Control statements

The control statements are used in the C language help a user to specify a program control's flow. In simpler words, the control statements help users specify the order of execution of the instructions present in a program.

Control Statements Types Used in C Language

The C language provides support for the following set of statements in its program:

1. **If Statements**
2. **Switch Statement**
3. **Conditional Operator Statement**
4. **Go to Statement**
5. **Loop Statements**

For Loop:

For loop allow a set of instructions to be executed repeatedly until the condition is true, and then terminate when the condition is false.

Program to print name five time using for loop:

```
#include<stdio.h>
void main()
{
Int I;
for(i=1;i<=5;i++)
printf("Prakash /t");
}
```

Output: **Prakash Prakash Prakash Prakash Prakash**

Program to print factorial of a number using for loop **Discussed at end of page**	**Program to print number is prime or not using for loop:**

Program to print sum of natural numbers using for loop

```
#include<stdio.h>
Void main()
{
Int i,
printf("enter any number\n");
scanf("%d",&n);
for(i=1;i<=5;i++)
{
sum=sum+i;
}
printf("sum of series is %d",sum);
}
```

Output:
enter any number
5
Sum of series is 15.

While Loop: While Loop

A **while** loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a **while** loop in C programming language is −

```
while(condition) {
   statement(s);
}
```

Program to print 1 to 10 using while:

```
#include<stdio.h>

void main()

{

int i=1;

while(i<=10)

{

printf("%d /t",i);

i++;

}
```

Output:1 2 3 4 5 6 7 8 9 10

Program to reverse a number using while loop

```c
#include <stdio.h>

void main() {

  int n, rev = 0, rem;

  printf("Enter an integer: ");
  scanf("%d", &n);

  while (n> 0) {
    remainder = n % 10;
    rev = reverse * 10 + rem;
    n =n/ 10;
  }

  printf("Reversed number = %d",
rev);


}
```

Output:
Enter an integer:123
Reversed number: 321

Program to find palindraome of a number using while:

```c
#include <stdio.h>
void main() {
  int n, rev = 0, rem, z;
    printf("Enter an integer: ");
    scanf("%d", &n);
    z = n;

        while (n != 0) {
        rem = n % 10;
        rev = rev * 10 + rem;
        n =n/ 10;
    }

    // palindrome if orignal and
reversed are equal
    if (z == rev)
        printf("%d is a
palindrome.", z);
    else
        printf("%d is not a
palindrome.", z);

}
```

Output

Enter an integer:111
111 is a palindrome number

Program to find armstrong number using while loop:

```
#include<stdio.h>
#include<conio.h>
 void main()
{
int n,num,r,ans=0;
clrscr();
 printf("Enter a positive integer: ");
scanf("%d", &num);
n=num;
 while(num>0)
{
r=num%10;
ans=ans+r*r*r;
num=num/10;
}

if(ans==n)
printf("%d is an Armstrong number.",n);
else
printf("%d is not an Armstrong number.",n);
getch();
}
```

Output:

```
Enter a positive integer 152
153 is an Armstrong number.
```

do while Loop: A **do...while** loop is similar to a while loop, except the fact that it is guaranteed to execute at least one time.

Syntax

The syntax of a **do...while** loop in C programming language is −

```
do {
   statement(s);
} while( condition );
```

while and do while Loop Difference:

Parameters	while	do-while
Checking of Condition	It first needs to check the condition, and only then can we execute the statement(s).	The execution of statement(s) occurs at least once. After that, the checking of the condition occurs
Semicolon	while(condition) No semicolon is present at the end of the while loop.	while(condition); A semicolon is present at the end of the do-while loop.
Requirement of Brackets	We don't require any brackets if there is only a single statement.	We would always require brackets in this case.
Controlling	The while loop is an entry-controlled type of loop.	The do-while loop is an exit-controlled type of loop.

Example: **Int i=7;** **while(i<=6)** **{** **print("hello");** **}** **Output:** **Blank or no value**	**Example:** **int i=7;** **do** **{** **print("hello");** **}** **while(i<=6);** **Output: hello**

Nested Loops in C:

A **nested loop** means a loop statement inside another loop statement. That is why nested loops are also called "**loop inside loops**". We can define any number of loops inside another loop.

