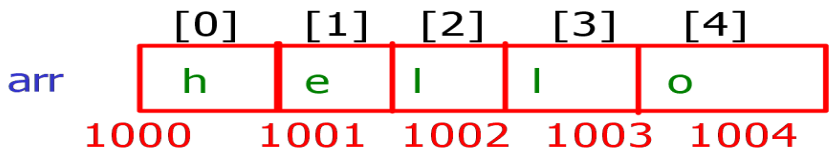
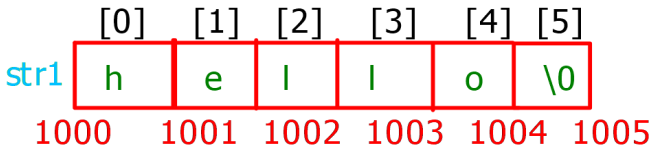
Strings

- String is a sequence of characters terminated by null character '\0' or array of characters terminated by null character '\0'.
- Strings are used to store the text/message information in the application like name, address, company name, institute name etc.
- Normally array of character is declared as,
    - char arr[5] = {'h','e','l','l','o'};
    - Memory will be allocated 5 bytes in stack like below

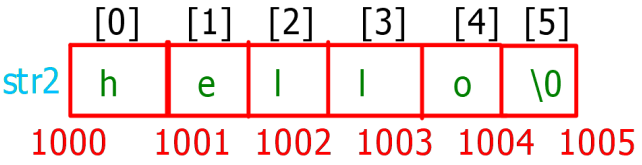


- But above array is not string, it has to be end by null character like below example
- Below are the ways of declaring the string in C language

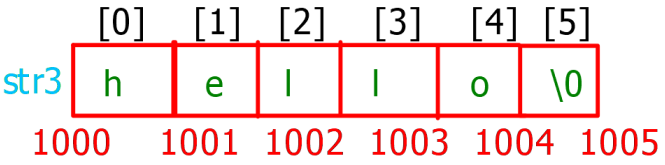
1. **char str1[6] = {'h','e','l','l','o','\0'};**

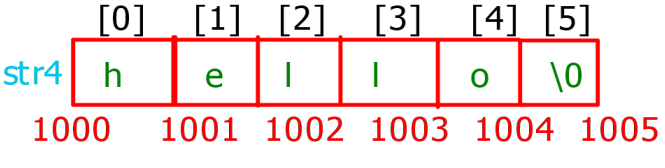


2. **char str2[] = {'h','e','l','l','o','\0'};**
    a. As, you learned in the arrays, size can be skipped if array is initialised
    b. But see to it that array is terminated by '\0'



3. **char str3[] = {"h" "e" "l" "l" "o" "\0"};**
    a. Here, each character is treated as string and concatenate with the each like, h\0+e\0+l\0+l\0+o\0+\0 = hello\0+\0 = hello\0
    b. So, whenever " " is used, the compiler will add a null character at the end.



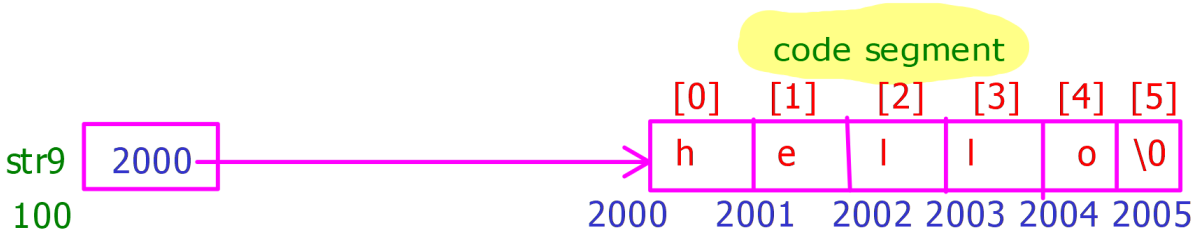


4. **char str5[6] = {“Hello”};**
    a. Here, because of double quotes, the compiler will add null character at the end explicitly.

