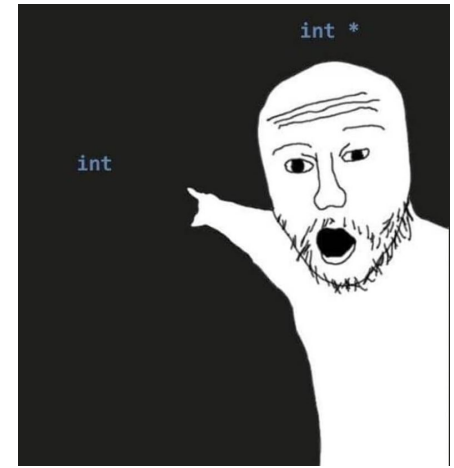




POINTERS



Sumaiyah Zahid

ADDRESS SPACE



Name = FAST

Value / Occupants = Students

Address = St-4, Sector 17-D, National Hwy 5, Karachi

ADDRESS SPACE

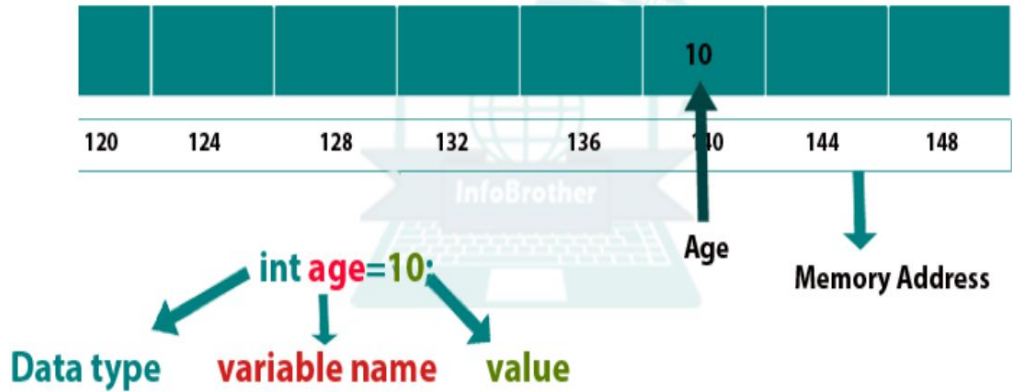
```
int x = 10;
```

Name = x

Value = 10

Address = Any memory location

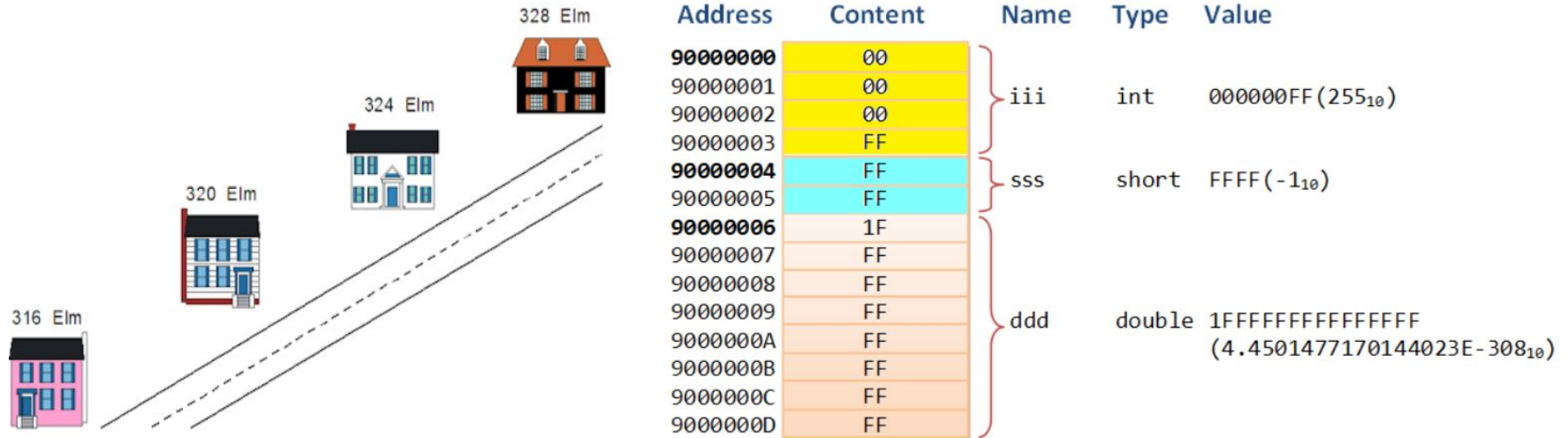
You can access that memory location using **&** operator.



