forbeginners.c   tutorials.out

CSE1062 | CCS1063 ‘Practicals’ {

[Fundamentals of Computer Programming]

< Tutorial Session 03 - Control Structures >

}

CONTROL STRUCTURES

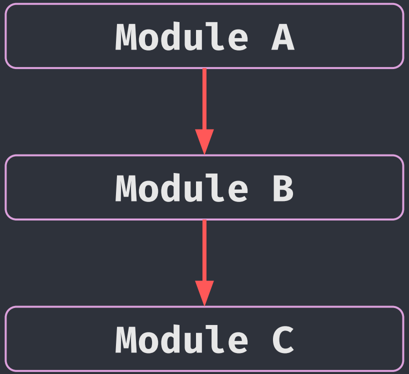
Control Structures are just a way to specify **flow of control** in programs. It basically analyzes and chooses in which **direction** a program flows based on certain **parameters** or **conditions**. There are three basic types of logic, or flow of control, known as:

- Sequence logic, or sequential flow
- Selection logic, or conditional flow
- Iteration logic, or repetitive flow

https://www.geeksforgeeks.org/control-structures-in-programming-languages/

Sequential Logic (Sequential Flow)

“Sequential logic as the name suggests follows a **serial** or **sequential** flow in which the flow depends on the series of instructions given to the computer.”



https://www.geeksforgeeks.org/control-structures-in-programming-languages/

Selection Logic (Conditional Flow)

Selection Logic simply involves a **number of conditions** or parameters which decides one out of several written modules. The structures which use these type of logic are known as **Conditional Structures**. These structures can be of three types:

- Single Alternative (IF STATEMENT)
- Double Alternative (IF ELSE STATEMENT)
- Multiple Alternatives (IF ELSE IF STATEMENT)

https://www.geeksforgeeks.org/control-structures-in-programming-languages/

Iteration Logic (Repetitive Flow)

Iteration Logic (Repetitive Flow)

The Iteration logic employs a loop which involves a **repeat statement** followed by a module known as the body of a loop.
The two types of these structures are:

- Repeat-For Structure
- Repeat-While Structure

https://www.geeksforgeeks.org/control-structures-in-programming-languages/

IF Statement

```
  if (condition) {
   action
  }
```

IF Statement…

```
#include <stdio.h>
void main(){
    int j;
    scanf("%d" ,&j);
    printf("your number is=%d\n", j);
    if(j>0){
        printf("it is a positive number");
        }
    }
```

IF Else Statement

```
if(condition) {
 action 1
} else {
 action 2
}
```

IF Else Statement…

```
#include <stdio.h>
void main(){
    int j;
    scanf("%d" ,&j);
    printf("your number is=%d\n", j);
    if(j>0){
        printf("it is a positive number");
        }
    else{
        printf("it is a negative number");
        }
    }
```

Nested IF

```
if (condition 1) {
    Statement 01
} else if (condition 2) {
    Statement 02
} else {
    Statement 03
    }
```

Nested IF…

```c
#include <stdio.h>
void main(){
    int marks = 65;
    if(marks >= 80)
        printf("Grade A");
    else if(marks >= 60)
        printf("Grade B");
    else if(marks >= 40)
        printf("Grade C");
    else
        printf("Grade D");
}
```

Switch case

The switch statement is a construct that is used when many conditions are being tested for

When there are many conditions, it becomes too difficult and complicated to use the if and else if constructs

The keyword default is executed when none of the conditions being tested for in the switch statement are met or executed

The break statement must be used after each condition because if it were not used than all the conditions from the one met will be executed

Switch case…

```
switch (variable) {
    case expression1:
        statement
        break;
    case expression2:
        statement
        break;
    case expression2:
        statement
        break;
    default:
        statement
        brake;
    }
```

Switch case…

```
    #include <stdio.h>
    void main(){
        char fruit;
        printf("Which one is your favorite fruit:\n");
        printf("a) Apples\n");
        printf("b) Bananas\n");
        printf("c) Cherries\n");
        scanf("%c", &fruit);
        switch (fruit){
            case 'a': printf("You like apples\n"); break;
            case 'b':printf("You like bananas\n");break;
            case 'c':printf("You like cherries\n");break;
            default:printf("You entered an invalid choice\n");
        }
    }
```

While statement

The “while” provides a mechanism for repeating C statements while a condition is true

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

While statement…

```
#include <stdio.h>
void main(){
    int j= 0;
    while (j<=10){
        printf("*");
        j++;
    }
}
```

