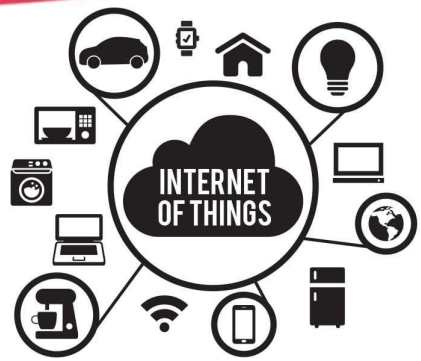
SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution



Department of Information Technology

23CST101 - PROBLEM SOLVING and C PROGRAMMING

I B.Tech. IT/ I SEMESTER

UNIT II : C PROGRAMMING BASICS

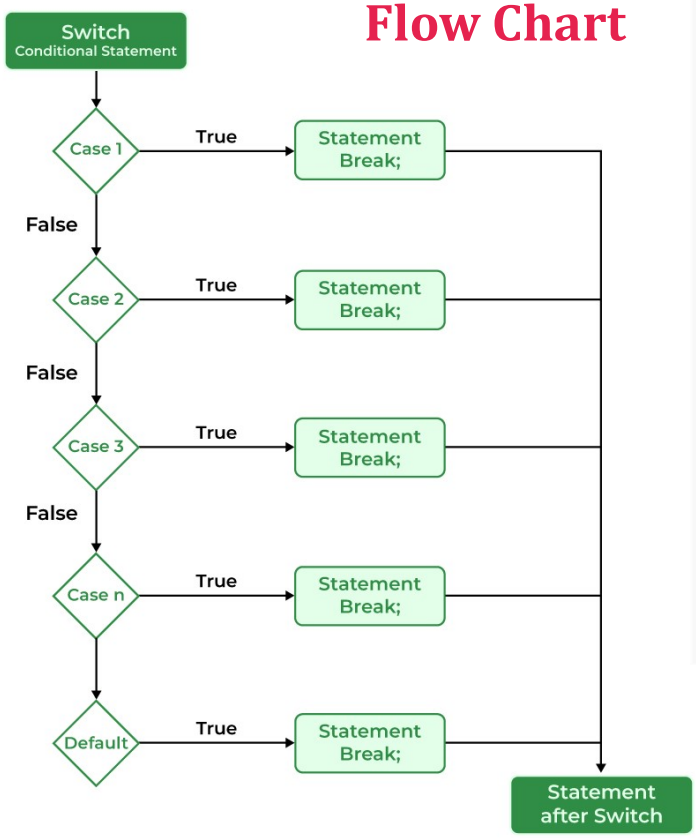
Introduction to 'C' Programming –Fundamental rules – Structure of a 'C' program – Compilation and Linking processes –Constants, Variables, keywords, Identifier, Delimiters – Declaring and Initializing variables – Data Types – Operators and Expressions –Managing Input and Output operations – Decision Making and Branching –Looping

Switch Statement

- Switch case statement evaluates a given expression and based on the evaluated value(matching a certain condition), it executes the statements associated with it.
- Basically, it is used to perform different actions based on different conditions(cases).

Switch

Flow Chart




```
switch(expression)
{
case value1: statement_1;
             break;
case value2: statement_2;
             break;
.
.
.
case value_n: statement_n;
              break;

default: default_statement;
}
```

Rules

1.In a switch statement, the "**case value**" must be of "**char**" and "**int**" type.

2.There can be one or N number of cases.

3.The values in the case must be **unique**.

4.Each statement of the case can have a break statement. It is optional.

