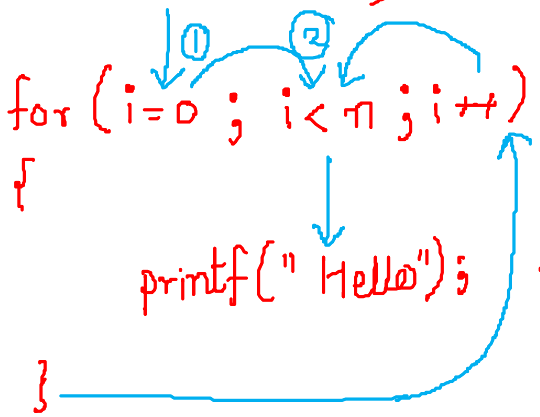
for loop :- int i;

```
for (i=0 ; i<n ; i++)
{
    printf(" Hello");
}
```



Output :-

```
Hello
Hello
Hello
```

```
for ( i=10 ; i>5 ; i--);
{
  printf("Hello");
}
```

1 time

3)
```
for (i = 1 ; i <= 7 ; i--)
{
  printf("Hello");
}
```

i $\boxed{1}$ <= 7

1)
```
for (i=1 ; i<5 ; i++)
{
   printf( "Hello");
}
```

i $\boxed{\cancel{1}}$ n $\boxed{5}$

~~2~~ ~~3~~ ~~4~~ 5

1 2 3 4 } 4 times

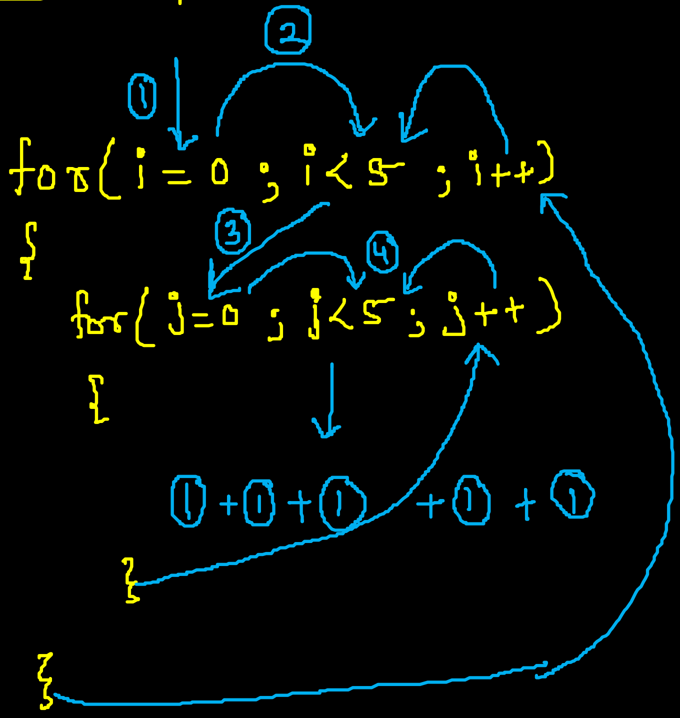
2)
```
for(i=0 ; i<=5 ; i++)
{
 printf ("Hello");
}
```

i $\boxed{\cancel{0}}$ n $\boxed{5}$

~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

1 2 3 4 5 6 } 6 times

# Nested for loop:-



```
for( i = 0 ; i< 5 ; i++)
{
   for( j=0 ; j<5 ; j++ )
   {

      1 + 1 + 1 + 1 + 1
   }
}
```

| i | j | |
|---|---|---|
| $\boxed{\cancel{0}}$ | $\boxed{\cancel{0}}$ | 0 |
| 1 | $\cancel{1}$ | |
| | $\cancel{2}$ | |
| | $\cancel{3}$ | |
| | $\cancel{4}$ | |
| | 5 | |