5. Below are the other ways of creating strings.

```
char str6[] = {"Hello"};
char str7[6] = "Hello";
char str8[] = "Hello";
```

6. char *str9 = “hello”;
    - Pointer to an characters
    - These kind of initialisation is called as string literals
    - Memory for the string literals are allocated in code/text segment



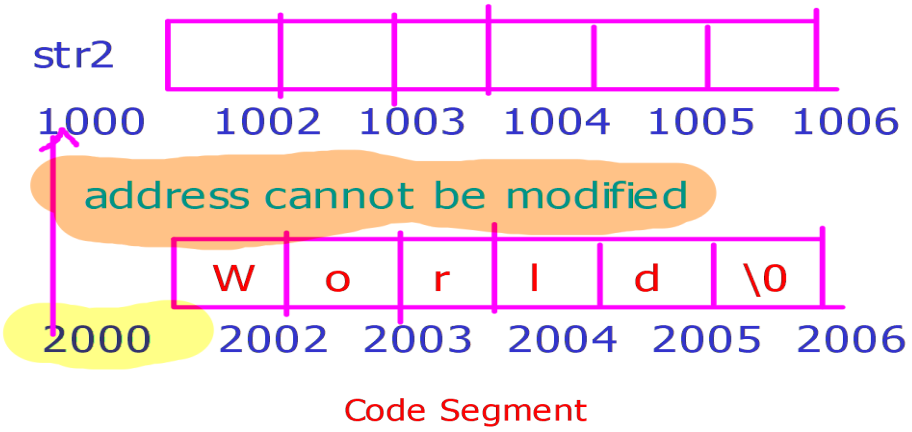
    - Code segment is read only memory
    - So, if you try to modify the string literals which will result in segmentation fault error
    - Str[2] = ‘E’ is not allowed

Invalid ways of creating strings

1. char str3[] = {"h","e","l","l","o","\0"};
    - invalid creation/initialization of a string - error
    - This will be treated as an array of strings, to create an array of string 2D array is used, which will be learned later in the module.
2. char str[ ];
    - Invalid, because the size of the array is mandatory.

3. char str2[6];
   Str2 = “World”;
    - It's a compile time error because array address will be fixed
    - This declaration is creating string in code segment and returning the address of that string, Array address cannot be modified as shown below

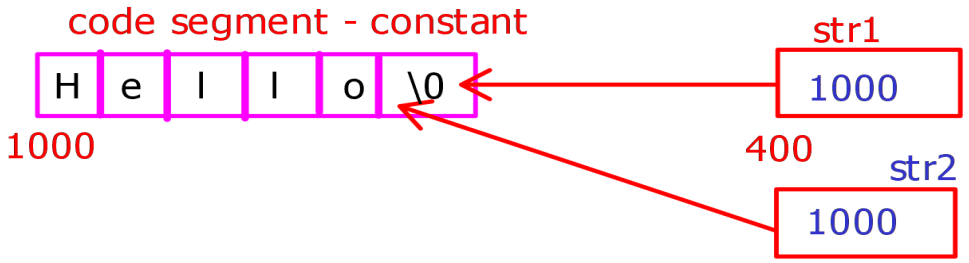
- So, it will result in a compile time error.



