Strings in C

PRESENTED BY AIML 2ND SEM STUDENTS

RAJYALAKSHMI 1RV22AI041
SAFIYA FARHEEN 1RV22AI048

ANANTH ATHREYA 1RV22AI007
ADITYA TEKRIWAL 1RV22AI003

Table of contents

01

Introduction

A string is a collection of characters which terminate with a null character \0.

Size of a string = no. of elements + 1

Eg: char A[5]= “Apple” ; □
      char B[6]= “Apple” ; ✔

A	p	p	l	e	\0

.   char C[10]= “Apple” ; ✔

A	p	p	l	e	\0	\0	\0	\0	\0

End of string

End of character array

Declaration & Initialization

Declaration- introducing a variable & its datatype. **Initialization**- assign initial value to the declared variable

Initializing a string in C :

1) **As a character array**

Syntax : char str[size] = “string” ;

```
Eg: char s1[]   = “Hello” ;
     char s2[6]= “Hello” ;
```

- *Can be modified*
- *Points only to 1st index*

2) **As a pointer to a string literal**

Syntax : char*p = “string” ;

```
Eg: char* s3 = “Goodbye”
```

- *Cannot be modified*
- *Can point to any index*

1) **A list of characters**

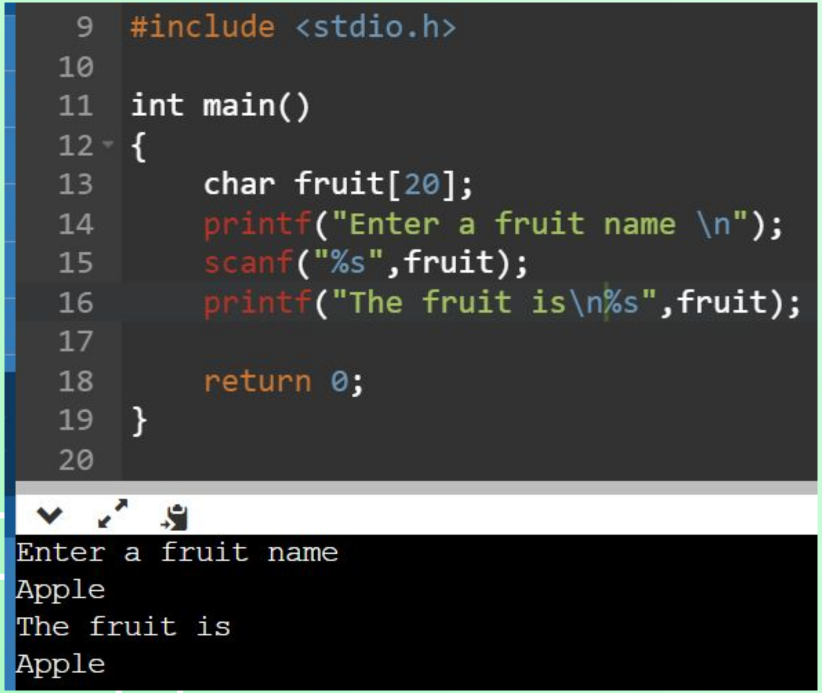
```
Eg: char s4[] ={ ‘a’, ‘b’, ‘c’, ‘d’, ‘\0’}
```