| | | |
|---|---|---|
| i = 0 | j [0 - 4] = 5 | |
| i = 1 | j [0 - 4] = 5 | |
| ⋮ | | |
| i = 4 | j [0-4] = 5 | |
| i = 5 | Break | $\boxed{25}$ |

```
for( i = 1 ; i< 5 ; i++ )
{
   printf("1");

   for( j = 1 ; j < 5 ; j++ )
   {
      printf("2");

   }
   printf("3");

}
```

| i | j |
|---|---|
| $\boxed{\cancel{1}}$ | $\boxed{\cancel{1}}$ |
| 2 | $\cancel{2}$ |
| | $\cancel{3}$ |
| | $\cancel{4}$ |
| | 5 |

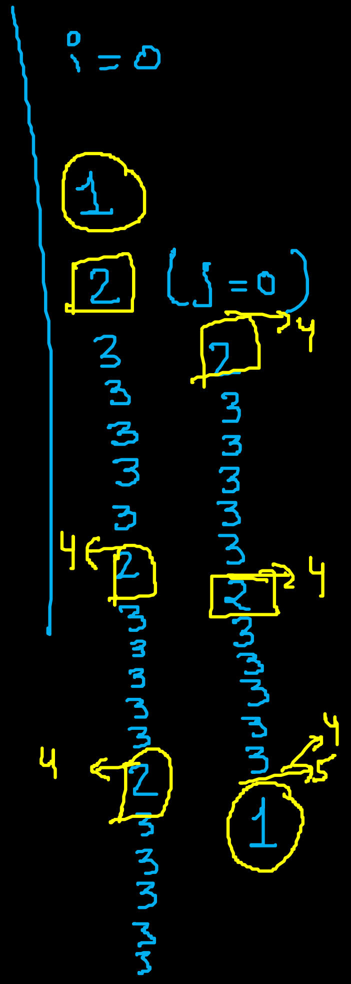
1 2 2 2 2 3 — i = 1

1 2 2 2 2 3 — i = 2

1 2 2 2 2 3 — i = 3

1 2 2 2 ~~3~~ 2 3 — i = 4

```
for ( i=0 ; i< 5 ; i++)   ⟶   i = [0 - 4] = 5 times
{
    printf(" 1");   ⟶   (5 times)
    for (j =0 ; j< 5 ; j++)   ⟶   j = [0-4] = 5 times
    {
        printf ("2");   ⟶   5 X 5 = (25 times)
        for (K=0 ; K < 5 ; k++)   ⟶   k = [0-4] = 5 times
        {
            printf ("3");   ⟶   5 X 5 X 5 = (125 times)
        }
        printf (" 4");   ⟶   25 times
    }
    Printf (" 5 ");   ⟶   5 times
}
```



```
for( i=0 ; i<5 ; i++ )
{
    printf ("1");
}
```

} 5 times

+

```
for( j=0 ; j<5 ; j++)
{
    printf ("2");
}
```

} 5 times

$\boxed{11111\ 22222} \longrightarrow$ 10 times

1-D $\rightarrow$ single for loop

```
for (r=0 ; r<4 ; r++)        → 5
{                                x
  for (c=0 ; c<3 ; c++)      → 5
  {
      printf ("1");          → 25
  }
}
```

2-D

|  | c=0 | 1 | 2 |
|---|---|---|---|
| r=0 | 1 | 1 | 1 |
| 1 | 2 | 2 | 2 |
| 2 | 3 | 3 | 3 |
| 3 | 4 | 4 | 4 |

no of col = 3

x

no. of row = 4

no. of cells = 12

$$
\begin{array}{c|ccc}
 & c=0 & c=1 & c=2 \\
\hline
r=0 & 1 & 1 & 1 \\
r=1 & 2 & 2 & 2 \\
r=2 & 3 & 3 & 3 \\
r=3 & 4 & 4 & 4
\end{array}
$$

row $\longrightarrow$

column $\downarrow$

row major

12

```
for(r=0; r<4 ; r++)
{
    for(c=0; c<3; c++)
        printf("%d", r+1);
        printf("\n");
}
```

$$
\begin{array}{ccc|l}
c=0 & c=1 & c=2 & \\
\hline
1 & 2 & 3 & \rightarrow r=0 \\
1 & 2 & 3 & \rightarrow r=1 \\
1 & 2 & 3 & \rightarrow r=2 \\
1 & 2 & 3 & \rightarrow r=3
\end{array}
$$

