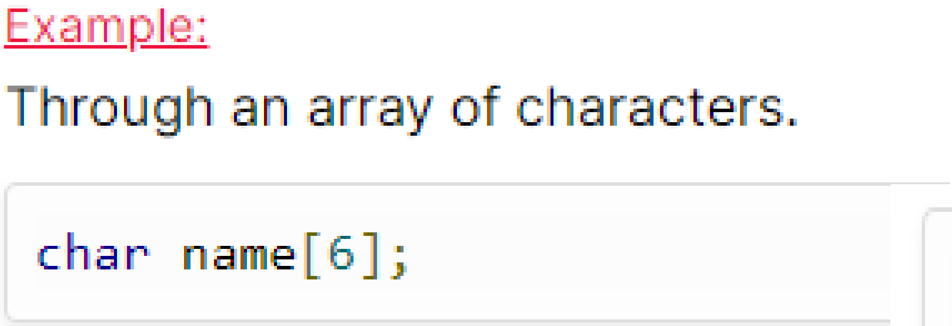
Strings in C Language

Module 2 Part – 2

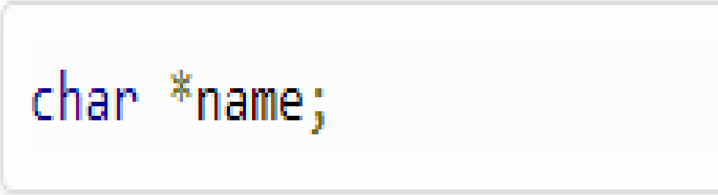
String Declaration and assigning values



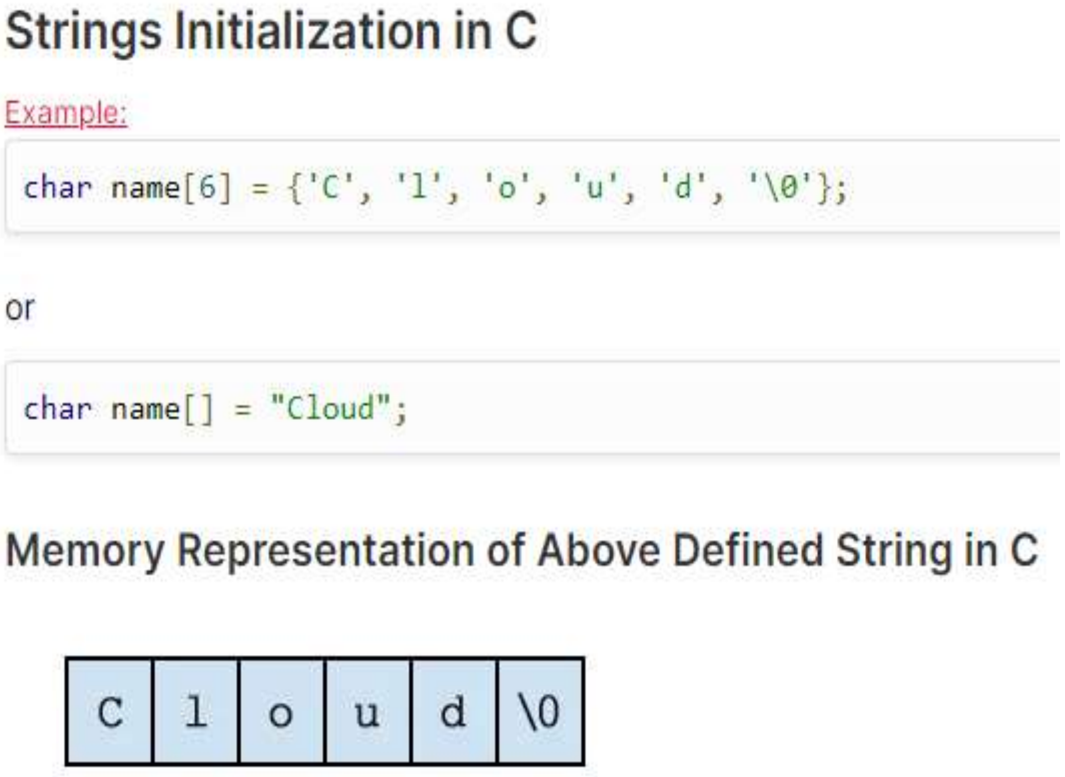
Example:

Through an array of characters.

```
char name[6];
```



```
char *name;
```



Strings Initialization in C

Example:

```
char name[6] = {'C', 'l', 'o', 'u', 'd', '\0'};
```

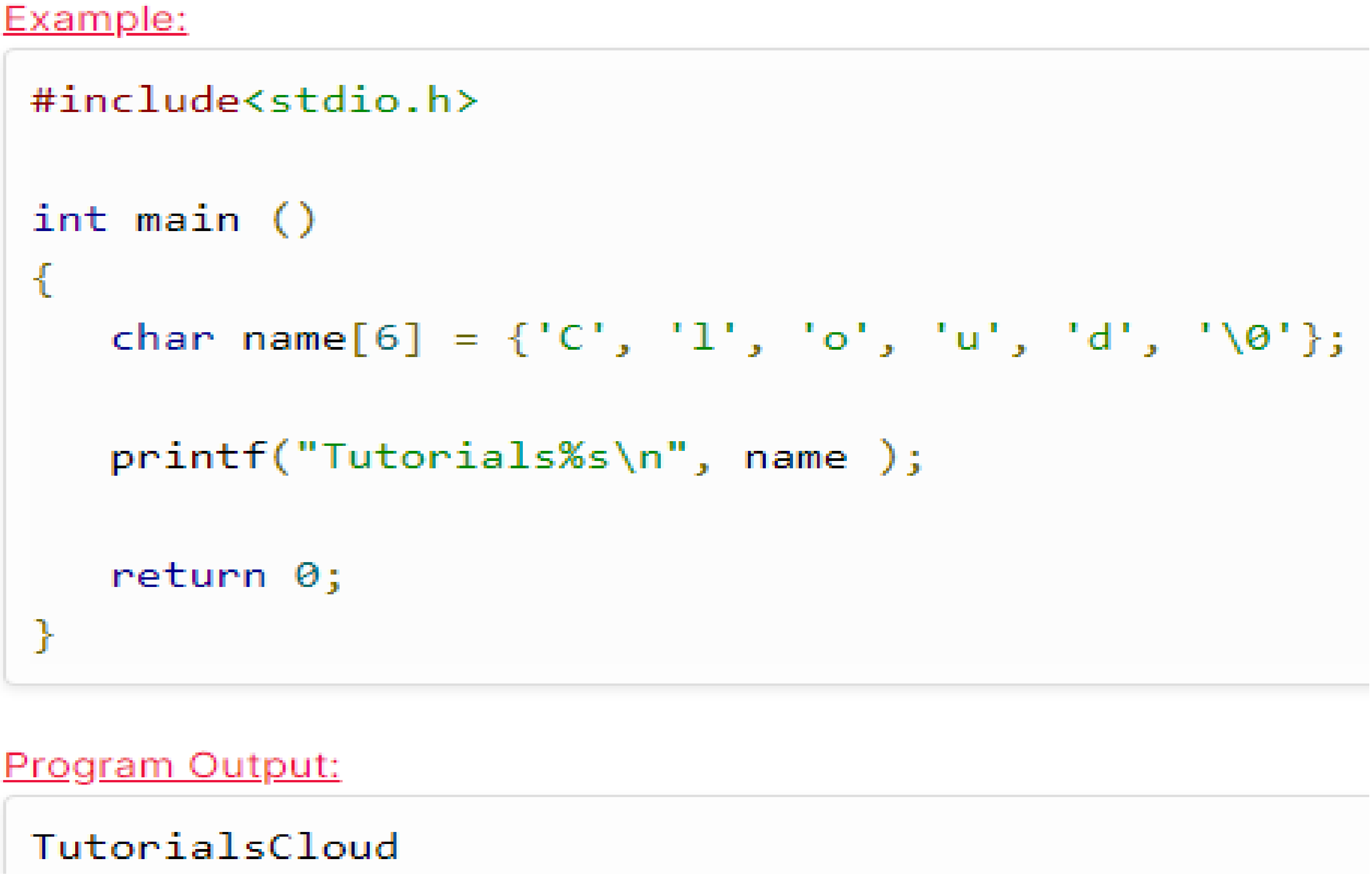
or

```
char name[] = "Cloud";
```

Memory Representation of Above Defined String in C

C	l	o	u	d	\0

Example:-



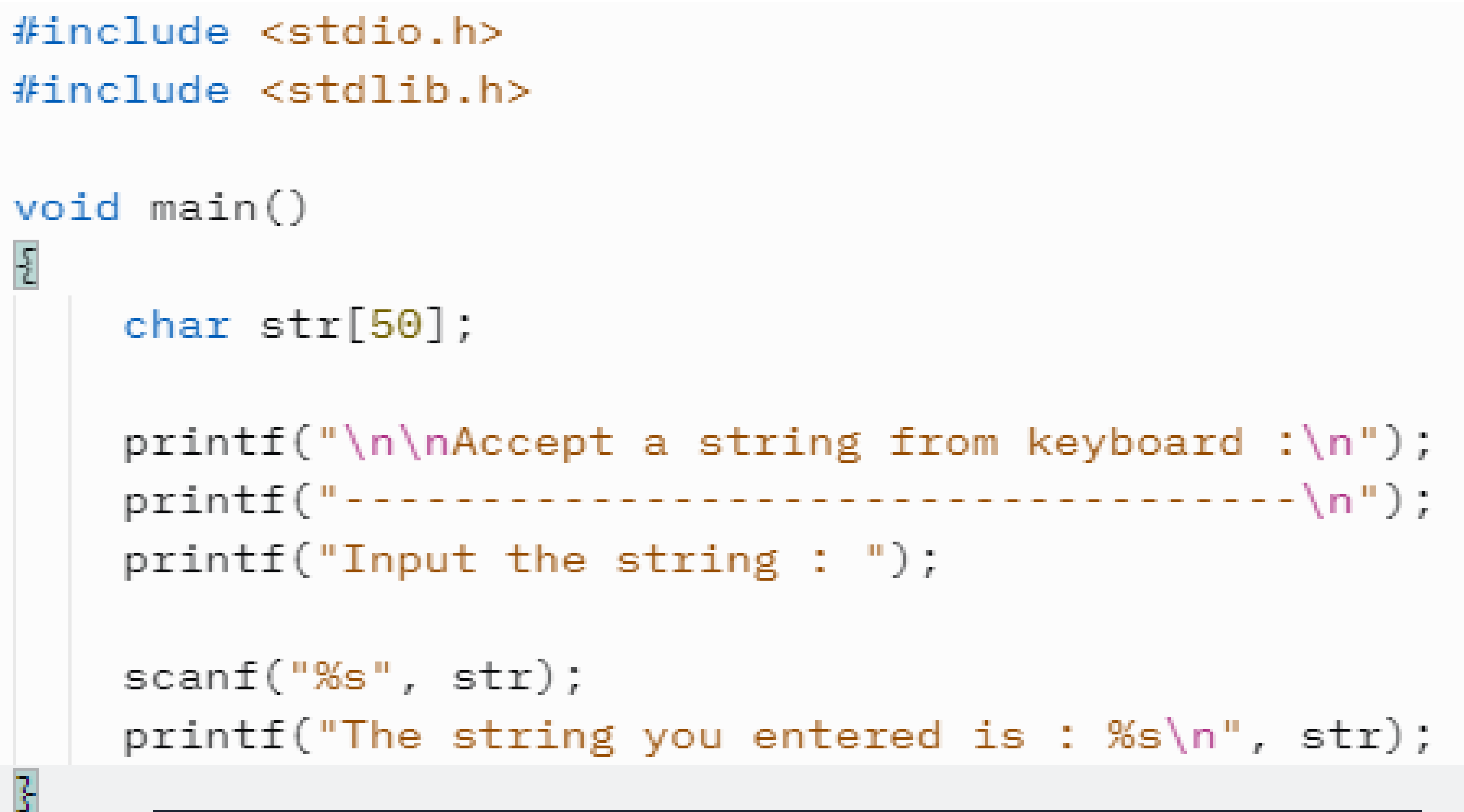
Example:

```
#include<stdio.h>

int main ()
{
   char name[6] = {'C', 'l', 'o', 'u', 'd', '\0'};

   printf("Tutorials%s\n", name );

   return 0;
}
```

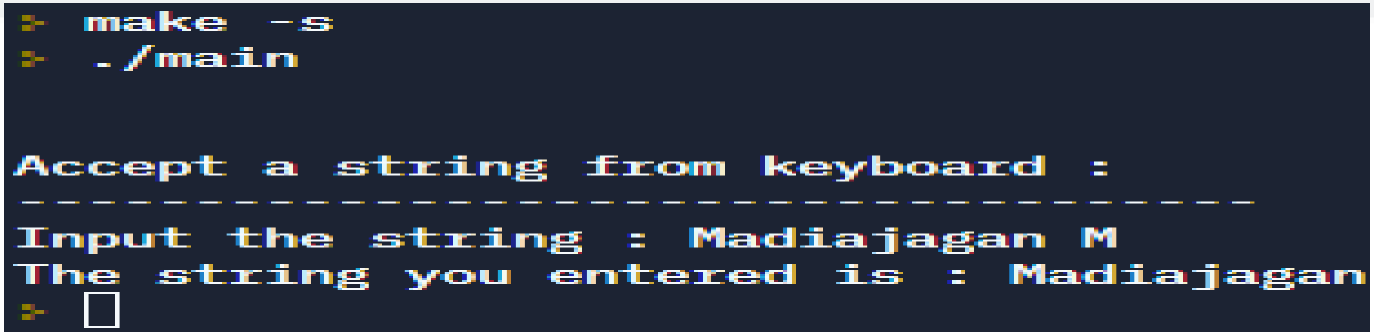
Program Output:

```
TutorialsCloud
```

Example: String

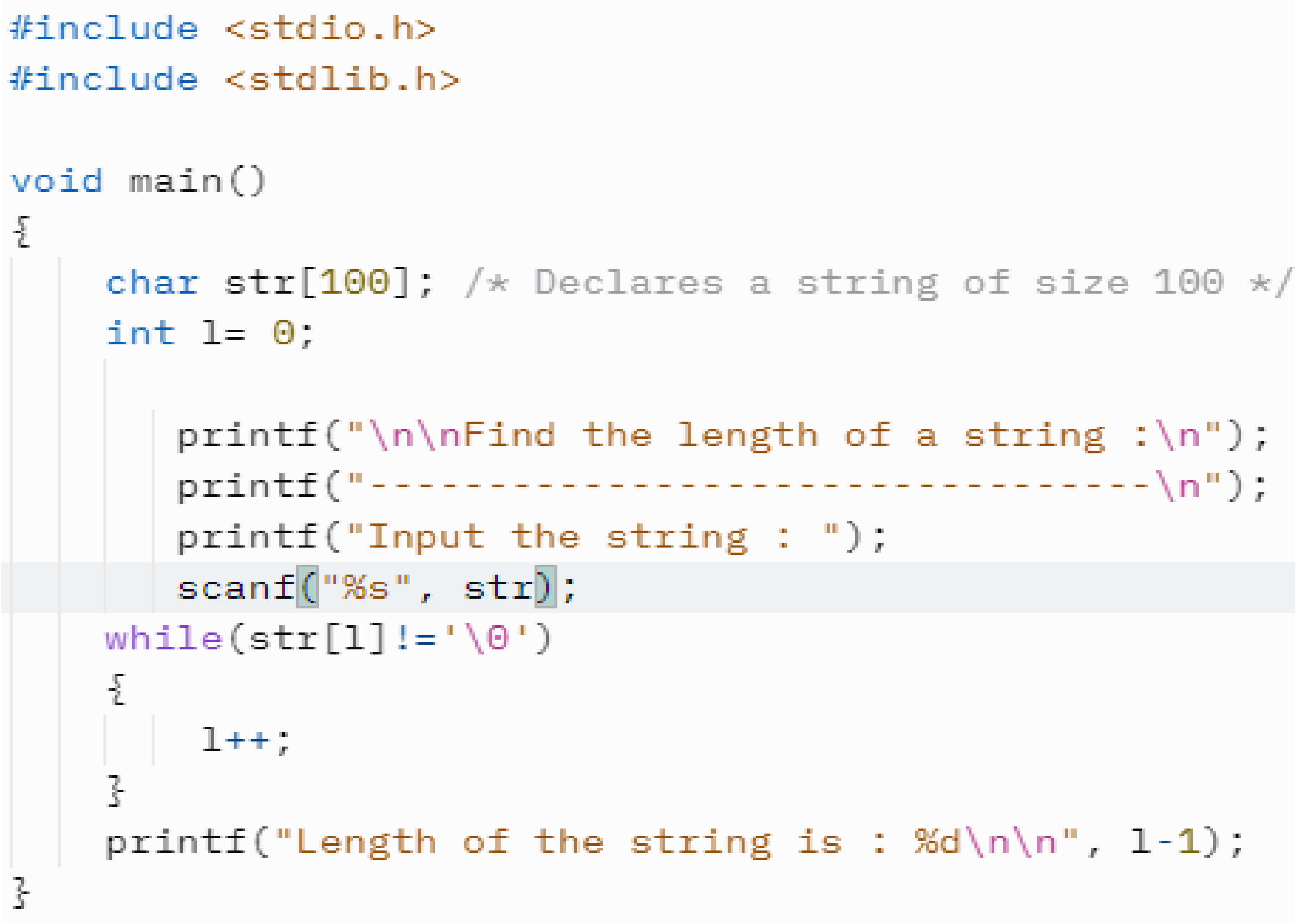


```c
#include <stdio.h>
#include <stdlib.h>

void main()
{
    char str[50];

    printf("\n\nAccept a string from keyboard :\n");
    printf("-----------------------------------\n");
    printf("Input the string : ");

    scanf("%s", str);
    printf("The string you entered is : %s\n", str);
}
```



