# CC-112L

# Programming Fundamentals

## Laboratory 10

### Introduction to Programming, Algorithms and C

### (PRE-LAB-10 CONTENT)

**Version: 1.0.0**

**Release Date: 09-05-2025**

**Department of Information Technology University of the Punjab**

**Lahore, Pakistan**

## Pre Lab Contents:

- Learning Objectives
- Required Resources
- General Instructions

## Background and Overview

- Functions

## Learning Objectives:

### Arrays – II:

*C programs/tasks to practice/test the following concepts:*

i. Searching Arrays
ii. Sorting Arrays
iii. Multidimensional Arrays
iv. Variable Length Arrays

### Strings and Formatted I/O:

*C programs/tasks to practice/test the following concepts:*

i. Fundamentals of Strings and Characters
ii. Character Handling Library
iii. String Conversion Functions
iv. Standard Input/Output Library Functions
v. String Manipulation Functions
vi. Comparison Functions
vii. Streams
viii. Formatted Output with printf
ix. printf Format Flags
x. Printing Literals and Escape Sequences
xi. Formatted Input with scanf

**Resources Required**:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022
- Code editor (Code::Blocks, Dev C++, VS Code)

# General Instructions

- Write and run each example in the lab.
- Modify examples to test different inputs and edge cases.
- Check array bounds carefully.
- Always #include required header files (<stdio.h>, <ctype.h>, <string.h>, <stdlib.h>).

| Teachers | | |
|---|---|---|
| **Course Instructor** | **Hafiz Anzar** | anzar@pucit.edu.pk |
| **Teacher Assistants** | **Rimsha Majeed** | bitf22m029@pucit.edu.pk |
| | **Maheen Fatima** | bitf22m031@pucit.edu.pk |
| | **Momna Muzaffar** | bcsf22m021@pucit.edu.pk |
| | **Zainab Mehmood** | bcsf22m038@pucit.edu.pk |
| | **Khadija tul Kubra** | bitf22m025@pucit.edu.pk |
| | **Inam ul Haq** | bitf22m017@pucit.edu.pk |
| | **Saqib Ali Javaid** | bcsf22m016@pucit.edu.pk |
| | **Hafiz Subhan** | bcsf22m043@pucit.edu.pk |
| | **Muhammad Saad** | bitf23m003@pucit.edu.pk |

# Lab Topics

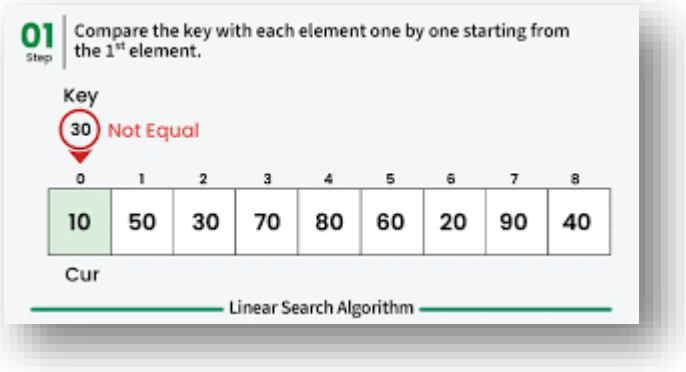
## Searching Arrays

**Definition:** Searching means finding whether a specific value (called a “key”) exists in an array and locating its position.

**Linear Search:**

- It’s the simplest and most universal search method.
- Works on both sorted and unsorted arrays.
- Useful when the dataset is small or when you have no control over the data order.

