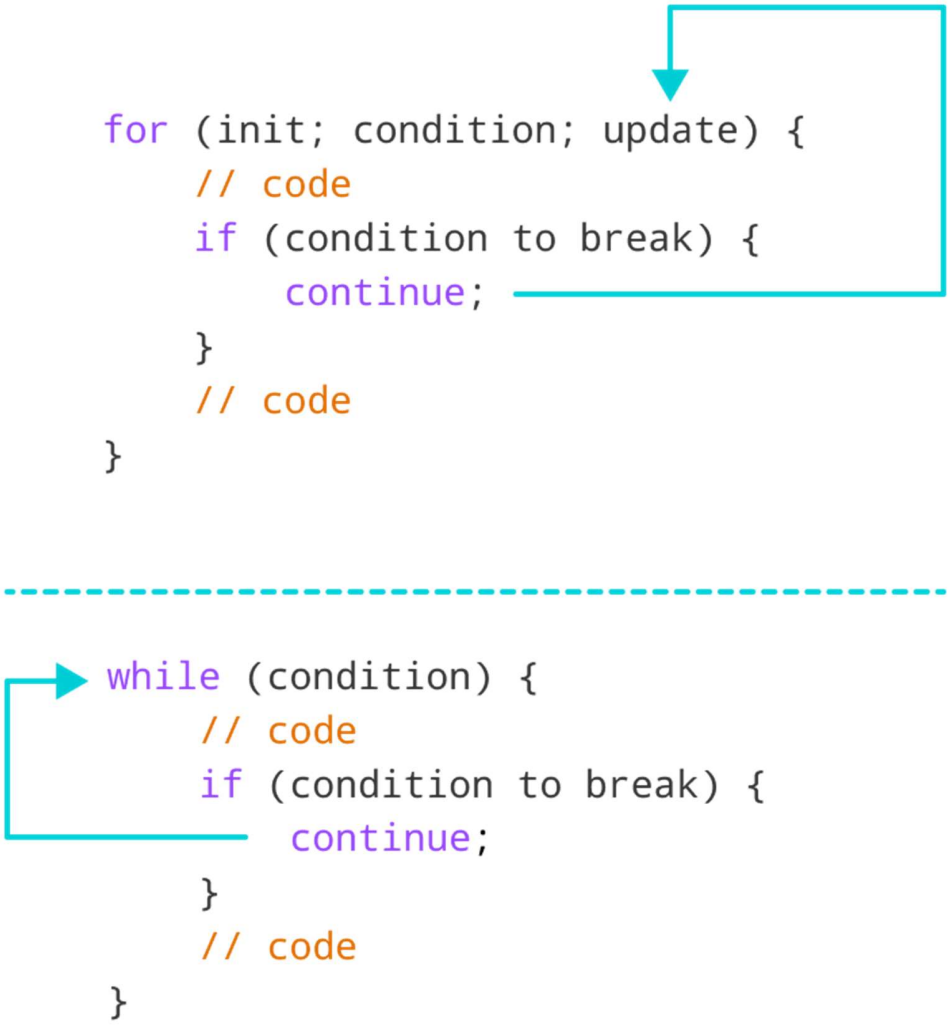
```
while (condition) {
    // code
    if (condition to break) {
        break;
    }
    // code
}
```

Working of C++ continue Statement

In computer programming, the `continue` statement is used to skip the current iteration of the loop and the control of the program goes to the next iteration.

The syntax of the `continue` statement is:

continue;



main.cpp

```cpp
// Online C++ compiler to run C++ program online
# include <iostream>
using namespace std;
int main() {
    char op;
    float num1, num2;
    cout << "Enter operator: +, -, *, /: "; cin >> op;
    cout << "Enter two operands: "; cin >> num1 >> num2;
    switch(op) {
        case '+':
            cout<<num1<<" + "<<num2<<" = "<<num1+num2;  break;
        case '-':
            cout<<num1<<" - "<<num2<<" = "<<num1-num2;  break;
        case '*':
            cout<<num1<<" * "<<num2<<" = "<<num1*num2;  break;
        case '/':
            if(num2==0){    cout<<"operation is not Possible Zero Error!!!!!";    break;  }
            cout<<num1<<" / "<<num2<<" = "<<num1/num2;    break;
        default:
//If the operator is other than +,-,* or /, error message is shown
            cout << "Error! operator is not correct";    break;
    }
    return 0;
}
```

Output

```
/tmp/xmvb6BtOw4.o
Enter operator: +, -, *, /: /
Enter two operands: 12
4
12 / 4 = 3
```

In the above program, we are using the `switch...case` statement to perform addition, subtraction, multiplication, and division.