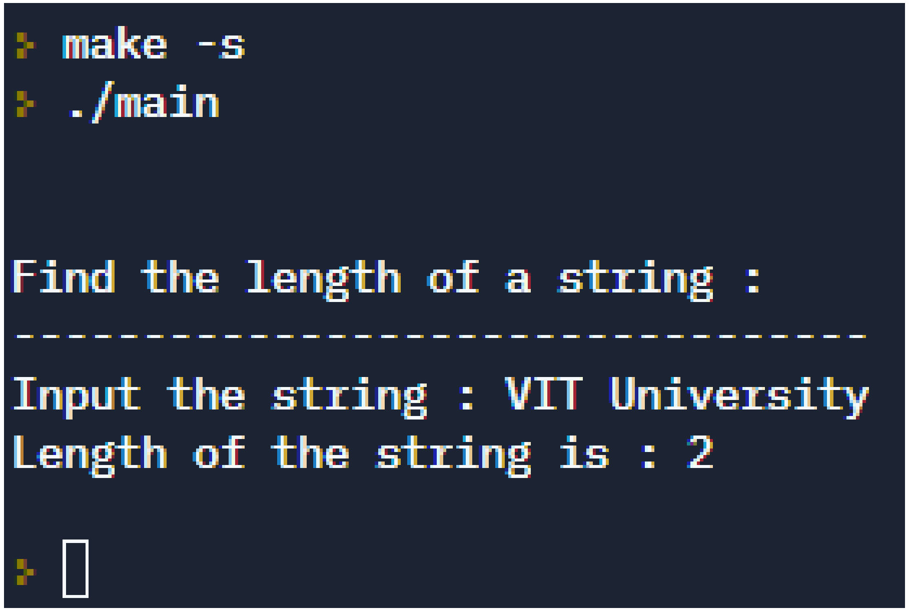
```
> make -s
> ./main


Accept a string from keyboard :
-----------------------------------
Input the string : Madiajagan M
The string you entered is : Madiajagan
> 
```

Write a program in C to find the length of a string without using library function



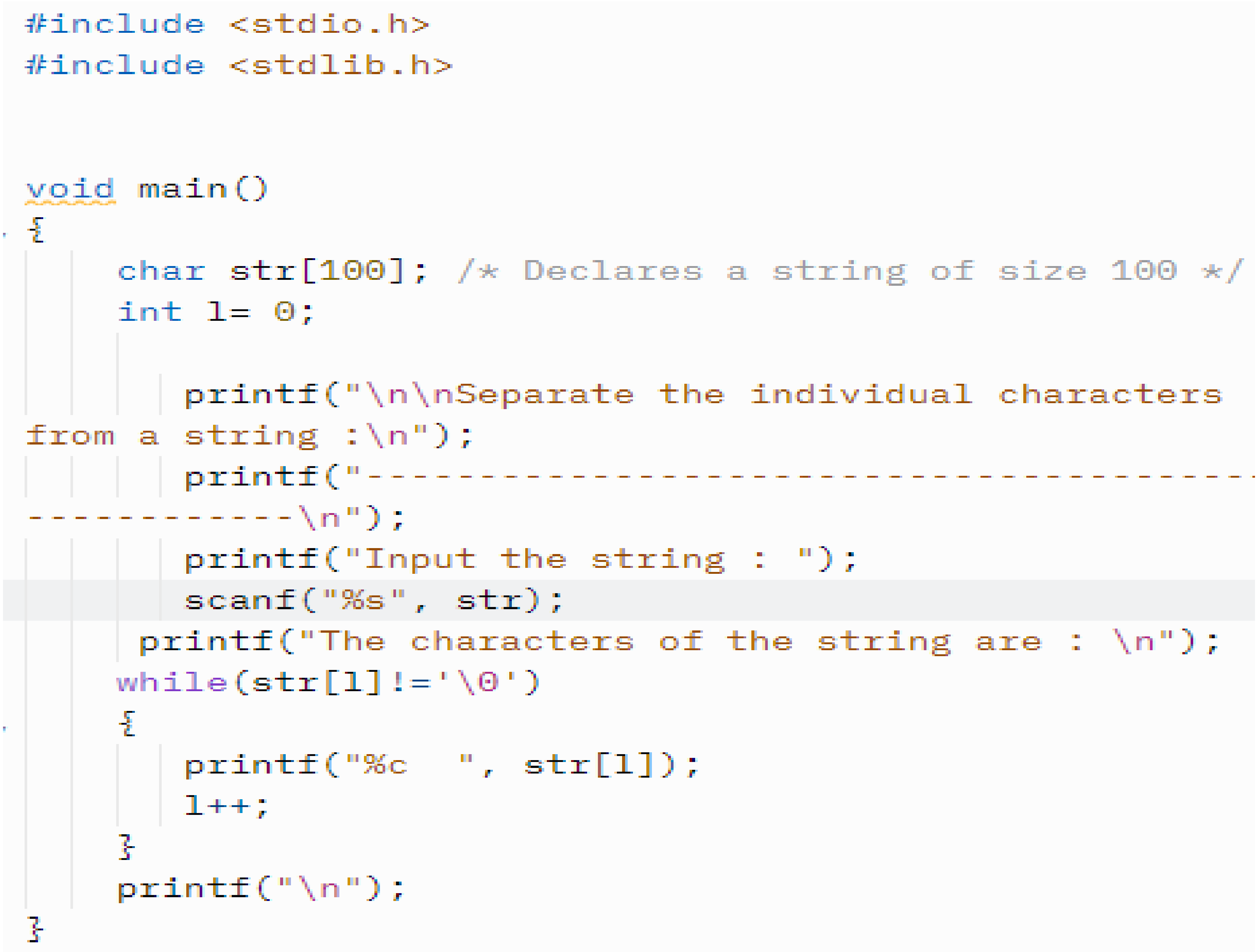
```c
#include <stdio.h>
#include <stdlib.h>

void main()
{
    char str[100]; /* Declares a string of size 100 */
    int l= 0;

       printf("\n\nFind the length of a string :\n");
       printf("---------------------------------\n");
       printf("Input the string : ");
       scanf("%s", str);
    while(str[l]!='\0')
    {
        l++;
    }
    printf("Length of the string is : %d\n\n", l-1);
}
```

Output:-



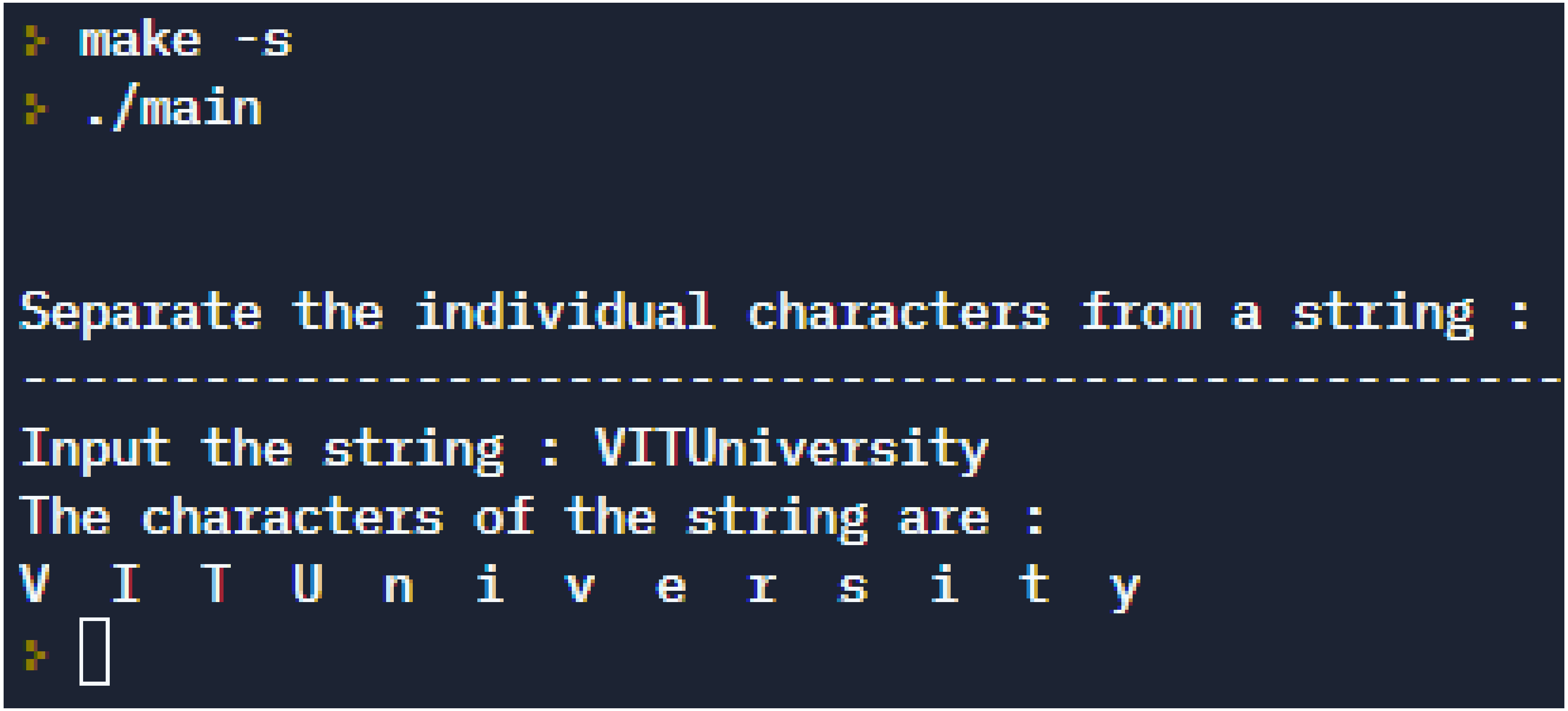
```
› make -s
› ./main


Find the length of a string :
---------------------------------
Input the string : VIT University
Length of the string is : 2

› 
```