**Example :** Check each element one by one (works for unsorted arrays). i.e: You have a list of roll numbers: 101, 102, 103, 104, 105.You want to check if roll number 103 is in the list. You look at eachitem one by one until you find 103.




```c
#include <stdio.h>
int main() {
    int arr[] = {10, 20, 30, 20, 50} ,  key = 20, found = 0;
    int size = sizeof(arr) / sizeof(arr[0]);
    for(int i = 0; i < size; i++) {
        if(arr[i] == key) {
            printf("Element %d found at index %d\n", key, i);
            found = 1;
        }
    }
    if(!found)
        printf("Element %d not found\n", key);
    return 0;
}
```

**Binary Search:** Only for sorted arrays, divides search space in half each time, much faster.

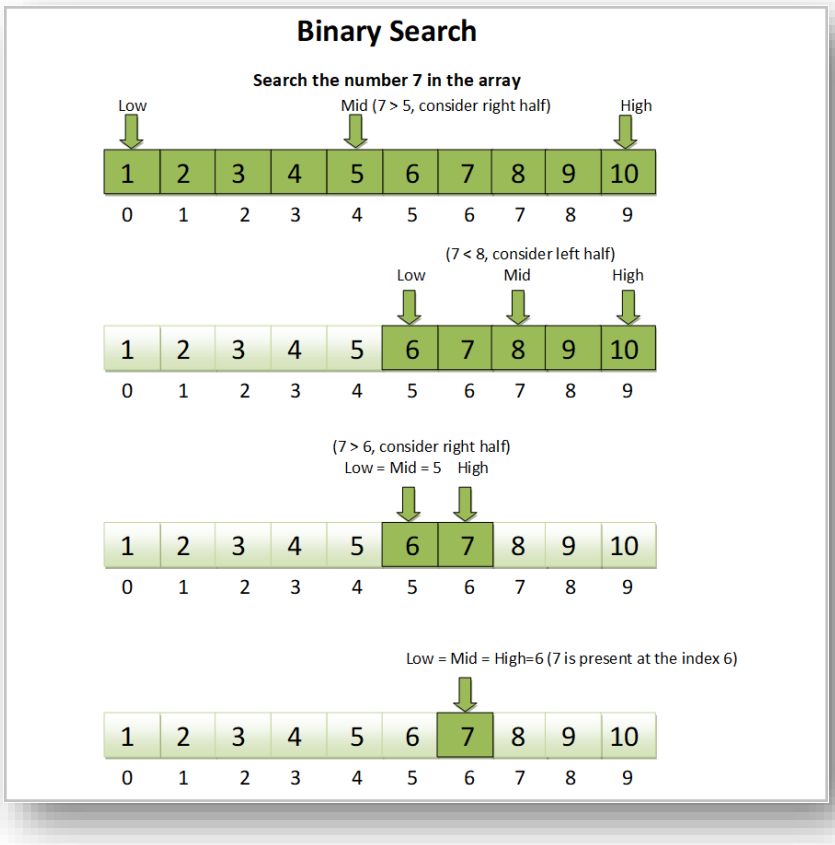
- Much faster than linear search for **large, sorted arrays**.
- Reduces the search space by half at every step.
- Ideal for applications where search operations happen frequently on sorted data.

**Example (Binary Search):**

In a sorted list of marks: 50, 60, 70, 80, 90, you want to find 70.

You first check the middle (70). If it's the number you want, you stop.

If you searched for 60, you would move to the left half (50, 60) and continue.




**For Linear Search:** https://www.geeksforgeeks.org/linear-search/

**For Binary Search:** https://www.geeksforgeeks.org/binary-search/

```c
#include <stdio.h>
int binarySearch(int arr[], int size, int key) {
    int low = 0, high = size -1;
    while(low <= high) {
        int mid = (low + high) / 2;
        if(arr[mid] == key) return mid;
        else if(arr[mid] < key) low = mid +1;
        else high = mid -1;
    }
    return -1;
}
int main() {
    int arr[] = {10,20,30,40,50}, key = 30;
    int size = sizeof(arr)/sizeof(arr[0]);
    int index = binarySearch(arr, size, key);
    if(index != -1)
        printf("Element %d found at index %d\n", key, index);
    else
        printf("Element %d not found\n", key);
    return 0;
}
```

## Sorting Arrays

**Definition:** Sorting means arranging the elements of an array in a particular order (usually ascending or descending).

