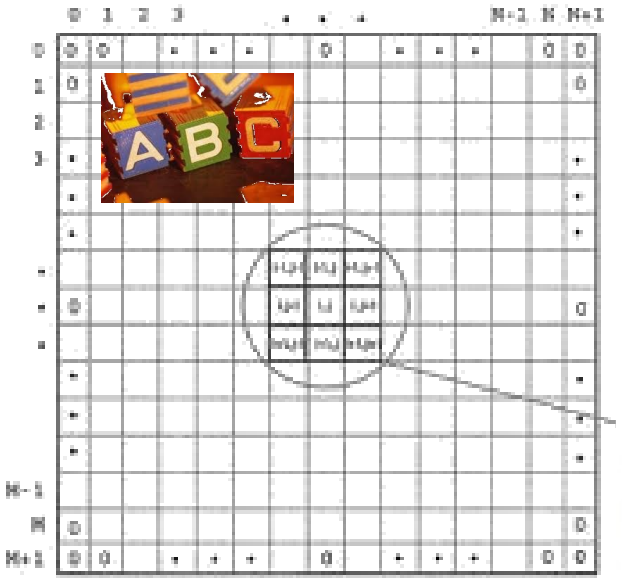


S11_1

CHARACTER ARRAYS
STRINGS

Objectives

To learn and appreciate the following concepts

- Strings definition, declaration, initialization
- Reading Strings
- Programs using strings

Session outcome

At the end of session student will be able to

- Declare and initialize strings
- Write programs using strings

Strings

Definition

- A string is an array of characters.
- Any group of characters (except double quote sign) defined between double quotation marks is a constant string.
- Character strings are often used to build meaningful and readable programs.

The common operations performed on strings are

✓ Reading and writing strings

✓ Combining strings together

✓ Copying one string to another

✓ Comparing strings to another

✓ Extracting a portion of a string ..etc.

Strings

Declaration and initialization

```
char string_name[size];
```

The size determines the number of characters in the string_name.

For example, consider the following array:

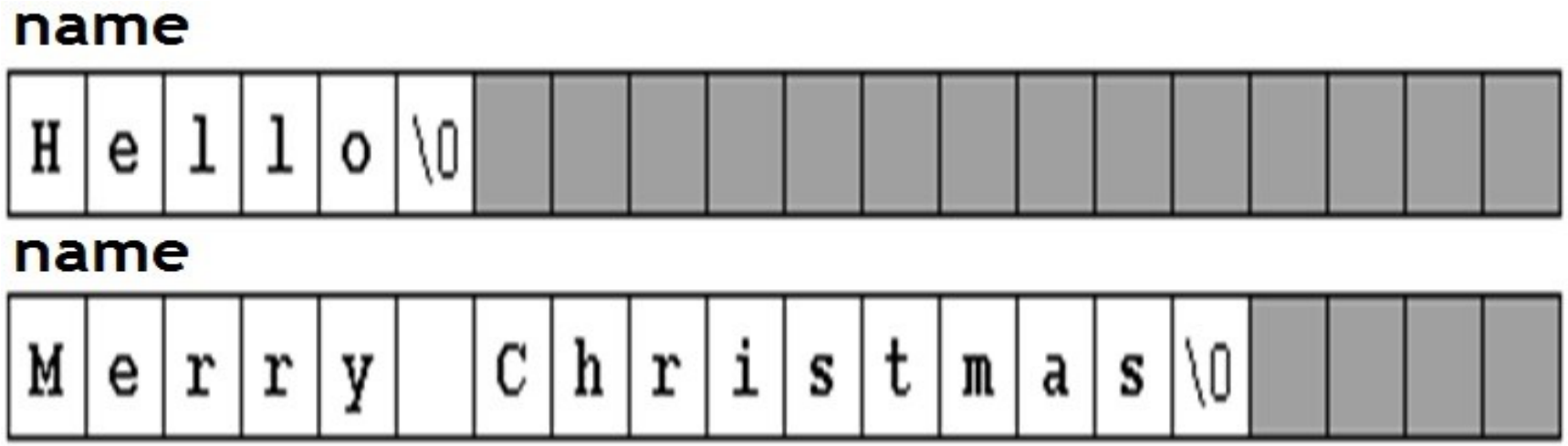
```
char name [20];
```

is an array that can store up to 20 elements of type char.

It can be represented as:

Strings

✓The character sequences "**Hello**" and "**Merry Christmas**" represented in an array *name* respectively are shown as follows :



Initialization of null-terminated character sequences

- **arrays of characters** or **strings** are ordinary arrays that follow the same rules of arrays.

For example

To initialize an array of characters with some predetermined sequence of characters one can initialize like any other array:

char myWord[ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };

Initialization of null-terminated character sequences

- Arrays of char elements have an additional methods to initialize their values: **using string literals**
- **“Manipal ”** is a constant string literal.