2) **A list of strings**

```
Eg: char* s5={“cherry” , “kiwi” , “mango”}
```

Reading strings from terminal

Scanf() with %s

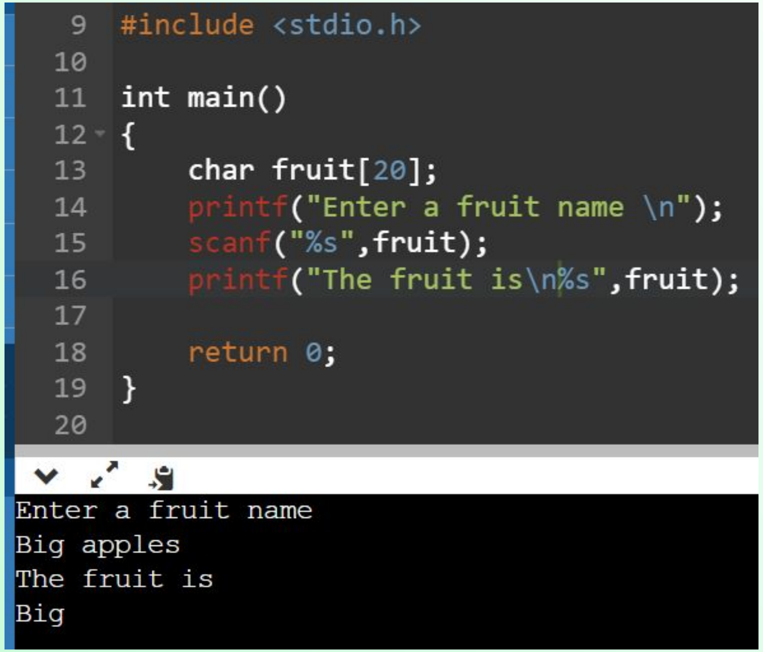


```c
#include <stdio.h>

int main()
{
    char fruit[20];
    printf("Enter a fruit name \n");
    scanf("%s",fruit);
    printf("The fruit is\n%s",fruit);

    return 0;
}
```

```
Enter a fruit name
Apple
The fruit is
Apple
```



```c
#include <stdio.h>

int main()
{
    char fruit[20];
    printf("Enter a fruit name \n");
    scanf("%s",fruit);
    printf("The fruit is\n%s",fruit);

    return 0;
}
```

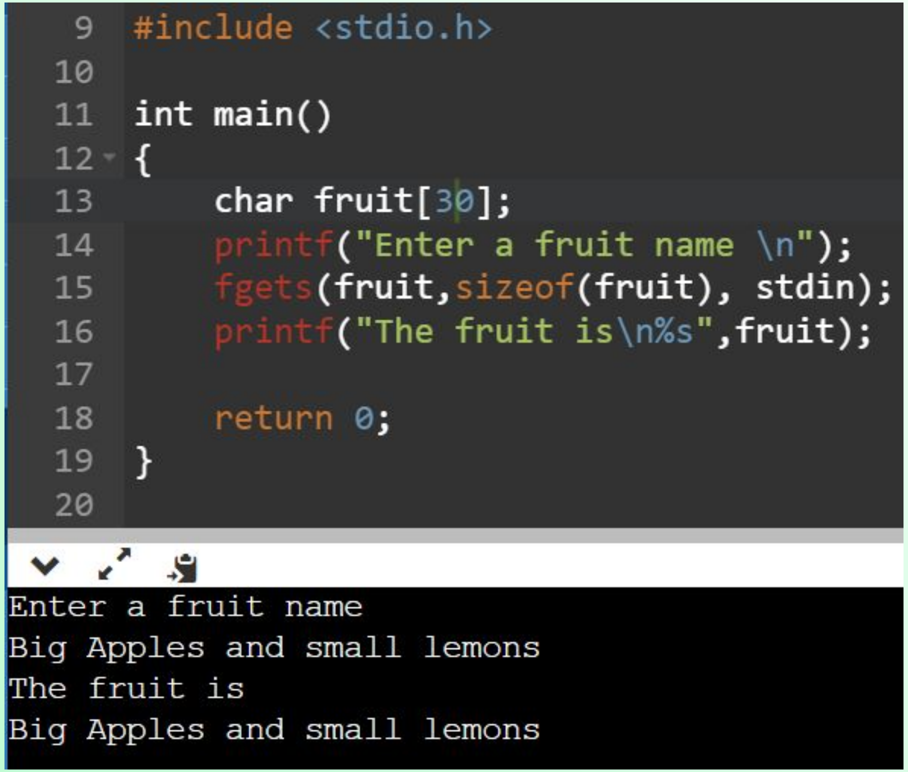
```
Enter a fruit name
Big apples
The fruit is
Big
```

