Anjuman — I — Islam's

M.H SABOO SIDDIK COLLEGE OF ENGINEERING

8, Saboo Siddik Polytechnic Rd., Byculla, Mumbai — 400 008.

Department of Basic Science and Humanity

Module-03: Cheat Sheet – Functions & Storage Classes

Subject: C Programming **Sem: I**

3.1 Function

1. Introduction of Function

- A function is a block of code that performs a specific task.
- It helps in code reusability, modularity, and maintainability.

2. Main Function

```
int main() {
   // program starts executing here
   return 0;
}
```

- Every C program must have a main() function.
- Execution starts from main().

3. Defining a Function

```
return_type function_name(parameters) {
   // body of the function
   return value;
}
```

✅ **Example:**

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

4. Accessing (Calling) a Function

```
int result = add(5, 3);
printf("%d", result);
```

5. Function Prototype

- Declaration of function before its use.
- Tells the compiler about the function name, return type, and parameters.

```
int add(int, int); // function prototype
```

6. Passing Arguments to a Function

- Call by Value: Copy of variable is passed (changes don't affect original).
- Call by Reference: Address is passed (changes affect original).

```
void swap(int *x, int *y) {
   int temp = *x;
   *x = *y;
   *y = temp;
}
```

7. Recursion

- A function calling itself to solve a smaller sub-problem.

```
int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n * factorial(n - 1);
}
```

➞ **Used in problems like factorial, Fibonacci, and tree traversal.**

◇ 3.2 Storage Classes

Storage Class	Scope	Lifetime	Default Value	Storage	Keyword Example
auto	Local (inside function)	Till function ends	Garbage	Memory	auto int x = 10;
register	Local	Till function ends	Garbage	CPU Register (if available)	register int i;
static	Local/Global	Till program ends	0	Memory	static int count = 0;
extern	Global	Till program ends	0	Memory	extern int a;

Usage Tips

- auto is default for local variables.
- register is used for frequently used variables (like loop counters).
- static retains value between function calls.
- extern is used to access a global variable defined in another file.

Types of Functions

In C programming, functions can be classified into **four types** based on whether they **take arguments** and/or **return a value**.

🔹 1️⃣ Function with Arguments but No Return Value

📘 **Concept:**

Function takes input parameters but doesn't return any value to the calling function.

✅ **Syntax:**

```
void function_name(data_type parameter1, data_type parameter2) {
    // body of the function
}
```

✅ **Example:**

```
#include <stdio.h>
void add(int a, int b) {
    printf("Sum = %d", a + b);
}

int main() {
    add(5, 10); // passing arguments
    return 0;
}
```

🧩 **Output:**

Sum = 15

🔹 2️⃣ **Function with Arguments and Return Value**

📘 **Concept:**

Function takes parameters and returns a value back to the calling function.

✅ **Syntax:**

```
return_type function_name(data_type parameter1, data_type parameter2) {
   // body of the function
   return value;
}
```

✅ **Example:**

```
#include <stdio.h>
int add(int a, int b) {
   return a + b;
}

int main() {
   int result = add(10, 20);
   printf("Sum = %d", result);
   return 0;
}
```

🧩 **Output:**

Sum = 30

🔹 3️⃣ **Function without Arguments and without Return Value**

📘 **Concept:**

Function doesn't take any input or return any output. It performs a task internally.

✅ **Syntax:**

```
void function_name(void) {
   // body of the function
}
```

✅ **Example:**

```
#include <stdio.h>
void message() {
   printf("Hello, World!");
}

int main() {
   message();
   return 0;
}
```

🧩 **Output:**

Hello, World!

🔹 4️⃣ Function without Arguments but with Return Value

📘 **Concept:**

Function doesn't take any parameters but returns a value to the calling function.

