Higher School of Applied Sciences of Tlemcen
1-year computer science course

Functions

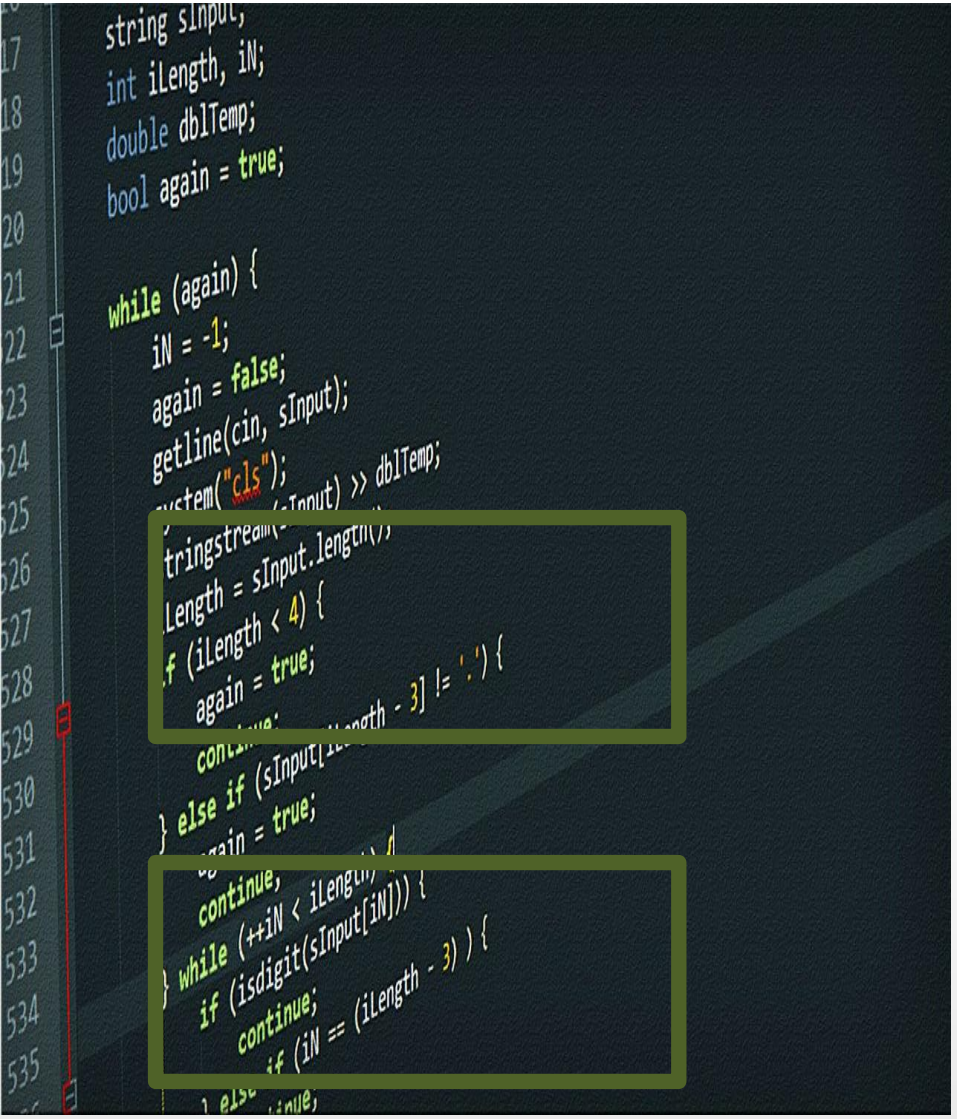
Part 1

Presented by:
Dr. Imane NEDJAR

Part 1

Overview of Functions

Problem Statement



Some programs are:

- Very long and difficult to write and understand.
- Repetitive in code

Functions are used to perform **specific actions**, and they are **reused** multiple times

Definition

A function is a block of code that only **executes when it is called**.

