Programming Language Lab

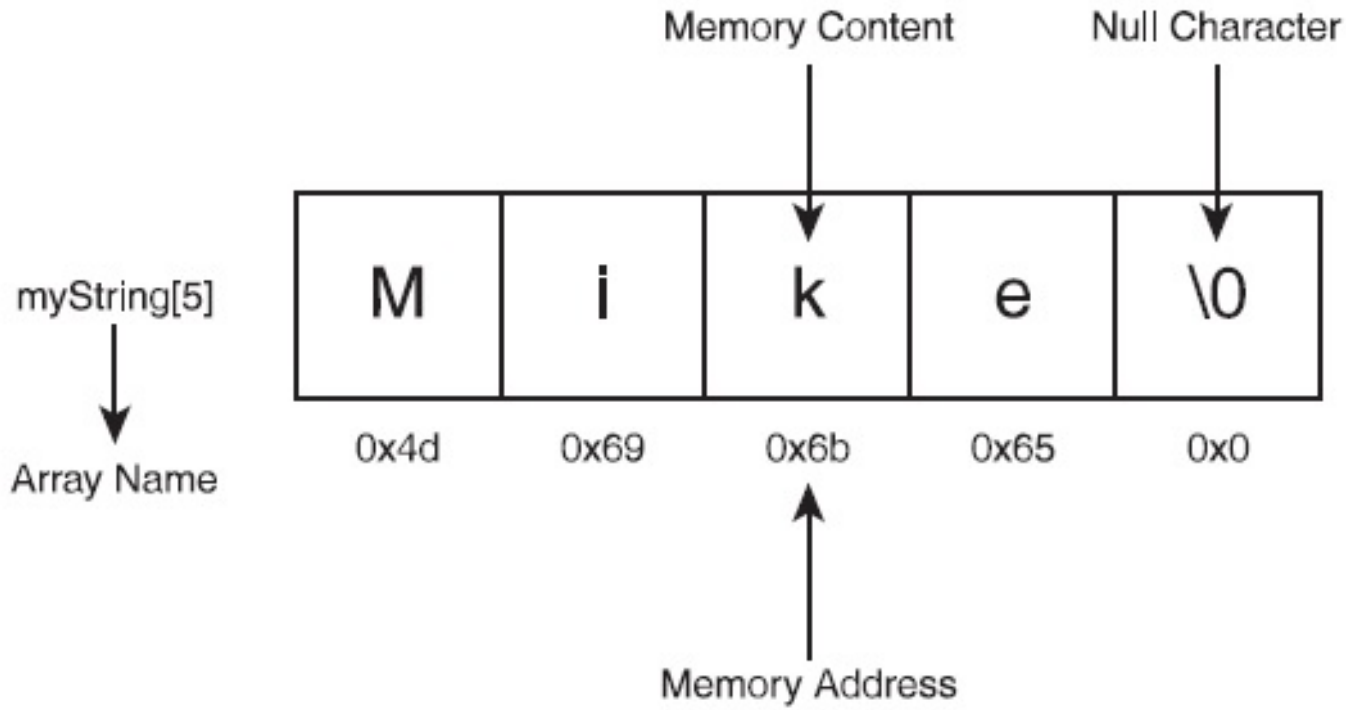
Bibhas Adhikari

IIT Kharagpur

November 3, 2020

Strings

- Strings are groupings of letters, numbers or any other characters
- In C, *strings* are created and initialized using a character array and a terminating NULL character, for instance:
  $char\ myString[5] = \{'M','i','k','e','\backslash 0'\};$



