Chapter 8 - Strings

A String is a 1-D character array terminated by a null character ('\0')
A null character is used to denote the termination of a string characters are stored in a contiguous memory locations.

Initializing Strings

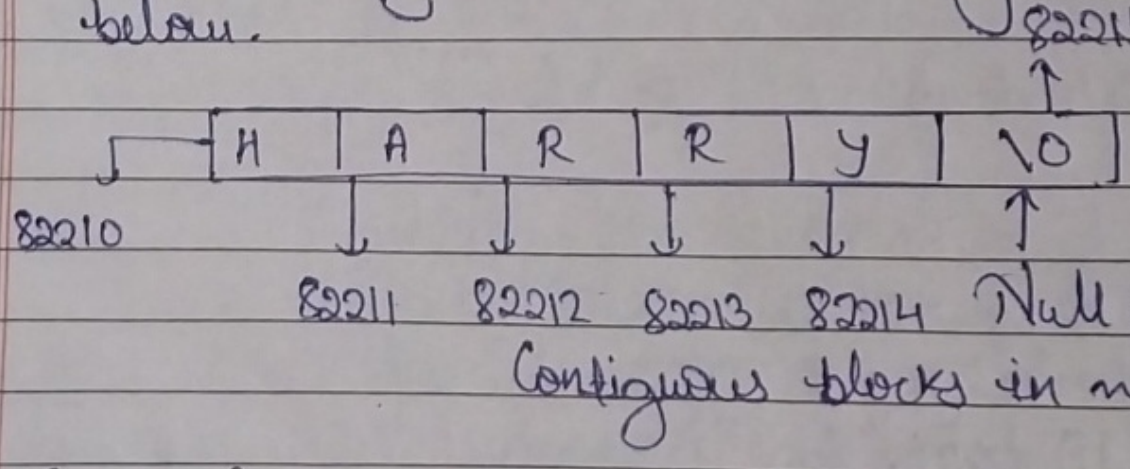
Since Strings is an array of characters it can be initialized as follows:

```
char s[] = {'H', 'A', 'R', 'R', 'Y', '\0'};
```

There is another shortcut for initializing String in C language.

Strings in Memory

A string is stored just like an array in the memory as shown below.



Quick Quiz

Create a String using double quotes and print its content using a loop

```
#include <stdio.h>
int main(){
    char name[] = {'J', 'o', 'h', 'n', '\0'};
    char *ptr = &name[0];

    for(int i=0; i<4; i++){
        printf("%c", *(ptr+i));
    }
    printf("\n");
    for(int i=0; i<4; i++){
        printf("%c", name[i]);
    }
    return 0;
}
```

Printing Strings

A string can be printed character by character using printf and %c

But there is another convenient way to print Strings in C

```
char st[] = "HARRY";
printf("%s", st);
```

Taking String input from the user

We can use %s with scanf to take string input from the user

```
char st[50];
scanf("%s", st);
```

scanf automatically adds a null character when the Enter key is pressed.

Note:-

(1) The string should be short enough to fit into the array.

(2) scanf cannot be used to input multi-word strings with spaces.

gets() and puts()

gets() is a function which can be used to receive a multi-word string

```
char st[30];
gets(st); // The Entered string is stored in st
```

multiple gets() calls will be needed for multiple strings.

Likewise, puts can be used to output a string

```
puts(st); // prints the string & places the cursor on the next line
```

Declaring A String using pointers

we can declare strings using pointers.

char *ptr = "Harry";

this tells the compiler to store the string in memory and assigned address is stored in char pointer

Note:-

(1) Once a string is defined using char st[] = "Harry" it cannot be reinitialized to something else

(2) A string defined using pointers can be reinitialized

ptr = "Rohan";

Standard library functions for strings

C provides a set of standard library functions for string manipulation.

Some of the most commonly used string functions are:-

STRLEN()

This function is used to count the number of character in the string including the null ('\0') characters.

int length = strlen(st);

these functions are declared under <string.h> header file

STRCPY()

This function is used to copy the content of second string into first string passed to it

```
char Source[] = "Harry";
char target[30];
strcpy(target, Source); // target now contains "Harry"
```

target string should have enough capacity to store the source string.