ADDRESS SPACE

Like houses, each variable has a unique address that gets larger as you move along the street or through memory.

The contents of a variable, like the occupants of a house, can change over time, but the address of a variable, like the address of a house, is fixed and does not change (unless the house is picked up by a tornado and carried over a rainbow, which almost never happens).



```
#include <stdio.h>
int main()
{
    int a=0;
    char b='k';
    double c=345678666;
    float d=4.56;
    printf("Printing the values\n");
    printf("int = %d\nchar = %c\ndouble = %lf\nfloat = %f\n",a,b,c,d);

    printf("Printing the addresses\n");
    printf("int = %d\nchar = %d\ndouble = %d\nfloat = %d \n", &a, &b,
&c, &d);

    return 0;
}
```

```
Printing the values
int = 0
char = k
double = 345678666.000000
float = 4.560000
```

```
Printing the addresses
int = 6487580
char = 6487579
double = 6487568
float = 6487564
```

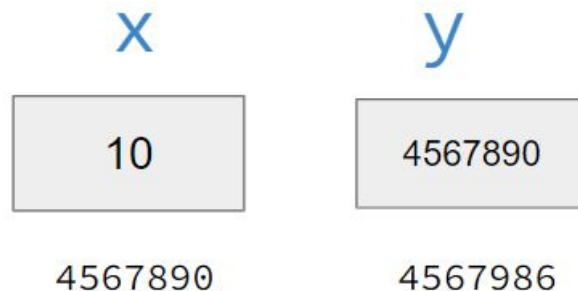
POINTERS DEFINITION

To hold any address of variable, computer needs another storage location.

Pointer is a variable which holds address of another variable.

```
int x = 10;
```

```
int *y = &x;
```



POINTERS DEFINITION

& Address Operator

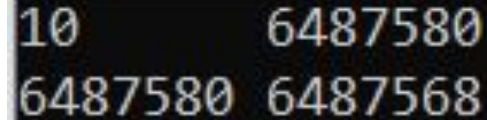
* Dereferencing Operator

```
int x = 10;
```

```
int *y = &x; or int *y; y = &x;
```

```
printf("%d %d\n", x, y);
```

```
printf("%d %d\n", &x, &y);
```



10	6487580
6487580	6487568

A memory dump visualization showing two rows of data. The first row contains the value '10' and the address '6487580'. The second row contains the address '6487580' and another address '6487568'.

POINTERS DEFINITION

Interestingly we can access value of x through y variable too.

y = holds the address of x
*y = pointer accessing value
of x

```
printf("%d %d\n", y, *y);
```




```
#include <stdio.h>
int main()
{
    int a=0;           int *ap;
    char b='k';        char *bp;
    double c=345678666;double *cp;
    float d=4.56;      float *dp;

    ap = &a; bp = &b; cp = &c; dp=&d;
    printf("Printing the values\n");
    printf("int = %d\nchar = %c\ndouble = %lf\nfloat = %f\n", *ap, *bp,
*cp, *dp);

    printf("Printing the addresses\n");
    printf("int = %d\nchar = %d\ndouble = %d\nfloat = %d \n", ap, bp,
cp, dp);

    return 0;
}
```

```
Printing the values
int = 0
char = k
double = 345678666.000000
float = 4.560000
```

```
Printing the addresses
int = 6487548
char = 6487547
double = 6487536
float = 6487532
```

```
#include <stdio.h>
int main()
{
    int fno, sno, *ptr, *qtr, sum;

    printf(" Input the first number : ");
    scanf("%d", &fno);
    printf(" Input the second  number : ");
    scanf("%d", &sno);
    ptr = &fno;
    qtr = &sno;

    sum = *ptr + *qtr;

    printf(" The sum of the entered numbers is : %d\n\n",sum);

    return 0;
}
```

WHY POINTERS?

- Increases the execution speed of program.
- Used for dynamic memory allocation.
- Pass by reference.
- Pointers makes possible to return more than one value in functions.
- To access variables that are declared outside the functions.
- Strings and arrays are more efficient with pointers.

PRACTISE EXERCISE

1. The length & breadth of a rectangle and radius of a circle are input through the keyboard. Write a program to calculate the area & perimeter of the rectangle, and the area & circumference of the circle using pointers.
2. Write a program that asks the user to enter two numbers, obtains them from the user and prints their sum, product, difference, quotient and remainder using pointers.



