**Naming Convention in C++**

Names in the program are the key to program readability. If the name is appropriate in a program, then everything fits together and relationships are clear, meaning is derivable. C++ uses CamelCase as a practice for writing names of methods, variables, classes, packages, and constants.

*CamelCase is a naming convention where a name is formed of multiple words that are joined together as a single word with the first letter of each of the word capitalized.*

Below are the naming conventions of C++ programming. They must be followed while writing code in C++ for good maintenance, readability, and understanding of the program.

**Type 1: Classes and Class Attributes Names**

* The class name should be a noun.
* Use upper case letters as word separators, and lower case for the rest of the word.
* The first character in the class name must be in upper case.
* No underscores (‘\_’) are permitted in the class name.

*class PerimeterRectangle*

*class FingerprintScanner*

* The private attribute name in class should be prepended with the character ‘m’.
* After prepending ‘m’, the same rules will be followed for the name as that for the class name.
* Character ‘m’ also precedes other name modifiers also. For example, ‘p’ for pointers.

*class PerimeterRectangle  
{  
    public:  
    int perimeter;  
    private:  
    int mLength;  
    int mWidth;  
}*

**Type 2: Functions and Function Argument Names**

Usually, every function in C++ performs one or more actions, so the name of the function should clearly hint at what it does. Each method/ function name should begin with a verb.

* Suffixes are sometimes useful. For example,
  + **Count-**the current count of the counter.
  + **Key-**the key value.
* Prefixes are sometimes useful. For example,
  + **get-**get value.
  + **set-**set value.

The same name convention is used as that for the class names.

*int getValue();*

*int SolveEquation();*

The first character of function/ method argument names should be lowercase.  Each word should also begin with a capital letter.

*int PerimeterRectangle(int lengthRectangle, int widthRectangle)*

**Type 3: Variables**

When the variables are declared dynamically using the **new** keyword or if the variables are declared as class attributes then they take memory from the heap and when the variables are created in a C++ program, the memory is allocated from the program stack.

* The variable name should begin with an alphabet.
* Digits may be used but only after the alphabet.
* No special symbols can be used in variable names except for the underscore(‘\_’).
* No keywords can be used for variable names.

*int total\_cost;*

*int length;*

**Pointer variables**should be prepended with ‘p’ and place the asterisk ‘\*’ close to the variable name instead of the pointer type.

*int \*pName;*

*int \*pAge, address; // Here only pAge is a pointer variable*

**Reference variables** should be prepended with ‘r’. This helps to differentiate between the method returning a modifiable object and the same method returning a non-modifiable object.

**Static variables** should be prepended with ‘s’.

*static int sCount;*

**Type 4: Constant**

The global constants should be all capital letters separated with ‘\_’.

*const double TWO\_PI = 6.28318531;*

**Type 5: File Naming;**

* No special character is allowed in the file name except for underscore (‘\_’) and dash (‘-‘).
* The file name should end with the .c extension in the end or should end with the .cpp extension.
* Do not use filenames that already exist in /user/include. or any predefined header file name.

*helloworld.c       // Valid*

*hello\_world.cpp    // Valid*

*hello-world.cpp   // Valid*

*hel-lo\_world.cpp  // Valid*

*hello\* world.cpp  // Not Valid*

*iostream.cpp // Not Valid*

*hello123@world.cpp// Not Valid*