5.The default Statement is also optional.

switch

```
void main()
{
    int var = 1;
   switch (var) {
     case 1:
        printf("Case 1 is Matched.");
        break;
     case 2:
        printf("Case 2 is Matched.");
        break;
     case 3:
        printf("Case 3 is Matched.");
        break;
     default:
        printf("Default case is Matched.");
        break;
}}
```

```
void  main()
{
    int var = 2;
    // switch case without break
   switch (var) {
    case 1:
       printf("Case 1 is executed.\n");
    case 2:
       printf("Case 2 is executed.\n");
    case 3:
       printf("Case 3 is executed.");
    case 4:
       printf("Case 4 is executed.");
  }
 }
```

C Program to print the day of the week using a switch case

```
int main()
{
   int day = 2;
    printf("The day with number %d is ", day);
   switch (day) {
    case 1:
       printf("Monday");
       break;
    case 2:
       printf("Tuesday");
       break;
    case 3:
       printf("Wednesday");
       break;
    case 4:
       printf("Thursday");
       break;
    case 5:
       printf("Friday");
       break;
    case 6:
       printf("Saturday");
       break;
    case 7:
       printf("Sunday");
       break;
    default:
       printf("Invalid Input");
       break;
    }
   return 0;
}
```

C Program to create a simple calculator using switch

```
int main()
{
      char choice;
      int x, y;

   while (1) {
      printf("Enter the Operator (+,-,*,/)\nEnter x to "
            "exit\n");
      scanf(" %c", &choice);

      // for exit
      if (choice == 'x') {
         exit(0);
      }

      printf("Enter the two numbers: ");
      scanf("%d %d", &x, &y);

      // switch case with operation for each operator
      switch (choice) {
       case '+':
          printf("%d + %d = %d\n", x, y, x + y);
          break;

       case '-':
          printf("%d - %d = %d\n", x, y, x - y);
          break;

       case '*':
          printf("%d * %d = %d\n", x, y, x * y);
          break;
       case '/':
          printf("%d / %d = %d\n", x, y, x / y);
          break;
       default:
          printf("Invalid Operator Input\n");
      }
   }
   return 0;
}
```

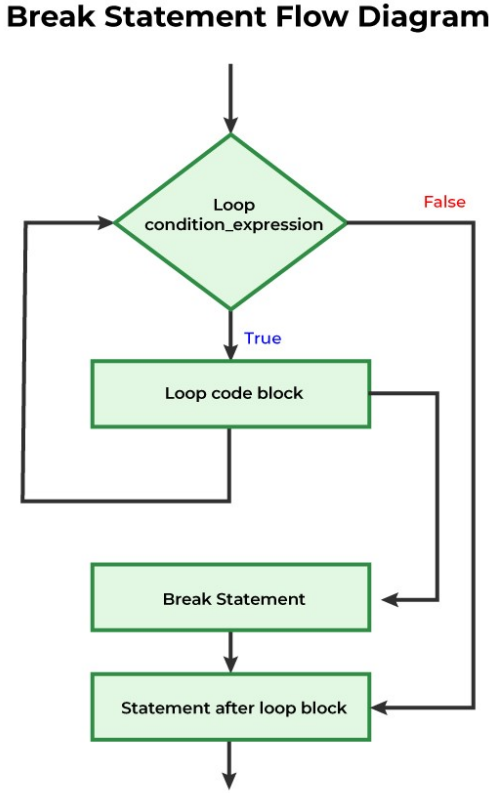
Jump Statement

- jump statements are used to jump from one part of the code to another altering the normal flow of the program.
- They are used to transfer the program control to somewhere else in the program.

There are 4 types of jump statements

- break
- continue
- goto
- return

break



Syntax

```
break;
```

The break statement exits or terminates the loop or switch statement based on a certain condition, without executing the remaining code.

break

```
#include <stdio.h>
int main()
{
   int i;
   // for loop
   for (i = 1; i <= 10; i++) {

      // when i = 6, the loop should end
      if (i == 6) {
         break;
      }
      printf("%d ", i);
   }
   printf("Loop exited.\n");
   return 0;
}
```

Output

1 2 3 4 5 Loop exited.

