Unit 6
Strings & Input/Output In C

STRINGS

INTRODUCTION TO STRINGS

In C, a string is a sequence of characters terminated with a null character ('\0'). It is essentially a one-dimensional array of characters.

```
char name[] = "Ram"; // Automatically adds '\0' at the end
```

Here, the memory allocation looks like:

| 'P' | 'r' | 'i' | 'n' | 'c' | 'e' | '\0' |

- Strings are not a built-in data type in C.
- They are manipulated using character arrays and pointers.
- Stored in contiguous memory locations.
- Null-terminated: This is how the compiler knows where the string ends.

DECLARING AND INITIALIZING STRINGS

```
// Declaration with size
char name[10] = "Ajmer";
```

```
// Declaration without size (compiler counts characters + 1 for '\0')
char name[] = "Ajmer";
```

```
// Declaration using character array
char name[] = {'A','j','m','e','r','\0'};
```

INPUT AND OUTPUT OF STRINGS

Using *scanf*() and *printf*()

```
char name[20];
scanf("%s", name);   // Takes input till whitespace
printf("%s", name);
```

Problem

- Cannot read strings with spaces.

Solution

- Use *gets*() and *puts*() *(Note: gets() is unsafe and deprecated in modern compilers)*

```
char name[100];
gets(name);     // Reads full line including spaces
puts(name);
```

Better alternative: *fgets*()

```
fgets(name, sizeof(name), stdin); // Safer input
```

STRING LIBRARY FUNCTIONS (IN $<string.h>$)

Function	Description
strlen()	Returns length of string
strcpy()	Copies one string to another
strcat()	Concatenates two strings
strcmp()	Compares two strings
strrev()	Reverses a string
strlwr()	Converts string to lowercase
strupr()	Converts string to uppercase
strchr()	Finds first occurrence of a character
strstr()	Finds first occurrence of a substring

Example

```
#include <stdio.h>
#include <string.h>

int main() {
  char s1[20] = "Hello";
  char s2[20] = "World";

  strcat(s1, s2); // s1 becomes "HelloWorld"
  printf("%s\n", s1);

  return 0;
}
```

STRING MANIPULATION FUNCTIONS

• **Copying Strings**	`strcpy(dest, src);`
• **Comparing Strings**	`strcmp(s1, s2);`
• **Concatenating Strings**	`strcat(s1, s2);`
• **Finding Length**	`strlen(s);`
• **Finding a Character**	`strchr(s, 'a');`
• **Finding Substring**	`strstr(s1, s2);`

CONVERSION BETWEEN STRINGS AND NUMBERS

Function	Purpose
atoi()	Convert string to integer
atof()	Convert string to float
atol()	Convert string to long
itoa()	Convert integer to string (non-standard)
sprintf()	Formats string like printf

```
char str[10] = "1234";
int num = atoi(str); // num = 1234
```

FILE INPUT/OUTPUT

WHAT IS A FILE?

A file is a storage area on disk used to store data permanently, unlike variables which lose their values after program termination.

FILE OPERATIONS IN C

Operation	Function
Create a file	fopen()
Read from file	fscanf(), fgets()
Write to file	fprintf(), fputs()
Close a file	fclose()

FILE OPENING MODES

Mode	Description
"r"	Open for reading. File must exist.
"w"	Open for writing. Creates new file.
"a"	Append to file. Creates if doesn't exist.
"r+"	Read and write. File must exist.
"w+"	Create file for read and write.
"a+"	Read and append.

FILE POINTERS

```
FILE *fp;
fp = fopen("file.txt","r");
```

WRITING TO A FILE

```
FILE *fp;
fp = fopen("myfile.txt","w");
fprintf(fp,"Hello File!");
fclose(fp);
```

READING FROM A FILE

```
FILE *fp;
char ch;
fp = fopen("myfile.txt","r");
while ((ch = fgetc(fp)) != EOF) {
printf("%c",ch);
}
fclose(fp);
```

READING/WRITING STRINGS

```
fputs("Hello", fp);
fgets(str, size, fp);
```

FILE STATUS AND ERROR HANDLING

Function	Description
feof()	Checks end of file
ferror()	Checks for error during file operation
perror()	Prints error message
rewind()	Sets file pointer to beginning

BINARY FILES

Binary files store data in raw format (0s and 1s). Use:

- *fread*() to read
- *fwrite*() to write

```
fwrite(&var, sizeof(var), 1, fp);
fread(&var, sizeof(var), 1, fp);
```

File Positioning Functions

Function	Purpose
ftell(fp)	Returns current position of file pointer
fseek(fp, offset, origin)	Moves pointer to specific location
rewind(fp)	Sets pointer to beginning

```
fseek(fp, 0, SEEK_SET); // Beginning of file
fseek(fp, 0, SEEK_END); // End of file
```

SAMPLE PROGRAM: Copy Contents of One File to Another

```
#include <stdio.h>

int main() {
  FILE *f1, *f2;
  char ch;

  f1 = fopen("source.txt", "r");
  f2 = fopen("destination.txt", "w");

  while ((ch = fgetc(f1)) != EOF) {
    fputc(ch, f2);
  }

  fclose(f1);
  fclose(f2);

  printf("File copied successfully!");
  return 0;
}
```