scanf() reads the string input until it encounters a blank space

Reading strings from terminal

fgets() with %s

It reads the string including blank spaces



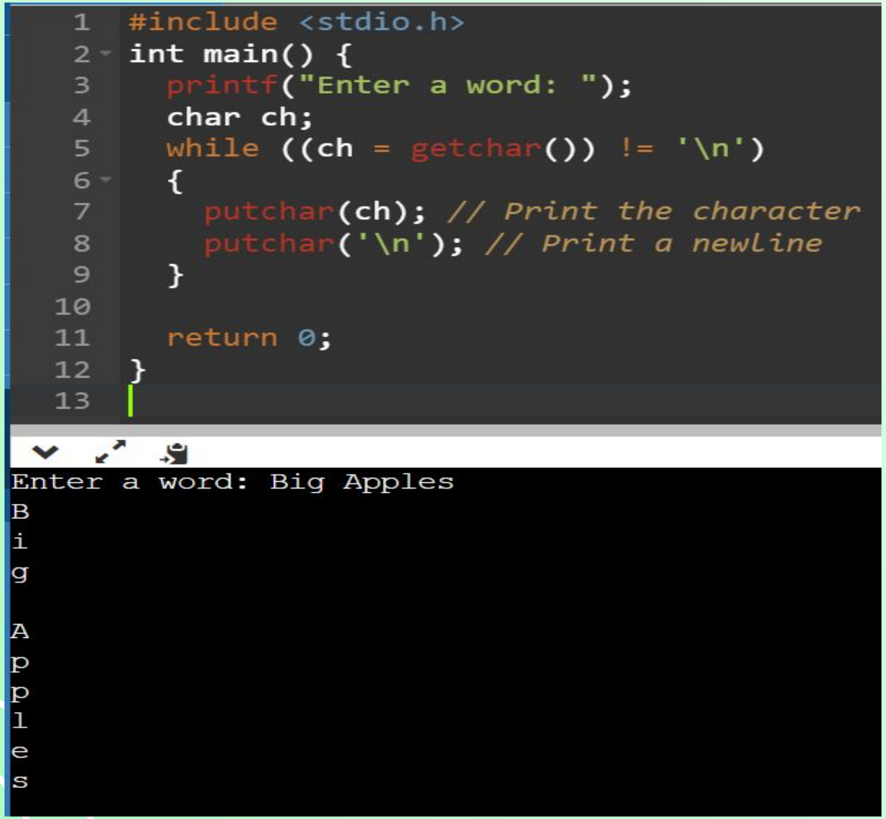
```c
#include <stdio.h>

int main()
{
    char fruit[30];
    printf("Enter a fruit name \n");
    fgets(fruit,sizeof(fruit), stdin);
    printf("The fruit is\n%s",fruit);

    return 0;
}
```

```
Enter a fruit name
Big Apples and small lemons
The fruit is
Big Apples and small lemons
```

Reading strings from terminal



```c
#include <stdio.h>
int main() {
  printf("Enter a word: ");
  char ch;
  while ((ch = getchar()) != '\n')
  {
    putchar(ch); // Print the character
    putchar('\n'); // Print a newline
  }

  return 0;
}
```

```
Enter a word: Big Apples
B
i
g

A
p
p
l
e
s
```

getchar()

It reads the string characters, one by one.

Writing Strings to screen

In C programming, you can use the printf function to write a string to the screen (standard output). The printf function is part of the standard C library and is used to format and print data to the console. Here's how you can use it to print a string:

```
#include <stdio.h>

int main() {
   char myString[] = "Hello, world!";  // Your string

   printf("%s\n", myString);  // Print the string followed by a newline character

   return 0;
}
```

We can also specify the precision with which the above array is displayed. For example:

```
printf("%15.5s",myString)
```

The above line indicates that the first five characters are to be printed in a field width of 15 columns.

Arithmetic operations on characters

In C programming, characters are actually represented as integers using their ASCII (or Unicode) values. This means that you can perform arithmetic operations on characters just like you would on integers.

ASCII value of : ‘A’ is 65
ASCII value of : ‘Z’ is 90
ASCII value of : ‘a’ is 97
ASCII value of : ‘z’ is 122

Possible ways of Manipulation:

Way1: Displays ASCII value

char x=’b’;

printf(“%d”,x); //Displays result as 98

Way2: Displays Character value

char x=’b’;

printf(“%c”,x); //Displays result as b

Way3: Displays next ASCII value

char x=’b’+2;

printf(“%d”,x); //Displays result as 100

Way4: Displays next Character value

char x=’b’+2;

printf(“%c”,x); //Displays result as d

Way5: Displays difference between 2 ASCII in integer

char x=’z’-’a’;

printf(“%d”,x); //Displays result as 25

Way6: Displays difference between 2 ASCII in char

char x=’z’-’a’;

printf(“%c”,x); //Displays result as z-a

Atoi Function:

The atoi function in C is used to convert a string (character array) representing an integer into its corresponding integer value.