LOOPY LOOP

Sumaiyah Zahid

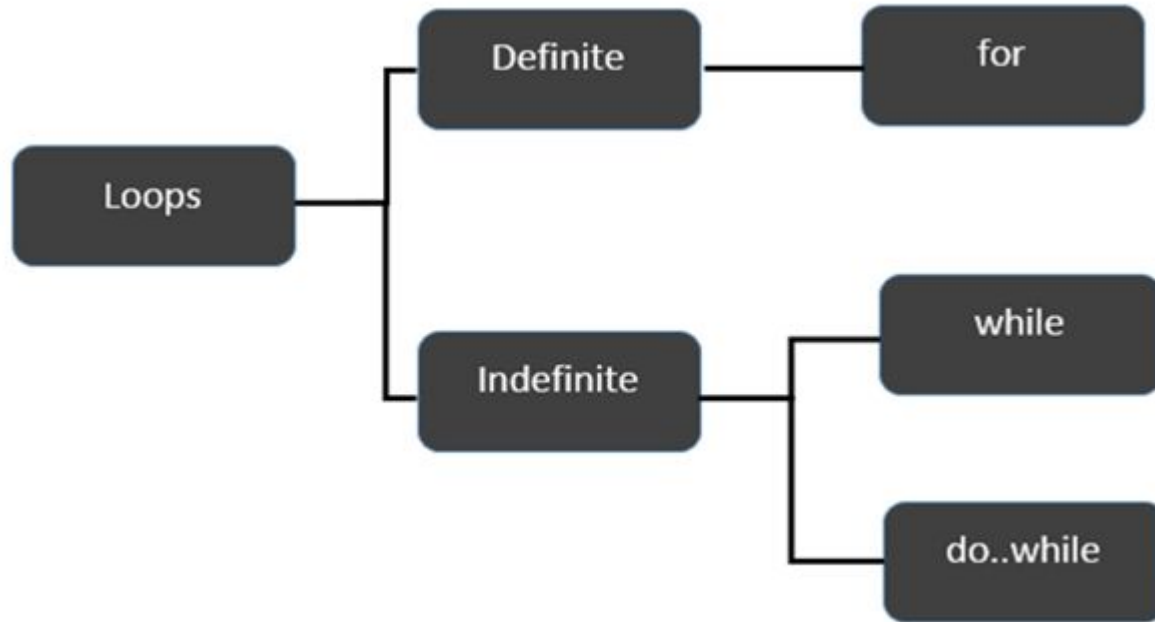
LOOP WORLD

1. Start
2. Eat
3. Sleep
4. Code
5. Go to Line 2
6. End

The real power of computers is in their ability to repeat an operation or a series of operations many times.

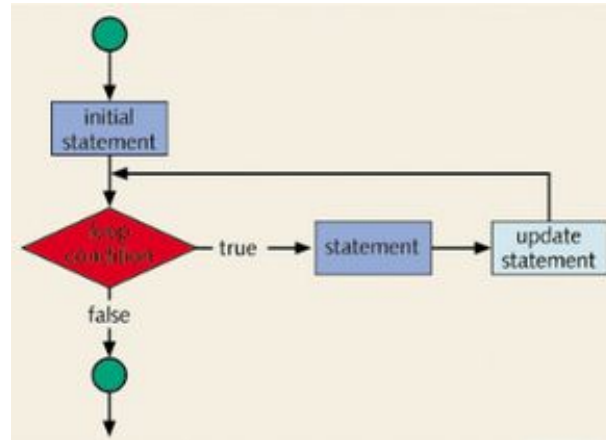
When action is repeated many times, the flow is called a loop.

