Computer Programming Fundamentals

Handouts No. 5

1. Math Functions
2. Operators (Part 2)
    a. Comparison Operators
    b. Logical Operators
3. Flow control (Part 1)
    a. Conditions
        i. if else
4. Errors in C++

1. Math Functions

C++ allows us to perform mathematical operations through the functions defined in <cmath> header file. The <cmath> header file contains various methods for performing mathematical operations. The commonly used functions of cmath header file are given below.

Function	Description
round()	This function returns the nearest integer value of the double argument passed to this function. If decimal value is from ".1 to .4", it returns integer value less than the argument. If decimal value is from ".5 to .9", it returns the integer value greater than the argument.
floor()	This function returns the nearest integer which is less than or equal to the argument passed to this function.
ceil()	This function returns nearest integer value which is greater than or equal to the argument passed to this function.
sqrt()	This function is used to find square root of the argument passed to this function.
pow()	This is used to find the power of the given number.

Example

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double f;

    f = round(3.5);
    cout<<"round => "<<f<<endl;
    f = floor(3.9);
    cout<<"floor => "<<f<<endl;
    f = ceil(3.1);
    cout<<"ceil => "<<f<<endl;
    f = sqrt(25);
    cout<<"sqrt => "<<f<<endl;
    f = pow(2,3);
    cout<<"pow => "<<f<<endl;
    return 0;
}
```

cmath also contains some constants like M_PI with hold the mathematical π value, which is 3.14159265358979323846

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   cout<<" PI = "<<M_PI<<endl;
    return 0;
}
```

2. Operators (Part 2)

2.1. Comparison Operators

The comparison operators compare two values and return either true or false, represented as 1 or 0. They are mainly used to specify conditions, which are expressions that evaluate to either true or false.

```
int x = (2 == 3);                   /* 0 - equal to */
x = (2 != 3);                       /* 1 - not equal to */
x = (2 > 3);                        /* 0 - greater than */
x = (2 < 3);                        /* 1 - less than */
x = (2 >= 3);                       /* 0 - greater than or equal to */
x = (2 <= 3);                       /* 1 - less than or equal to */
```

2.2. Logical Operators

The logical operators are often used together with the comparison operators. Logical and (&&) evaluates to true if both the left and right sides are true, and logical or (||) is true if either the left or right side is true. For inverting a Boolean result there is the logical not (!) operator. Note that for both "logical and" and "logical or" the right-hand side will not be evaluated if the result is already determined by the left-hand side.

2.2.1. Logical AND (&&)

Logical AND takes two operands and returns true only if both of the operands are true as in the following table.

Note: false is represented by 0 and true is represented by 1.

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

2.2.2. Logical OR (||)

Logical OR takes two operands and returns false only if both of the operands are false as in the following table.

Note: false is represented by 0 and true is represented by 1.

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

2.2.3. Logical NOT (!)

Logical Not takes single operand and returns the opposed value as in the following table

Note: false is represented by 0 and true is represented by 1.

A	! A
0	1
1	0

Example

```
int a=10,b=20,c=30,x;
x = b < c && b > a ;       // x = 1
x = c == b || c == a ;     // x = 0
x = ! (b > a) ;              // x = 0
```

Operator Precedence

In C++, expressions are normally evaluated from left to right. However, when an expression contains multiple operators, the precedence of those operators decides the order in which they are evaluated. The order of precedence can be seen in the following table, where the operator with the lowest precedence will be evaluated first. This same order also applies to many other languages, such as C++ and C#.

`() [] x++ x--`
`! ++x --x`
`* / %`
`+ -`
`< <= > >=`
`== !=`
`&&`
`||`
`= op=`
`,`

3. Flow Control

3.1. Conditions

Conditional statements are used to execute different code blocks based on different conditions.