- But with a pointer it's possible because a pointer is capable of holding the address of any memory segments.
- So, the below declaration holds true for pointers.
    - char *str;
      Str = “World”

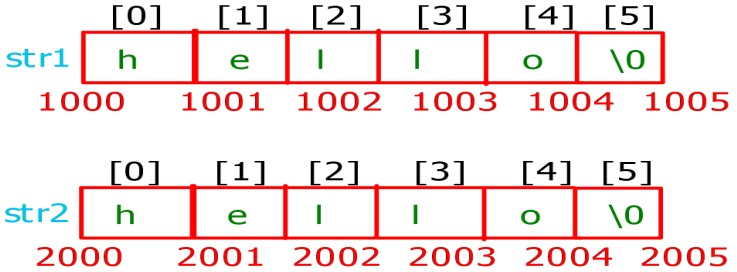
Shareable Strings

- Consider below example
    - char str1 = “Hello”:
      char str2 = “Hello”;



- If the string literals are having same collection of characters (case sensitive), then because of read only memory instead of allocating new memory compiler will make pointer to point to the same memory location
- This is known as shareable memory.
- Sharing is only valid in case of string literals and pointers. With array the memory layout looks like below:

str1[6] = “hello”;
str2[6] = “hello”;



String Methods

1. strlen(str)
    a. Strlen function returns the length of the string
    b. Difference between strlen and sizeof is that sizeof returns the number of bytes including the null character but strlen returns length excluding null character.
    c. Strlen returns an unsigned integer value.

```
size_t my_strlen(char *str)
{
    int count =
    while(*(str+count++));
    return count-1;
}
```

⇒

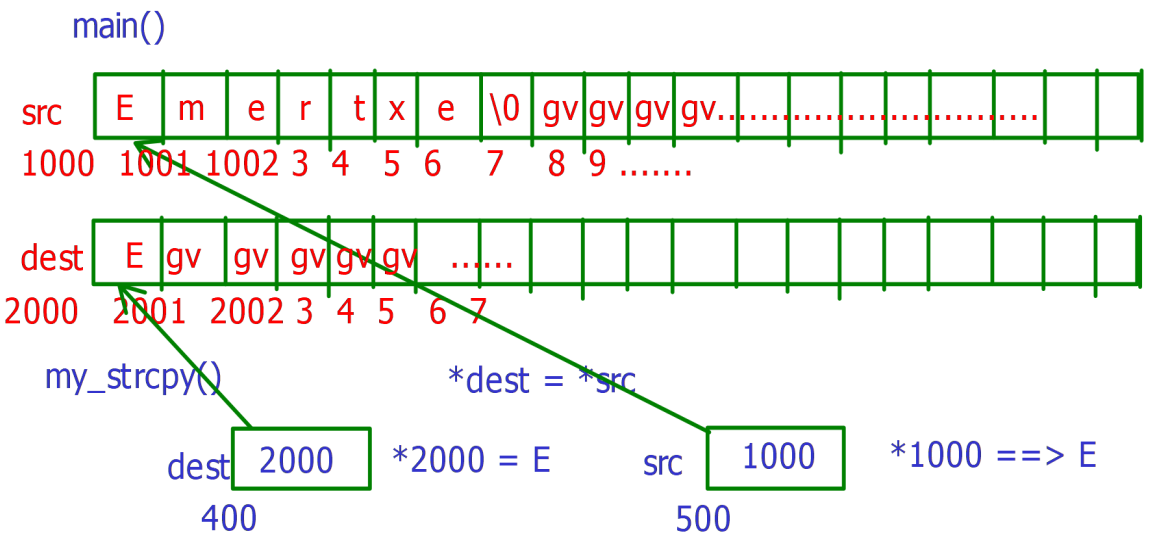
1.. *(1000+0++) - *1000 - h count=1

2. *(1000+1++) - *1001 - i count=2

3. *(1000+2++) - *1002 - \0 count = 3

2. String copy - **char *strcpy(char *_dest_, const char *_src_);**
    a. Strcpy is used to copy a string from one to another.
    b. Strcpy will copy character by character like below

```
void my_strcpy(char *dest, char *src)
{
        while(*src)
        {
                *dest = *src;
                src++;
                dest++;
        }
        *dest = *src; //to copy null character at the end
}
```



3. String Compare - **`int strcmp(const char *s1, const char *s2);`**
    a. Strcmp is used to compare the 2 strings.
    b. Strcmp will return 0 → if both the strings are equal
        i. Returns < 0 if first string has leeser ASCII value than second string
        ii. Returns > 0 if first string has greater ASCII than second string