LOOP LOGIC STRUCTURE



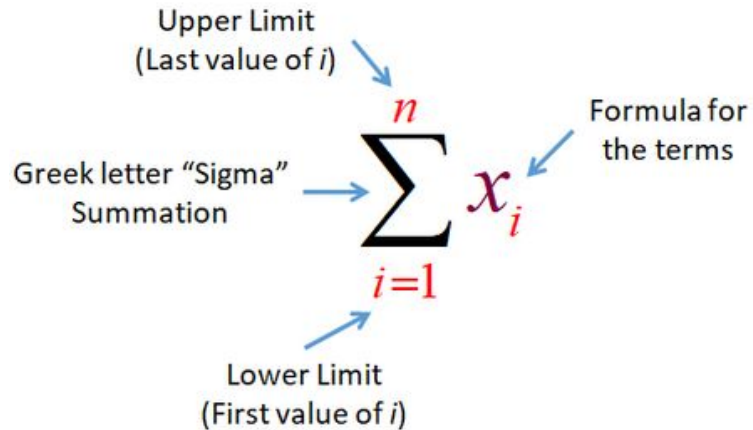
FOR LOOP

`for(initial statement; loop condition; update statement)`



FOR LOOP

Sigma Notation



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

Sitesbay.com

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

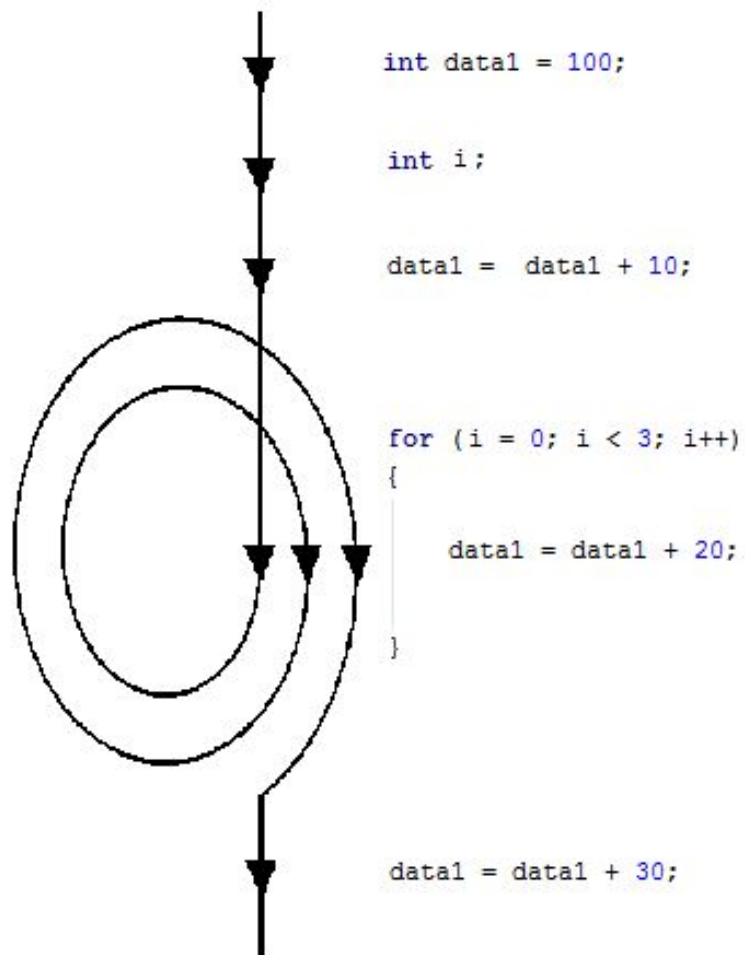
FOR LOOP EXECUTION



The for loop executes as follows:

1. The initial statement executes.
2. The loop condition is evaluated. If the loop condition evaluates to `true`
 - a. Execute the `for` loop statement.
 - b. Execute the update statement.
3. Repeat Step 2 until the loop condition evaluates to `false`.

The initial statement usually initializes a variable is the first statement to be executed and is executed only once.

```
#include <stdio.h>
int main()
{
    int i;
    for (i=0; i<5; i++)
    {
        printf("Hello World\n");
    }
    return 0;
}
```



simple	compound
<pre>for(i = 1; i <= 5; i++) printf("Output of stars.\n"); printf("*");</pre>	<pre>for(i = 1; i <= 5; i++) { printf("Output of stars.\n"); printf("*\n"); }</pre>
 <pre>Output a line of stars. Output a line of stars. Output a line of stars. Output a line of stars. Output a line of stars. *</pre> <pre>for(i = 1; i <= 5; i++); printf("*");</pre> <pre>*</pre>	 <pre>Output a line of stars. * Output a line of stars. * Output a line of stars. * Output a line of stars. * Output a line of stars. *</pre>

- All three statements can be omit—initial statement, loop condition, and update statement.

```
for(;;)  
    printf("Hello\n");
```

```
Hello  
Hello  
.  
.
```

Infinite output;
press break to
stop the program
running

- Count backward

```
for(i = 10; i >= 1; i--)  
    printf("%d ", i);
```

```
10 9 8 7 6 5 4 3 2 1
```

- Increment (or decrement) the loop control variable

```
for(i = 1; i <= 20; i = i + 2)  
    printf("%d ", i);
```

```
1 3 5 7 9 11 13 15 17 19
```



Suppose we want to add five numbers (say to find their average).