Syntax:

```
for ( initialization; condition; increment ) {


   for ( initialization; condition; increment ) {


      // statement of inside loop
   }


   // statement of outer loop
}
```

Nested for loop Examples:

```
include <stdio.h>
void main()
{
 int i, j, n;
 printf ("enter number\n");
 scanf ("%d",&n);
 for(i=1;i<=n;i++)
 {
  for (j=1;j<=i;j++)
  {
   printf ("%d",j);
  }
  printf ("\n");
 }
}
```

Output:

```
enter number 5
1
12
123
1234
12345
```

```
#include<stdio.h>
void main()
{
int i,j,term;
clrscr();
printf("Enter the number of row you
want to see");
scanf("%d",&term);
for (i = term ; i >= 1 ; i--)
{
for(j = 1 ; j <= i ; j++)
{
printf("%d",j);
}
printf("\n");
}

}
```

Output:

```
12345
 1234
 123
 12
 1
```

Key Differences Between Break and Continue Statement

Break	Continue
The break statement is used to exit from the loop.	The continue statement is not used to exit from the loop.
The break statement results in the control transfer out of the loop where the break keyword appears.	Continue statement results in skipping the loop's current iteration and continuing from the next iteration.
Both switch and loop statements are used with the break Statements.	Only loops are permitted with the continued expression.
Syntax: **break;**	Syntax: **continue;**

Break and Continue Example:

```
#include <stdio.h>
void main() {
 int i;
  for (i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  printf("%d\t", i);
 }
 }
Output :0 1 2 3
```

```
#include <stdio.h>
void main() {
 int i;
  for (i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  printf("%d\t", i);
 }
 }
Output :0 1 2 3 5  6 7 8 9
```

For loops programs

Program for factorial of a number

```
1. #include<stdio.h>
2. void main()
3. {
4.  int i,fact=1,number;
5.  printf("Enter a number: ");
6.   scanf("%d",&number);
7.     for(i=1;i<=number;i++){
8.       fact=fact*i;
9.   }
10.           printf("Factorial of %d is: %d",number,fact);
11.         }
```

Output:Enter a number 5

Factorial of 5 is 120

Program for finding prime number in c:

Program to Check Prime Number

```c
#include <stdio.h>

Void main() {

  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d", &n);

  if (n == 0 || n == 1)
    flag = 1;

  for (i = 2; i <= n / 2; ++i) {

    if (n % i == 0) {
      flag = 1;
      break;
    }
  }

  if (flag == 0)
    printf("%d is a prime number.", n);
  else
    printf("%d is not a prime number.", n);


}
```

Output

```
Enter a positive integer: 29
29 is a prime number
```

Program to print fabnosis series using for loop:

```
#include<stdio.h>

void main()
{
    int n = 10;
    int a = 0, b = 1;

    // printing the 0th and 1st term
    printf("%d, %d",a,b);

    int c;

    // printing the rest of the terms here
    for(int i = 2; i < n; i++){
        c = a + b;
printf("%d, ",c);
        a = b;
        b = c;

    }


}
```

Output

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

go to statement in c: The **C goto statement** is a jump statement which is sometimes also referred to as an **unconditional jump** statement. The goto statement can be used to jump from anywhere to anywhere within a function.

Syntax:

```
Syntax1      |   Syntax2
-------------------------
goto label;  |    label:
.            |    .
.            |    .
.            |    .
label:       |    goto label;
```

```
#include<stdio.h>

void main()
{
int i,num;
printf("enter any number");
Scanf("%d",&num)
if (num % 2 == 0)
        goto even;
    else
        goto odd;

even:
    printf("%d is even", num);
        return;
odd:
    printf("%d is odd", num);
}
```

Output: enter any number 15

15 is odd

Programs for practice:

Q1.Program to print table of a number .