Predefined Functions

Example

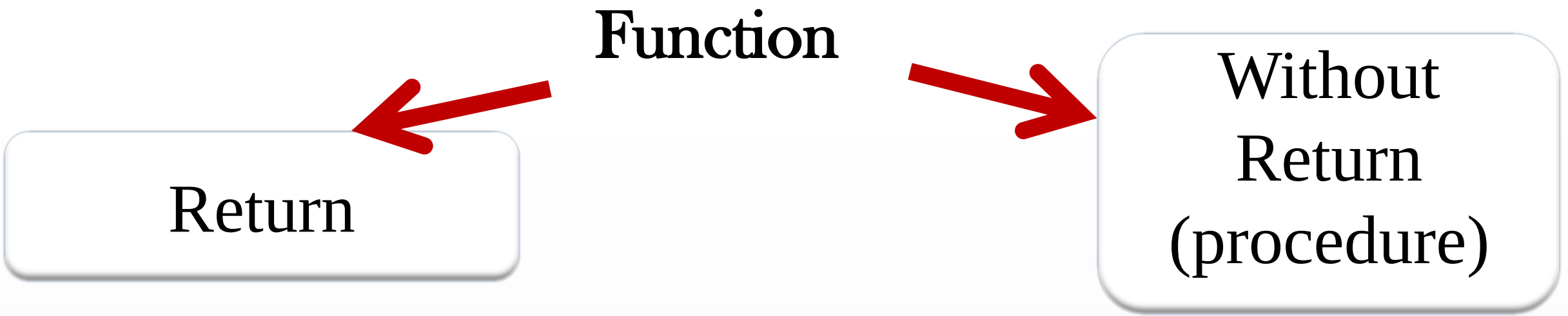
- main ()
- printf()
- scanf()
- gets()
- getchar()

Creating a Function

Specify the function name, followed by parentheses **()** and curly braces **{}**.

```
name_function ()
  {
// The code
 }
```

Type des fonctions



Return _Type name-fonction ()

void name-fonction ()

```
int  myFunction() {
 int x=3, y=4;
 return x+y;}
```

```
void myFunction() {
  printf("Hello world!");
}
```

Function Execution

*Functions will be executed when they are called.
*To call a function, write the function name followed by two parentheses () and a semicolon.

```
void myFunction() {
  printf("Hello world");
}


int main() {
  myFunction ();  // The function call
  return 0;}
```

Hello world

Function Execution

```
void myFunction() {
  printf("Hello world");
}

int main() {
  myFunction ();
  myFunction ();
  myFunction ();
   return 0;}
```

```
Hello world
Hello world
Hello world
```

Parameters and Arguments

The parameters of a function

Parameters are variables passed to functions

```
ReturnType  name_of_Function  (parameter1, parameter2, parameter3)
{
  // code
}
```

```
void  name_of_Function  (parameter1, parameter2, parameter3) {
  // code
}
```

Parameters and Arguments

The arguments of a function

```
void myFunction(char First_name[])
{
 printf("Hello %s\n", First_name);
}

int main() {
  myFunction ("Mohamed");
  myFunction ("Hanan");
  myFunction ("Yasmin");
  return 0;}
```

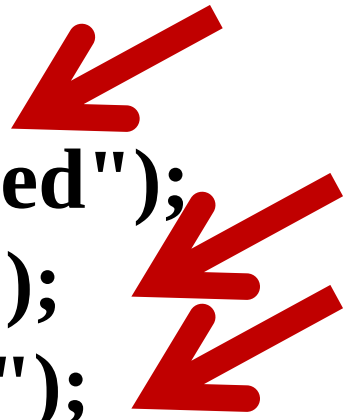
```
Hello Mohamed
Hello Hanan
Hello yasmin
```

Parameters and Arguments

Les arguments d'une fonction

```
void myFunction(char First_name [])
{
 printf("Hello %s\n", First_name);
}

int main() {
  myFunction ("Mohamed");
  myFunction ("Hanen");
  myFunction ("Yasmin");
  return 0;}
```



Parameter: First_name

Arguments: Mohamed, Hanen, yasmin

Parameters and Arguments

The multiple parameters of a function

```
void myFunction(char First_name [],int age ) {
 printf("Hello %s, you have %d \n", First_name ,age);
}
int main() {
  myFunction ("Mohamed",35);
  myFunction ("Hanan",36);
  myFunction ("Yasmin",13);
  return 0;}
```

```
Hello Mohamed you have 35
Hello Hanan you have 36
Hello yasmin you have 13
```

Parameters and Arguments

Passing arrays as function parameters

```
void myFunction(int t[5]) {
 for (int i = 0; i < 5; i++)
 {printf("%d\n", t[i]);}}
int main() {
int t [5] = {10, 20, 30, 40, 50};
myFunction (t);
 return 0;}
```

```
10
20
30
40
50
```

Functions

Return a value

```
int  myFunction(int x) {
  return 5 + x;}

int main() {
printf("Result is: %d", myFunction (3));
  return 0;
}
```

Result is: 8

Functions

Return a value

```
int  myFunction(int x,int y) {
  return y+ x;}

int main() {
printf("Result is: %d", myFunction (2,5));
  return 0;
}
```