From what you have learned so far, you could proceed as follows.

```
scanf("%d %d %d %d %d",&num1,&num2,&num3,&num4,&num5);
```

```
sum = num1+num2+num3+num4+num5;
```

```
average = sum/5;
```

Suppose we wanted to add and average 100, or 1000, or more numbers. We would have to declare that many variables, and list them again in scanf statement, and perhaps, again in the output statement.

```
#include <stdio.h>
int main()
{
    int sum=0,i, new_num,avg;
    for (i=0; i<5; i++)
    {
        printf("Enter Number ");
        scanf("%d",&new_num);
        sum=sum+new_num;
    }
    avg=sum/5;
    printf("The Sum is %d \n",sum);
    printf("The Average is %d \n", avg);
    return 0;
}
```


Summation
(capital sigma)

$$\sum_{n=0}^4 3n$$

```
sum = 0;  
for( n=0; n<=4; n++ )  
    sum += 3*n;
```

Product
(capital pi)

$$\prod_{n=1}^4 2n$$

```
prod = 1;  
for( n=1; n<=4; n++ )  
    prod *= 2*n;
```

HOME EXERCISE

Using for loop print a multiplication table of any user given number.

Input integer: 2

$$2*1=2$$

$$2*2=4$$

$$2*3=6$$

...

HOME EXERCISE

Write a program in C to display the cube of the number upto given an integer.

Input number of terms : 5

Expected Output :

Number is : 1 and cube of the 1 is :1

Number is : 2 and cube of the 2 is :8

Number is : 3 and cube of the 3 is :27

Number is : 4 and cube of the 4 is :64

Number is : 5 and cube of the 5 is :125

WHILE LOOP

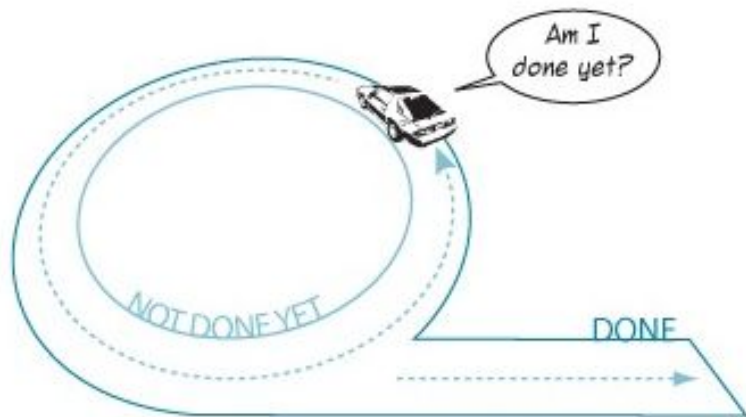
Also known as conditional loop.

```
while (not succeed)
```

```
{
```