Q2.Program to print pattern

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

Q3.Program to print

*

* *

* * *

* * * *

* * * * *

Q4.Program to print pattern

```
#include <stdio.h>
void main()
{
   int i, j;

    int rows = 3;

   char character = 'A';

      for (i = 1; i <= rows; i++)
   {
      for (j = 1; j <= i; j++)
      {
         printf("%c ",character);
           character++;
      }
      printf("\n");
   }

}
```

Output

```
A
B  C
D  E  F
```

Array in C: An array is defined as the collection of similar type of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.

Advantage of C Array

1) Code Optimization: Less code to the access the data.

2) Ease of traversing: By using the for loop, we can retrieve the elements of an array easily.

3) Ease of sorting: To sort the elements of the array, we need a few lines of code only.

Declaration of C Array

We can declare an array in the c language in the following way.

data_type array_name[array_size];

Now, let us see the example to declare the array.

```
int marks[5];
```

1)Program to find sum and average of an array

```
#include<stdio.h>

void main()
{
   int arr[100], size, i, sum = 0,avg;
     printf("Enter array elements\n");
for(i = 0; i < 5; i++)
 scanf("%d",&arr[i]);
   for(i = 0; i < size; i++){
        sum = sum + arr[i];
}
avg=sum/5;
       printf("Sum of the array = %d\n",sum);
    printf("Average of the array = %d\n",avg);
}
```

Output:

```
Enter five marks of student
1 2 4 5 6
Sum of the marks = 18
Average of the marks = 3
```

Program to find minimum element in an array.

```
#include <stdio.h>
void main() {
  int n;
  int a[100],small;
  printf("Enter the number of
elements (1 to 100): ");
  scanf("%d", &n);
printf(“enter numbers\n”);
  for (int i = 0; i < n; ++i) {
       scanf("%d", &a[i]);
  }
  // storing the smallest number to
arr[0]
small=a[0];
  for (int i = 1; i < n; ++i) {
    if (a[i] < small) {
      small= a[i];
    }
  }
  printf("smallest element = %d",
small);

}
```

Output:

```
Enter the number of elements (1 to
100): 3
enter numbers
12 2 45
smallest element = 2
```

Program to find minimum element in an array.

```
#include <stdio.h>
void main() {
  int n;
  int a[100],large;
  printf("Enter the number of
elements (1 to 100): ");
  scanf("%d", &n);
printf(“enter numbers\n”);
  for (int i = 0; i < n; ++i) {
       scanf("%d", &a[i]);
  }
  // storing the largest number to
arr[0]
large=a[0];
  for (int i = 1; i < n; ++i) {
    if (a[i] >large) {
      large= a[i];
    }
  }
  printf("largest element = %d",
large);
}
```

Output:

```
Enter the number of elements (1 to
100): 3
enter numbers
12 67 3
largest element = 67
```

Multidimensional Array:

A multi-dimensional array can be termed as an array of arrays that stores homogeneous data in tabular form. We can perform matrix operations using multidimensional array.

Examples:

Two dimensional array: int two_d[10][20];

Size of Multidimensional Arrays:

The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

For example:

The array **int x[10][20]** can store total (10*20) = 200 elements.

Program to transpose elements of a matrix using arrays:

```
#include <stdio.h>
void main() {
   int a[2][3],i,j;
   printf("enter elements \n");
   for(i=0;i<2;i++)
   {
      for(j=0;j<3;j++)
      {
       scanf("%d",&a[i][j]);
      }
}
      printf("entered matrix\n");
   for(i=0;i<2;i++)
   {
      for(j=0;j<3;j++)
      {
       printf("%d\t",a[i][j]);
      }
      printf("\n");
   }
   printf("Transpose of matrix\n");
   for(i=0;i<3;i++)
   {
      for(j=0;j<2;j++)
      {
       printf("%d\t",a[j][i]);
      }
      printf("\n");
   } }
```

Output: elements

1 2 4 5 6 7

entered matrix

1 2 4

5 6 7

Transpose of matrix

1 5

2 6

4 7

Program to multiply matrix in 2d array

```
#include<stdio.h>

void main()
{
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
printf("enter the number of row=");
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
printf("enter the first matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}

printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
}
}   }
```

```
//for printing result
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}   }
```

Output:

```
enter the number of row=2
enter the number of column=2
enter the first matrix element=
1 2 3 4
enter the second matrix element=
5 6 7 8
multiply of the matrix=
19      22
43      50
```

Structure: Structure is a collection of different data type.Structure is declared using struct keyword.

