Control Statements - II

hitbullseye

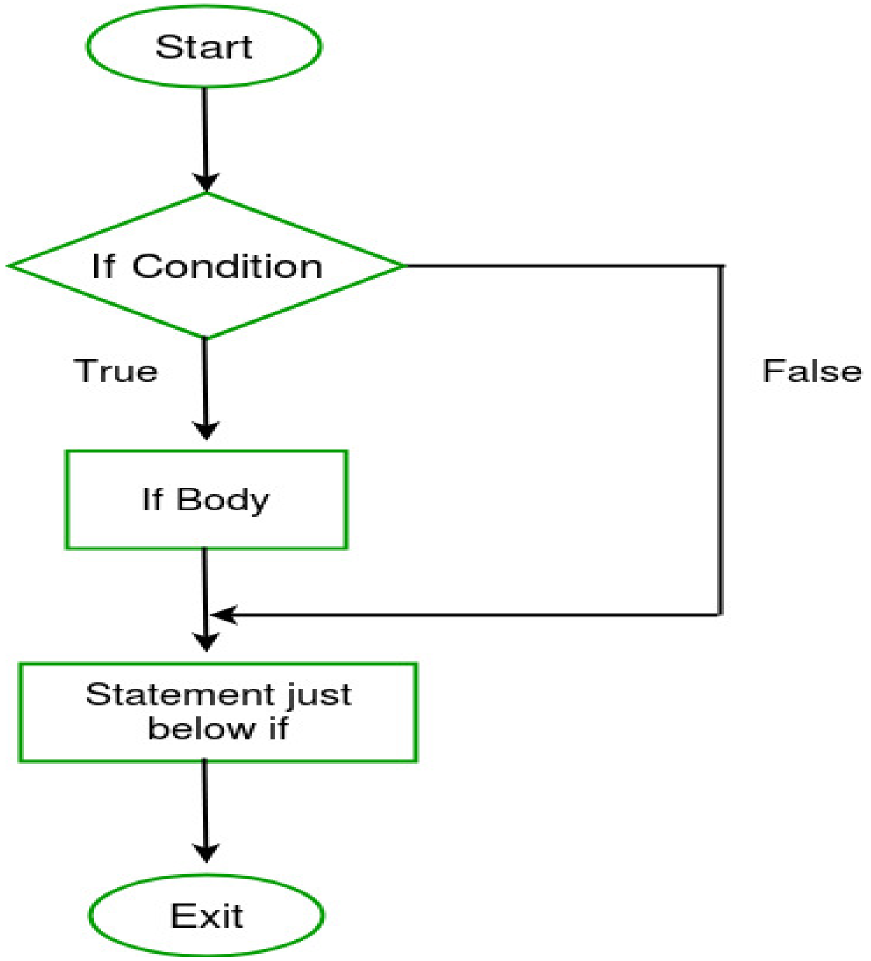
If Statements

If Statement:

- **if statement is the most simple decision-making statement.**
- **It is used to decide whether a block of statements will be executed or not.**
- **If the condition is true, then a certain block of code will execute.**

The Syntax is:

```
if (condition x)
{
Statement1;
Statement 2;
}
```



Program:

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

hitbullseye

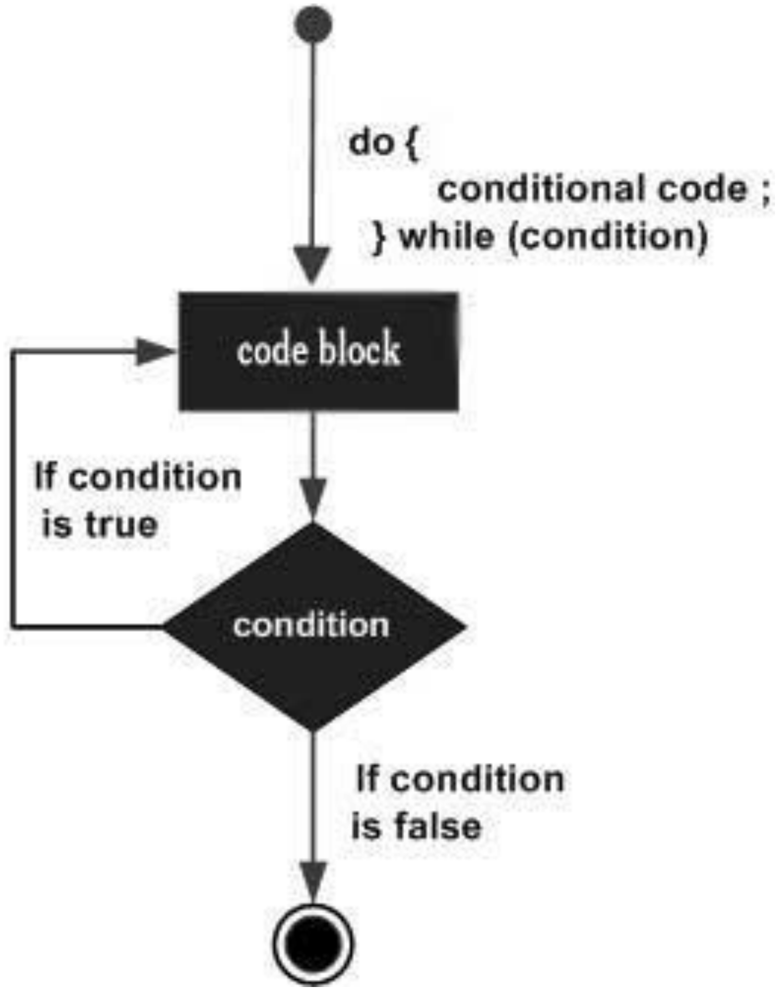
do-while loop

do-while loop:

- **Do-while loop in C is similar to the while loop except that the condition is always executed after the body of a loop.**
- **It is also called an exit-controlled loop.**
- **The body of do-while loop is executed at least once.**

Syntax is :

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



hitbullseye

Program:

```
#include<stdio.h>
int main()
{
        int num=1;
        do{
                printf ("%d\ ",4*num);
                num ++;
        }while(num <=10);
        return 0;
}
```

Output: 4 8 12 16 20 24 28 32 36 40

for loop

for loop:

- **for loop is a more efficient loop in C programming.**
- **This loop is used when you know many types you want to execute the block of code.**

Syntax is :

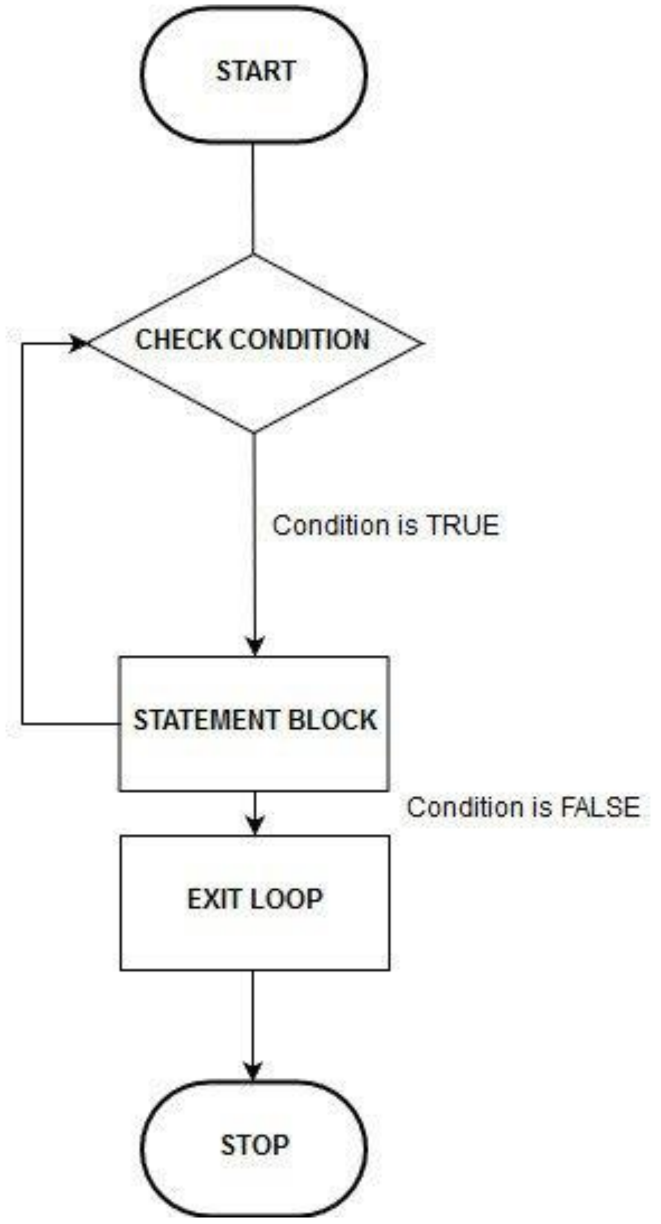
```
for(Expression 1; Expression 2; Expression 3)
{
//code
}
```