- **Bubble Sort:** Swap adjacent elements if out of order.
- **Selection Sort:** Find the minimum element and put it in the right place.
- **Insertion Sort:** Insert elements into the correct position step by step.

### Bubble Sort

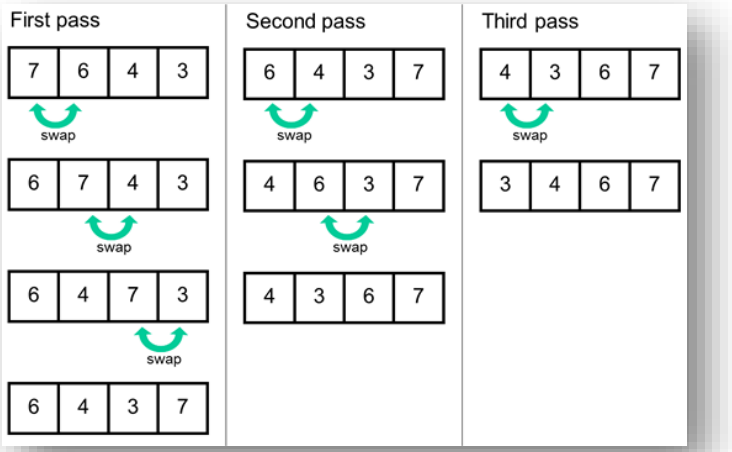
**Why we use it:**

- It's simple and easy to understand.
- Good for **learning sorting fundamentals** and small datasets.
- Useful when the array is almost sorted (it can finish early in some implementations).

**Example:** You have cards numbered 5, 3, 2, 4, 1.

You compare each neighboring pair and swap them if they are out of order.

You repeat this process until all cards are arranged as 1, 2, 3, 4, 5.




```c
#include <stdio.h>
void bubbleSort(int arr[], int n) {
    for(int i =0; i< n-1; i++)
        for(int j =0; j< n-i-1; j++)
            if(arr[j]> arr[j+1]) {
                int temp = arr[j];
                arr[j]= arr[j+1];
                arr[j+1]= temp;
            }
}
int main() {
    int arr[] = {5,1,4,2,8}, n=5;
    bubbleSort(arr,n);
    printf("Sorted array: ");
    for(int i=0; i<n; i++) printf("%d ", arr[i]);
    return 0;
}
```

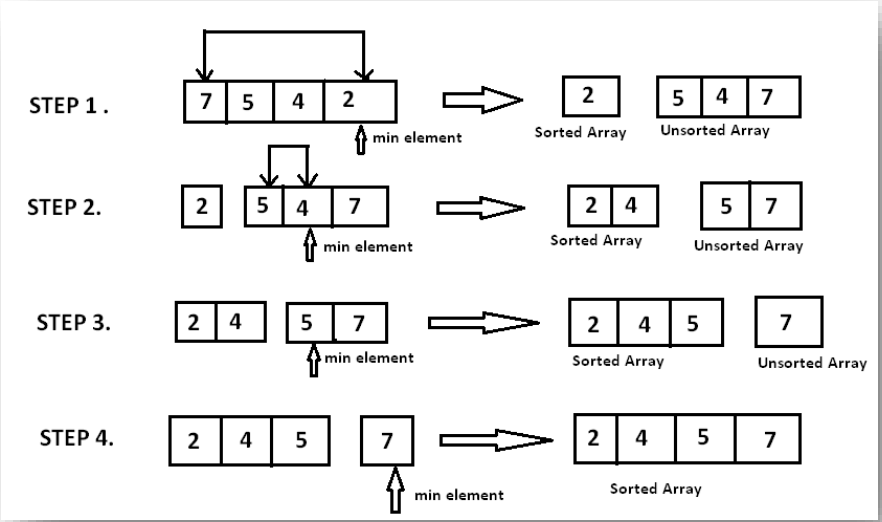
**For Detail on** : https://www.geeksforgeeks.org/bubble-sort-algorithm

## Selection Sort examples

From the same set of unsorted elements, you find the smallest one and place it at the start. Then you find the next smallest and place it in the second position, and so on.