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

int main() {
   char str1[] = "12345";
   char str2[] = "42abc";  // Non-numeric characters after digits

   int value1 = atoi(str1);
   int value2 = atoi(str2);

   printf("String 1: %s\n", str1);
   printf("Integer value 1: %d\n", value1);

   printf("String 2: %s\n", str2);
   printf("Integer value 2: %d\n", value2);

   return 0;
}
```

OUTPUT:

```
String 1: 12345
Integer value 1: 12345
String 2: 42abc
Integer value 2: 42
```

Significance:

1. Can convert any string of numbers into integer value that can perform the arithmetic operations like integers.
2.Header file- **stdlib.h**

Operation on strings

- Finding the length of a string
- Converting characters of a string into uppercase characters
- Converting characters of a string into lowercase characters
- Concatenating two strings to form a new string
- Appending a string to another string
- Comparing 2 strings
- Reversing a string

Finding the length of a string

- For every string, the ending character is ‘\0’.
- This fact can be used to find the length of a string.
- For finding the length of the string, we count the number of characters which are not ‘\0’.

Algorithm

```
Let str be a string and len be length of the string(to be found)
Step 1: Set i as 0
Step 2: While str[i] is not ‘\0’, repeat step 3
Step 3: set i as i+1
            [end of while]
Step 4: set len as i
Step 5: end
```

Program

```
#include<stdio.h>
int main()
{
        char str[100];
        int i,len;
        printf(“Enter a string”);
        gets(str);
        i=0;
        while(str[i] != ’\0’)
                i+=1;
        len=i;
        printf(“The length of the string is %d”,len);
        return 0;
}
```

Converting characters of a string into uppercase characters

- In the previous slides, we have seen that the ASCII values of A-Z ranges from 65 to 90, and a-z ranges from 97-122.
- While converting the uppercase letters to lowercase, we subtract 32 from its ascii value.

Algorithm

```
Let str be a string. resstr is the resultant uppercase string
Step 1: set i as 0
Step 2: while str[i] is not ‘\0’, repeat step 3
Step 3: if str[i]>=’a’ and str[i]<=’z’
            Set resstr[i]=str[i]-32
        Else
            Set resstr[i]=str[i]
            [end of if]
        Set i=i+1
            [enf of while]
Step 4: Set resstr[i] as ’\0’
Step 5: Exit
```

Code

```
#include<stdio.h>
int main()
{
	char str[100],resstr[100];
	int i;
	printf("Enter a string\n");
	gets(str);
	i=0;
	while(str[i]!='\0')
	{
		if(str[i]>='a'&&str[i]<='z')
			resstr[i]=str[i]-32;
		else
			resstr[i]=str[i];
		i++;
	}
	resstr[i]='\0';
	printf("The uppercase string is:");
	puts(resstr);
	return 0;
}
```

Converting characters of a string into lowercase characters

- While converting the uppercase letters to lowercase, we add 32 to its ascii value.

Algorithm

Let str be a string. resstr is the resultant uppercase string

```
Step 1: set i as 0
Step 2: while str[i] is not ‘\0’, repeat step 3
Step 3: if str[i]>=’A’ and str[i]<=’Z’
              Set resstr[i]=str[i]+32
            Else
              Set resstr[i]=str[i]]
              [end of if]
            Set i=i+1
              [end of while]
Step 4: Set resstr[i] as ’\0’
Step 5: Exit
```

Code

```
#include<stdio.h>
int main()
{
        char str[100],resstr[100];
        int i;
        printf("Enter a string\n");
        gets(str);
        i=0;
        while(str[i]!='\0')
        {
                if(str[i]>='A'&&str[i]<='Z')
                        resstr[i]=str[i]+32;
                else
                        resstr[i]=str[i];
                [End of if]
                i++;
        }
        resstr[i]='\0';
        printf("The uppercase string is:");
        puts(resstr);
        return 0;
}
```

Concatenating two strings to form a new string

- Consider 2 strings str1 and str2. newstr is the resultant string.
- First, we add the characters of str1 in order to newstr, then the characters of str2.

Algorithm