Write a program in C to separate the individual characters from a string.



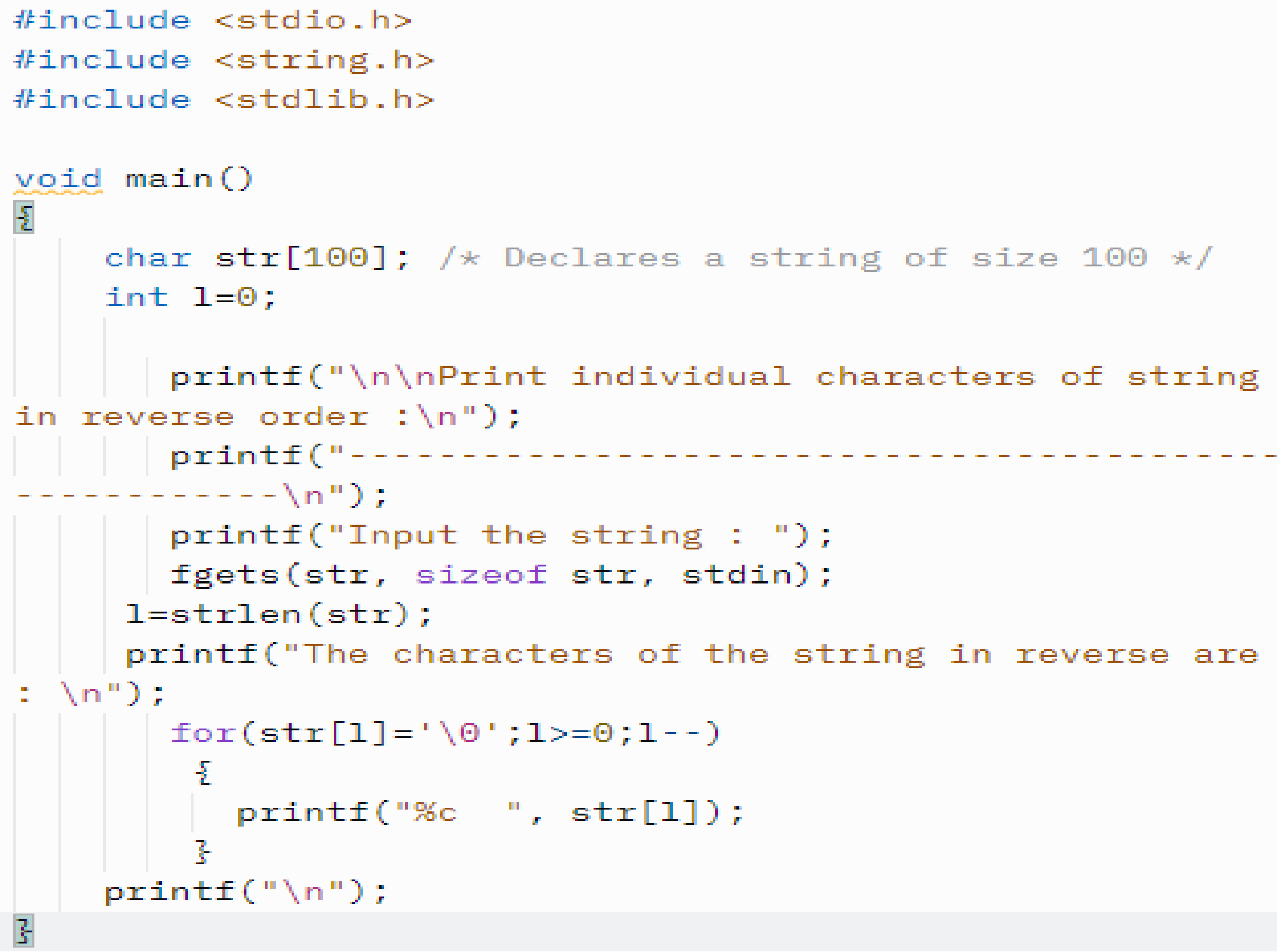
```
#include <stdio.h>
#include <stdlib.h>


void main()
{
    char str[100]; /* Declares a string of size 100 */
    int l= 0;

       printf("\n\nSeparate the individual characters
from a string :\n");
       printf("----------------------------------------
------------\n");
       printf("Input the string : ");
       scanf("%s", str);
     printf("The characters of the string are : \n");
    while(str[l]!='\0')
    {
       printf("%c  ", str[l]);
       l++;
    }
    printf("\n");
}
```

Output:



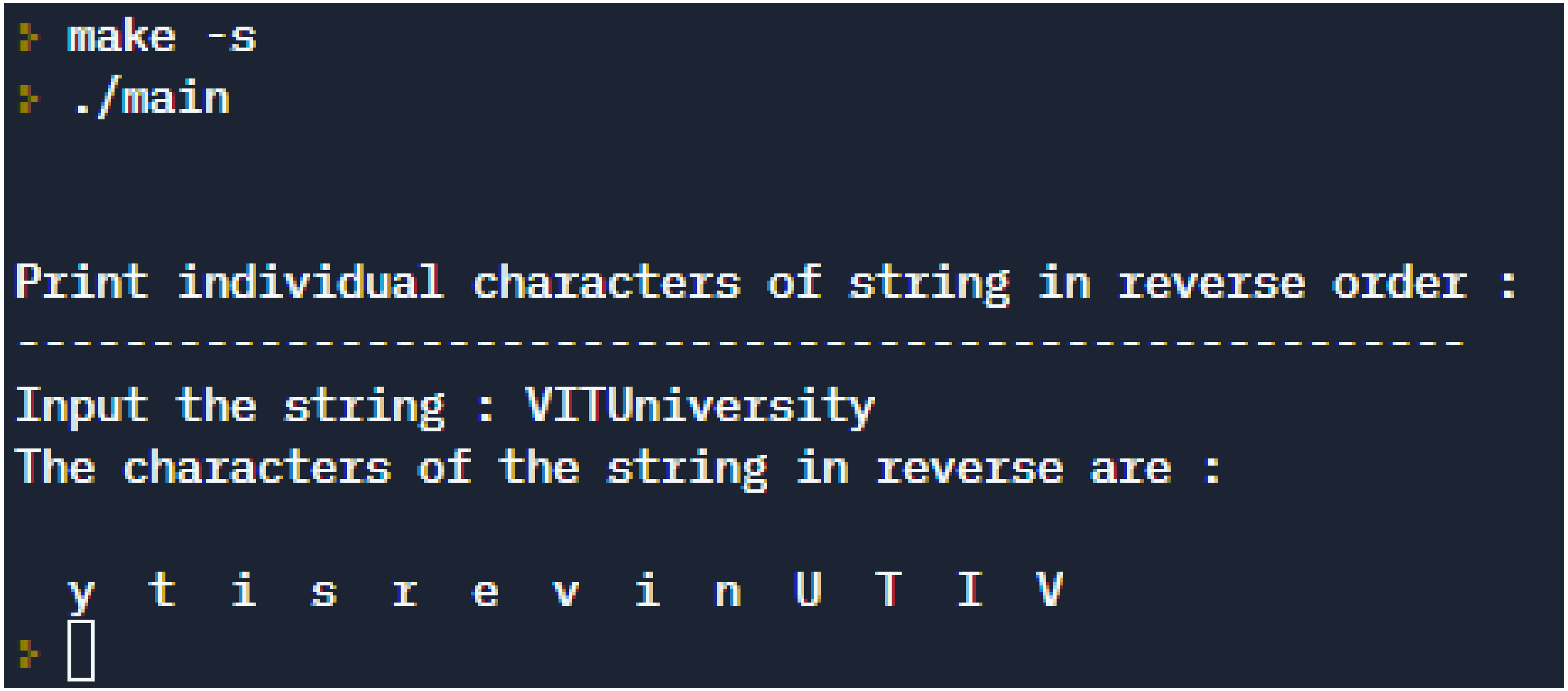
```
› make -s
› ./main


Separate the individual characters from a string :
----------------------------------------------------
Input the string : VITUniversity
The characters of the string are :
V  I  T  U  n  i  v  e  r  s  i  t  y
› ▯
```

Write a program in C to print individual characters of string in reverse order.



