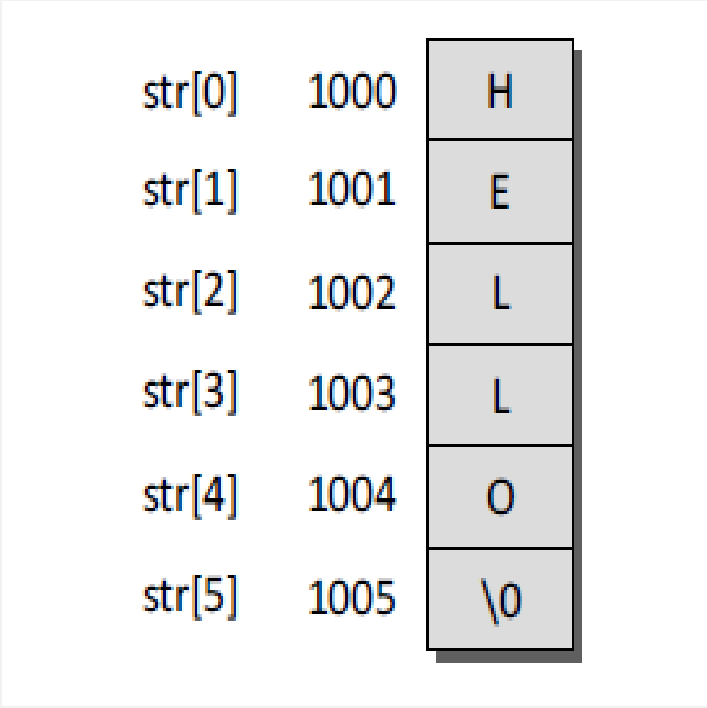
Strings

Introduction

- A string is a null-terminated character array. This means that after the last character, a null character (‘\0’) is stored to signify the end of the character array.
- The general form of declaring a string is
- char str[size];
- For example if we write,
- char str[] = “HELLO”;
- We are declaring a character array with 5 characters namely, H, E, L, L and O. Besides, a null character (‘\0’) is stored at the end of the string. So, the internal representation of the string becomes- HELLO’\0’. Note that to store a string of length 5, we need 5 + 1 locations (1 extra for the null character).

Introduction

- The name of the character array (or the string) is a pointer to the beginning of the string.



Reading Strings

If we declare a string by writing

char str[100];

Then str can be read from the user by using three ways

use scanf function

using gets() function

using getchar()function repeatedly

- The string can be read using scanf() by writing

  scanf(“%s”, str);

- The string can be read by writing

  gets(str);

  gets() takes the starting address of the string which will hold the input. The string inputted using gets() is automatically terminated with a null character.

Reading Strings

- The string can also be read by calling the getchar() repeatedly to read a sequence of single characters (unless a terminating character is entered) and simultaneously storing it in a character array.

```
i=0;
getchar(ch);
while(ch != '*')
{       str[i] = ch;
        i++;
        getchar(ch);
}
```

Writing Strings

The string can be displayed on screen using three ways

- use printf() function
- using puts() function
- using putchar()function repeatedly

The string can be displayed using printf() by writing

```
printf("%s", str);
```

The string can be displayed by writing

```
puts(str);
```

Writing Strings

The string can also be written by calling the putchar() repeatedly to print a sequence of single characters.

```
i=0;
while(str[i] != '\0*)
{       putchar(str[i]);
        i++;
}
```

Suppressing Input

- scanf() can be used to read a field without assigning it to any variable. This is done by preceding that field's format code with a *. For example, given:
- scanf("%d*c%d", &hr, &min);
- The time can be read as 9:05 as a pair. Here the colon would be read but not assigned to anything.
- **Using a Scanset**
- The ANSI standard added the new **scanset** feature to the C language. A scanset is used to define a set of characters which may be read and assigned to the corresponding string. A scanset is defined by placing the characters inside square brackets prefixed with a %.

Suppressing Input

- **int main()**
  {
  char str[10];
  printf("\n Enter string: ");
  scanf("%[aeiou]", str);
  printf("The string is : %s", str);
  return 0;
  }
- The code will stop accepting character as soon as the user will enter a character that is not a vowel.
- However, if the first character in the set is a ^ (caret symbol), then scanf() will accept any character that is not defined by the scanset. For example, if you write
- scanf("%[^aeiou]", str);

Length

- The number of characters in the string constitutes the length of the string.
- For example, LENGTH(“C PROGRAMMING IS FUN”) will return 20. Note that even blank spaces are counted as characters in the string.
- LENGTH(‘0’) = 0 and LENGTH(‘’) = 0 because both the strings does not contain any character.

```
Step 1: [INITIALIZE] SET I = 0
Step 2: Repeat Step 3 while STR[I] != NULL
Step 3:       SET I = I + 1
        [END OF LOOP]
Step 4: SET LENGTH = I
Step 5: END
```