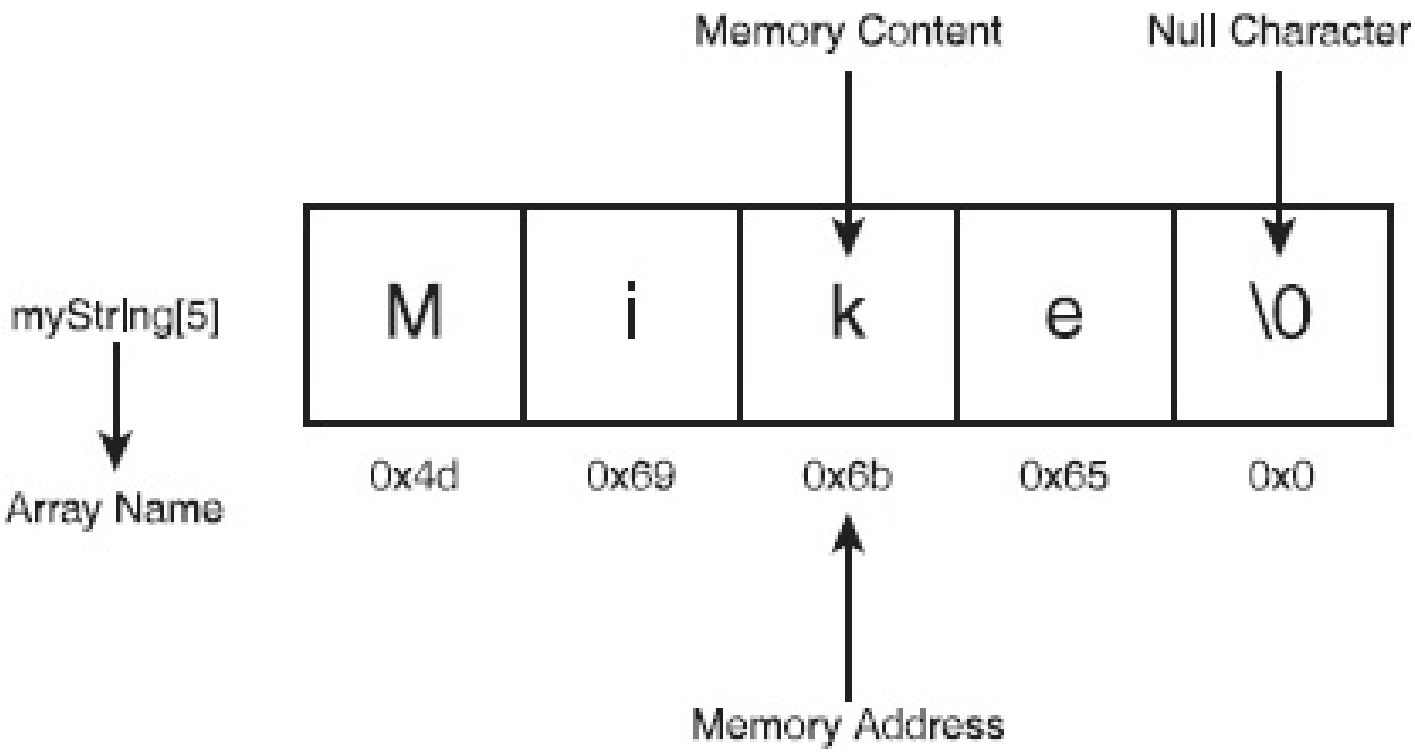
Strings

- However, the variable *myString* can be created and initialized with a string literal which is a grouping of characters enclosed in quotation marks: *char myString[ ]* = "Mike";
- Strings are implemented as pointers to the memory, that is, string names are just pointers that point to the first character's memory address in the string. For example, *char *myString = "Mike";*

```
char *mynameString = "Bibhas";
int x;
printf("\nThe pointer variable's value is: %p\n", *mynameString);
printf("\nThe pointer variable points to: %s\n", mynameString);
printf("\nThe memory locations for each character are: \n\n");
for(x = 0; x < 7; x++)
printf("%p\n", mynameString[x]);
```

String


*myString
0x4d
Pointer variable contains memory
address of first element in array


Memory Content
Null Character
myString[5]
Array Name
M
i
k
e
\0
0x4d
0x69
0x6b
0x65
0x0
Memory Address

Strings: printing

```
main()
{
char color[12] = {'\0'};
printf("Enter your favorite color:  ");
scanf("%s", color);
printf("\nYou entered: %s", color);
}
```

Strings: Array string

Using 1-d array:

```
main( )
{
char *strNames[5] = {0};
int x;
strNames[0] = "Apple";
strNames[1] = "Ball";
strNames[2] = "Cat";
strNames[3] = "Dog";
strNames[4] = "Elephant";
printf("\nNames in pointer array of type char:\n\n");
for (x = 0; x < 5; x++)
printf("%s\n", strNames[x]);
}
```

