IT 1010 – Note for Lecture 05

Repetition control structure

Why do we need repetition in programming?

Repetition is needed in programming for several reasons:

- To simplify code: If we have a task that needs to be repeated multiple times, we can use a repetition control structure to write the code once and then repeat it as needed. This makes the code more concise and easier to read and understand.
- To be more efficient: Instead of writing out the same code multiple times, we can use a repetition control structure to execute the code once and then repeat it as needed. This can save us a lot of time and effort, especially if we need to repeat the code many times.

Here are some examples of how repetition is used in programming:

- Counting: We can use a repetition control structure to count the number of times a certain condition is met. For example, we can use a while loop to count the number of times a user clicks a button.
- Accumulating running totals: We can use a repetition control structure to accumulate a running total of a series of numbers. For example, we can use a for loop to calculate the sum of the first 100 positive integers.
- Accepting input data: We can use a repetition control structure to accept input data from a user. For example, we can use a do...while loop to ask a user to enter their name until they enter a valid name.

What are the Repetition Control Structure used in C language?

In C language, repetition control structures are used to repeat a block of code a certain number of times.

There are three types of repetition control structures in C:

- while loop: The while loop repeats a block of code if a certain condition is true. The syntax for a while loop is:

```
while (condition)
{
  // block of code to be repeated
```

}

For example, the following code will print the numbers from 1 to 10:

```
int i = 1;
while (i <= 10)
{
  printf("%d\n", i);
  i++;
}
```

- do...while loop: The do...while loop is similar to the while loop, but the block of code is executed at least once, even if the condition is false. The syntax for a do...while loop is:

do
{
// block of code to be repeated
} while (condition);

For example, the following code will print the numbers from 1 to 10, even if the variable i is already greater than 10:

```
int i = 10;

do {
  printf("%d\n", i);
  i--;
} while (i >= 1);
```

- for loop: The for loop is the most versatile repetition control structure in C. It allows you to specify the initial value, the condition, and the update expression in a single statement. The syntax for a for loop is:

for (initialization; condition; increment/decrement)
{
// block of code to be repeated
}

For example, the following code is equivalent to the previous example with the while loop:

```
for (int i = 1; i <= 10; i++)
{
  printf("%d\n", i);
```

```
}
```

Repetition control structures are a powerful tool for writing efficient code. They can be used to solve a wide variety of problems, such as counting, accumulating running totals, and accepting input data.

What is meant by counter control repetition?

A counter control loop is a type of repetition control structure that repeats a block of code a specific number of times. The number of times the loop repeats is controlled by a counter variable.

The counter variable is initialized to a starting value, and then it is incremented or decremented by a specific amount each time the loop repeats. The loop continues to repeat until the counter variable reaches a predetermined value.

Here is a simple example of a counter control loop in C language:

```
int counter = 1;
while (counter <= 10)
{
  printf("%d\n", counter);
  counter++;
}
```

This code will print the numbers from 1 to 10, one per line. The counter variable is initialized to the value 1.

The while loop repeats the block of code inside the loop if the value of the variable counter is less than or equal to 10. The printf() function prints the value of the variable counter to the console. The counter++ statement increments the value of counter by 1.

In this example, the counter variable is initialized to the value 1. The loop then repeats if the counter variable is less than or equal to 10. Each time the loop repeats, the value of the counter variable is incremented by 1. This means that the loop will repeat 10 times, printing the numbers from 1 to 10.

Counter control loops are a powerful tool for repeating code a specific number of times. They are often used to solve problems that involve counting, accumulating running totals, and accepting input data.

Here are some other examples of how counter control loops can be used in C language:

- To count the number of times a certain condition is met.
- To accumulate a running total of a series of numbers.
- To accept input data from a user.
- To print a table of values.
- To perform a calculation a specific number of times.

Counter control loops are a versatile tool that can be used to solve a wide variety of problems.

What is meant by sentinel control loop in C language?

A sentinel control loop in C language is a repetition control structure that repeats a block of code until a specific value, known as a sentinel value, is encountered. The sentinel value is a value that is used to signal the end of the loop.

Here is a simple example of a sentinel control loop in C language:

```
int number;

//Gets the first input before entering to the loop


printf("Enter a number: ");
scanf("%d", &number);


while (number != -1)
{
  printf("Enter a number: ");
  scanf("%d", &number);
}
```

This code will prompt the user to enter a number. The loop will continue to repeat if the user does not enter the sentinel value, which is -1 in this case. Once the user enters the sentinel value, the loop will end, and the program will print the message "The loop has ended."

Sentinel control loops are a useful tool for repeating code until a specific condition is met. They are often used to solve problems where the number of times the loop should repeat is not known in advance.

Here are some other examples of how sentinel control loops can be used in C language:

- To read a series of numbers from a user until the user enters a sentinel value.
- To process a list of data items until the end of the list is reached.
- To perform a calculation until a specific result is obtained.
- To search for a specific value in a list of data items.

Sentinel control loops are a versatile tool that can be used to solve a wide variety of problems.

Use of nested loops in C language

Nested loops in C language are a powerful tool that can be used to repeat code multiple times in a complex way. A nested loop is a loop that is placed inside another loop. The inner loop will execute once for each iteration of the outer loop.

Here is a simple example of a nested loop in C language:

```
int i, j;

for (i = 0; i <= 10; i++)
{
  for (j = 0; j <= 10; j++)
  {
    printf("%d %d\n", i, j);
  }
}
```

This code will print the numbers from 0 to 10, ten times each. The outer loop will iterate 10 times, and the inner loop will iterate 10 times for each iteration of the outer loop. This means that the code will print the numbers from 0 to 10 100 times in total.

Nested loops are often used to solve problems that involve multidimensional arrays or matrices. For example, we can use nested loops to print the elements of a 2D array, or to calculate the determinant of a matrix.

Here are some other examples of how nested loops can be used in C language:

- To print a spiral pattern.
- To draw a shape, such as a square or a triangle.

Nested loops are a versatile tool that can be used to solve a wide variety of problems.