STRCAT()

This function is used to concatenate two strings

```
char S1[12] = "Hellow";
char S2[] = "Harry";

Strcat(S1, S2); // S1 now contains "Hellow Harry"
                <no space in between>
```

STRCMP()

This function is used to compare two strings. It returns 0 if the strings are equal a negative value if the first string's mismatching character's ASCII value is less than the second.

String's corresponding mismatching character, and a positive value otherwise

```
Strcmp("jar", "joke");
Strcmp("joke", "jar");
```

Chapter 8 - Practice Set

Q13 which of the following is used to appropriately read a multi-word string

1. gets()
2. fputs()
3. fprintf()
4. scanf()

3 gets()

Q29 write a program to take string as an input from the user %c and %s confirm that the strings are equal

```
#include <stdio.h>
int main(){
    char str1[15];
    char str2[15];
    scanf("%s", str);
    printf("you Entered: %s\n", str);
    printf("Enter a string: ");
    for (int i=0; i<15; i++){
        scanf("%c", &str[i]);
        fflush(stdin); // clear the input Buffer
}
str[15] = '\0'; // Ensure null termination
```

```
printf("you Entered: %s\n", str);
return 0;
}
```

Q33. write your own version of strlen function from <string.h>

```
#include <stdio.h>
int main(){
char str[] = "HARRY";
int i=0, count;
char c = str[i];
while (c != '\0'){
    c = str[i];
    i++;
}
count = i-1;
printf("%d", count);
return 0;
}
```

Q4 write a function slice() to slice a string. It should change the original string such that it is now the sliced string. take 'm' and 'n' as the start and ending position for slice.

```
#include <stdio.h>
char* slice(char str[], int m, int n)
    int i=0, count;
```

```
char *ptr1 = &Str[m];
char *ptr2 = &Str[n];

Str = ptr1;
str[n] = '\0';
return Str;
}
int main(){
    char str[] = "Harry bhai";
    printf("%s", slice(str, 1, 7));
    return 0;
}
```

Q5) write your own version of strcpy function from <string.h>

```
#include <stdio.h>
int main(){
    char str1[100], str2[100];
    printf("Enter first string: ");
    fgets(str1, sizeof(str1), stdin);

    for(int i=0; str[i] != 'p'; i++){
        str2[i] = str1[i];
        str2[i+1] = '\0';
    }
    printf("the second string is: %s\n", str2);
    return 0;
}
```

5) write a program to Encrypt a string by adding 1 to ascii value of its characters

3

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

int main(){
    char str[] = "Mera sara paisa takiye
                  ke neeche block poly nee h
for (int i=0; i< strlen(str); i++)
    {
        str[i] = str[i] +1;
    }
    printf("%s", str);
    return 0;
    }
```

Q 7 write a program to decrypt the string Encrypted using Encrypt function ( ) in problem 6.

y

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

int main(){
    char str[] = "- - - Encrypted code - -
    for (int i=0; i< strlen(str); i++)
    {
        str[i] = str[i]-1;
    }
    printf("%s", str);
    return 0;
    }
```

8) write a program to count the occurrence of a given character in a string.

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

int main(){
    char c = 'r';
    int count = 0;
    char str[] = "Harry";
    for (int i=0; i<strlen(str); i++)
    {
        if (str[i] == c){
            count++;
        }
    }
    printf("%d", count);
    return 0;
}
```

9) write a program to check whether a given character is present in a string or not.

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

int main(){
    char c = 'd';
    int contains = 0;
    char str[] = "Harry";
    for (int i=0; i<5; i++)
    {
        printf("This is a nice character\n");
    }
```

```
for (int i=0; i < strlen(str); i++)
{
    if (str[i] == c){
        contains = 1;
        break; // this break statement will exit the loop once the character is found!
    }
}

if (contains) {
    printf("yes it contains!\n");
}
else {
    printf("Doesnot contain!\n");
}

return 0;
}
```