While statement…

```c
#include <stdio.h>
void main(){
    int j=1, f;
    printf("Enter Three Numbers\n");
    while (j<=3){
        printf("Enter %d Number =", j);
        scanf("%d", &f);
        printf("Number is =%d\n", f);
        j++;
    }
}
```

While statement...

```c
#include <stdio.h>
void main(){
    int x = 0;
    while((x<=5)){
        int y=0;
        while (y<x+1){
            printf("*");
            y++;
        }
        printf("\n");
        x++;
    }
}
```

Output:

```
*
**
***
****
*****
******
```

The do while statement

The do {} while statement allows a loop to continue whilst a condition evaluates as TRUE (non-zero). The loop is executed at least once

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

The do while statement…

```c
#include <stdio.h>
void main(){
    int j=-10;
    while (j>0){
        printf("number is %d in while\n",j);
        j++;
    }
    printf("end of while\n");

    j=-10;
    do{
        printf("number is %d in do while\n",j);
        j++;
    } while (j>0);
    printf("end of do while\n");
}
```

For loop

The for loop can execute a block of code for a fixed or given number of times.

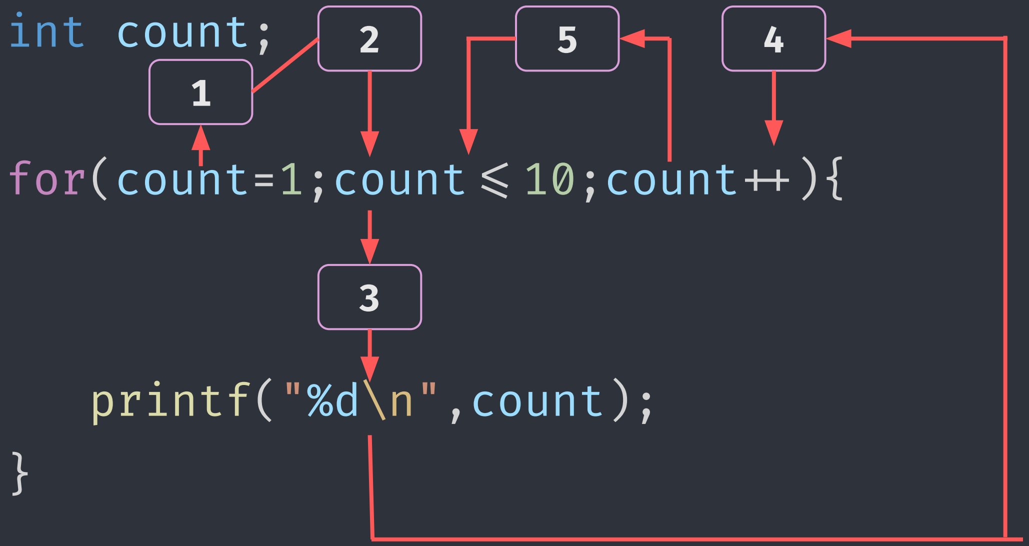
```
for (initializations; test conditions; increment value) {
    block of code;
}
```

For loop...

```c
#include <stdio.h>
void main(){
    int count;

    for(count=1;count<=10;count++){


        printf("%d\n",count);
    }
}
```



For loop...

```c
#include <stdio.h>
void main(){
    int a, b;
    for (a=1; a<=5;++a){
        for (b=1;b<=a;++b){
            printf ("*");
        }
        printf ("\n");
    }
}
```

Output:

```
*
**
***
****
*****
```

For loop...

```
#include <stdio.h>
void main()
{
int count, number=0;
for (count = 1;number<=1000 && count<=5;
count=count+1)
{
printf("%d\n", count);
printf("Enter a number? \n");
scanf("%d", &number);
}
}
```

For loop…

```c
#include <stdio.h>
void main()
{
int i, j, k;
for (i = 0,j=5,k=-1; i<10;i++,j++,k--)
{
printf(" value of i=%d \n", i);
printf(" value of j=%d \n", j);
printf(" value of k=%d \n \n", k);
}
}
```

Break

The break key word forces immediate exit from the nearest enclosing loop.

Break...

```c
#include <stdio.h>
void main(){
    int i, j;
    for (i=0;i<10;i++){
        printf("Enter a Number=" );
        scanf("%d", &j);
        if(j<=0){
            break;
        }
        printf("Number is positive \n\n");
    }
}
```

Continue

The continue keyword forces the next iteration of the nearest enclosing loop

Continue…

```c
#include <stdio.h>
void main(){
    int i, j;
    for (j=1;j<10;j++){
        if(j%3==0){
            continue;
        }
        printf("j=%d\n", j);
    }
}
```

```
Thanks; {
    ‘Do you have any questions?’

        < bgamage@sjp.ac.lk >

}
```