How This Program Works

1. We first prompt the user to enter the desired operator. This input is then stored in the `char` variable named `oper`.
2. We then prompt the user to enter two numbers, which are stored in the float variables `num1` and `num2`.
3. The `switch` statement is then used to check the operator entered by the user:
    - If the user enters `+`, addition is performed on the numbers.
    - If the user enters `-`, subtraction is performed on the numbers.
    - If the user enters `*`, multiplication is performed on the numbers.
    - If the user enters `/`, division is performed on the numbers.
    - If the user enters any other character, the default code is printed.

Notice that the `break` statement is used inside each `case` block. This terminates the `switch` statement.
If the `break` statement is not used, all cases after the correct `case` are executed.

05. Leap Year

To understand this example, you should have the knowledge of the following C++ programming topics:

- C++ if, if...else and Nested if...else

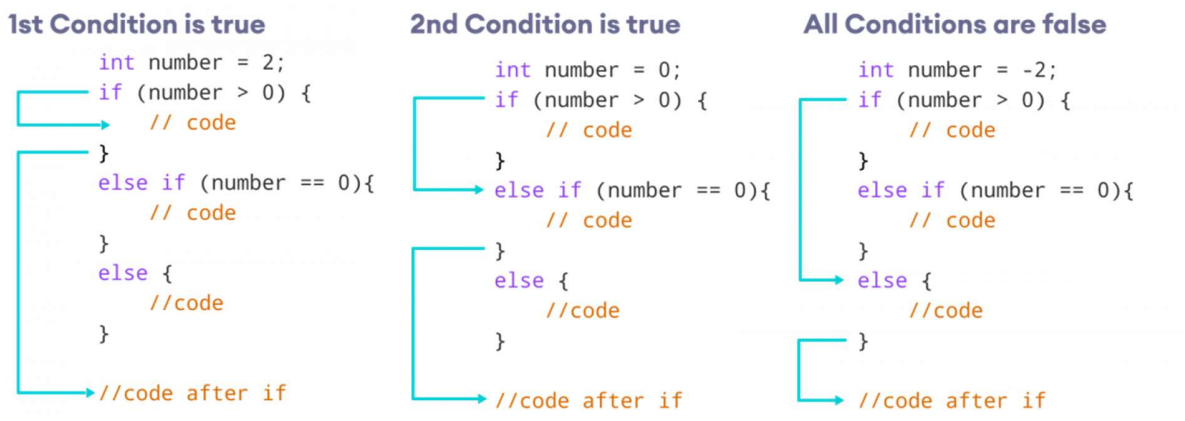
C++ if...else...else if statement

The `if...else` statement is used to execute a block of code among two alternatives. However, if we need to make a choice between more than two alternatives, we use the `if...else if...else` statement.
The syntax of the `if...else if...else` statement is:

```
    if (condition1) {
      // code block 1
    }
    else if (condition2){
      // code block 2
    }
    else {
      // code block 3
}
```

Here,

- If `condition1` evaluates to `true`, the `code block 1` is executed.
- If `condition1` evaluates to `false`, then `condition2` is evaluated.
- If `condition2` is `true`, the `code block 2` is executed.
- If `condition2` is `false`, the `code block 3` is executed.



Note: There can be more than one else if statement but only one if and else statements.

All years which are perfectly divisible by 4 are leap years except for century years (years ending with 00) which is leap year only it is perfectly divisible by 400.

For example: 2012, 2004, 1968 etc are leap year but, 1971, 2006 etc are not leap year. Similarly, 1200, 1600, 2000, 2400 are leap years but, 1700, 1800, 1900 etc are not.

In this program below, user is asked to enter a year and this program checks whether the year entered by user is leap year or not.

main.cpp | Run | Output | Clear

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;

int main() {
    int year;

    cout << "Enter a year: ";
    cin >> year;

    if (year % 4 == 0) {
        if (year % 100 == 0) {
            if (year % 400 == 0)
                cout << year << " is a leap year.";
            else
                cout << year << " is not a leap year.";
        }
        else
            cout << year << " is a leap year.";
    }
    else
        cout << year << " is not a leap year.";

    return 0;
}
```

```
/tmp/xmvb6Bt0w4.o
Enter a year: 2078
2078 is not a leap year.
```

Here, we have used nested `if` statements to check whether the year given by the user is a leap year or not.
First, we check if `year` is divisible by 4 or not. If it is not divisible, then it is not a leap year.
If it is divisible by 4, then we use an inner `if` statement to check whether `year` is divisible by 100.
If it is not divisible by 100, it is still divisible by 4 and so it is a leap year.

We know that the century years are not leap years unless they are divisible by 400.

So, if `year` is divisible by 100, another inner `if` statement checks whether it is divisible by 400 or not.
Depending on the result of that innermost `if` statement, the program determines whether `year` is a leap year or not.