5/02/2023

Functions

It has a well defined task. Ex → only print, find a ^ b etc.

It is a program which is linked with a well defined task.

Suppose that we want to print our name 5 times.

```
for (int i=0; i<5; i++) {
        cout << "Bhavya";
}
```

This can be achieved by the above code. Now suppose I want to do the above task multiple times & then we will copy paste the same code again & again which will make our code lengthy and bulky.

Now if there is a mistake in that particular snippet, then we have to rectify it again & again for the same code snippet.

There is no readability as just to print the name, we are writing so much lengthy codes.

To solve all the above issues, function plays a major role.

Creating a function

```
Void printName () {

    for (int i=0; i<5; i++) {
            cout << "Bhavya";
    }

}
```

Now in main(), we can call this

function.

```
int main () {
        printName () ;   → Function call / invoke
}
```

Syntax of creating a function

```
                          ┌ Should be logical
return_type   function_name (          ) {
    // Function body                ↳ input parameters
}
```

→ If function does not return anything, then void can be used.

→ return_type can be any data type

→ function_name should be logical names

→ Suppose we want to find v from eq^n $v^2 = u^2 + 2as$, hence u, a & s will be the i/p parameters.

→ Logic of the function will come inside the function body.

Note → There is possibility that there are no input parameters.

```
                 ┌ function name
Ex →  int   main ( )     { // Function body }
       ↓          ↓
return type    no i/p parameters
```

```
int main () {           }
                        }  We need to follow this
    return 0 ;          }  practice.
}                       }
```

return 0 means that the main function has

been successfully executed.

0 → success in main().

Note → We can easily fix the bug by modifying the function. We can reuse the same code again and again with the help of function.

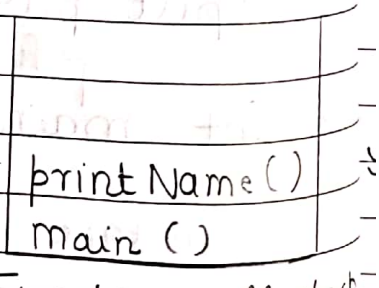
→ Practical (actually happening in memory)

Function call stack

Stack is a kind of data structure i.e data is stored in a specific way. Ex → Plates piled up, books piled up etc. The plate ke pt last will be picked up first & hence this is the concept of LIFO i.e last in first out.

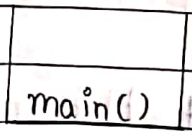
Function call stack tells that which function calls have gone, which function has called other function, local variables of function and what the function will return.

```
main() {

    printName();

}
```

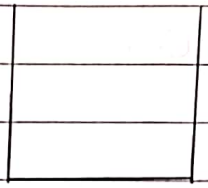


Function call stack

This is done after printName() has successfully been executed



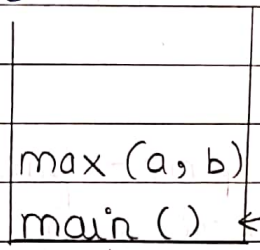
After the main() is fully executed, it returns 0 & gets out of the function call stack.



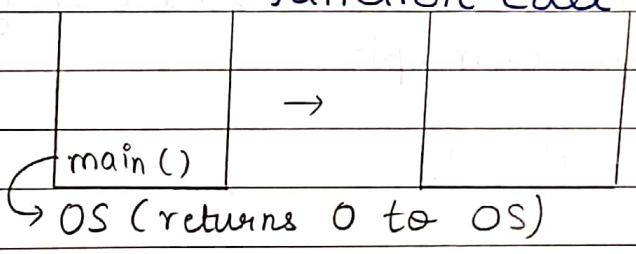
Function call stack.

Ex→
```
main () {
    int a = 1 ;
    int b = 2 ;
    cout << max (a, b) ;
    return 0 ;
}
```

```
int max (int a, int b) {
    if (a > b)
        return a ;
    else
        return b ;
}
```



2 max function returns 2 to main() & gets out of function call stack.



Note→ main() returns 0 to operating system as OS only starts the program.

Ques! Function to add two numbers.

```
int sum (int a, int b) {
    return (a+b) ;
}
```

a & b are i/p parameters

100
5 a

different address ← 600
5 a

Note →
```
main() {
    int a = 5;
    print(a);
    return 0;
}

void print(int a) {
    cout << a;
}
```

a of main() & print() is different i.e having different address. A copy of i/p parameter in print function is created. This concept is known as pass by value. Also variable a will finish as soon as the print() function is over.

void print(int a) { } → This is the syntax of pass by value.

Any changes done to a inside print function won't be reflected in the original a of main() function. This is valid only for pass by value concept.

↳ (copy creation)

How to see address of any variable?

int a = 5;

cout << &a; → Prints address of a

↳ This operator is known as address of operator.

We will be getting a hexadecimal value which is the address of variable a.

Note → main() {
add(a, b); → This gives an error as add is defined below it.
}
int add(int a, int b);

Soln to this is

int add(int a, int b); → declaration
main() { } → function call
int add(int a, int b) { } → definition

We have to make sure that atleast we have declared the function before invoking the function.

Ques 2 Find max of 3 numbers.

```
int max(int a, int b, int c) {

    if (a>b && a>c) { return a; }
    else if (b>a && b>c) { return b; }
    else { return c; }

}
```

Ques 3 Counting from 1 to n.

```
void printCounting(int n) {

    for (int i = 1; i <= n; i++) {
        cout << i << " ";
    }
}
```

Here we used void as we don't have to return anything.

Ques4 Function of student & grade problem

marks >= 90 → A
marks >= 80 → B
marks >= 70 → C
marks >= 60 → D
marks < 60 → E

Note → void print (int n); print (n);
↳ Parameter ↳ Argume

```
char grade (int marks) {

    if (marks >= 90) {return 'A';}
    else if (marks >= 80) { return 'B';
    else if (marks >= 70) {return 'C';}
    else if (marks >= 60) {return 'D';}
    else { return 'E'; }

}
```

This code can be written more neatly with the help of switch statement.

```
switch (marks/10) {
    case 10:
    case 9:
        ;

}
```

Ques5 Sum of even number upto n.

```
int printEven (int n) {   int sum = 0;
    for (int i = 1; i <= n ; i++) {
        if (i % 2 == 0)
            sum = sum + i;
    }
    return sum;
}
```

Try to avoid % operator as it is heavy operation. Use bitwise AND operator instead.

Alternative

```
int printEven (int n) {
        int sum = 0;
        for (int i = 2; i <= n ; i = i + 2) {
            sum = sum + i;
        }
        return sum;
}
```