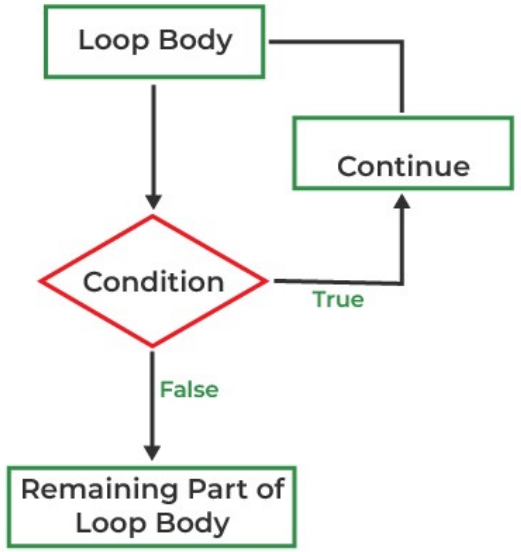
Continue

Syntax

```
continue;
```

used to skip the remaining code after the continue statement within a loop and jump to the next iteration of the loop.

When the continue statement is encountered, the loop control immediately jumps to the next iteration, by skipping the lines of code written after it within the loop body.



Continue

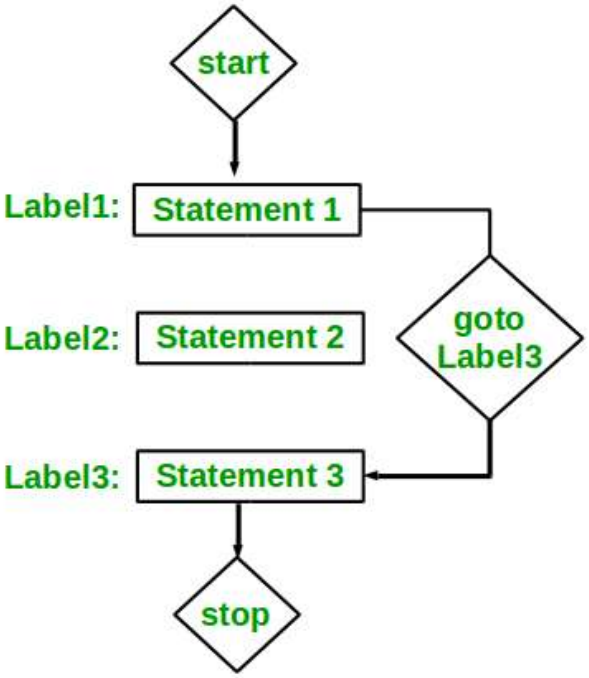
```
#include <stdio.h>

int main()
{
   int i;
   // loop
   for (i = 0; i < 5; i++) {
      if (i == 2) {
         // continue to be executed if i = 2
         printf("Skipping iteration %d\n", i);
         continue;
      }
      printf("Executing iteration %d\n", i);
   }
   return 0;
}
```

Output

```
Executing iteration 0
Executing iteration 1
Skipping iteration 2
Executing iteration 3
Executing iteration 4
```

goto



Syntax

```
goto label;
.
.
label:
//code
```

Used to jump to a specific point from anywhere in a function.

It is used to transfer the program control to a labeled statement within the same function.

goto

```
#include <stdio.h>
 // function to check even or not
void check(int num)
{
   if (num % 2 == 0)
      // jump to even
      goto even;
   else
      // jump to odd
      goto odd;
even:
   printf("%d is even", num);
   // return if even
   return;
odd:
   printf("%d is odd", num);
}
 int main()
{
   int num = 26;
   check (num);
   return 0;
}
```

Output

26 is even

return

used to terminate the execution of a function and return a value to the caller. It is commonly used to provide a result back to the calling code.