- **Expression 1 is executed once before the execution of the code block.**
- **Expression 2 defines the condition for executing the code block.**
- **Expression 3 is executed (every time) after the code block has been executed.**

Program

```
include<stdio.h>

int main()
{
        int i;
        for(i=1;i<=10;i++)
        {
            printf("%d\ ",i);
        }
        return 0;
}
Output:  1 2 3 4 5 6 7 8 9 10
```



Jumping Statements

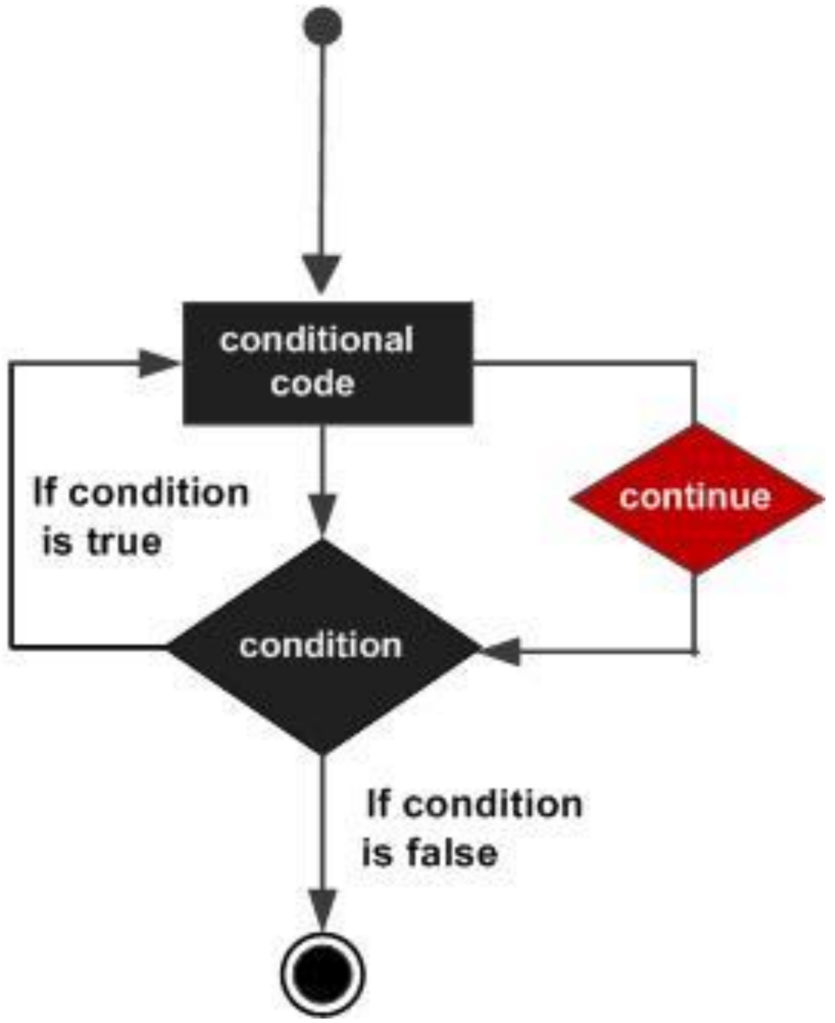
Continue:

- **Continue forces the next iteration of the loop to take place, skipping any code in between.**
- **The statements which is present between the continue statement and the end of the loop aren't executed.**
- **The continue statement skips some lines of code inside the loop and then continues with the next iteration.**
- **Continue statement is not used to exit from the loop.**

Program

```
#include<stdio.h>
int main()
{
int a;

for (a = 0; a < 10; a++) {
    if (a == 4) {
      continue;
    }
    printf("%d\ ", a);
  }
  return 0;
}
Output: 0 1 2 3 5 6 7 8 9
```

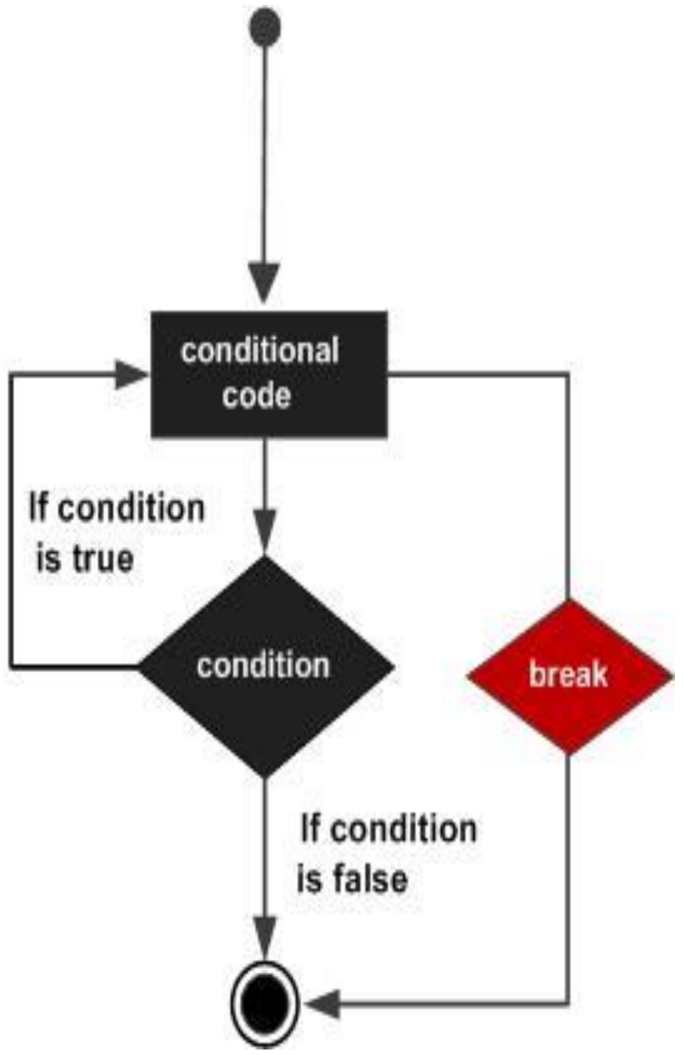


hitbullseye

Break

Break:

- **Break is used to break up the loop.**
- **The break statement is used inside loop and the loop is immediately terminated.**
- **The break statement breaks the loop one by one if it is used in nested loop.**
- **Break statement is used to exit from the loop.**



hitbullseye

Program

```
#include <stdio.h>
int main () {
   int m= 10;
while( m < 20 ) {
      printf("value of m: %d\n ", m);
      m++;
      if( m> 15) {
break;
      }}
   return 0;}
```

hitbullseye

Output is :

```
value of m: 10
value of m: 11
value of m: 12
value of m: 13
value of m: 14
value of m: 15
```

Return

Return:

It ends the execution of the function and returns control where the function has started.

