Unit 3: Control Structure in C

Control structures in C are used to control the flow of execution of a program. They are mainly divided into:

- **Selective (Decision Making) Structures**
- **Looping (Iterative) Structures**

3.1 Selective Structure

Selective structures are used to make decisions based on conditions.

3.1.1 If Statement

◇ Syntax:

```
if(condition)
{
    // statements
}
```

◇ Flow:

If the condition is true (non-zero), statements inside the block execute.

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if(num > 0)
    {
        printf("Number is positive");
    }

    return 0;
}
```

3.1.2 If-Else Statement

◇ Syntax:

```
if(condition)
{
    // true block
}
else
{
    // false block
}
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(num % 2 == 0)
    {
        printf("Even number");
    }
    else
    {
        printf("Odd number");
    }
    return 0;
}
```

3.1.3 Nested If-Else Statement

When an `if` or `else` contains another `if-else` inside it.

◇ Syntax:

```
if(condition1)
{
    if(condition2)
    {
        // statements
    }
    else
    {
        // statements
    }
}
else
{
    // statements
}
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if(num > 0)
    {
        if(num % 2 == 0)
        {
            printf("Positive Even Number");
        }
        else
        {
            printf("Positive Odd Number");
        }
    }
    else
    {
        printf("Number is negative or zero");
    }

    return 0;
}
```

3.1.4 Switch Statement

Used when multiple conditions are based on a single variable.

◇ Syntax:

```
switch(expression)
{
    case value1:
        // statements
        break;

    case value2:
        // statements
        break;

    default:
        // statements
}
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int day;
    printf("Enter number (1-3): ");
    scanf("%d", &day);

    switch(day)
    {
        case 1:
            printf("Monday");
            break;

        case 2:
            printf("Tuesday");
            break;

        case 3:
            printf("Wednesday");
            break;

        default:
            printf("Invalid choice");
    }

    return 0;
}
```

3.1.5 Conditional Operator (?:)

Also called **Ternary Operator**.

◇ Syntax:

```
condition ? expression1 : expression2;
```

If condition is true → expression1 executes
If false → expression2 executes

◇ Example Program:

```
#include <stdio.h>
```

```
int main()
{
    int a = 10, b = 20;
    int max;

    max = (a > b) ? a : b;

    printf("Maximum number is %d", max);

    return 0;
}
```

3.2 Looping Structure

Loops are used to execute a block of code repeatedly.

3.2.1 While Loop

◇ Syntax:

```
while(condition)
{
    // statements
}
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int i = 1;

    while(i <= 5)
    {
        printf("%d\n", i);
        i++;
    }
    return 0;
}
```

3.2.2 Do-While Loop

Executes at least once because condition is checked after execution.

◇ Syntax:

```
do
{
    // statements
} while(condition);
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int i = 1;

    do
    {
        printf("%d\n", i);
        i++;
    } while(i <= 5);

    return 0;
}
```

3.2.3 For Loop

Used when number of iterations is known.

◇ Syntax:

```
for(initialization; condition; increment/decrement)
{
    // statements
}
```

◇ Example Program:

```
#include <stdio.h>

int main()
{
    int i;

    for(i = 1; i <= 5; i++)
    {
        printf("%d\n", i);
    }

    return 0;
}
```

3.2.4 Nested Loops

Loop inside another loop.

◇ Example Program (Print Pattern):

```
#include <stdio.h>

int main()
{
    int i, j;

    for(i = 1; i <= 3; i++)
    {
        for(j = 1; j <= 3; j++)
        {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

3.2.5 Loop Interrupts

Loop control statements:

1 break

Terminates loop immediately.

```
#include <stdio.h>

int main()
{
    int i;

    for(i = 1; i <= 10; i++)
    {
        if(i == 5)
            break;

        printf("%d\n", i);
    }

    return 0;
}
```

2 continue

Skips current iteration.

```
#include <stdio.h>

int main()
{
    int i;

    for(i = 1; i <= 5; i++)
    {
        if(i == 3)
            continue;
        printf("%d\n", i);
    }

    return 0;
}
```

3️⃣ goto (Rarely Used)

```
#include <stdio.h>

int main()
{
    int i = 1;

start:
    if(i <= 3)
    {
        printf("%d\n", i);
        i++;
        goto start;
    }

    return 0;
}
```

Full Structure of a C Program

```
#include <stdio.h>      // Header files

// Global variable declaration (optional)

int main()              // Main function
{
    // Variable declaration
    int a = 5;

    // Statements
    printf("Value of a is %d", a);
    return 0;             // End of program
}
```