✅ **Syntax:**

```
return_type function_name(void) {
   // body of the function
   return value;
}
```

✅ **Example:**

```
#include <stdio.h>
int getNumber() {
   int num = 25;
   return num;
}

int main() {
   int n = getNumber();
   printf("Number = %d", n);
   return 0;
}
```

🧩 **Output:**

```
Number = 25
```

🧩 **Quick Summary Table**

Type	Takes Arguments	Returns Value	Example Function
1. With arguments, no return	✅ Yes	❌ No	void add(int a, int b)
2. With arguments, with return	✅ Yes	✅ Yes	int add(int a, int b)
3. Without arguments, no return	❌ No	❌ No	void message()
4. Without arguments, with return	❌ No	✅ Yes	int getNumber()

🧠 C Programming Practice – Functions & Storage Classes

Q1. Define and call a function to calculate the area of a circle.

✅ **Problem Statement:**

Write a program to define a function area() that takes radius as argument and prints the area of a circle.

✅ **Code:**

```
#include <stdio.h>
#define PI 3.14

void area(float r) {      // function definition
   float result = PI * r * r;
   printf("Area of circle = %.2f", result);
}

int main() {
   float radius = 5.0;
   area(radius);          // function call
   return 0;
}
```

✅ **Output:**

```
Area of circle = 78.50
```

Q2. Function with argument and return value – find maximum of two numbers.

✅ **Problem Statement:**

Write a program using a function max() that accepts two integers and returns the greater one.

✅ **Code:**

```
#include <stdio.h>

int max(int a, int b) {      // function with return value
   if (a > b)
      return a;
   else
      return b;
}

int main() {
   int x = 15, y = 25;
   int result = max(x, y);
   printf("Maximum = %d", result);
   return 0;
}
```

✅ **Output:**

```
Maximum = 25
```

Q3. Function without argument and without return value – print greeting message.

✅ **Problem Statement:**

Create a function greet() that displays “Welcome to C Programming!”.

✅ **Code:**

```
#include <stdio.h>

void greet() {              // no argument, no return
   printf("Welcome to C Programming!");
}

int main() {
   greet();
   return 0;
}
```

✅ **Output:**

```
Welcome to C Programming!
```

Q4. Function with recursion – find factorial of a number.

✅ **Problem Statement:**

Write a recursive function fact() to calculate the factorial of a number.

✅ **Code:**

```
#include <stdio.h>

int fact(int n) {
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1);
}

int main() {
   int num = 5;
   printf("Factorial = %d", fact(num));
   return 0;
}
```

✅ **Output:**

```
Factorial = 120
```

Q5. Demonstrate auto storage class.

✅ **Problem Statement:**

Write a program to show the default behavior of auto variable.

✅ **Code:**

```
#include <stdio.h>

void display() {
    auto int x = 10;       // auto variable
    printf("Value of x = %d\n", x);
    x++;
}

int main() {
    display();
    display();
    display();
    return 0;
}
```

✅ **Output:**

```
Value of x = 10
Value of x = 10
Value of x = 10
```

🧩 (Each time, x is re-created when function runs again.)

Q6. Demonstrate static storage class.

✅ **Problem Statement:**

Write a program to show how a static variable retains its value between function calls.

✅ **Code:**

```
#include <stdio.h>

void counter() {
    static int count = 1;    // static variable
    printf("Count = %d\n", count);
    count++;
}

int main() {
    counter();
    counter();
    counter();
    return 0;
}
```

✅ **Output:**

```
Count = 1
Count = 2
Count = 3
```

🧩 (Static variable retains its value between function calls.)

Q7. Demonstrate extern storage class.

✅ **Problem Statement:**

Write a program to use an external global variable inside a function.

✅ **Code:**

```
#include <stdio.h>

int num = 100;          // global variable

void show() {
   extern int num;     // using external variable
   printf("Number = %d", num);
}

int main() {
   show();
   return 0;
}
```

✅ **Output:**

Number = 100

Q8. Demonstrate register storage class.

✅ **Problem Statement:**

Write a program using a register variable for faster access.

✅ **Code:**

```
#include <stdio.h>

int main() {
   register int i;
   for (i = 1; i <= 5; i++) {
      printf("%d ", i);
   }
   return 0;
}
```

✅ **Output:**

1 2 3 4 5

🧩 (Variable i may be stored in CPU register for faster access.)