12

column major

```
for(c=0 ; c<3 ; c++)
{
    for(r=0; r<4; r++)
        printf("%d", c+1);
        printf("\n");
}
```

|  | j=0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| i=0 | * |  |  |  |  |
| 1 | * | * |  |  |  |
| 2 | * | * | * |  |  |
| 3 | * | * | * | * |  |
| 4 | * | * | * | * | * |

$j = [0 \ldots\ldots i] \Rightarrow \boxed{j <= i}$

```
for(i=0 ; i<=4 ; i++)
{
    for(j=0; j<=i; j++)
    {
        printf("*");
    }
    printf("\n");
}
```

|  |  |
|---|---|
| i = 0 | j = 0 |
| i = 1 | j = 0, 1 |
| i = 2 | j = 0,1,2 |
| i = 3 | j = 0,1,2,3 |
| i = 4 | j = 0,1,2,3,4 |

```
j=0  1  2  3  4
             *     i=0
          *  *      1
       *  *  *      2
    *  *  *  *      3
 *  *  *  *  *      4
```

```
for( i=0 ; i<=4;i++)
{
   for( j=0 ; j<=4 ; j++)
   {
      if ( j >=4-i)
         printf(" * ");
      else
         printf("  ");
   }
   printf(" \n");
}
```

| | | |
|---|---|---|
| $i = 0$ | $j = 4$ | $(4-0) = 4$ |
| $i = 1$ | $j = 3, 4$ | $(4-1) = 3$ |
| $i = 2$ | $j = 2, 3, 4$ | $(4-2) = 2$ |
| $i = 3$ | $j = 1, 2, 3, 4$ | $(4-3) = 1$ |
| $i = 4$ | $j = 0, 1, 2, 3, 4$ | $(4-4) = 0$ |

$(j >= 4-i)$

break , Continue

```
for( i = 0 ; i< 10 ; i++)
{
    if( i == 4)
        break;
}
```

$\longrightarrow$ it will break the loop inside which it is present

i = 0, 1, 2, 3, 4

```
for ( i = 0 ; i< 10 ; i++)
{
   if ( i == 4)
   continue ;
   printf(" Hello");

}
```

i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Skip

9 times

WAP to print table of a number given by the user.

$2 \times 1 = 2$

$2 \times 2 = 4$

$2 \times 3 = 6$

$\vdots$

$\vdots$

```
*  *  *  *  *
*  *  *  *
*  *  *
*  *
*
```

```
for(i = 0 ; i<5 ;i++)
{
  for(j=0 ; j<=4-i ; j++)
        printf(" *");
        Printf("\n");
  }
```

```
j=0  1  2  3  4  5  6  7  8
             *                 i=0
          *  *  *              =1
       *  *  *  *  *           =2
    *  *  *  *  *  *  *        =3
 *  *  *  *  *  *  *  *  *     =4
```