  For example,

  char result[14] =“Manipal”;
- **Double quoted** (") strings are literal constants whose type is in fact a null-terminated array of characters.

So string literals enclosed between double quotes always have a null character (**'\0'**) automatically appended at the end.

Initialization

- **Initialization**:

```
char myWord [ ] = { 'H', 'e', 'l', 'l', 'o', '\0' };

char myWord [ ] = "Hello";
```

- In both cases the array of characters myword is declared with a size of 6 elements of type char:

✓The 5 characters that compose the word "**Hello**" plus a final null character ('**\0**') which specifies the end of the sequence and that,

✓In the second case, when using double quotes (") null character ('**\0**') is appended automatically.

Example

```
#include<stdio.h>

int main()    {

   char greeting[]="Hello ";

   char yourname[80];

   printf("enter your name:");

   scanf("%s",yourname);

   printf("%s,%s",greeting, yourname);

  return 0;   }
```

```
enter your name:David
Hello ,David
Process returned 0 (0x0)   execution time : 24.636 s
Press any key to continue.
```

Example

```
Enter a string:
Manipal
Manipal
```

```
#include <stdio.h>
int main()
{
 const int MAX = 80;     //max characters in string
 char str[MAX];           //string variable str
 printf("Enter a string: \n");
 scanf("%s",str);    //put string in str
 printf("%s",str);   //display string from str
 return 0;
}
```

Reading Embedded Blanks

To read everything that you enter from the keyboard until the ENTER key is pressed (including space).

Syntax:

```
gets(string) ;
```

To write/display that you entered.

Syntax:

```
puts(stringname) ;
```

Example

```
#include <stdio.h>
int main()
{
    const int MAX = 80;    //max characters in string
    char  str[MAX];   //string variable str
    printf("\nEnter a string: ");
    gets(str);
    printf(" the string is \n");
    puts(str);
    return 0;
}
```

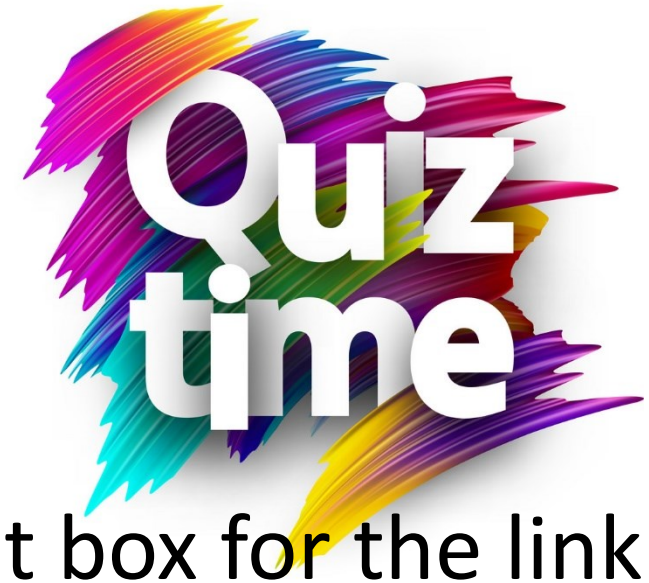
```
Enter a string: Technology
 the string is
Technology
```

The function will continue to accept characters until enter key is pressed.

Reading multiple lines: Example

```
#include <stdio.h>

int main() {

const int MAX = 2000;  //max characters in string

char str[MAX];  //string variable str

printf("\nEnter a string:\n");

scanf("%[^#]",str); //read characters to str until a # character is encountered

printf("You entered:\n");

printf("%s",str);

return 0;

}
```

The function will continue to accept characters until termination key (#) is pressed.



Go to posts/chat box for the link to the question

submit your solution in next 2 minutes

The session will resume in 3 minutes

Count the number of characters in a string

```
#include <stdio.h>
int main()
{
const int Max = 100;
char sent[Max];
int i=0, count=0;
printf("enter sentence \n");
gets(sent);
puts(sent);
```

```
while(sent[i]!='\0')
{
 count++;
 i++;
}
```

```
printf(" \n no of characters =  %d", count);
return 0;
}
```

```
enter sentence
Manipal
```

```
enter sentence
Manipal
Manipal

 no of characters = 7
```

Count the number of words in a sentence

```
#include <stdio.h>
int main()
{
const int MAX = 100;
char sent[MAX];
int i=0,count=1;
printf("enter sentence \n");
gets(sent);
printf("\n");
```

```
while(sent[i]!='\0')
{
 if ( ( sent[i] ==' ' ) && ( sent[i+1]!=' '  ) )
  count++;
  i++;
}
printf("\n no. of words =%d", count);
return 0;
}
```

```
enter sentence
Manipal Institute of Technology
```

```
enter sentence
Manipal Institute of Technology


 no. of words =4
```

Summary

- Strings definition, declaration, initialization
- Reading Strings
- Programs using strings