**Why we use it:**

- It's simple and predictable.
- Requires **minimum number of swaps** compared to bubble sort.
- Useful when swap operations are costly, but comparisons are cheap.




```c
#include <stdio.h>
void selectionSort(int arr[], int n) {
    for(int i=0;i<n-1;i++) {
        int min_idx=i;
        for(int j=i+1;j<n;j++)
            {
                if(arr[j]<arr[min_idx])
                {  min_idx=j; }
            }
        int temp=arr[min_idx]; arr[min_idx]=arr[i]; arr[i]=temp;
    }
}
int main() {
    int arr[]={29,10,14,37,13},n=5;
    selectionSort(arr,n);
    printf("Sorted array: ");
    for(int i=0;i<n;i++) printf("%d ",arr[i]);
    return 0;
}
```

For detail on Selection sort : https://www.geeksforgeeks.org/selection-sort-algorithm-2/

## Variable Length Arrays (VLAs)

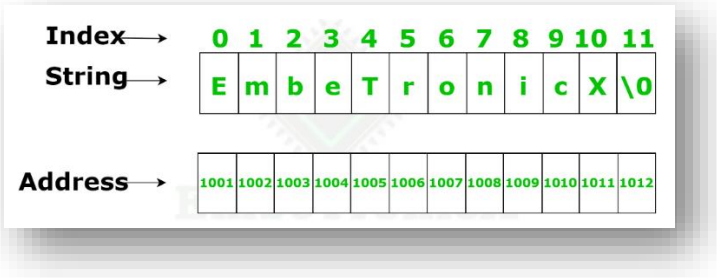
A variable length array is array whose size is determined at runtime(instead of at compile time).

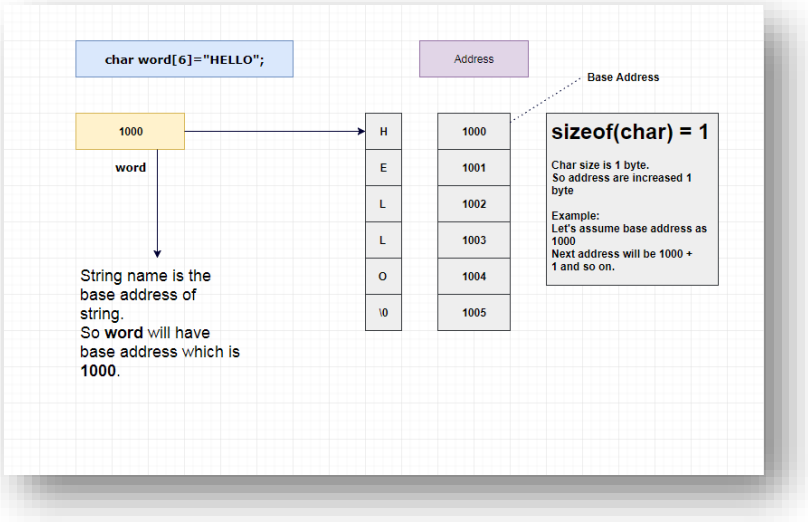
Array size decided at runtime. **Syntax:** int arr[n]; after reading n. You can pass VLAs to functions.

**Why we use it:**

- Allows dynamic memory use without manually managing memory like with `malloc`.
- Lets you handle situations where array size is unknown at compile time.

**Example:** You ask the user: "How many students are in your class?"
They answer: "10." Your program creates an array for 10 students, and if next time they answer "20," it creates an array for 20 students.