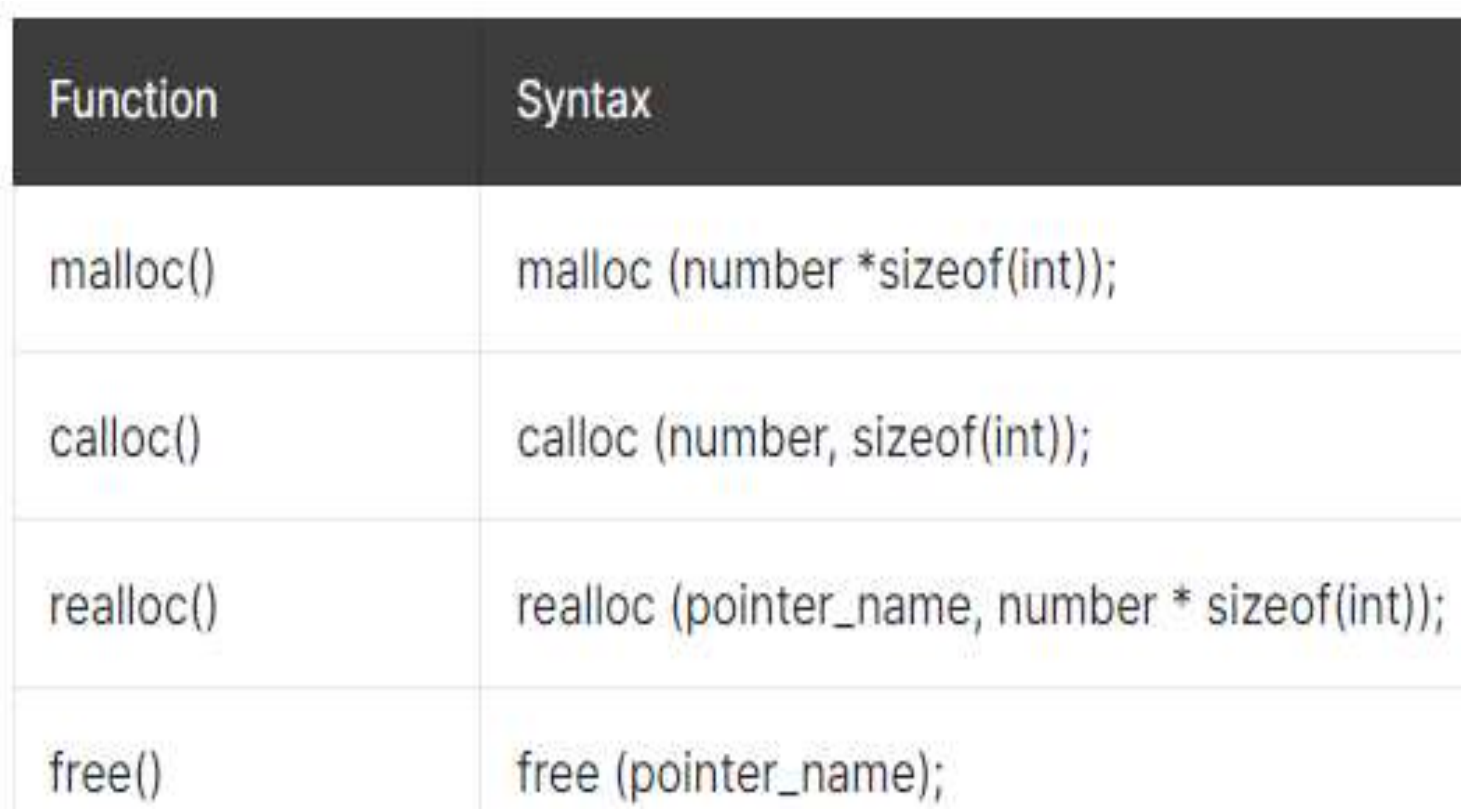
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void main()
{
    char str[100]; /* Declares a string of size 100 */
    int l=0;

       printf("\n\nPrint individual characters of string
in reverse order :\n");
       printf("-----------------------------------------
------------\n");
       printf("Input the string : ");
       fgets(str, sizeof str, stdin);
     l=strlen(str);
     printf("The characters of the string in reverse are
: \n");
       for(str[l]='\0';l>=0;l--)
        {
          printf("%c  ", str[l]);
        }
    printf("\n");
}
```

Output:-



```
› make -s
› ./main


Print individual characters of string in reverse order :
------------------------------------------------------
Input the string : VITUniversity
The characters of the string in reverse are :

