

INCREMENT/DECREMENT OPERATORS

C supports both increment (add 1 to) and decrement (subtract 1 from) operators. These are :

- Increment ++
- Decrement --

e.g. **count++;** is the same as **count = count + 1;**

Similarly **count--;** is the same as **count = count – 1;**

ITERATION (REPETITION or LOOPING)

Programming construct that allows the repeating of a task. The number of repetitions can take one of three forms :

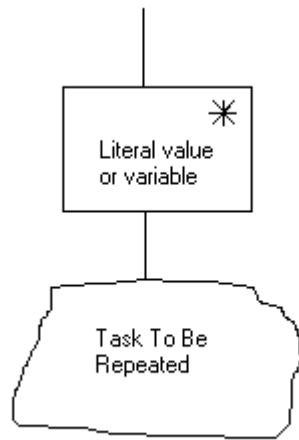
- **Fixed** - do the task a set number of times e.g. 100 times
- **Endless** - never stop doing the task - typically used in control systems e.g. heart pacemakers, etc
- **Continuously Evaluated** - a condition is checked to determine if the task is to be repeated again. This takes two forms :
 - **0 or more times** - while a condition exists, perform the task. The condition is checked on entry and thus the task may not be completed at all
 - **1 or more times** - similar to above but the condition is checked after the task is completed one time.

IMPLEMENTATION IN C

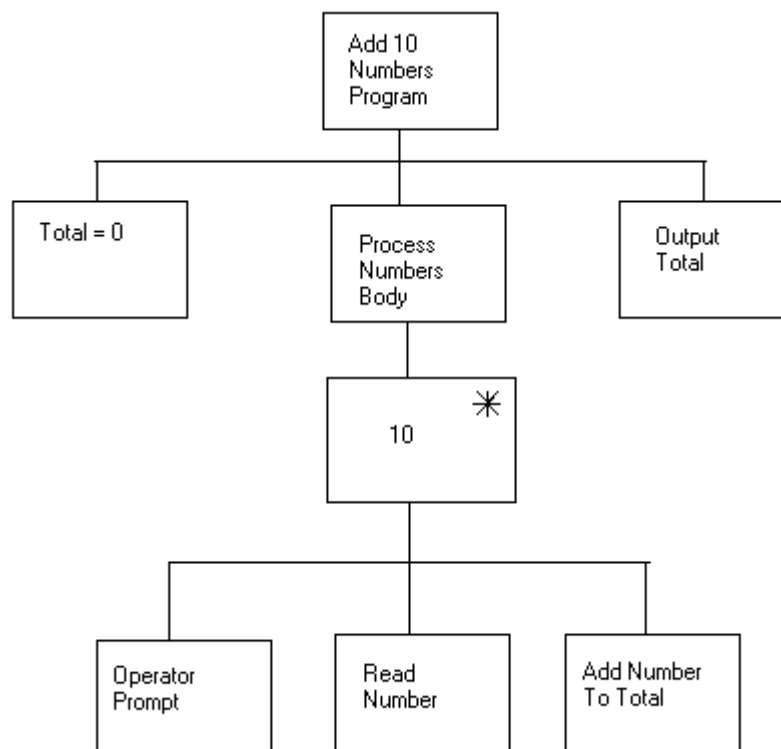
Type of iteration	Implementation In C
Fixed	for loop
Endless	Modified while loop
0 or more times	while loop
1 or more times	do .. while loop

FIXED LOOP

1. Design Template In Structure Charts



Example : Produce a program that adds 10 numbers.



2. Implementation In C

- Implemented as a **'for'** loop
- Computer needs a variable (memory location) called the loop control to hold the count of the number of times the required task has been completed.
- The loop control must be of a countable data type i.e. a whole number (integer) so is typically a data type **'int'** but could be of data type **'char'** as each character has an unique number (ASCII value).
- Skeleton templates :

Single Action in Task Body

```
for (          )
    statement;
```

Multi Action In Task Body

```
for (          )
{
    statement 1;
    statement 2;
    |
    statement n;
}
```

- Basic information in control line :

```
for ( set loop control to ; expression when ; modify loop control )
      start value           false exits loop
```

