week 3_3:

strings

rajin teaches programming

INTRODUCTION TO STRINGS:

Strings are **arrays** of **characters** in C programming that represent sequences of characters. They are used to **store** and **manipulate textual** data such as **names**, **sentences**, and **more.** Understanding strings is **crucial** for developing applications that involve **text processing.**

– DECLARATION AND INITIALIZATION:

Strings in C are represented as **arrays** of **characters terminated** by a null character `'\0'`, which indicates the end of the string.

STRINGS

```
char stringName[size]; //Declaration
char stringName[] = “Hello”; //Initialization
```

– THE INPUT PROBLEM:

There are **many**, many ways to take an **input** of **strings** in C, and generally, the **typical** **scanf()** function is considered “**unsafe**” due to the potentiality of **memory leaks.** Thus, **alternative methods** have been introduced to take an **input** of a string. **Any** one is applicable, but even **still**, **scanf()** is the **easiest** and most **preferred** way of taking string inputs. Normally, it cannot take strings with spaces, but we will look at a solution later on.

INPUT METHOD

```
char str[100];
printf("Enter a string: ");
scanf("%s", str);
```

input using scanf() and %s (cannot take strings with spaces)

INPUT METHOD

```c
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
```

input using fgets() (can take strings with spaces)

INPUT METHOD

```c
char str[100];
printf("Enter a string: ");
gets(str);
```

input using gets() (not recommended for memory vulnerabilities)

INPUT METHOD

```c
char str[100];
printf("Enter a string: ");
scanf("%[^\n]", str);
```

input using scanf() (can take strings with spaces because of \n)

– ACCESSING STRING ELEMENTS:

Individual characters of a **string** can be **accessed** using the **array index** notation.

we can leave the size of the array unspecified if we "hard"-code the string.

Developer -

```c
#include<stdio.h>

int main() {

    char str[] = "Clark";

    char firstCharacter = str[0];

    printf("%c\n", firstCharacter);

    return 0;
}
```

the firs character is accessed, in this case, "C".

– STRING OUTPUT:

Unlike taking input, **printing strings** are a lot more **simple.** They can be printed to the **standard output** using the `printf()` **function** with the `%s` **format specifier.**

STRING MANIPULATION FUNCTIONS:

C provides a set of **library functions** for performing various **operations** on **strings,** such as **copying, concatenating** (joining), **comparing,** and **searching.**

function name	purpose
strcpy(destination, source)	Copies the string pointed to by `source` into the array pointed to by `destination`.
strcat(destination, source)	Concatenates the string pointed to by `source` onto the end of the string pointed to by `destination`.
strlen(string)	Returns the length of the string.
strcmp(string1, string2)	Compares two strings and returns an integer less than, equal to, or greater than zero if `string1` is found, respectively, to be less than, equal to, or greater than `string2`.

Examples of using each function will be shown in class and discussed further in depth.

CONCLUSION:

Strings are **essential** in C programming for **handling textual data.** Understanding **string manipulation functions** and **techniques** is vital for developing robust applications**.** Further practice will be done to improve your programming skills and ensure you are comfortable with both strings and arrays**.**

next class 4_1:

recursion

rajin teaches programming

reach out for classes at: rajin.khan2001@gmail.com