Converting Characters of a String into Upper Case

- In memory the ASCII code of a character is stored instead of its real value. The ASCII code for A-Z varies from 65 to 91 and the ASCII code for a-z ranges from 97 to 123. So if we have to convert a lower case character into upper case, then we just need to subtract 32 from the ASCII value of the character.

```
Step1:  [INITIALIZE] SET I=0
Step 2: Repeat Step 3 while STR[I] != NULL
Step 3:          IF STR[I] >= 'a' AND STR[I] <= 'z'
                         SET Upperstr[I] = STR[I] - 32
                 ELSE
                         SET Upperstr[I] = STR[I]
                 [END OF IF]
                 SET I = I + 1
         [END OF LOOP]
Step 4: SET Upperstr[I] = NULL
Step 5: EXIT
```

Concatenating two Strings to form a New String

IF S1 and S2 are two strings, then concatenation operation produces a string which contains characters of S1 followed by the characters of S2.

```
Step 1: [INITIALIZE] I = 0 and J = 0
Step 2: Repeat Steps 3 to 4 while str1[i]! = NULL
Step 3:          SET new_str[J] = str1[I]
Step 4:          Set I = I+1 and J = J+1
        [END of LOOP]
Step 5: SET I=0
Step 6: Repeat Steps 6 to 7 while str2[i]! = NULL
Step 7:          SET new_str[J] = str2[I]
Step 8:          Set I = I+1 and J = J+1
        [END of LOOP]
Step 9: SET new_str[J] = NULL
Step 10: EXIT
```

Appending

- Appending one string to another string involves copying the contents of the source string at the end of the destination string. For example, if S1 and S2 are two strings, then appending S1 to S2 means we have to add the contents of S1 to S2. so S1 is the source string and S2 is the destination string. The appending operation would leave the source string S1 unchanged and destination string S2 = S2+S1.

```
Step 1: [INITIALIZE] SET I =0 and J=0
Step 2: Repeat Step 3 while Dest_Str[I] != NULL
Step 3:   SET I + I + 1
          [END OF LOOP]
Step 4: Repeat Steps 5 to 7 while Source_Str[J] != NULL
Step 5:   Dest_Str[I] = Source_Str[J]
Step 6:   SET I = I + 1
Step 7:   SET J = J + 1
        [END OF LOOP]
Step 8: SET Dest_Str[J] = NULL
Step 9: EXIT
```

Comparing Two Strings

If S1 and S2 are two strings then comparing two strings will give either of these results-

- S1 and S2 are equal
- S1>S2, when in dictionary order S1 will come after S2
- S1<S2, when in dictionary order S1 precedes S2

```
Step 1: [INITIALIZE] SET I=0, SAME =0
Step 2: SET Len1 = Length(STR1), Len2 = Length(STR2)
Step 3: IF len1 != len2, then
           Write "Strings Are Not Equal"
        ELSE
           Repeat while I<Len1
              IF STR1[I] == STR2[I]
                    SET I = I + 1
              ELSE
                    Go to Step 4
              [END OF IF]
           [END OF LOOP]
        IF I = Len1
              SET SAME = 1
              Write "Strings are equal"
        [END OF IF]
Step 4:  IF SAME = 0
           IF STR1[I] > STR2[I], then
              Write "String1 is greater than String2
           ELSE IF STR1[I] < STR2[I], then
              Write "String2 is greater than String1"
           [END OF IF]
        [END OF IF]
Step 5: EXIT
```

Reversing a String

- If S1= “HELLO”, then reverse of S1 = “OLLEH”. To reverse a string we just need to swap the first character with the last, second character with the second last character, so on and so forth.

```
Step 1: [INITIALIZE] SET I=0, J= Length(STR)  1
Step 2: Repeat Steps 3 and 4 while I < J
Step 3:      SWAP( STR(I), STR(J))
Step 4:      SET I = I + 1, J = J   1
       [END OF LOOP]
Step 5: EXIT
```

Array of Strings

- Now suppose that there are 20 students in a class and we need a string that stores names of all the 20 students. How can this be done? Here, we need a string of strings or an array of strings. Such an array of strings would store 20 individual strings. An array of string is declared as,

  char names[20][30];

- Here, the first index will specify how many strings are needed and the second index specifies the length of every individual string. So here, we allocate space for 20 names where each name can be maximum 30 characters long.

Array of Strings

- Let us see the memory representation of an array of strings. If we have an array declared as,

  char name[5][10] = {“Ram”, “Mohan”, “Shyam”, “Hari”, “Gopal”};

Name[0]	R	A	M	'\0'					
Name[1]	M	O	H	A	N	'\0'			
Name[2]	S	H	Y	A	M	'\0'			
Name[3]	H	A	R	I	'\0'				
Name[4]	G	O	P	A	L	'\0'			

Write a Program to Read and Print the Names of N Students of a Class

```
#include<stdio.h>
#include<conio.h>
main()
{
        char names[5][10];
        int i, n;
        clrscr();

        printf("\n Enter the number of students : ");
        scanf("%d", &n);
        for(i=0;i<n;i++)
```

Write a Program to Read and Print the Names of N Students of a Class

```
        {
                printf("\n Enter the name of %dth student : ", i+1);
                gets(names[i]);
        }
        printf("\n Names of the students are : \n");
        for(i=0;i<n;i++)
                puts(names[i]);
        getch();
        return 0;
}
```

Thank You!