Program:

```
#include <stdio.h>
void Print()
{
printf("Welcome to C");
}
 int main()
{
   Print();
   return 0;     }
```

Output: Welcome to C

goto

goto:

- goto statement is unconditional statement.
- When program reaches a goto statement, execution immediately jumps, to the location specified by the goto statement.
- By using goto we can jump to line of code with in a same file.

goto

Program:

```
#include <stdio.h>>
void main()
{      int num=7;
if (num % 2 == 0)
        goto even;
    else
        goto odd;
even:   printf("%d is even\n", num);
odd:  printf("%d is odd\n", num);
}
```

Output: 7 is odd

hitbullseye

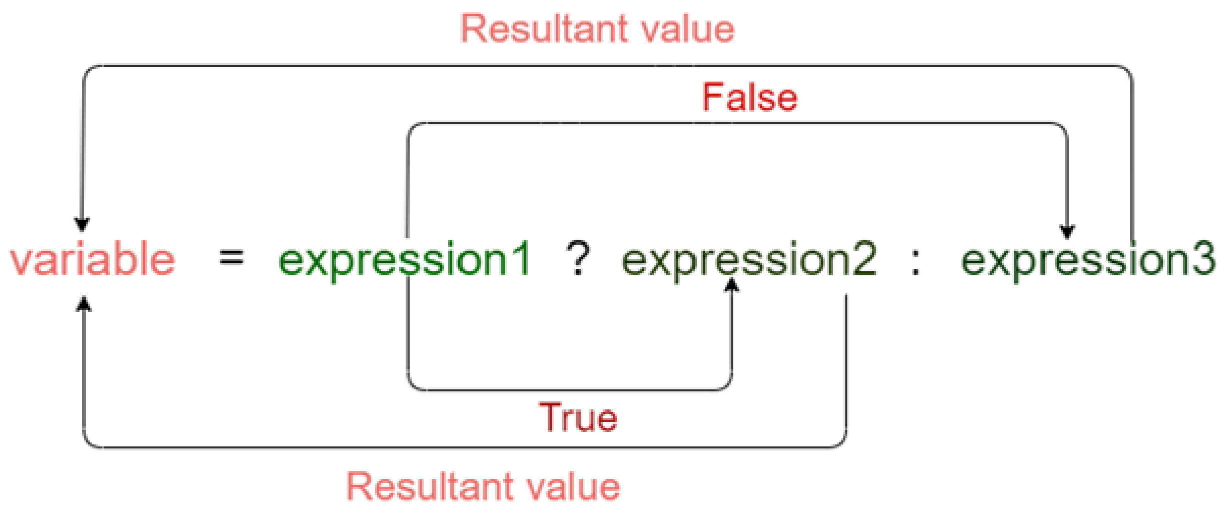
Conditional Operator

This operator is also known as ternary operator.

Syntax:
variable = Exp1(condition) ? Exp2 : Exp3

Conditional Operator has three arguments.

- **Argument of comparison**
- **The result upon a false comparison**
- **The result upon a true comparison**



- If the expression1 is a Boolean condition that is either true or false.
- If the expression1 results true, then the expression2 will execute.
- The expression2 is true only when it returns a non-zero value.
- If the expression1 returns false, then the expression3 will execute.
- The expression3 is false only when it returns zero value.

Program:

```
#include <stdio.h>
int main(void)
{ int x, y, z, min;
    x = 100;
    y = 200;
    z = 50;
    min = (x<y && x<z) ? (x) : (y<z) ? (y) : (z) ;
    printf (“Min value : %d \n", min);
    return 0;  }
Output: Min value : 50
```

References:

Student reference link:

https://www.javatpoint.com

https://www.tutorialspoint.com/cprogramming

https://www.tutorialspoint.com/index.htm

https://www.javatpoint.com/conditional-operator-in-c

THANK YOU

hitbullseye