LOOPS

LOOPS

- Loops are used when a task needs to be number of times.
  - Printing numbers from 1 to 100
  - Printing even numbers from 1 to 50
- Loops can be categorized as :
  - Pre Tested or Entry Controlled Loops:
    - ***while*** loop, ***for*** loop
  - Post Tested Loops or Exit Controlled Loops :
    - ***do-while*** loop

while LOOP

- Syntax :

```
while(expression)
{
    statements;
}
```

where expression is logical expression, which results in true(non-zero) or false(0)

for LOOP

- Syntax:

  for (initialization; condition; updation)

  {

  statements;

  }

iniatialization : initialize parameters

condition : It is logical expression. Must result true for execution of ***for*** loop.

updation : increment or decrement value of counter

do-while LOOP

- Syntax:

do

{

statements;

}***while*** (condition);

- Condition is checked at the end of the loop.
- Body of loop always execute at least once.

NESTED LOOP

- A loop can be a part of another loop.
  - Inner loop is nested in outer loop

```
for (i=1; i<10; i++)
{
   for (j=1; j<10; j++)
   {
   }
}
```

- Inner loop executes completely for each value of outer loop.

break and ***continue*** STATEMENT

- ***break*** transfers the control to the statement following the loop in which it is written.
- ***break*** can be used to handle errors or exceptional condition.
- ***continue*** transfers the control to the beginning of the loop in which it is written.
- In loop, when control goes to ***continue*** it skips that iteration.

Using two ***breaks***

```
for (…….)
{
    for(……)
    {
        …
        if(calamity)
        break;          //breaks from inner loop
    }
    if(disaster)
    break;              //breaks from outer loop
```

goto STATEMENT

- Control can be transferred to some other part of the program by using this statement.
- Syntax :

  goto label;

  where label is an identifier used to label the target statement to which control will be transferred.

  label : statements;
- goto is used to exit from deeply nested loops, since break can exit from only one loop at a time.

Comma OPERATOR

- It allows more than one initialization and updation in ***for*** loop.
- ***for*** (expr1**,** expr2 ; condition; expr3);
- ***for*** (expr1 ; condition; expr2**,** expr3);
- ***for*** (expr1**,** expr2 ; condition; expr3**,** expr4);
- Only one condition is allowed, multiple conditions can be given using logical operators.

TYPES OF ERRORS

- Syntax Errors
  - Statement Missing, Unknown identifier
- Logical Errors
  - Counter not incremented, Wrong Expression
- Run-time Errors
  - Division by zero
  - Dynamic memory allocation failed
- Linker Errors
  - Function definition missing