- Example : Add 10 numbers

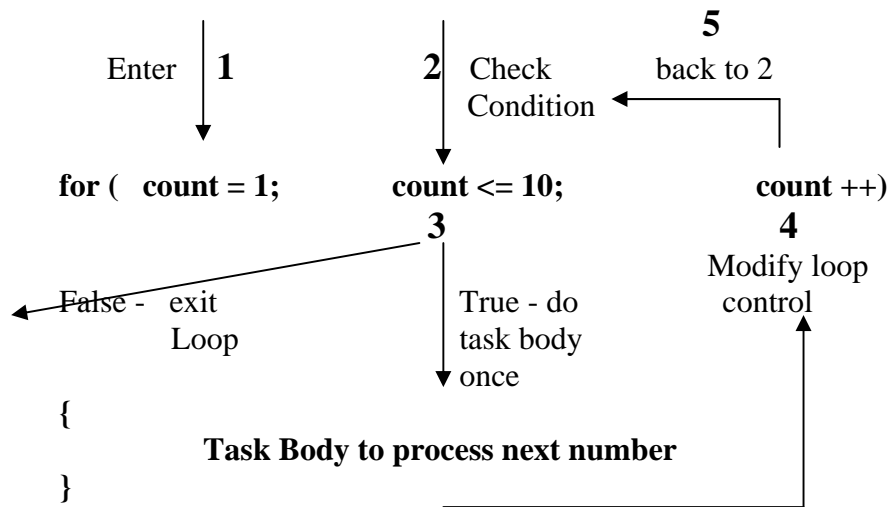
```
/****** GLOBAL VARIABLES *****/
*/
int count, /*Loop control - count of numbers processed*/
    number, /* Number input from the keyboard*/
    total; /*Total value of the input numbers*/

/****** MAIN PROGRAM *****/
*/
void main (void)
{
    total = 0;
    |
    for ( count = 1; count <= 10; count ++ )
    {
        printf("\n\nPlease input number %d : ", count);
        scanf("%d", &number);
        total = total + number;
    }

    printf("\n\nThe total of the 10 numbers was %d", total);

/*******/
```

- Basic operation :



- DO NOTS :

- Put semi-colon at end of the control line

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

- Modify the loop control inside the task body

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

- Use the value of the loop control outside the loop (it may not be what you expect; i.e. it is platform dependant - some platforms leave it at the exit value but others change it).

- DO :

- Use the value of the loop control inside the loop if required

```
for ( count = 1; count <= 10; count++)
{
printf("\n\nPlease input number %d : ", count);
```

- Use meaningful names for the loop control variable

e.g. the requirement is to process data for the 24 hours in the day

```
for ( hours = 1; hours <= 24; hours ++)
```

is more meaningful than

```
for ( count = 1; count <= 24; count ++)
```

e.g. input the annual sales for years 1990 to 2000 inclusive

could be implemented as

```
for (count = 1; count <=11; count++)
```

but far better and much more readable as

```
for (years = 1990; years <= 2000; years ++ )
```

FIXED LOOP EXERCISES

1. Write a program which reads and adds 5 integer numbers and then outputs their average.
2. Write a program which reads and multiplies 6 real numbers. Output the product to the screen.
3. Write a program to print a " 4 times table " in the form :

1 x 4 = 4
2 x 4 = 8
3 x 4 = 12
etc.

4. Modify exercise 3 to read an integer, say n, and outputs a "n times table ".
5. Write a program which reads in 6 integer numbers and outputs :
 - The number of positive numbers.
 - The number of negative numbers.
 - The sum of the positive numbers.
 - The sum of the negative numbers.
6. Write a program which reads in 10 characters and outputs the number of occurrences of the letter a ('A' or 'a').
7. Write a program that asks for :

Cost of item
Money handed over.

on 3 occasions.

Output is the change in the appropriate number of coins (£1, 50p, etc) with assumption that the maximum of each denomination is available.

After each output, allow the operator to control the program via the selection of a key after a suitable operator prompt.

FIXED LOOP EXERCISE ALGORITHMS

1. Set total = 0

Do 5 times

Operator prompt to ask for an integer number Read number Add number to total

Calculate average

Output average

2. Set product = 1

Do 6 times Operator prompt to ask for a real number Read number Update running product

Output product

4. Operator prompt ask for which times table
Read number

Output line of table

 - 12 times

5. Set positive count to 0
Set negative count to 0
Set positive sum to 0
Set negative sum to 0

Do 6 times

Operator prompt to ask for integer number Read number

If number positive

 Add 1 to positive count

 Add number to positive sum

Else

 Add 1 to negative count

 Add number to negative sum

Output positive count

Output negative count

Output positive sum

Output negative sum

6. Total of letter a = 0

Do 10 times

Operator prompt to ask for character
Read character

If character = a
Increment total of letter a

Output total of letter a

7. Do 3 times

Operator prompt to ask for cost of item
Read Cost

Operator prompt to ask for money given
Read money given

Output change (starting with highest denomination)

Operator prompt to press key
Read key