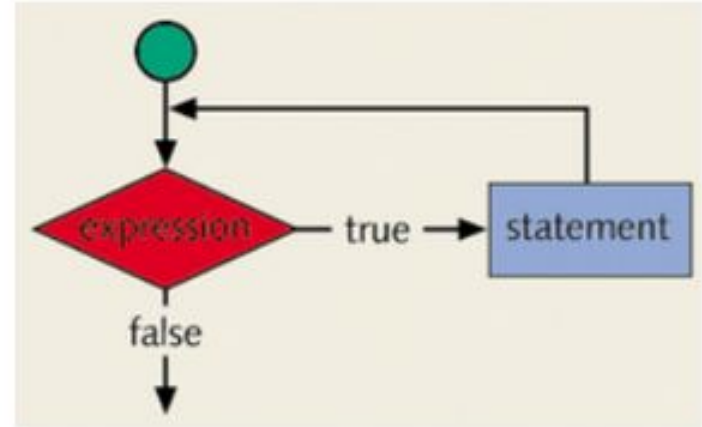
```
    keep_trying();
```

```
}
```



WHILE LOOP

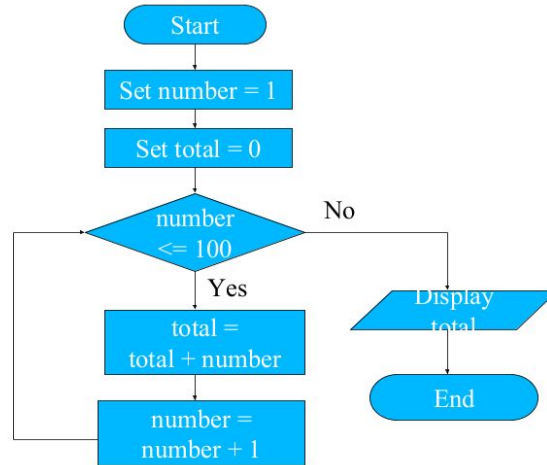
```
while(programmer==alive)  
{  
  if(day==normal)  
    coding();  
  else if(day==special_event)  
    coding();  
  else  
    coding();  
}
```



```
#include <stdio.h>
int main()
{
    int i=0;
    while (i<5)
    {
        i++;
        printf("Hello World\n");
    }
    return 0;
}
```

PRACTICE QUESTION

Make a C program using while loop which calculates the sum of 100 integers 1, 2, 3, ..., 100.



```
#include <stdio.h>
int main()
{
    int i=0, sum=0;
    while (i<100)
    {
        i++;
        sum=sum+i;
    }
    printf("The Sum is %d.",sum);
    return 0;
}
```


PRACTICE QUESTION

Make a C program using while loop which calculates the sum of n integers until sum is less than 100.

```
#include <stdio.h>
int main()
{
    int i=0, sum=0;
    while (sum<=100)
    {
        i++;
        sum=sum+i;
    }
    printf("The Sum is %d.",sum);
    return 0;
}
```

PRACTICE QUESTION

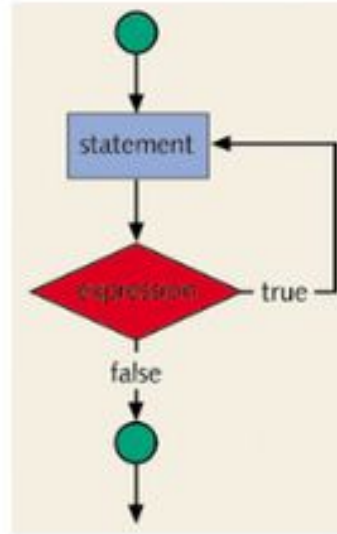
Make a C program to convert decimal number to binary.

```
#include <stdio.h>
int main()
{
    int num,a;
    printf("Enter decimal number :");
    scanf("%d", &num);
    while (num!=0)
    {
        a=num%2;
        printf("%d",a);
        num=num/2;
    }

    return 0;
}
```

DO- WHILE LOOP

```
do  
{  
    keep_trying();  
}  
while (not succeed);
```



```
while (not edge) {  
    run();  
}
```

```
do {  
    run();  
} while (not edge);
```



WHAT'S THE OUTPUT?

```
i = 11;  
while(i <= 10)  
{  
    printf("%d ",i);  
    i++;  
}
```

```
i = 11;  
do  
{  
    printf("%d ",i);  
    i++;  
}  
while(i <= 10);
```



PRACTICE QUESTION

Write a C program using do while to keep asking for a number until you enter a +ve number.


```
#include <stdio.h>
int main()
{
    int num;
    do
    {
        printf("Enter any number :");
        scanf("%d", &num);
    }
    while (num < 0);
    return 0;
}
```

PRACTICE QUESTION

1. Make a C program which ask the user for the password until it matches with the actual one.
2. Make a C program which ask the user for the password. If it doesn't match in 3 trials, it shouldn't ask again.

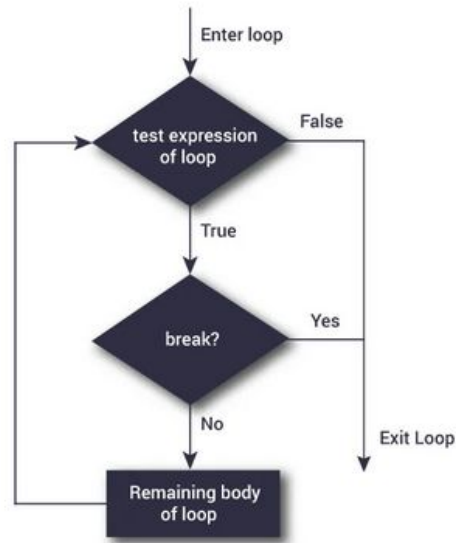
HOME ASSIGNMENT

1. Make a C program to print first 50 even numbers.
2. Make a C program to read an age of 15 person & find out how many of them fall under :
 - a) Still a baby- age 0 to 5
 - b) Attending school - age 6 to 17
 - c) Adult life-age 18 & over