**For Detail on Variable length Array:**

https://www.youtube.com/watch?app=desktop&v=JW3Vg0xpJLY

**Example: Declare, fill, print 1D & 2D VLA**

```c
#include <stdio.h>
void print1D(int n, int arr[n]) {
    for(int i=0;i<n;i++){ printf("%d ",arr[i]); }
    printf("\n");
}
void print2D(int r, int c, int arr[r][c]) {
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++) printf("%d ",arr[i][j]);
        printf("\n");
    }
}
int main() {
    int n,r,c;
    printf("Enter size of 1D array: "); scanf("%d",&n);
    int arr1[n];
    for(int i=0;i<n;i++) { arr1[i]=i+1; }
    printf("1D VLA: "); print1D(n,arr1);

    printf("Enter rows and cols for 2D array: ");
    scanf("%d%d",&r,&c);
    int arr2[r][c];
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            arr2[i][j]=i+j;
    printf("2D VLA:\n"); print2D(r,c,arr2);
    return 0;
}
```

# String

**Why we use them:**

- Strings are essential for handling **text data** names, messages, commands,file names etc.
- They let you store and process sequences of characters efficiently.
- Almost every program uses strings for input, output, or internal operations.

## Fundamentals of Strings and Characters

- **Definition:**

  A **string** is an array of characters ending with a null character \0. A **character** is a single letter, digit, or symbol..
- Can be accessed with char[] or char*.

**Example: String input, display, and character iteration**

```c
#include <stdio.h>
int main() {
    char str[50];
    printf("Enter a string: ");
    scanf("%s", str);  // stops at space
    printf("String: %s\n", str);
    printf("Characters: ");
    for(int i=0;str[i]!='\0';i++)
        printf("%c ", str[i]);
    return 0;
}
```

## Character Handling Library *(<ctype.h>)*

**Definition:** Provides functions to check and transform characters, like:

- isalpha() → check if a letter
- isdigit() → check if a digit
- toupper() → convert to uppercase
- tolower() → convert to lowercase

**Example: Count alphabets, digits, convert to uppercase**

```c
#include <stdio.h>
#include <ctype.h>
int main() {
    char str[]="He110World!";
    int letters=0,digits=0;
    for(int i=0;str[i]!='\0';i++)
    {
        if(isalpha(str[i])) { letters++;  }
        if(isdigit(str[i])) { digits++; }
        str[i]=toupper(str[i]);
    }
    printf("Uppercase: %s\nLetters: %d Digits: %d\n",
        str, letters, digits);
    return 0;
}

```

## String Conversion Functions *(<stdlib.h>)*

**Definition:** Functions to convert strings to numbers:

- atoi() → string to int
- atof() → string to float
- strol() → string to long integer

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    char s1[]="1234", s2[]="3.14";
    int x = atoi(s1);
    double y = atof(s2);
    printf("Integer: %d, Float: %.2f\n", x, y);
    return 0;
}
```

## Standard Input/Output Functions *(<stdio.h>)*

**Definition:** Functions for input and output, such as:

- fgets() → read string
- getchar() → read character
- putchar() → print character

**Example: Read string with spaces, read a character**

```c
#include <stdio.h>
int main() {
    char line[100];
    char ch;
    printf("Enter line: ");
    fgets(line, sizeof(line), stdin);
    printf("You entered: %s", line);
    printf("Enter one character: ");
    ch=getchar();
    printf("You entered: ");
    putchar(ch);
    return 0;
}
```

## String Manipulation Functions (<string.h>)

**Definition:** Functions to manipulate strings:

- strcpy() → copy
- strcat() → concatenate
- strlen() → length
- strcmp() → compare
- strcspn() → to remove new line character

**Example: Copy, concatenate, length**

```c
#include <stdio.h>
#include <string.h>
int main() {
    char s1[20]="Hello", s2[]="World";
    strcpy(s1,"Hi");  // s1 = "Hi"
    strcat(s1," "); strcat(s1,s2);  // s1 = "Hi World"
    printf("Combined: %s (Length: %lu)\n", s1, strlen(s1));
    return 0;
}
```

## String Comparison Functions

**Functions:** strcmp(), strncmp()

```c
#include <stdio.h>
#include <string.h>
int main() {
    char a[]="apple", b[]="apricot";
    if(strncmp(a,b,3)==0)
        printf("First 3 letters match\n");
    else
        printf("First 3 letters don't match\n");
    return 0;
}
```

## Streams, Formatted I/O, Escape Sequences

**Example: Print integers, floats, characters, and use escape sequences**