```
Step 1: Set i=0 and j=0
Step 2: while str1[i] is not ‘\0’, repeat step 3
Step 3: set newstr[j] as str1[i]
            Set i as i+1 and j as j+1
            [end of while]
Step 4: Set i as 0
Step 5: while str2[i] is not ‘\0’,repeat step 6
Step 6: set newstr[j] as str2[i]
            Set i as i+1 and j as j+1
            [end of while]
Step 7: set newstr[j] as ‘\0’
Step 8: Stop
```

Code

```
#include<stdio.h>
int main()
{
	char str1[100],str2[100],newstr[100];
	int i,j;
	printf("Enter a string\n");
	gets(str1);
	printf("Enter another string\n");
	gets(str2);
	i=0;j=0;
	while(str1[i]!='\0')
	{
		newstr[j]=str1[i];
		i++;j++;
	}
	i=0;
	while(str2[i]!='\0')
	{
		newstr[j]=str2[i];
		i++;j++;
	}
	newstr[j]='\0';
	printf("The concatenated string is:");
	puts(newstr);
return 0;
}
```

Appending a string to another string

- In this operation, we copy the contents of source_str at the end of the dest_str.

Algorithm

```
Step 1: Set i=0 and j=0
Step 2: while dest_str[i] is not ’\0’, repeat Step 3
Step 3: set i as i+1
            [end of while]
Step 4: while source_str[j] is not ‘\0’, repeat Step 5
Step 5: dest_str[i]=source_str[j]
            Set i as i+1 and j as j+1
            [end of while]
Step 6: Set dest_str[i] as ‘\0’
Step 7: End
```

Code

```
#include<stdio.h>
int main()
{
        char dest_str[100],source_str[100];
        int i,j;
        printf("Enter a string\n");
        gets(dest_str);
        printf("Enter another string\n");
        gets(source_str);
        i=0;j=0;
        while(dest_str[i]!='\0')
                i++;
        while(source_str[j]!='\0')
        {
                dest_str[i]=source_str[j];
                i++;j++;
        }
        dest_str[i]='\0';
        printf("The new string is:");
        puts(dest_str);
        return 0;
}
```

Comparing 2 strings

- 2 strings are said to be equal if all its characters at their indices are equal.

Algorithm

Let str1 and str2 be two strings. l1 is the length of str1 and l2 is the length of str2.

```
Step 1: if l1 not equal to l2,
                Write that strings are not equal
        Else
                For i from 0 to l1-1,repeat
                        If str1[i] is equal to str2[i]
                                continue
                        Else if str1[i]>str2[i]
                                Write that str1 is greater than str2
                                break
                        Else
                                Write that str2 is greater than str1
                                break
                        [End of if]
                [End of for]
        If i is l1
                Write that strings are equal
Step 2: Exit
```

Code

```
#include<stdio.h>
#include<string.h>
int main()
{
	char str1[100],str2[100];
	int i,l1,l2;
	printf("Enter string 1:");
	gets(str1);
	printf("Enter string 2:");
	gets(str2);
	l1=strlen(str1);
	l2=strlen(str2);
	if(l1!=l2)
		printf("The strings are not equal");
	else
	{
		for(i=0;i<l1;i++)
		{
			if(str1[i]==str2[i])
				continue;
			else if(str1[i]>str2[i])
			{
				printf("str1 is greater than str2");
				break;
			}
			else
			{
				printf("str2 is greater than str1");
				break;
			}
		}
		if(i==l1)
			printf("The strings are equal");
	}
}
```

Reversing a string-Algorithm

```
Consider a string str with length n. revstr is the reverse string
Step 1: Set i as 0 and j as n-1
Step 2: while i<n, repeat
                set revstr[i] as str[j]
                Set i as i+1 and j as j-1
            [end of for]
Step 3: set revstr[i] as ‘\0’
Step 4: Exit
```

Code

```
#include<stdio.h>
#include<string.h>
int main()
{
	char str[100],revstr[100];
	int i,j,len;
	printf("Enter a str:");
	gets(str);
	len=strlen(str);
	i=0;j=len-1;
	while(i<len)
	{
		revstr[i]=str[j];
		i++;j--;
	}
	revstr[i]='\0';
	printf("The reversed string is ");
	puts(revstr);
	return 0;
}
```