Syntax

return expression;

return

```
#include <stdio.h>
 int add(int a, int b)
{
   int sum = a + b;
   return sum; // Return the sum as the result of the
            // function
}
 void printMessage()
{
   printf("Welcome to IT\n");
   return; // Return from the function with no value (void)
}
int main()
{
   int result = add(5, 3);
   printf("Result: %d\n", result);

   printMessage();

   return 0;
}
```

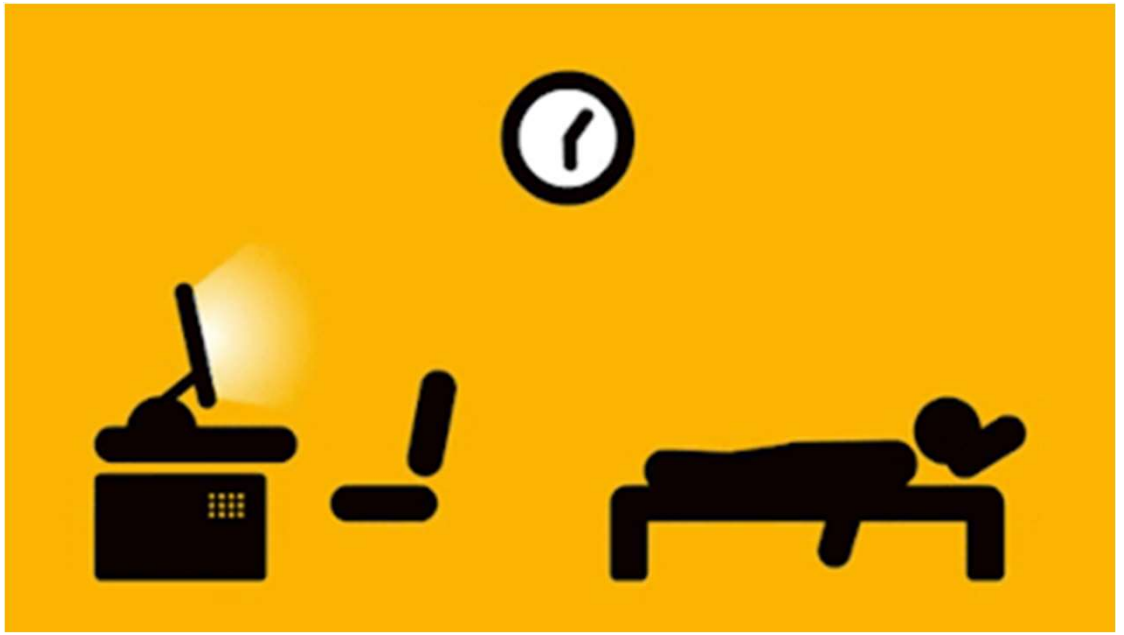
Output

8
Welcome to IT

Looping

- It is defined as a block of statement which is repeatedly executed for certain number of times
- Loop Variable
- Initialization
- Increment / Decrement

Repetition



HAI
HAI
HAI
HAI
HAI
HAI
HAI
HAI

You Will Want to display HAI More Than Once

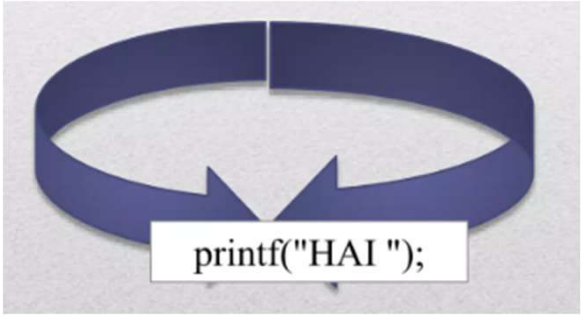
```
printf("HAI");
printf(" HAI ");
printf(" HAI ");
printf(" HAI ");
printf(" HAI ");
printf(" HAI ");
printf(" HAI ");
printf(" HAI ");
```

No good programmer does this!

You Could Code The 'HAI' Like This

LOOP

- Don’t need to write this code 8 times.
- A loop is a piece of code which allows you to repeat some code more than once without having to write it out more than once.