$$\boxed{(4-i <= j)\ \&\&\ (j <= 4+i)}$$

| | |
|---|---|
| i = 0 | j = 4 |
| i = 1 | j = 3, 4, 5 |
| i = 2 | j = 2, 3, 4, 5, 6 |
| i = 3 | j = 1, 2, 3, 4, 5, 6, 7 |
| i = 4 | j = 0, 1, 2, 3, 4, 5, 6, 7, 8 |

```
for( i= 0 ; i< 5 ; i++)
{
  for( j = 0 ;  j <= 4+i ; j++)
  {
        if ( 4-i <= j)
              printf(" *") ;
        else printf("  ");
  } printf("\n");
}
```

```
j =  0  1  2  3  4  5  6  7  8  9
     *  *  *  *  *  *  *  *  *  *     i = 0
     *  *  *  *           *  *  *  *    = 1
     *  *  *                 *  *  *    = 2
     *  *                       *  *    = 3
     *                             *    = 4
```

```
for(i=0; i<=4; i++)
{
 for(j=0; j<=9; j++)
  {
     if((j<=4-i) || (j>=5+i))
         printf("*");
     else
         printf(" ");
  }
}
```

i=0 , j = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

i=1 , j = (0, 1, 2, 3) (6, 7, 8, 9)

i=2 , j = (0, 1, 2) (7, 8, 9)

i=3 , j = (0, 1) (8, 9)

i=4 , j = (0) (9)

Prateek Jain
9555031137

```
j=0  1 2 3  4  5  6
                          i=0
        1
      1 2 1                =1
    1 2 3 2 1              =2
  1 2 3 4 3 2 1            =3
```

$i=0$ , $j = 3$

$i=1$ , $j = 2, 3, 4$

$i=2$ , $j = 1, 2, 3, 4, 5$

$i=3$ , $j = 0, 1, 2, 3, 4, 5, 6$

$(3-i <= j) \&\& (j <= 3+i)$

```
for (i=0 ; i<=3 ;i++)
{
  for (j=0, c=1; j <= 3+i ; j++)
   {
      if( 3-i <= j)
      {
         printf(" %d" , c);
         if( j < 3)
              c++;
         else
              c--;
      }
      else printf("   ");
} }
```

int c=1

```
j = 0   1   2   3   4
                        i = 0
 *
                          = 1
 *   *
                          = 2
 *       *
                          = 3
 *           *
 *   *   *   *   *        = 4
```

```
for (i = 0 ; i <= 4 ; i++)
{
  for (j = 0 ; j <= 4 ; j++)
  {
      if (j==0 || j == i || i == 4)
             printf(" *");
      else   print ("  ");
  }
     printf("\n");
}
```

```
i = 0,   j = 0
i = 1,   j = 0, 1
i = 2,   j = 0, 2
i = 3,   j = 0 , 3
i = 4,   j = 0,1,2,3,4
```

## Switch case:-

```
switch(        ) → int/Char
{
    case 1 :
                break;
    cal 2 :
                break;
    case 3 :
                break;
    default :
                break;
}
```

(1) Default get executed when you don't have matching case.

(2) we can't use continue in switch case

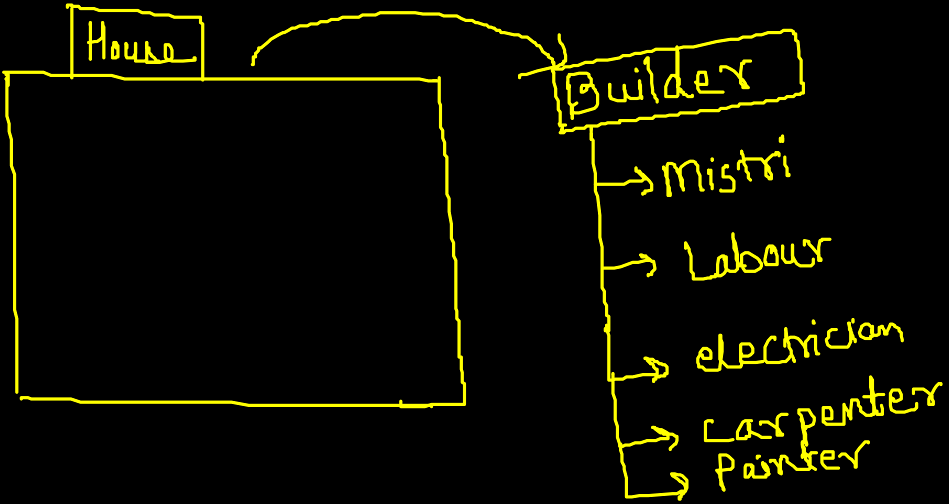
Goto Label:

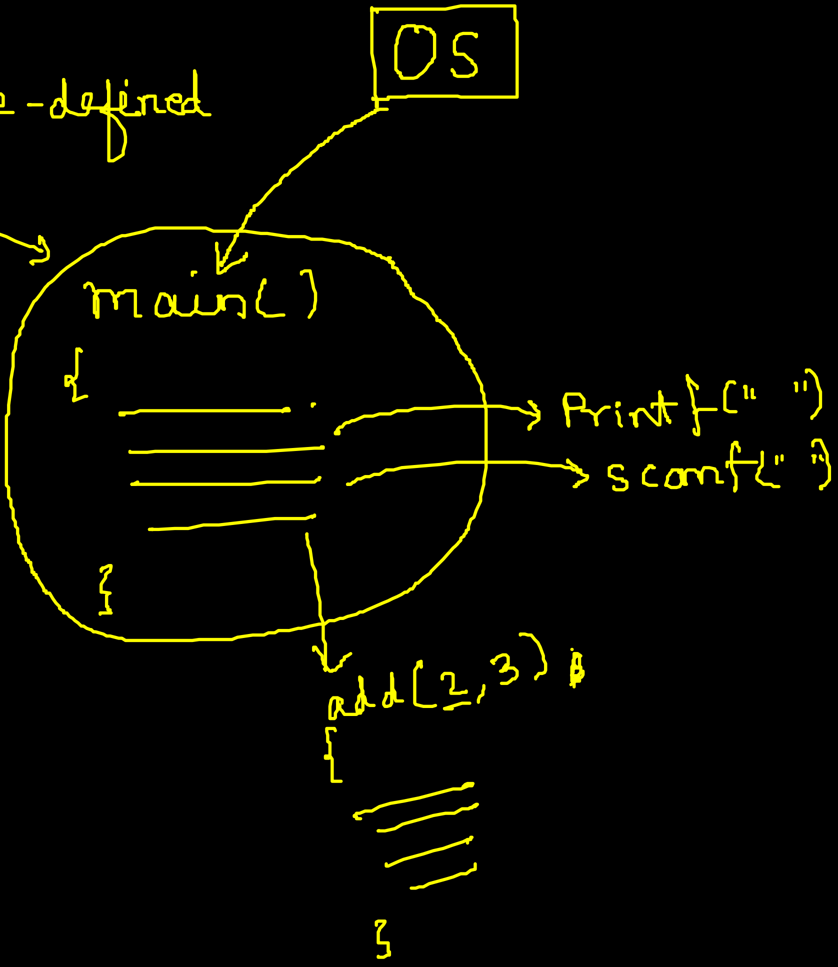
```
int main()
{
    L:  printf("Hello");

    goto L;
}
```

# What is Function

1) User defined    2) Pre-defined

```
int main()
{
    printf(      );  ──> Pre-defined
    scanf(       );  ──>

}
```