  y  t  i  s  r  e  v  i  n  U  T  I  V
› 
```

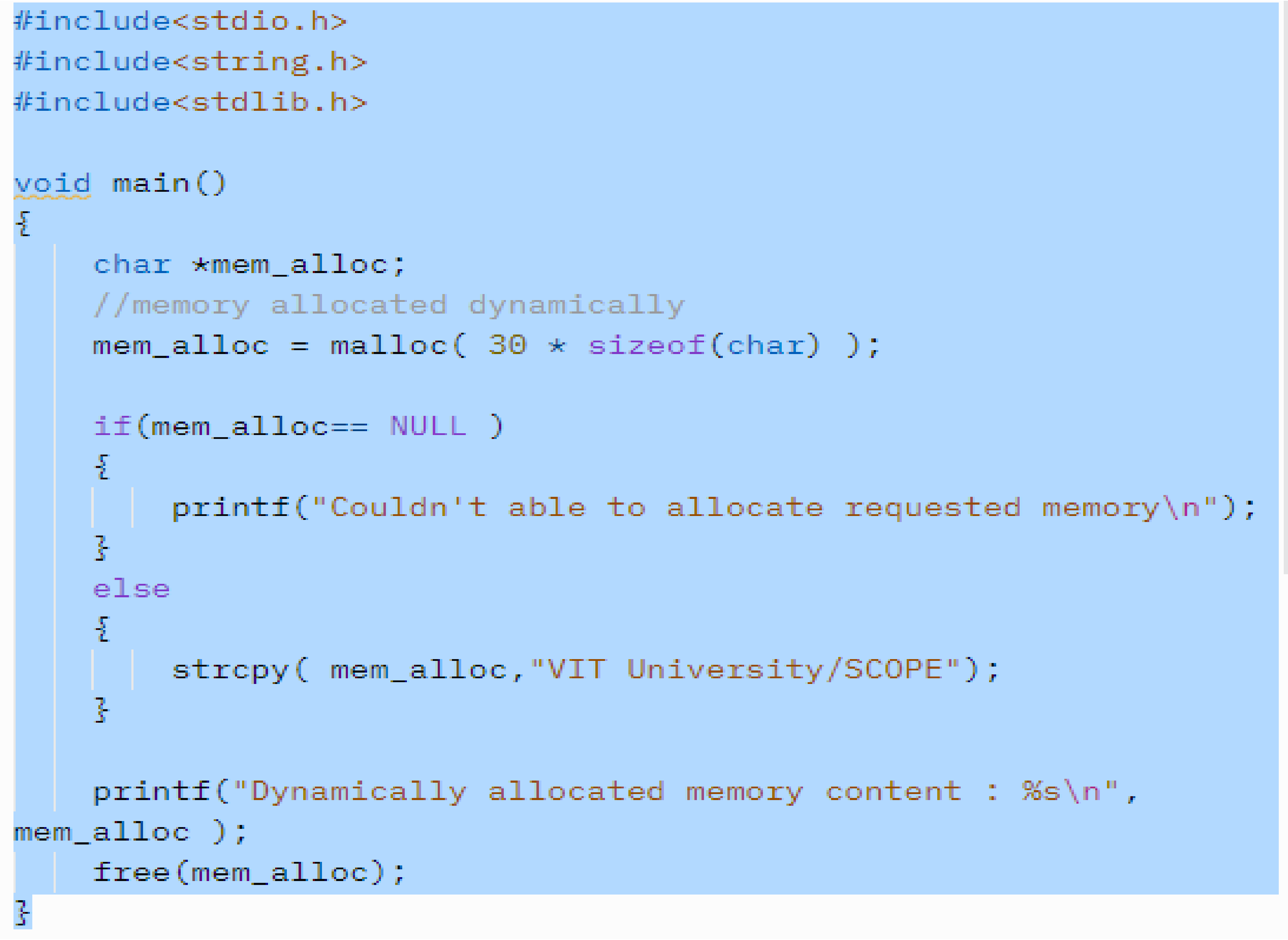
C - Dynamic memory allocation functions



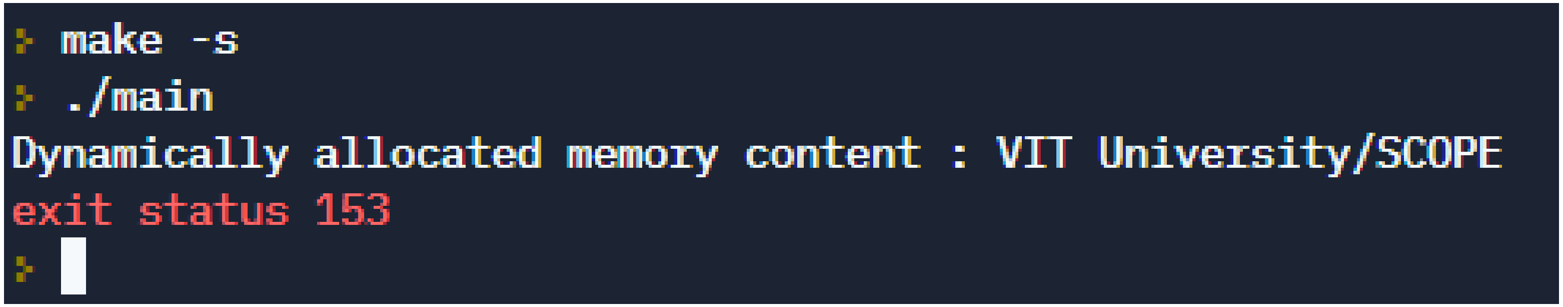
Function	Syntax
malloc()	malloc (number *sizeof(int));
calloc()	calloc (number, sizeof(int));
realloc()	realloc (pointer_name, number * sizeof(int));
free()	free (pointer_name);

malloc function

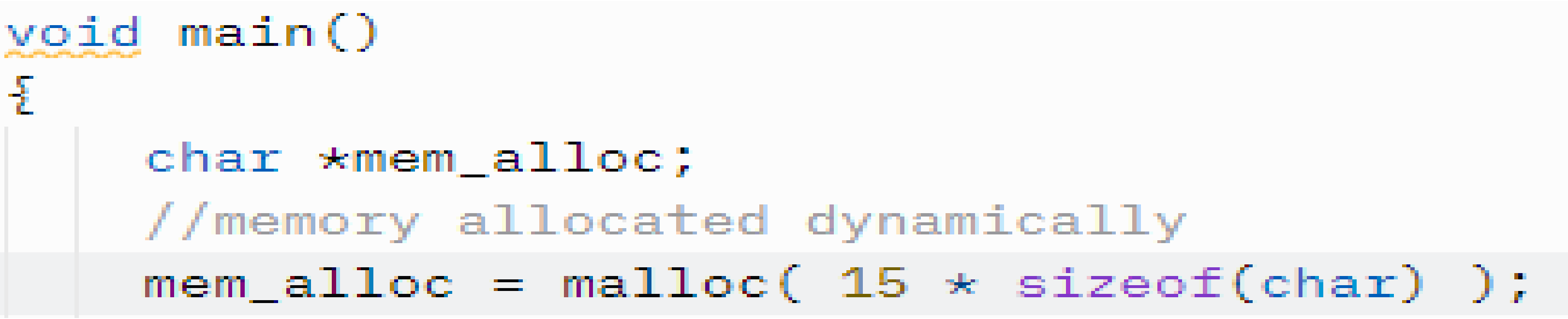
- malloc function is used to allocate space in memory during the execution of the program.
- malloc function does not initialize the memory allocated during execution. It carries garbage value.
- malloc function returns null pointer if it couldn't able to allocate requested amount of memory.



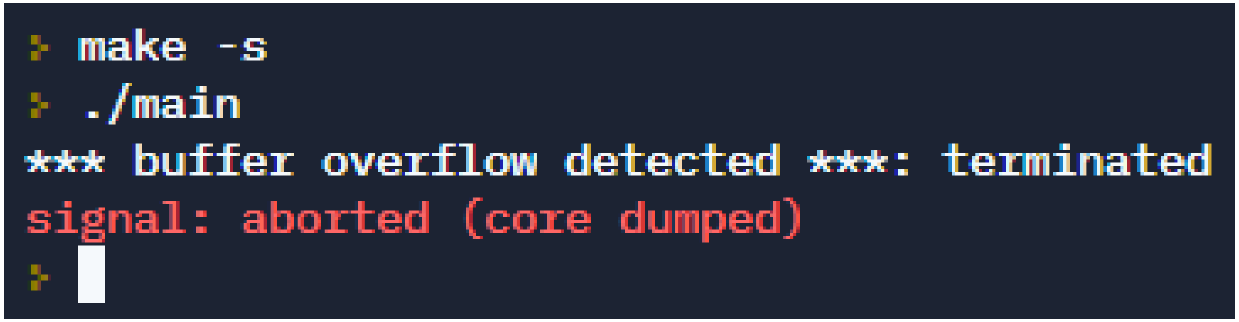
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void main()
{
    char *mem_alloc;
    //memory allocated dynamically
    mem_alloc = malloc( 30 * sizeof(char) );

    if(mem_alloc== NULL )
    {
        printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
        strcpy( mem_alloc,"VIT University/SCOPE");
    }

    printf("Dynamically allocated memory content : %s\n",
mem_alloc );
    free(mem_alloc);
}
```



```
> make -s
> ./main
Dynamically allocated memory content : VIT University/SCOPE
exit status 153
> 
```



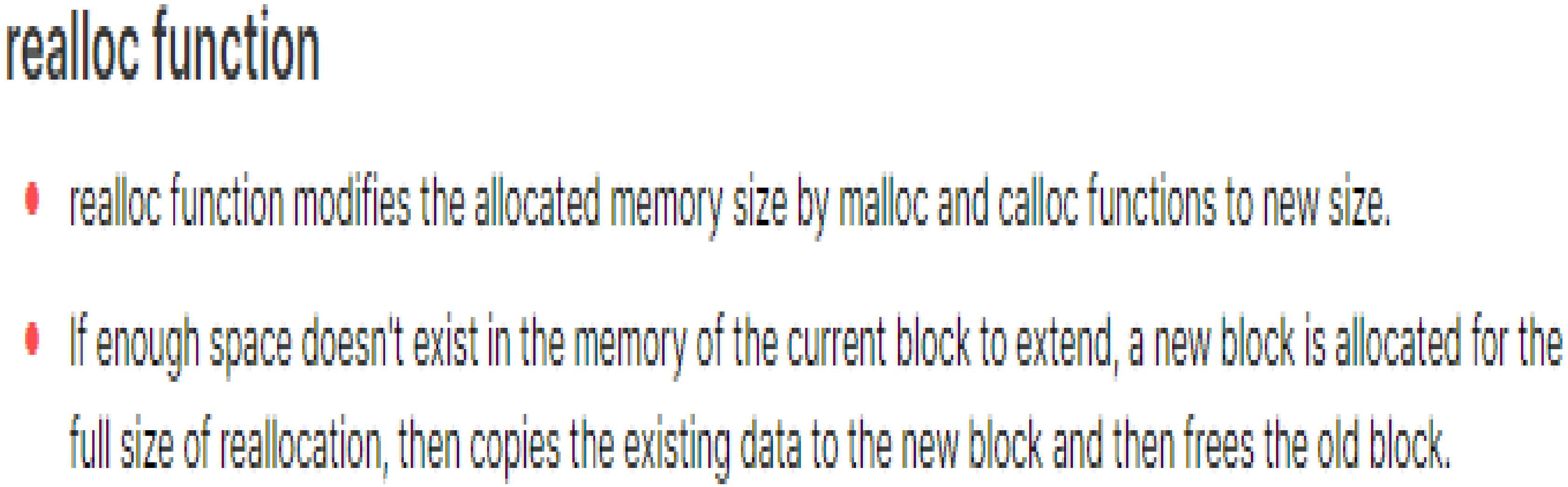
```c
void main()
{
    char *mem_alloc;
    //memory allocated dynamically
    mem_alloc = malloc( 15 * sizeof(char) );
```



```
> make -s
> ./main
*** buffer overflow detected ***: terminated
signal: aborted (core dumped)
> 
```

