Magno, John Paul A. BSIT-2C

WHAT IS LOOP IN C++?

Loops in C++ are used for repetitive activities and tasks, running the same code multiple times with different values. They are fundamental to programming, allowing for concise and efficient coding. A loop runs one or more lines of code based on certain conditions and then runs them again as long as those conditions are true. For example, if you need to calculate a series of equations with differing values, you could use a loop to do this efficiently without needing to write out each equation individually. Loops can also be used to iterate through data structures or search for specific values within an array or collection of objects. In summary, loops give programmers the power to automate repetitive tasks easily, saving time and energy when coding.

GIVE THE DIFFIRENCES TYPES OF LOOPING IN C++

For loop in C ++

The for loop in C ++ is an incredibly useful coding feature that allows for efficient iteration of operations for a specific number of times. It helps to reduce the amount of code needed for certain loops and also helps to make coding easier for developers. In contrast to other looping features, the for loop gives you more control over how many times your code should be executed, making it useful for virtually any situation. Furthermore, the for loop can be customized via different syntax parameters, giving coders even more control for greater efficiency and speed. With its helpful properties, the for loop can definitely make any C ++ coding journey smoother and more effective.

EX.

```
#include <iostream>

using class std;

int main()
  {
     for(int i=1;i<=10;i++)
       cout<<i <<"\n";
     }
  }
```

Infinite For loop in C++

Infinite For loop in C++ are frequently used to automate processes that need to be repeated numerous times. An Infinite For loop will continue running until the process has been satisfied or it is manually terminated by the user. This makes Infinite For loops particularly effective at quickly and consistently completing app-development tasks which may require a large amount of repetition. Infinite For loops also allow for easy maintenance since any changes made in the code will immediately affect every iteration of the loop, eliminating the need for multiple revisions. When working with Infinite For Loops, it is important to be careful and take precautions, as an infinite loop can cause performance issues or even crash your system if left unchecked.

EXAMPLE:

```
#include <iostream>

using class std;

int main ()

{

 for (; ;)

 {

   cout<<"Infinitive For Loop";

 }

}
```

While loop in C++

The while loop in C++ is an extremely useful tool for programming. It allows for the same block of code to continuously execute until a certain condition is satisfied. This means that tasks can be completed in an efficient manner, thereby saving time and resources. It could be used to create a counter or check if user input meets the desired criteria. Furthermore, complex algorithms and processes can be crafted with the help of a while loop and some other logical operators. Additionally, the ability to repeat code makes it easier for developers to modify their programs as their needs change and their requirements evolve over time. Ultimately, because of its versatility and ease of use, the while loop remains popular and widely used by programmers all around the world.

EXAMPLE:

```
#include <iostream>
```

```
using class std;

int main()

{

 int i=1;

        while(i<=10)

     {

          cout<<i <<"\n";

          i++;

     }

}
```

Nested While loop in C++

Nested while loop in C++ are a powerful programming tool that can be used to execute code multiple times by nesting a while loop within another while loop. This type of loop structure is beneficial when working with two-dimensional data sets, where a set of instructions must be repeated for each array element. Nested while loops essentially provide additional flexibility and possibility in C++ coding since they allow you to iterate through two or more arrays simultaneously. Nested loops can be considered “loops within loops”, making them an indispensable coding tool for the serious programmer.

EXAMPLE:

```
#include <iostream>

using class std;

int main ()

{

  int i=1;

    while(i<=3)

     {

       int j = 1;

 while (j <= 3)
```

```
        {
          cout<<i<<" "<<j<<"\n";
          j++;
        }
         i++;
      }
}
```

Infinite while loop in C++

Infinite while loop in C++ can be extremely powerful programming tools. They are designed to run until a specific condition is met - or not met, as the case may be for an infinite loop! Infinite while loops can be used for scenarios involving continuous calculations or if the programmer does not know how many times the loop should execute but wants to ensure that a certain condition is evaluated over and over again. While not always appropriate, infinite while loops can be a great way to create more efficient software programs.

EXAMPLE:

```
#include <iostream>
using class std;
int main ()
{
  while(true)
    {
      cout<<"Infinitive While Loop";
    }
}
```

Do-while loop in C++

Do-while loops are an essential part of the C++ programming language. They allow code to be repeated, with the loop only stopping once a defined condition has been satisfied. Do-while loops are different from other types of loops in C++, such as while and for loops, that use a specific test at the beginning to determine if the loop will be carried out. Do-while loops execute their first iteration before checking if the condition is met, allowing them to always run

at least once even if the condition is not true when it is first tested. Do-while loops can help programmers keep their code concise yet powerful when harnessing the power of repetition.

EXAMPLE:

```
#include <iostream>

using class std;

int main()

{

    int i = 1;

        do

        {

           cout<<i<<"\n";

           i++;

        }

        while (i <= 10) ;

}
```

Nested Do-while loop in C++

A nested do-while loop in C++ programming language is a complex structure created by nesting two consecutive do-while loops that allow a programmer to execute the set of commands inside the loop multiple times according to the assigned condition. Nested do-while loops come in handy when a certain number of coding instructions need to be executed more than once whatever the condition may be. The nested do-while loop allows a user to handle larger chunks of code thereby significantly improving the efficiency and execution speed of programs. In addition, the nested do-while loop also helps with process automation making tedious tasks easier for coders.

EXAMPLE:

```
#include <iostream>

using class std;

int main()

{
```

```
    int i = 1;
      do
      {
       int j = 1;
         do
         {
           cout<<i<<"\n";
           j++;
         }
           while (j <= 3) ;
           i++;
      }
      while (i <= 3) ;
}
```

Infinite Do-while loop in C++

Infinite Do-while loop in C++ are a useful tool to create an endless loop in code. These types of loops can be used for various purposes, such as continual polling of a server or automation of a task until a certain condition is met. Unlike for or while loops, the body code within an infinite do-while loop will always execute at least once before testing the condition. This makes them very beneficial for tasks that require certain steps to start, yet need to keep running perpetually. Infinite do-while loops should be used with caution, however, as it is possible for the body of the sentence to run infinitely if not coded properly, leading to system crashes or other catastrophic results.

EXAMPLE:

```
#include <iostream>

using class std;

int main()

{

 do
```

```
  {
    cout<<"Infinitive do-while Loop";
  } while(true);
}
```

1

[1] https://www.scholarhat.com/tutorial/cpp/loops-in-cpp-programming