Result is: 7

Functions

Return a value

```
int  myFunction(int x,int y) {
  return y+ x;}

int main() {
int r= myFunction(5, 3);
printf("Result is: %d", r);
  return 0;
}
```

Result is: 8

Functions

Function Declaration

```
type  nomFunction( parameter 1, parameter 2) ;
```

```
int  myFunction(int x , int y) ;
```

Functions

Function Declaration

```
int  myFunction(int x,int y) ;
int  myFunction(int x,int y) {
  return y+ x;}

int main() {
int r= myFunction(5, 3);
printf("Result is: %d", r);
  return 0;
}
```

Functions

Function Declaration

```
int  myFunction(int x,int y) ;
```

```
int main() {
int r= myFunction(5, 3);
printf("Result is: %d", r);
  return 0;
}
int  myFunction(int x,int y) {
  return y+ x;}
```

Functions

Function Declaration

```
void  myFunction() ;
int main() {
myFunction();
  return 0;
}
void  myFunction() {
 printf(" Hello");}
```

Functions

```
#include <stdio.h>
int a,b;
float d;
float div(int x, int y);
float div(int x, int y)
{
float z = x/y;
return z;}
int main(){
        scanf(" %d" ,&a);
        scanf(" %d ",&b);
        d= div(a,b);
return 0;}
```