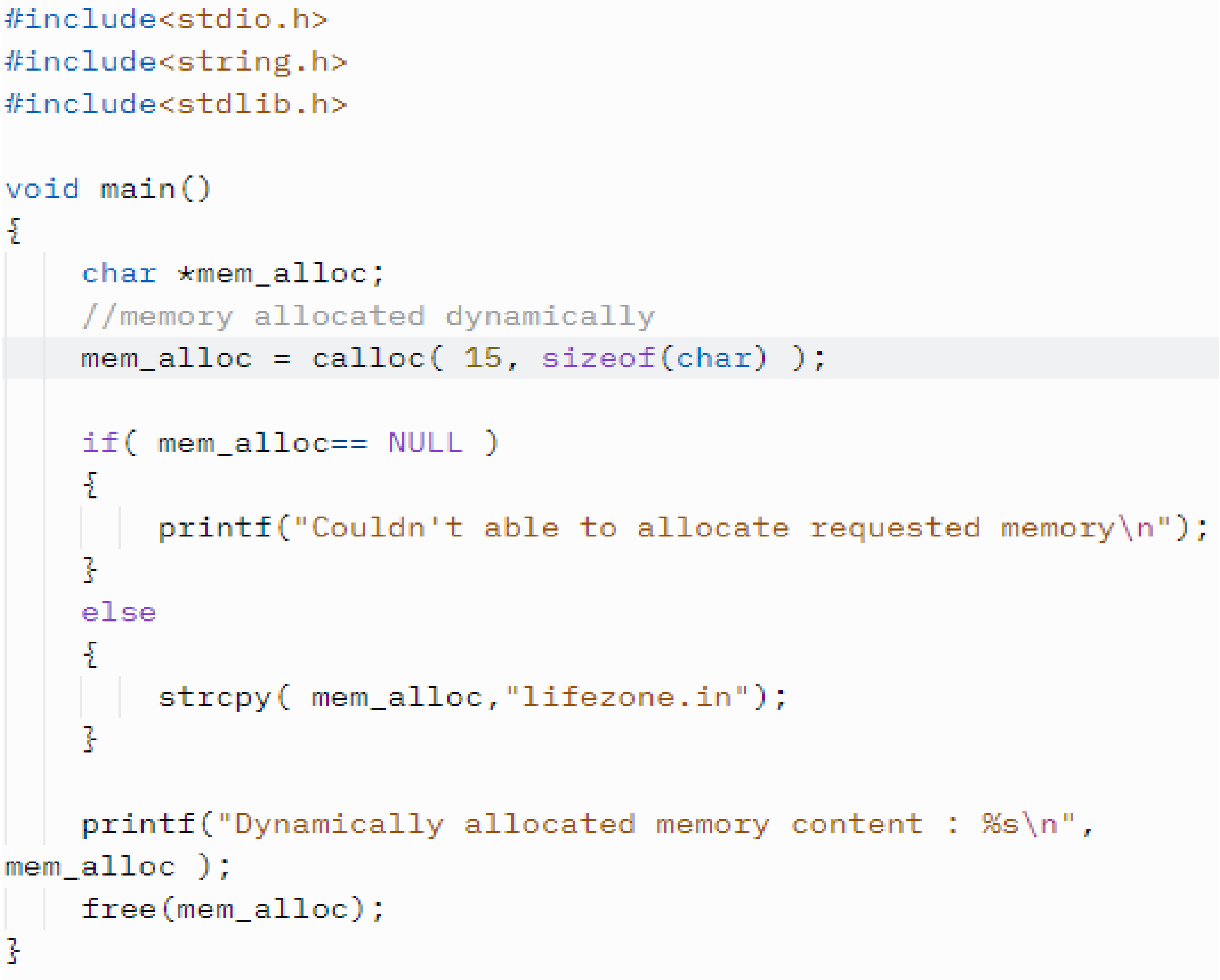
calloc function

- calloc () function and malloc () function is similar. But calloc () allocates memory for zero-initializes. However, malloc () does not.

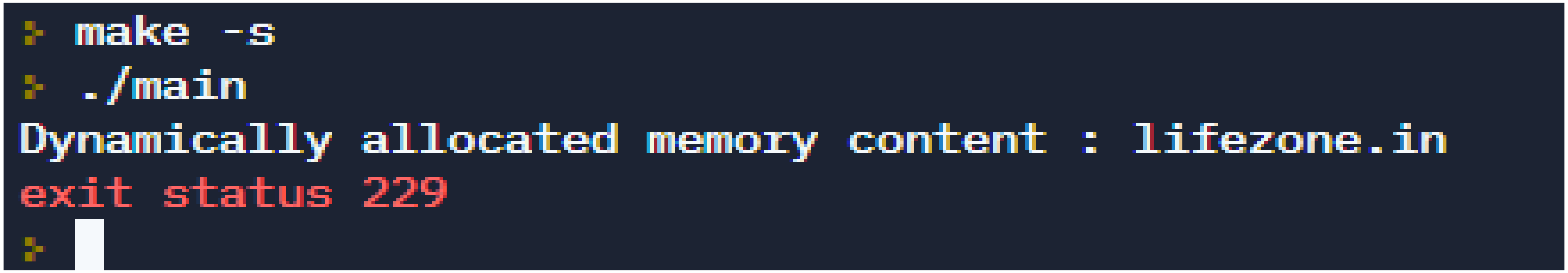


realloc function

- realloc function modifies the allocated memory size by malloc and calloc functions to new size.
- If enough space doesn't exist in the memory of the current block to extend, a new block is allocated for the full size of reallocation, then copies the existing data to the new block and then frees the old block.



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

void main()
{
    char *mem_alloc;
    //memory allocated dynamically
    mem_alloc = calloc( 15, sizeof(char) );

    if( mem_alloc== NULL )
    {
        printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
        strcpy( mem_alloc,"lifezone.in");
    }

