Date: March 7, 2023 Class 1: Basics of programming.

Class notes

A. Data types:

1. Integer (**int**)
   These are whole numbers that can be positive, negative, or zero. They are represented by the "int" keyword.

```
int x = 5;
```

2. Float/Decimal (float)
   These are numbers with a fractional part. They can be represented by the "float" and "double" keywords.

```
float pi = 3.14;
```

3. Boolean (bool)
   This data type is used to store either true or false values. A boolean variable can only hold one of two values, either true or false.

```
bool isAppleRed = true;
```

4. Character (char)
   This data type is used to store individual characters. A char variable can hold any single character, such as 'a' or '9'.

```
char c = 'a';
```

5. String (string)
   String is used to store a sequence of characters.
   The "string" data type is very useful for handling text data

```
string name = "NIT-W";
```

B. Printing things on screen.

In C++, cout is like a tool to display words and numbers on the computer screen. You can use it to show messages or output results from a program. To use cout, you type cout << followed by the message or data you want to display.

```
cout<<"Hello world!";

// Hello world

int A = 156;

cout<<A;

// 156
```

C. Operators:

1. Arithmetic operators

Arithmetic operators are used to perform mathematical operations such as addition, subtraction, multiplication, and division. These operators include + (addition), - (subtraction), * (multiplication), / (division), and % (modulus).

```
int A = 6;
int B = 3;
int sum = A + B; // 9
int sub = A - B; // 3
int mul = A * B; // 18
int div = A / B; // 2
int mod = A % B; // 0
```

2. Logical operators

Logical operators are used to combine two or more Boolean expressions and return a Boolean value based on the result of the combination. The logical operators in C++ are && (logical AND), || (logical OR), and ! (logical NOT).

```
bool isAbove6ft = true;
bool haveClearedTheExam = true;

if (isAbove6ft && haveClearedTheExam){
       // both condition satisfied
}
if (isAbove6ft || haveClearedTheExam){
       // any one of the condition satisfied.
}
if(!isAbove6ft){
       // is not above 6 ft.
}
```

3. Comparison operators

Comparison operators are used to compare two values and return a Boolean value (true or false) based on the result of the comparison. These operators include == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to).

```
if (raviAge == rajuAge){

       // ravi and raju are of same age.

}

if (raviAge != rajuAge){

       // ravi and raju are not of same age.

}

if (raviAge > rajuAge){

       // ravi's age is greater then raju.

}

if (raviAge < rajuAge){

       // ravi's age is lesser then raju.

}

if (raviAge >= rajuAge){
        // ravi’s age is greater or equal to raju.

}

if (raviAge <= rajuAge){

        // ravi’s age is lesser or equal to raju.

}
```

4. Assignment operators
   The basic assignment operator is =.

D. Conditional statements:

1. **If**
   In C++, an if statement is a conditional statement that allows you to execute certain code only if a certain condition is met. The syntax for an if statement is as follows:
   ```
   if (condition) {
       // code to execute if condition is true
   }
   ```
2. **If-else**
   You can also include an else statement with the if statement to execute code when the condition is false. The syntax for this is:

   ```
   if (condition) {
       // code to execute if condition is true
   }
   else {
       // code to execute if condition is false
   }
   ```

3. **else-if**
   You can also use multiple if statements together with else if statements. The syntax for this is:
   ```
   if (condition1) {
       // code to execute if condition1 is true
   }
   else if (condition2) {
       // code to execute if condition1 is false and condition2 is true
   }
   else {
   ```

```
    // code to execute if condition1 and condition2 are false
}
```

Here, if condition1 is true, the code inside the first set of curly braces will be executed, and if condition1 is false but condition2 is true, the code inside the second set of curly braces will be executed. If both condition1 and condition2 are false, the code inside the third set of curly braces will be executed.

4. **Switch**

In C++, switch is a control flow statement that allows you to select a block of code to execute based on the value of a given expression. The syntax for a switch statement is as follows:

```
switch (expression) {
    case value1:
        // code to execute if expression == value1
        break;
    case value2:
        // code to execute if expression == value2
        break;
    // add more cases as needed
    default:
  // code to execute if expression does not match any of the case values
       break;
```

Here, expression is a variable or an expression whose value will be compared against the values in the case statements. The case statements list possible values that the expression can take, and each case is followed by a block of code to execute if the expression matches the value of the case. The default statement is optional and specifies the code to execute if the expression does not match any of the case values.

When the switch statement is executed, the expression is evaluated, and then the code block corresponding to the matching case statement is executed. If there is no match, then the code block associated with the default statement is executed. The break statement is used to exit the switch block after the corresponding code has been executed.