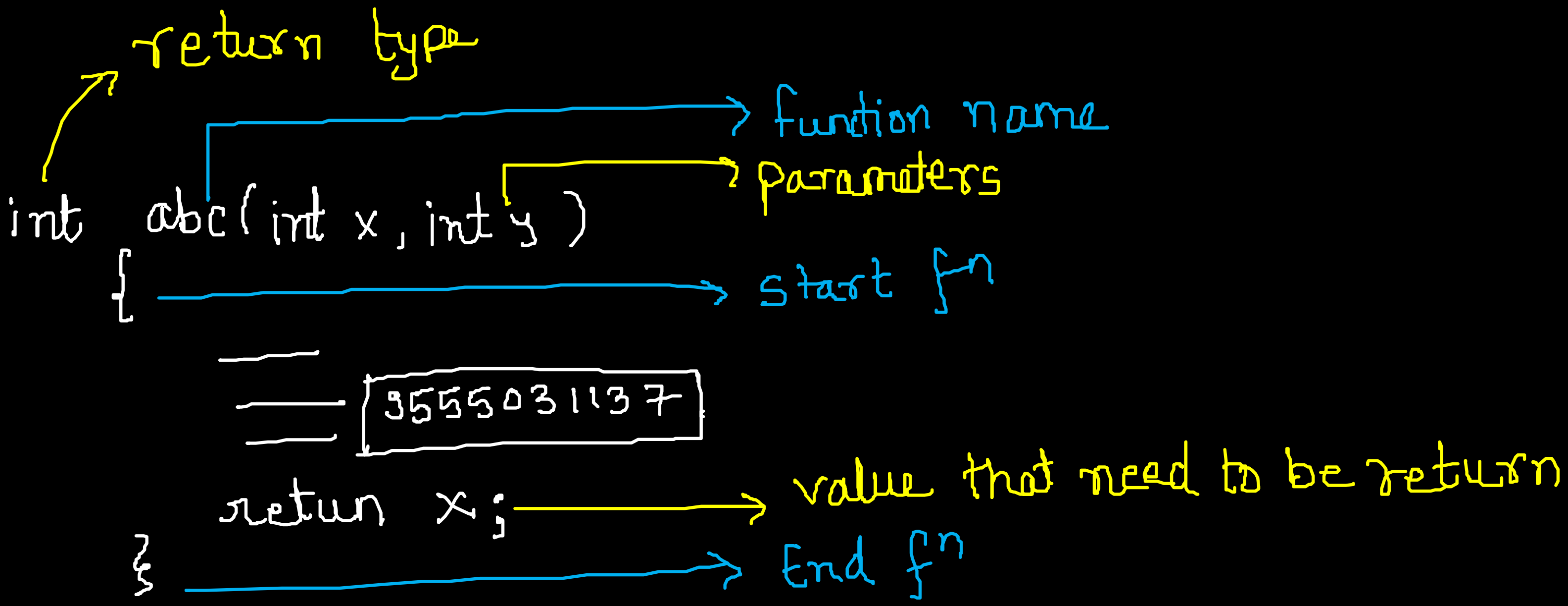


```c
int main()
{
 int a,b,c;
 printf("Enter a no");
 Scanf("%d %d", &a,&b);
  c = a+b;
  printf("Addition = %d", c);
  return 0;
}
```

1) Creation $\longrightarrow$ main()

2) Use/call $\longrightarrow$ printf/scanf

[main is a special userdefined $f^n$
$\because$ its name is predefined.]



```
int  abc( int x, int y )
{
    ____
    ____  9555031137
    ____
    retun x;
}
```

```
fun( )
{

}
```

```
void add( )
  {
     int a, b;
     printf("Enter 2 no");
     scanf("%d %d", &a,&b);
     printf("Addition =%d", a+b);
  }
```

void add( ) → function create / function defination

```
int main()
{
   add( );

   return 0;
}
```

int main() → TNRS

add( ); → function call

TNRN

## Function Declaration and Defination

1) function declaration $\longrightarrow$ int add( int x);

2) function call $\longrightarrow$ add( 2);

3) funtion defination $\longrightarrow$ int add( int x)

```
{
    ________
    ________
    ________
    ________
}
```