    For example
    Case 1: string1 = “Ram”
    string2 = “Ram”

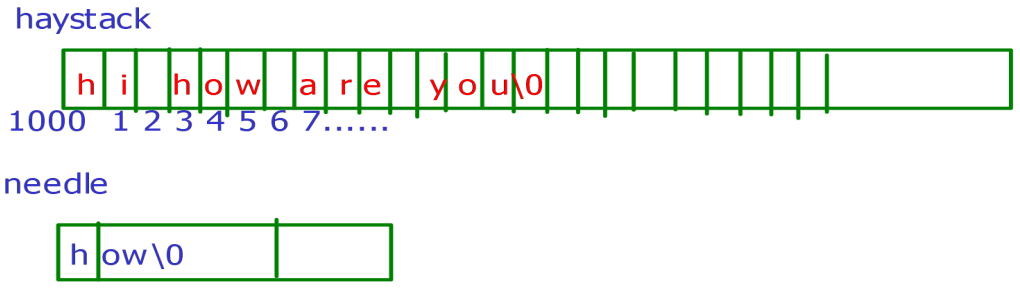
    ‘R’ is compared with ‘R’ using ASCII equivalent -> ‘R’ - ‘R’
    -> 82 - 82 = 0
    ‘a’ is compared with ‘a’ using ASCII equivalent -> ‘a’ - ‘a’
    -> 97 - 97 = 0
    ‘m’ is compared with ‘a’ using ASCII equivalent -> ‘m’ - ‘m’
    -> 109 - 109 = 0
    It's returning 0 at the end so, strings are equal.

    Case 2:
    string 1 = "Ram"
    string 2 = "Rama"

    ‘R’ is compared with ‘R’ using ASCII equivalent -> ‘R’ - ‘R’
    -> 82 - 82 = 0
    ‘a’ is compared with ‘a’ using ASCII equivalent -> ‘a’ - ‘a’
    -> 97 - 97 = 0
    ‘m’ is compared with ‘a’ using ASCII equivalent -> ‘m’ - ‘m’
    -> 109 - 109 = 0
    ‘\0’ is compared with ‘a’ using ASCII equivalent -> ‘\0’ - ‘a’
    -> 0 - 97 = - 97
    Here, strcmp return -97, so strings are not equal

- strcmp will compare character by character using ASCII equivalent value
- Once the character is mismatched, it will stop comparing
- If any one of the string reached to null character stop comparing

4. Substring search - char *strstr(const char *haystack, const char *needle);



a. Strstr will search for the substring in the given string
b. Here, needle is the substring and haystack is the string.
c. once there is a matching, it will return the starting address of how which is 1003 in the above example.
d. returns the base address of matching character
e. if matching needle is not found then it will return the NULL address

5. String concatenation - **`char *strcat(char *restrict dest, const char *restrict src);`**
   a. Strcat will merge one string with another
   b. It will concate the destination string with source

6. String token - **`char *strtok(char *restrict str, const char *restrict delim);`**
   a. Strtok will search for the token present in the string
   b. Once the token is found, replace that token with '\0'.
   c. For example,
      char str = "hi;':how are\you?/;bangalore"
      char token = ";"
   d. In the above example strtok will search for the ";" in the string whenever it finds the token it stops searching and returns the starting address from where it started searching for.
   e. To continue the search use loop and next time pass the Null as a first argument.
      Ex,

```
Res = strtok(str, token);
while(*str)
{
        printf("%s\n",res);
        strtok(NULL,token);
}
```

7. ASCII to integer - **`int atoi(const char *nptr);`**
   a. Atoi function converts given string to integer
   b. Takes input as string ang produce integer as output
   c. For example - input : "123"
      Output: 123

      Input: "123abc"
      Output: 123

      Input: abc123
      Output: 0