Strings: Array string

Using 2-d array:

```
main( )
{
char *colors[3][10] = {'\0'};
printf("\nEnter 3 colors seperated by spaces:  ");
scanf("%s %s %s", colors[0], colors[1], colors[2]);
printf("\nYour entered:  ");
printf("%s%s%s\n", colors[0], colors[1], colors[2]);
}
```

Strings: CONVERTING STRINGS TO NUMBERS

- *atof*—Converts a string to a floating-point number
- *atoi*—Converts a string to an integer
- \"-conversion operator

```
main( )
{
char *str1 = "123.79";
char *str2 = "55";
float x;
int y;
printf("\nString 1 is \"%s\"\n", str1);
printf("String 2 is \"%s\"\n", str2);
x = atof(str1);
y = atoi(str2);
printf("\nString 1 converted to a float is %.2f\n",x);
printf("String 2 converted to an integer is %d\n",y);
}
```

Strings: numerical arithmetic using strings

```
main()
{
char *str1 = "37";
char *str2 = "20";
int iResult;
iResult = atoi(str1) + atoi(str2);
printf("\nString 1 + String 2 is %d\n", iResult);
}
```

Strings: manipulation of strings

- *strlen( )* - part of the string-handling *library <string.h>*

```
#include <string.h>
main( )
{
char *str1 = "Bibhas";
char str2[ ] = "Adhikari";
printf("\nThe length of string 1 is %d\n",
strlen(str1));
printf("The length of string 2 is %d\n", strlen(str2));
}
```

- *tolower( ) and toupper( )* converting strings to either all uppercase or all lowercase with the help of the character-handling library *<ctype.h>*

```
#include <ctype.h>
main( )
{
char name1[ ] = "Bibhas";
char name2[ ] = "Adhikari";
convertL(name1);
convertU(name2);
}
void convertL(char *str) {
int x;
for (x = 0; x <= strlen(str); x++) str[x] = tolower(str[x]);
printf("\nFirst name converted to lower case is %s\n", str);
}
void convertU(char *str) {
int x;
for (x = 0; x <= strlen(str); x++) str[x] = toupper(str[x]);
printf("Last name converted to upper case is %s\n", str); }
```

Strings: manipulation of strings

- *strcpy( )* - copies the contents of one string into another string

```
# include <string.h>
main( )
{
char str1[11];
char *str2 = "C Language";
printf("\nString 1 now contains %s\n", strcpy(str1,
str2));
}
```

- *strcat( )* - concatenates or glues one string to another

```
main( )
{
char str1[40] = "Computer Science ";
char str2[] = "is applied mathematics";
printf("\n%s\n", strcat(str1, str2));
}
```

Strings: manipulation of strings

- *strcmp*( ) - to compare two strings for equality, character codes such as the ASCII character-coding system is udes to order the charadters

Sample Function	Return Value	Description
strcmp(string1, string2)	0	string1 is equal to string2
strcmp(string1, string2)	< 0	string1 is less than string2
strcmp(string1, string2)	> 0	string1 is greater than string2

- *strstr*( ) - considers two strings as arguments and searches the first string for an occurrence of the second string

```
main( )
{
char *str1 = "A";
char *str2 = "A";
char *str3 = "!";
printf("\nstr1 = %s\n", str1);
printf("\nstr2 = %s\n", str2);
printf("\nstr3 = %s\n", str3);
printf("\nstrcmp(str1, str2) = %d\n", strcmp(str1, str2));
printf("\nstrcmp(str1, str3) = %d\n", strcmp(str1, str3));
printf("\nstrcmp(str3, str1) = %d\n", strcmp(str3, str1));
if ( strcmp(str1, str2) == 0)
printf("\nLetter A is equal to letter A\n");
if ( strcmp(str1, str3) > 0)
printf("Letter A is greater than character !\n");
if ( strcmp(str3, str1) < 0)
printf("Character !  is less than letter A\n");
}
```

```
main( )
{
char *str1 = "Analyzing strings with the strstr()
function";
char *str2 = "ing";
char *str3 = "xyz";
printf("\nstr1 = %s\n", str1);
printf("2 = %s\n", str2);
printf("3 = %s\n", str3);
if ( strstr(str1, str2) != NULL )
printf("\nstr2 was found in str1\n");
else
printf("\nstr2 was not found in str1\n");
if ( strstr(str1, str3) != NULL )
printf("\nstr3 was found in str1\n");
else
printf("\nstr3 was not found in str1\n");
}
```