3.1.1. if else

The if statement will only execute if the expression inside the parentheses is evaluated to true. In C, this does not have to be a Boolean expression. It can be any expression that evaluates to a number, in which case zero is false and all other numbers are true.

```
int target = 10;
if (target == 10) {
    cout<<"Target is equal to 10";
}
```

another example:

```
int a = 1;
int b = 2;

if (a < b) {
    cout<<"A is smaller than B.";
}

if (a > b) {
    cout<<"A is greater than B.";
}
```

The if statement can have else clause, which will execute if previous conditions is false.

```
int a = 1;
int b = 2;

if (a < b) {
    cout<<"A is smaller than B.";
}
else {
    cout<<"A is greater than B.";
}
```

To test for other conditions, the if statement can be extended by any number of else/if clauses.

```
int a = 1;
int b = 2;

if (a < b) {
    cout<<"A is smaller than B.";
} else if (a == b) {
    cout<<"A is equal to B.";
} else {
    cout<<"a is greater than B.";
}
```

4. Errors in C++

Errors are the problems or the faults that occur in the program, which makes the behavior of the program abnormal, and experienced developers can also make these faults. Programming errors are also known as the bugs or faults, and the process of removing these bugs is known as debugging.

These errors are detected either during the time of compilation or execution. Thus, the errors must be removed from the program for the successful execution of the program.

There are mainly four types of errors exist in C++ programming:

1. Syntax error
   Syntax errors are also known as the compilation errors as they occurred at the compilation time, or we can say that the syntax errors are thrown by the compilers. These errors are mainly occurred due to the mistakes while typing or do not follow the syntax of the specified programming language. These mistakes are generally made by beginners only because they are new to the language. These errors can be easily debugged or corrected.
   For example:

   If we want to declare the variable of type integer,

   ```
   int a; // this is the correct form
   Int a; // this is an incorrect form.
   ```
   Commonly occurred syntax errors are:

   - If we miss the parenthesis (}) while writing the code.
   - Displaying the value of a variable without its declaration.
   - If we miss the semicolon (;) at the end of the statement.

2. Run-time error
   Sometimes the errors exist during the execution-time even after the successful compilation known as run-time errors. When the program is running, and it is not able to perform the operation is the main cause of the run-time error. The division by zero is the common example of the run-time error. These errors are very difficult to find, as the compiler does not point to these errors.

Let's understand through an example.

```
#include <iostream>
using namespace std;
int main()
{
    int a=2;
    int b=2/0;
    cout<<"The value of b is : "<< b <<endl;
    return 0;
}
```

3. Logical error
   The logical error is an error that leads to an undesired output. These errors produce the incorrect output, but they are error-free, known as logical errors. These types of mistakes are mainly done by beginners. The occurrence of these errors mainly depends upon the logical thinking of the developer. If the programmers sound logically good, then there will be fewer chances of these errors.

4. Semantic error

   Semantic errors are the errors that occurred when the statements are not understandable by the compiler.

   The following can be the cases for the semantic error:

   Undefined variable

```
#include <iostream>
using namespace std;

int main()

{

    a = 10;

    cout<<"The value of a is : "<< a <<endl;

   return 0;

}
```

   Use of an un-initialized variable.

```
int i;

i=i+2;
```

   Type compatibility

```
int b = "javatpoint";
```

Errors in expressions

```
int a, b, c;

a+b = c;
```

Exercises

1. Write C++ program that calculates the distance between two points
   Where the points are P1(X_1 , Y_1) and P2(X_2 , Y_2), and

   Distance = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

2. Write C++ program that checks if the input number is even or odd and prints the result.
3. Write C++ program that reads an integer number from the user and print the corresponding scale in the table below

90 – 100	Excellence
80 – 89	Very good
70 – 79	Good
60 -69	Average
50 – 59	Acceptable
0 – 49	Weak

4. Write C++ program that checks if the input is perfect square or not.
   for example, 25 is perfect square of 5 but 26 is not.
5. Write C++ program that checks if the input year number is perfect leap year or not.
   2020 is leap year but 2021 is not.
6. Write C++ program that represents a simple calculator which performs the main four arithmetic operations.
7. Write C++ program that calculates the number of hours, minutes and the remaining seconds in any enter number of seconds by the user.
   for example, if the input is 4000
   the output must be

```
 Hours: 1     Minutes: 6   Seconds: 40
```