Syntax for structure:

```
struct structureName {
dataType member1;
dataType member2; ...
};
```

Example: struct student

```
{
int rollno;
char name
char address
};
```

Array vs structure difference

Definition	Structure is a type of a container that holds variables of different types.	Array is a type of data structure that works as a container to hold variables of the very same type.
Allocation of Memory	In a structure, the memory allocation for the input data doesn't require being in consecutive memory locations.	The array stores the input data in a memory allocation of contiguous type

Program to display student data using structure;

```
#include<stdio.h>
struct student
{
int rollno;
char name[20];
char address[40];
};
void main()
{
struct student z;
printf(“enter rollno name and address of student \n”);
scanf("%d %s %s",&z.rollno,z.name,z.address);
printf("student details are:\n");
printf("Rollno=%d \t",z.rollno);
printf("Name=%s \t",z.name);
printf("Address=%s \t",z.address);
}
```

Output:

```
rollno 12
enter name sds
enter address sadas
student details are:
Rollno=12    Name=sds    Address=sadas
```

Union in c: Union in C is **a special data type available in C that allows storing different data types in the same memory location**. We can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple purposes.

Syntax:

```
union [union name]
   {
     member definition;
     member definition;
     ...
     member definition;
   };
```

Difference between structure and union:

Parameter	**Structure**	**Union**
Keyword	A user can deploy the keyword **struct** to define a Structure.	A user can deploy the keyword **union** to define a Union.
Accessing Members	A user can access individual members at a given time.	A user can access only one member at a given time.

Memory	struct student{
	int rollno;
	char name[20];
	}
	Total memory:24 bytes
		In the case of a Union, the memory allocation occurs for only one member with the largest .
		union student{
		int rollno;
		char name[20];
		}
		Total memory:20 bytes

Arrays with structures: Arrays can be used with structures .
Program to display three student data using array and structure:

```
#include<stdio.h>
struct student
{
int rollno;
char name[20];
char address[40];
};
void main()
{
   int i,n;
struct student z[20];
printf("enter how many students”);
scanf("%d",&n);
for (i=0;i<n;i++)
{
printf("Enter details of students %d:",i+1);
printf("enter rollno name and address\n");
scanf("%d%s%s",&z[i].rollno,z[i].name,z[i].address);

}

printf(“Students Details are\n");
for (i=0;i<n;i++)
{
printf("Rollno=%d ",z[i].rollno);
printf("Name=%s   ",z[i].name);
printf("Address=%s \n ",z[i].address);
}}
```

Output:

```
enter how many students 3
3
Enter details of students 1:enter rollno name and address
1 mukul lucknow
Enter details of students 2:enter rollno name and address
2 mahesh kanpur
Enter details of students 3:enter rollno name and address
3 rajesh delhi
Students Details are
Rollno=1 Name=mukul   Address=lucknow
 Rollno=2 Name=mahesh   Address=kanpur
 Rollno=3 Name=rajesh   Address=delhi
```

Character arrays in c:

C language provide 'char' data type to hold Alphabets and Special characters. Size of a single character is 1 Byte.

char names[] = {'a','b','c','d'};

Enumerated data type in c:

Enumeration or Enum in C is a special kind of data type defined by the user. It consists of constant integrals or integers that are given names by a user.

```
// An example program to demonstrate working
// of enum in C
#include<stdio.h>
 enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
 void main()
{
   enum week day;
   day = Wed;
   printf("%d",day);
   return 0;
}
```

Output:2

2

In the above example, we declared “day” as the variable and the value of “Wed” is allocated to day, which is 2. So as a result, 2 is printed.