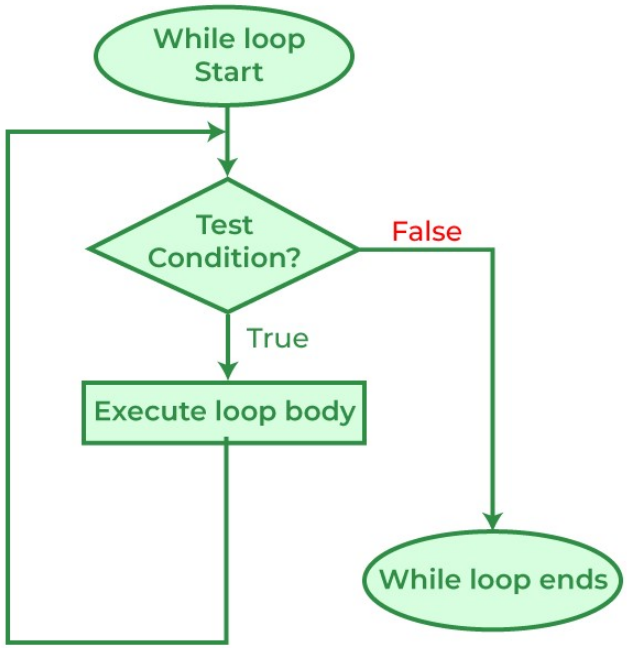


Looping Statement

- Loop provide a way to repeat a command and control – how many times they are repeated
- 3 Types
  - while
  - do while
  - for

while

Flow Chart



Syntax:

```
while( condition )
{
    ...
    block of statements;
    ...
}
```

If the test condition is true: the statement dets executed until the test condition becomes false

while Example

Print welcome to SNS College Technology 10 times

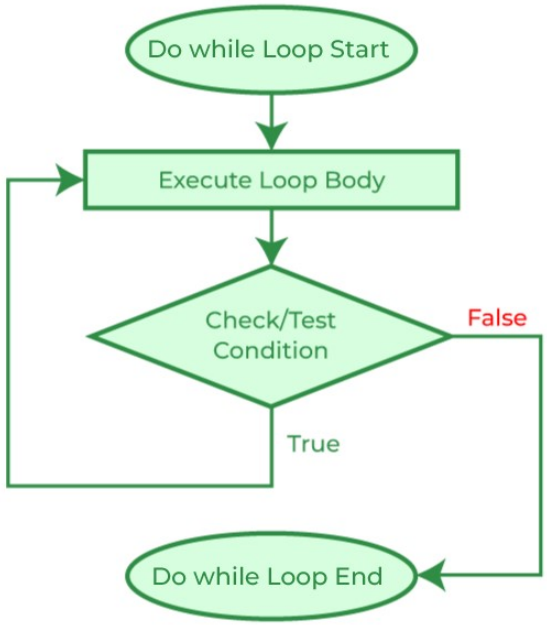
```
void main()
{
int count=0;
while(count<10)
{
printf("welcome to SNS College Technology");
count++;
}
getch();
}
```

do while

Syntax:

```
do
{
    ...
    block of statements;
    ...
} while( condition ) ;
```

Flow Chart



The body of a loop is always executed at least once. After the body is executed, then it checks the condition. If the condition is true, then it will again execute the body of a loop otherwise control is transferred out of the loop.

do while Example

Print welcome to SNS College Technology 10 times

```
void main()
{
int count=0;
do
{
printf("welcome to SNS College Technology");
count++;
} while(count<10);
getch();
}
```

for

Expression 1 Expression 2 Expression 3

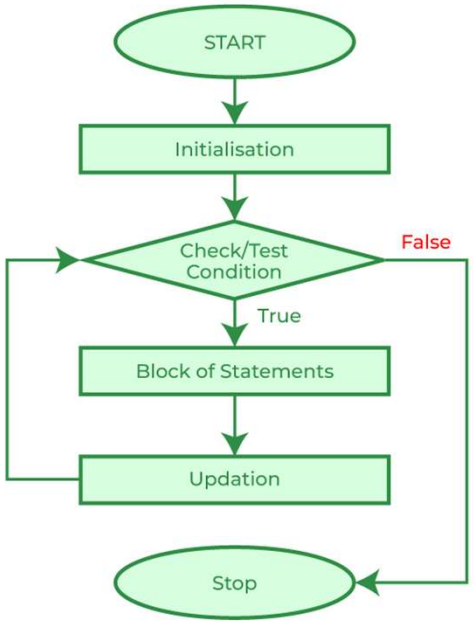
```
for(initialization; control expression; incrementing/decrementing)
{
    Statement 1;
    Statement 2;
    Statement 3;
}
```