BREAK

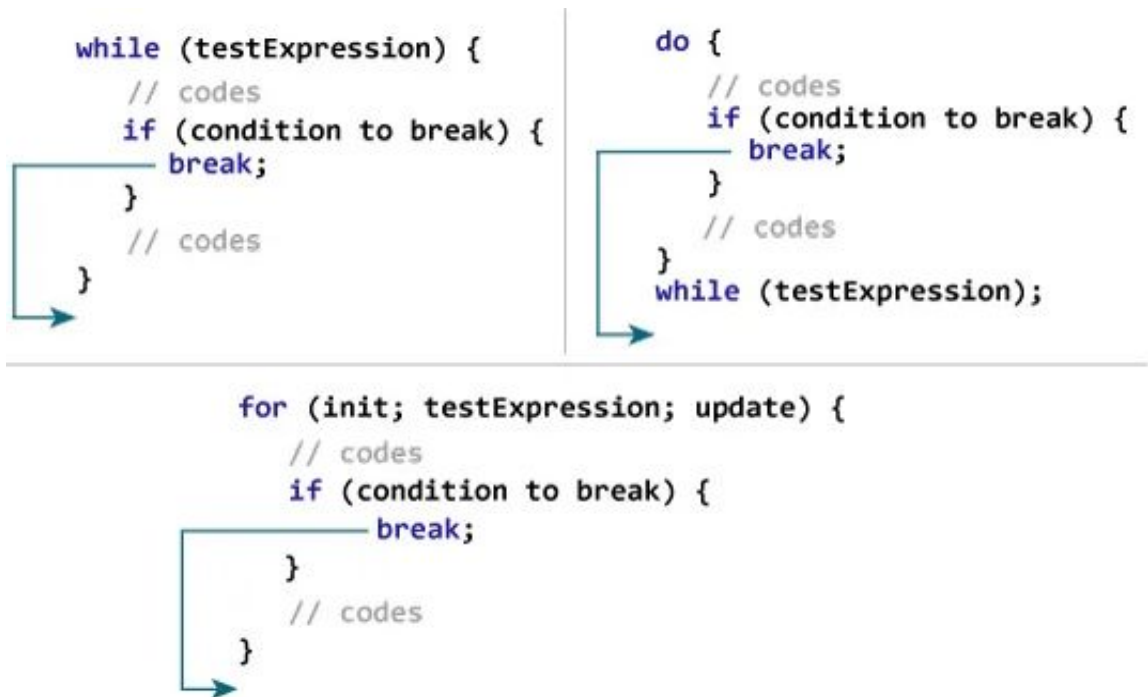
You can use the break statement in while, for, and do...while loops to alter the flow of control.

When the break statement executes in a repetition structure, it immediately exits from these structures.



```
while (not succeed)
{
    if(dead)
        break;
    keep_trying();
}
```

BREAK



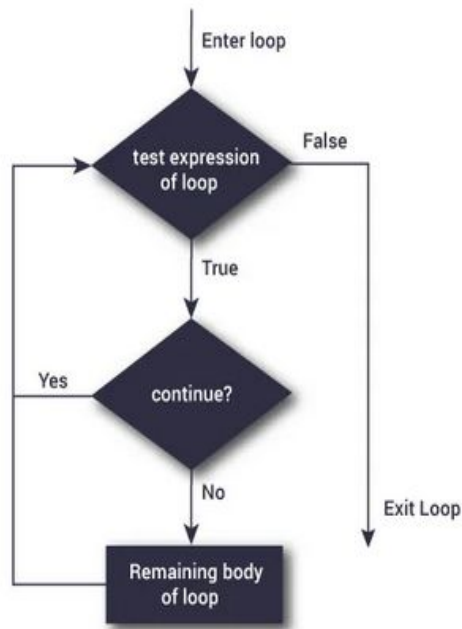
CONTINUE

The `continue` statement is used in while, for, and do-while structures.

When the `continue` statement is executed in a loop, it skips the remaining statements and proceeds with the next iteration of the loop.

In a while and do-while structure, the expression (that is, the loop-continue test) is evaluated immediately after the `continue` statement.

In a for structure, the update statement is executed after the `continue` statement, and then the loop condition (that is, the loop-continue test) executes.



CONTINUE

```
→ while (testExpression) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```

```
do {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
} → while (testExpression);
```

```
→ for (init; testExpression; update) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```

CAN YOU PRINT THIS USING FOR LOOP?