    printf("Dynamically allocated memory content : %s\n",
mem_alloc );
    free(mem_alloc);
}
```



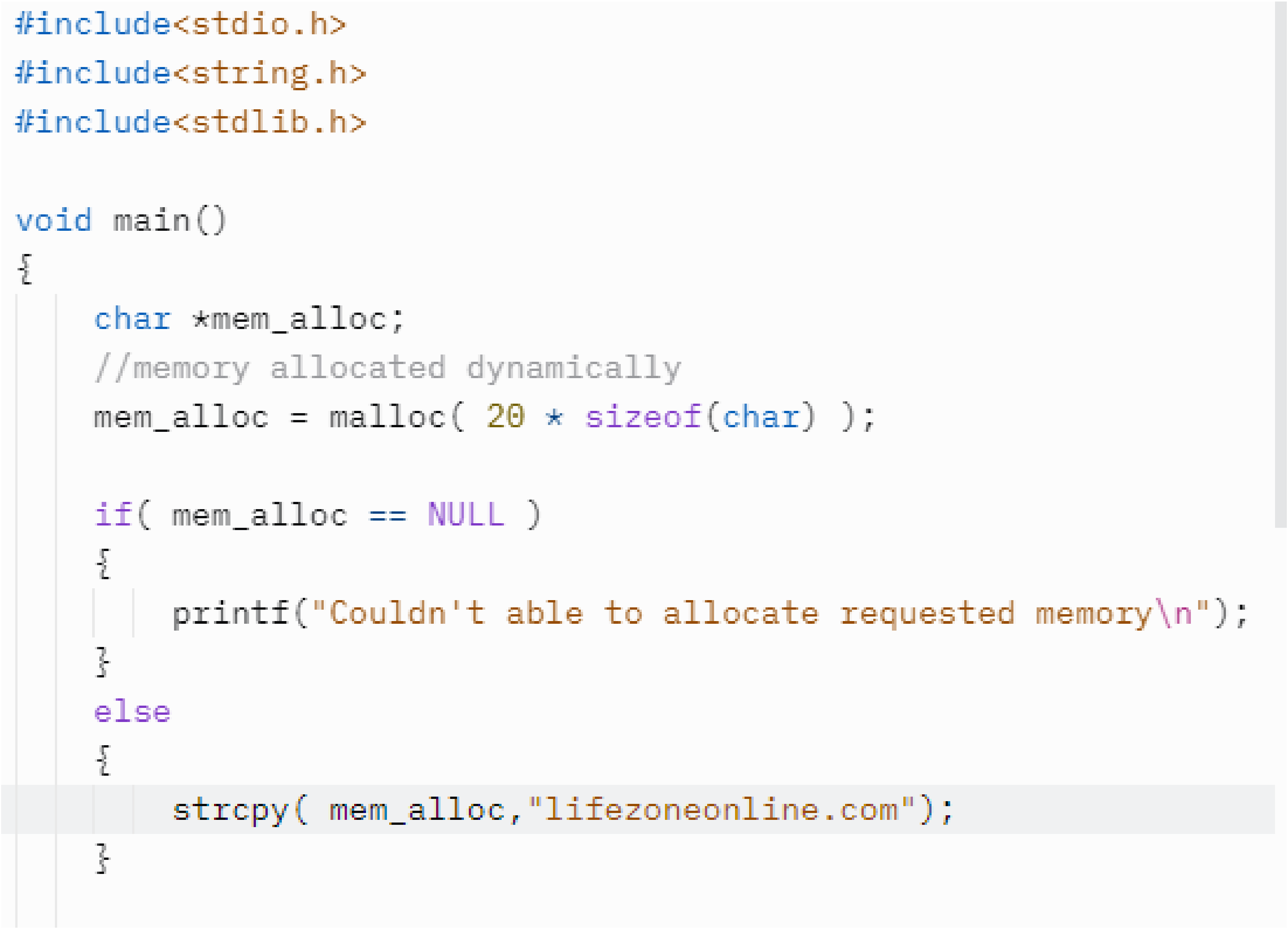
```
› make -s
› ./main
Dynamically allocated memory content : lifezone.in
exit status 229
› 
```

realloc function

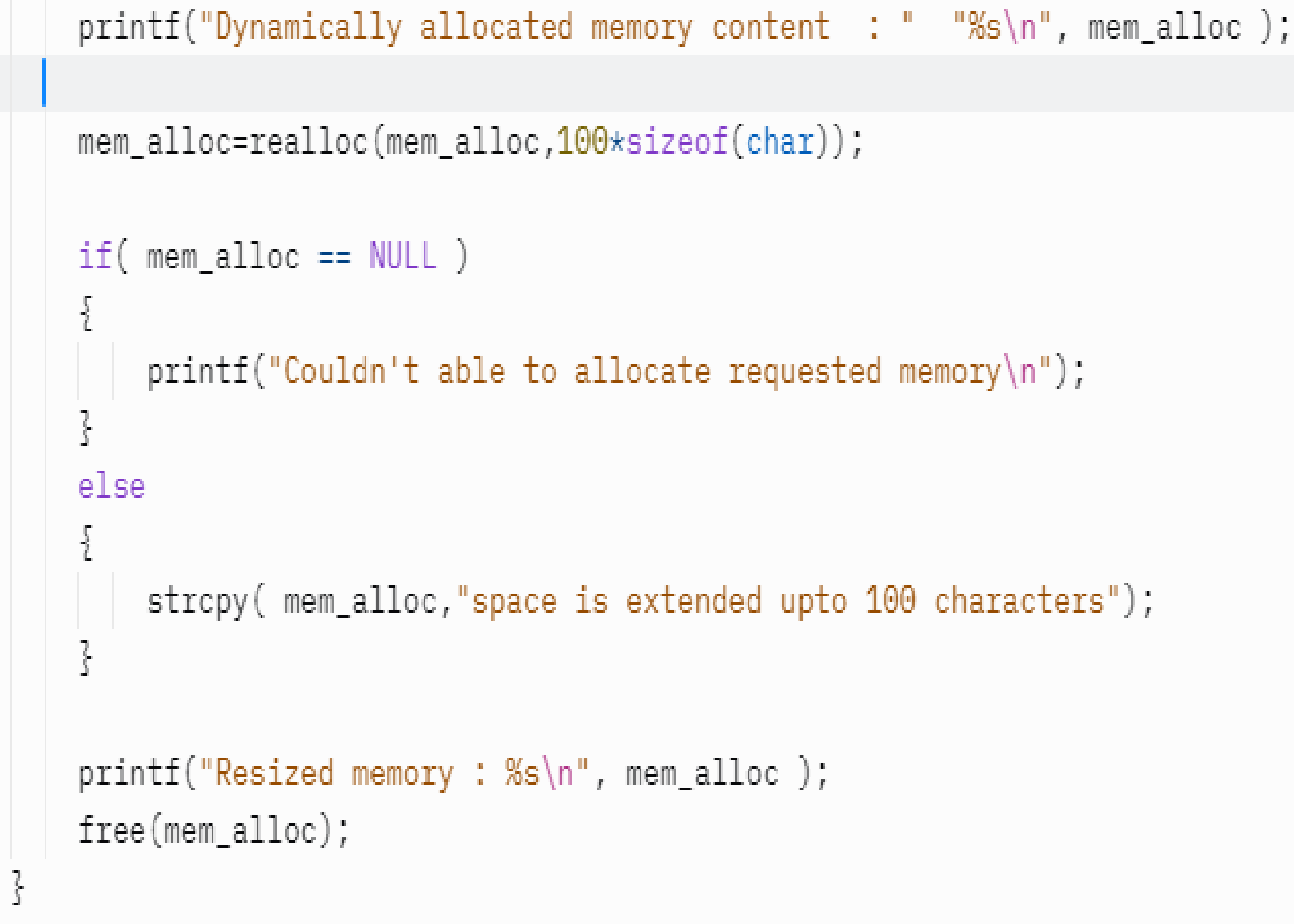
- realloc function modifies the allocated memory size by malloc and calloc functions to new size.
- If enough space doesn't exist in the memory of the current block to extend, a new block is allocated for the full size of reallocation, then copies the existing data to the new block and then frees the old block.

free function

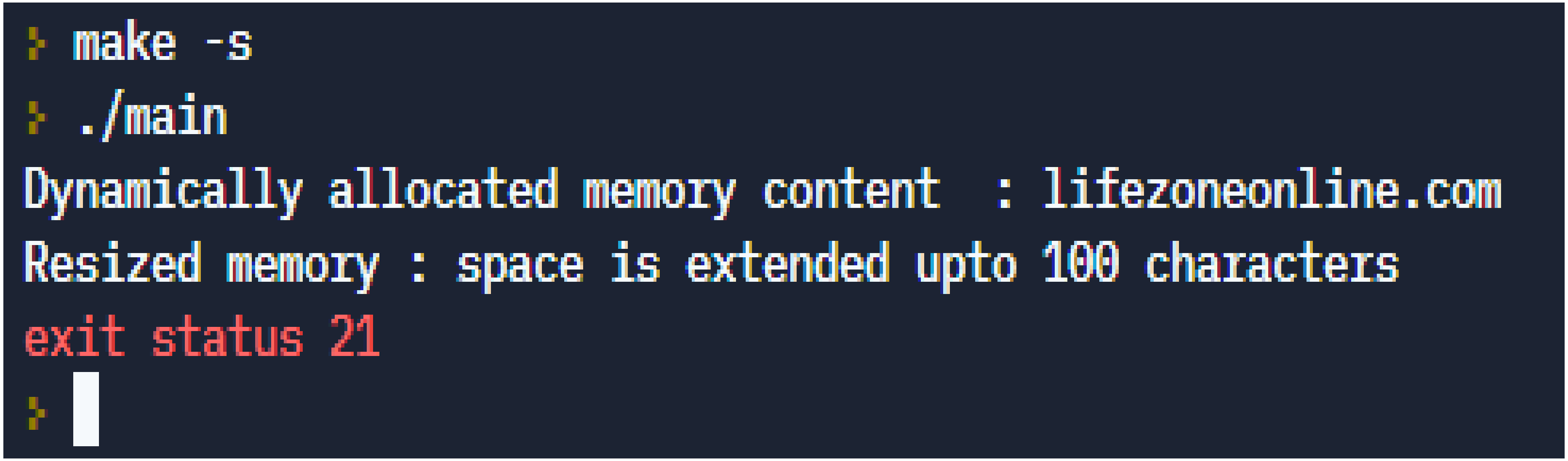
- free function frees the allocated memory by malloc (), calloc (), realloc () functions.



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

void main()
{
    char *mem_alloc;
    //memory allocated dynamically
    mem_alloc = malloc( 20 * sizeof(char) );

    if( mem_alloc == NULL )
    {
        printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
        strcpy( mem_alloc,"lifezoneonline.com");
    }

```



```
    printf("Dynamically allocated memory content  : "  "%s\n", mem_alloc );

    mem_alloc=realloc(mem_alloc,100*sizeof(char));

    if( mem_alloc == NULL )
    {
        printf("Couldn't able to allocate requested memory\n");
    }
    else
    {
        strcpy( mem_alloc,"space is extended upto 100 characters");
    }

    printf("Resized memory : %s\n", mem_alloc );
    free(mem_alloc);
}
```



```
› make -s
› ./main
Dynamically allocated memory content  : lifezoneonline.com
Resized memory : space is extended upto 100 characters
exit status 21
› 
```