Loop Body

```
for( a = 5;   a <= 10;   a++ )
```

Initilization — Condition — Increment (++) or Decrement (--)

Flow Chart

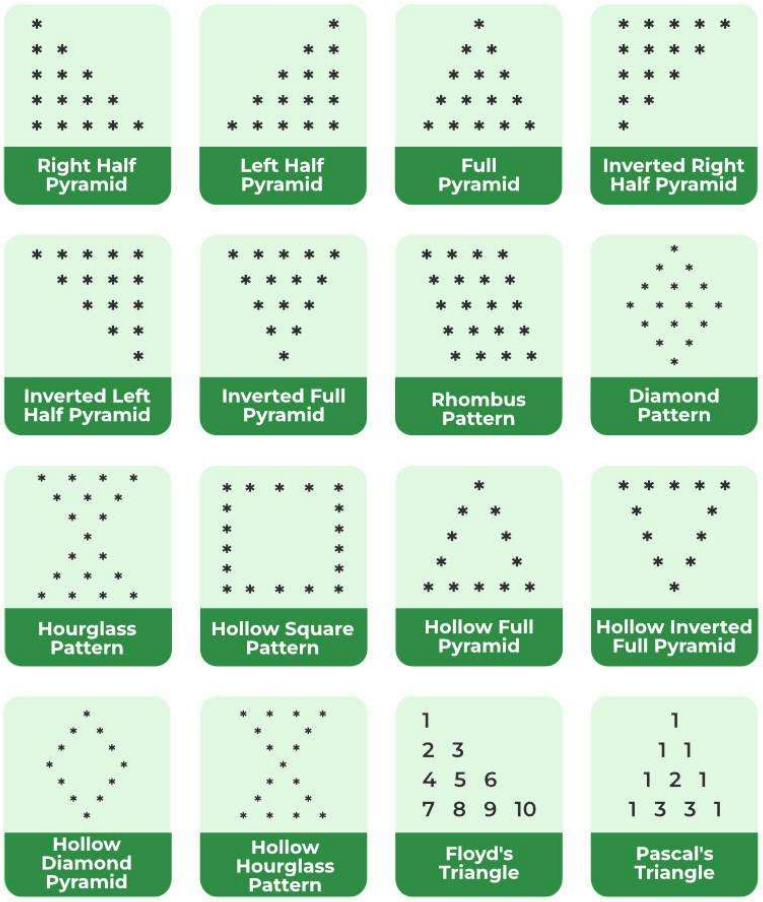


- The initial value of the for loop is performed only once.
- The condition is a Boolean expression that tests and compares the counter to a fixed value after each iteration, stopping the for loop when false is returned.
- The incrementation/decrementation increases (or decreases) the counter by a set value.

do while Example

Print welcome to SNS College Technology 10 times

```
void main()
{
int count;
for(count =0;count<10;count++)
{
printf("welcome to SNS College Technology");
}
getch();
}
```


Right Half Pyramid
Left Half Pyramid
Full Pyramid
Inverted Right Half Pyramid
Inverted Left Half Pyramid
Inverted Full Pyramid
Rhombus Pattern
Diamond Pattern
Hourglass Pattern
Hollow Square Pattern
Hollow Full Pyramid
Hollow Inverted Full Pyramid
Hollow Diamond Pyramid
Hollow Hourglass Pattern
1
2 3
4 5 6
7 8 9 10
Floyd's Triangle
1
1 1
1 2 1
1 3 3 1
Pascal's Triangle

References

- https://www.programiz.com/c-programming/c-operators

Thank You!