1

2

3

4

5

CAN YOU PRINT THIS USING FOR LOOP?

```
1      for( i=1; i<=5; i++)  
2  
3      {  
4          printf(“%d \n”, i);  
5      }
```

CAN YOU PRINT THIS USING FOR LOOP?

1 2 3 4 5

CAN YOU PRINT THIS USING FOR LOOP?

```
1 2 3 4 5    for( i=1; i<=5; i++)  
              {  
                printf("%d ", i);  
              }
```

CAN YOU PRINT THIS USING FOR LOOP?

1 2 3 4 5

1 2 3 4 5

CAN YOU PRINT THIS USING FOR LOOP?

```
1 2 3 4 5    for( i=1; i<=5; i++)  
               { printf(“%d ”, i);}  
  
1 2 3 4 5    printf ( “\n”);  
  
               for( i=1; i<=5; i++)  
               {   printf(“%d ”, i);}
```

CAN YOU PRINT THIS USING FOR LOOP?

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

CAN YOU PRINT THIS USING FOR LOOP?

```
for( i=1; i<=5; i++)  
    printf("%d ", i);  
printf ("\n");  
for( i=1; i<=5; i++)  
    printf("%d ", i);  
printf ("\n");
```

```
for( i=1; i<=5; i++)  
    printf("%d ", i);  
printf ("\n");  
for( i=1; i<=5; i++)  
    printf("%d ", i);  
printf ("\n");
```

IT'S A FOR OF FOR LOOP

```
for( j=1; j<=5; j++)  
{  
    for( i=1; i<=5; i++)  
        printf("%d ", i);  
    printf ("\n");  
}
```


CAN YOU PRINT THIS USING NESTED FOR LOOP?

row=1 1

row=2 1 2

row=3 1 2 3

Row=4 1 2 3 4

row=5 1 2 3 4 5

IT'S A FOR OF FOR LOOP

```
for( j=1; j<=5; j++)  
{  
    for( i=1; i<=j; i++)  
        printf("%d ", i);  
    printf ("\n");  
}
```

CAN YOU PRINT THIS USING NESTED FOR LOOP?

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

IT'S A FOR OF FOR LOOP

```
for( j=1; j<=5; j++)  
{  
    for( i=1; i<=6-j; i++)  
        printf("%d ", i);  
    printf ("\n");  
}
```

HOME ASSIGNMENT

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

CAN YOU PRINT THIS USING NESTED FOR LOOP?

```

        1
      1 2
    1 2 3  5-i
  1 2 3 4
1 2 3 4 5
```

IT'S A FOR OF FOR LOOP

```
for( j=1; j<=5; j++)  
{  
    for( i=1; i<=5-j; i++)  
        printf(" ", i);  
    for( i=1; i<=j; i++)  
        printf("%d", i);  
    printf ("\n");  
}
```

NESTED LOOPS

```
while(condition)
{
    printf("Outer loop");
    while(condition)
    {
        printf("Inner Loop");
    }
    printf("Outer loop");
}
```


CAN YOU PRINT THIS USING NESTED WHILE LOOP?

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

NESTED WHILE

```
int i=1,j;  
while(i<6)  
{  
    j=1;  
    while(j<i)  
    {  
        printf("%d",j);  
  
        j++;  
    }  
    printf("\n");  
    i++;  
}
```

CAN YOU PRINT THIS USING NESTED WHILE LOOP?

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

NESTED WHILE

```
int i=1,j;           // Main Loop
while(i<6)           // Outer Loop
{
    j=1;
    while(j<6-i)      // Inside loop 1
    {
        printf(" ");
        j++;
    }
    j=1;
    while(j<i)         // Inside Loop 2
    {
        printf("%d ",j);
        j++;
    }
    printf("\n");      // Outer Loop
    i++;
}
```

CAN YOU PRINT THIS USING NESTED WHILE LOOP?

$1 \quad 2(i) + 1 \quad // \quad 2(i) - 1$

123

12345

1234567

123456789

NESTED WHILE

```
int i=1,j;           // Main Loop
while(i<6)           // Outer Loop
{
    j=1;
    while(j<6-i)      // Inside loop 1
    {
        printf(" ");
        j++;
    }
    j=1;
    while(j<=2*i-1)    // Inside Loop 2
    {
        printf("%d",j);
        j++;
    }
    printf("\n");      // Outer Loop
    i++;
}
```

NESTED LOOPS

```
do
{
    printf("Outer loop");
    do
    {
        printf("Inner Loop");
    } while(condition);
    printf("Outer loop");
} while(condition);
```