```c
#include <stdio.h>
int main() {
    int x=10; float y=3.14;
    printf("Integer: %d, Hex: %x, Float: %.2f\n", x, x, y);
    printf("Line1\nLine2\nTab\tSpace\nQuote: \"Hello\"\n");
    return 0;
}
```

## Some Practice Questions with solution

**Palindrome Check**

```c
#include <stdio.h>
#include <string.h>
int main()
{
    char str[100];
    int i, len, flag = 1;
    printf("Enter string: ");
    scanf("%s", str);
    len = strlen(str);
    for (i = 0; i < len / 2; i++)
    {
        if (str[i] != str[len - i - 1])
        {
            flag = 0;
            break;
        }
    }
    if (flag)
        printf("Palindrome\n");
    else
        printf("Not a palindrome\n");
    return 0;
}
```

**Character Count**

```c
#include <stdio.h>
int main()
{
    char str[100];
    int count = 0;
    printf("Enter string: ");
    scanf("%s", str);
    for (int i = 0; str[i] != '\0'; i++)
    {
        count++;
    }
    printf("Character count: %d\n", count);
    return 0;
}
```

**Removal of a Character**

```c
#include <stdio.h>
int main()
{
    char str[100], ch;
    int i, j;
    printf("Enter string: ");
    scanf("%s", str);
    printf("Enter character to remove: ");
    scanf(" %c", &ch);
    for (i = 0, j = 0; str[i] != '\0'; i++)
        if (str[i] != ch)
        {
            str[j++] = str[i];
        }
    str[j] = '\0';
    printf("After removal: %s\n", str);
    return 0;
}
```

**Replacement of a Character**

```c
#include <stdio.h>
int main()
{
    char str[100], oldch, newch;
    printf("Enter string: ");
    scanf("%s", str);
    printf("Enter character to replace: ");
    scanf(" %c", &oldch);
    printf("Enter new character: ");
    scanf(" %c", &newch);
    for (int i = 0; str[i] != '\0'; i++)
        if (str[i] == oldch)
        {
            str[i] = newch;
        }
    printf("After replacement: %s\n", str);
    return 0;
}
```

# Pre-Lab-10 Tasks (Marks 40)

## Task-01 (Marks 5)

Write a C program to find the length of a string without using the strlen() function. The program should use a loop to count the number of characters until the null-terminator (\0) is encountered.

**Sample Output:** *Enter a string:* Hello, world!

**Length of the string:** 13

## Task-02 (Marks 10)

Write a C program that concatenates two user-input strings in two ways:

1. Using the built-in *strcat()* function from the C standard library.

2. Manually, without using any built-in concatenation functions (by using a loop).

**Sample output:**

*Enter first string: Hello*

*Enter second string: World*

**Using strcat():** HelloWorld

**Manual concatenation:** HelloWorld

## Task-03 (Marks 5)

Write a C program that takes a string as input from the user and removes all spaces from it. The program should display the string without any whitespace characters.

**Sample Output:**

*Enter a string: Hello World from C*

**String without spaces:** HelloWorldfromC

## Task-04 (Marks 5)

Write a C program that counts the number of vowels and consonants in a given string entered by the user. The program should ignore non-alphabetic characters (e.g., spaces, digits, punctuation) and consider both uppercase and lowercase letters.

**Sample Output:**

Enter a string: Hello World!

**Number of vowels:** 3

**Number of consonants:** 7

## Task-05 (Marks 15)

Write a C program to sort an array of integers in ascending order using the **Bubble Sort** algorithm. The program should:

1. Take the number of elements and the array elements as input from the user.

2. Sort the array in **ascending order** using the Bubble Sort technique.

3. Display the sorted array after sorting.

**Sample Output:**

Enter number of elements: 5

Enter elements: 34 7 23 32 5

**Sorted array: 5 7 23 32 34**

---

***Best of Luck!***

---