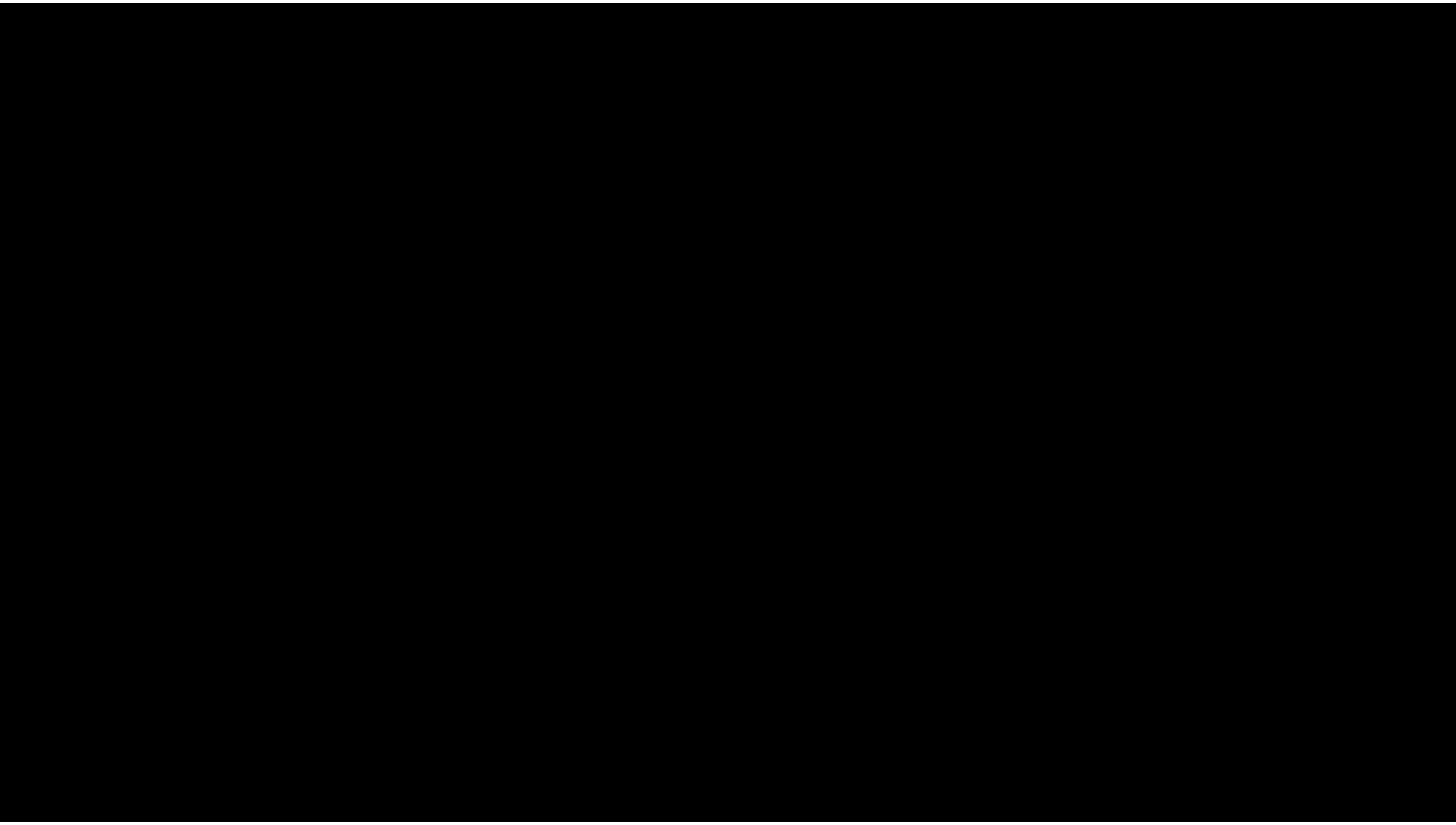
## Types of function

1) Take nothing & Return Nothing. $\longrightarrow$ void add( );

2) Take nothing & Return Something. $\rightarrow$ int add( );

3) Take something & Return Nothing. $\longrightarrow$ void add(int);

4) Take Something & Return Something. $\longrightarrow$ int add(int);



## Types of variable

## Advantages of Function