Declaration

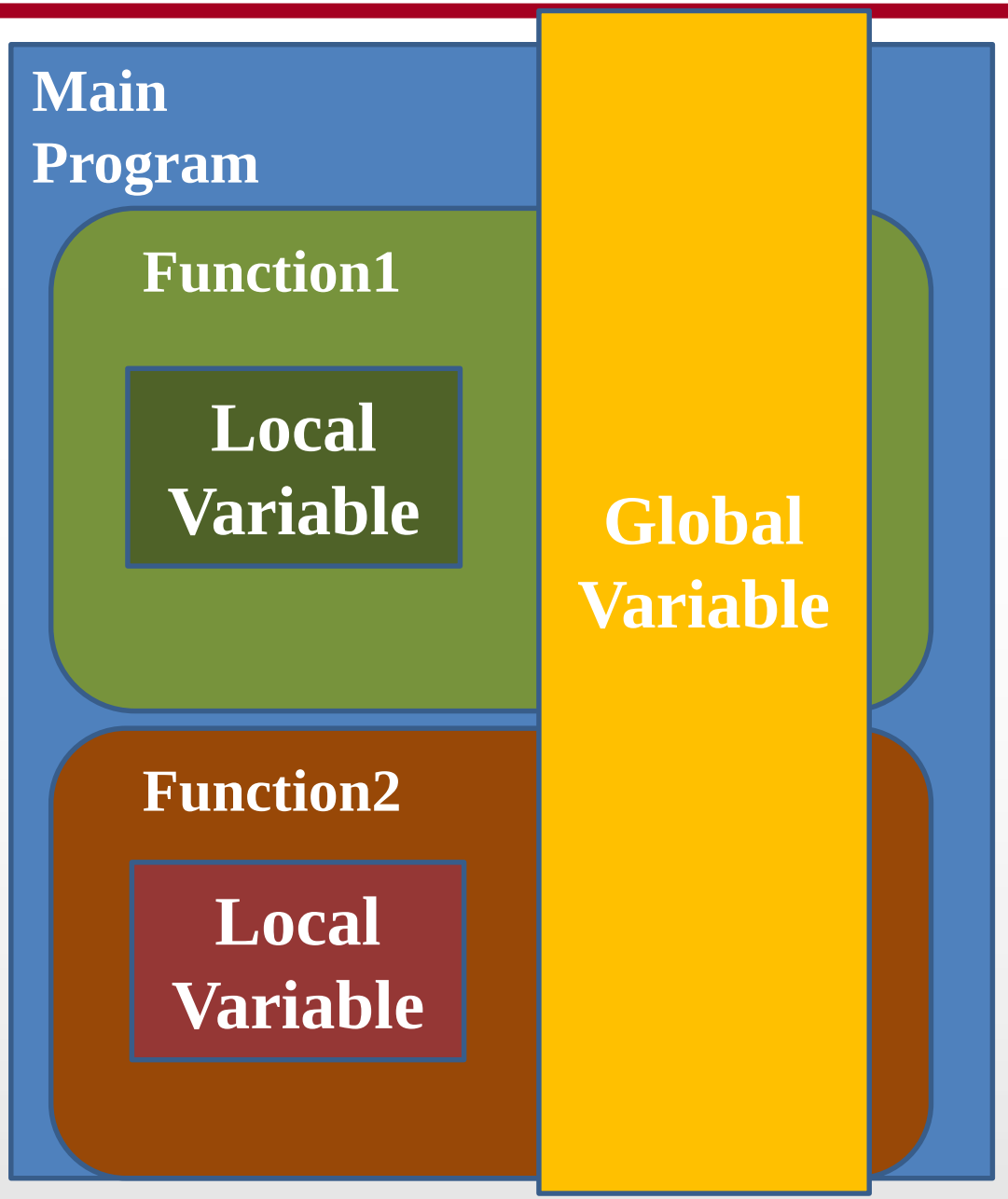
x,y: Parameters

Function Body

Function Call

a,b: Arguments

Global Variable vs. Local Variable



Global Variable vs. Local Variable

Local Variable

A **local variable** is a variable with local scope:

- Declared inside a function
- Can only be used within that function
- Destroyed at the end of the function's execution

Global Variable vs. Local Variable

Global Variable

A **global variable** is a variable with global scope:

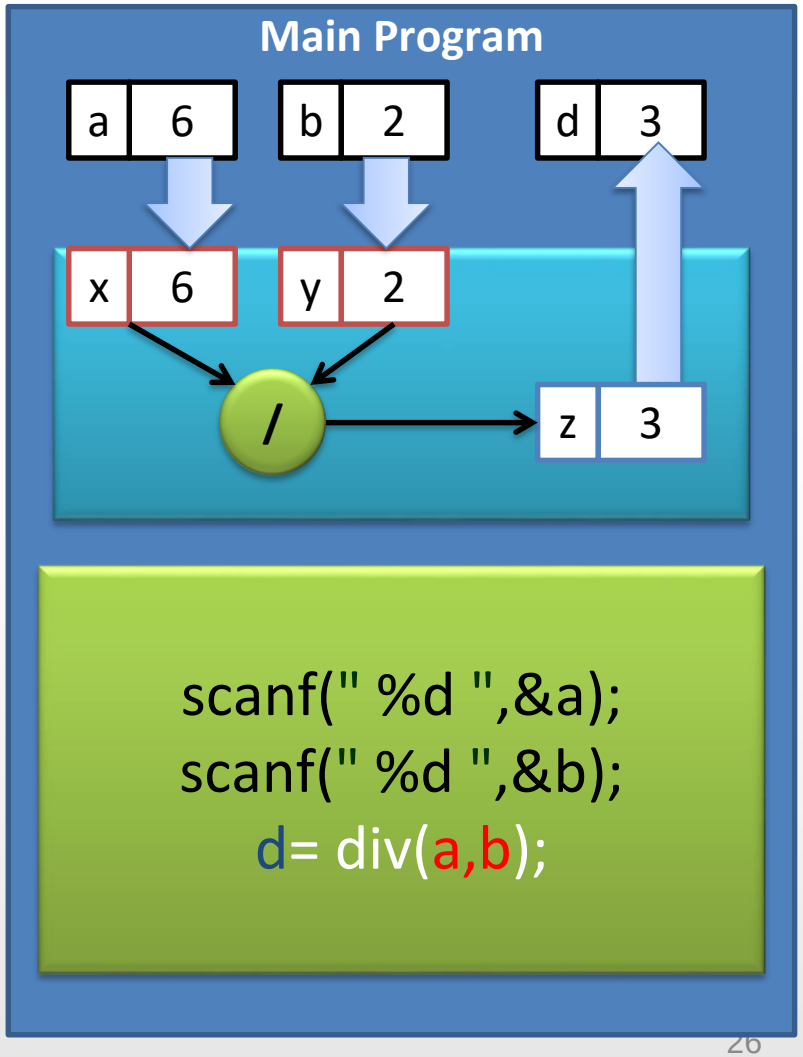
- Declared outside of functions
- Can be used by any function in the program
- Destroyed at the end of the program's execution

Global Variable vs. Local Variable

```
#include <stdio.h>
int a,b;
float d;
float div (int x, int y);
float div(int x, int y){
    float z = x/y;
return z;}
int main(){
        scanf(" %d ",&a);
        scanf(" %d ",&b);
        d= div(a,b);
return 0;}
```

Globals Variables

Locals Variables



26

Functions

```
#include<stdio.h>
int tab1[50],N1;
int tab2[50],N2;
int k;
void Display_Array (int t[],int N)
{
int i;
for(i=0;i<N;i++){
    printf("%d\n",i+1,t[i]);
    }
}
```

```
int main(){
    //Read tab1
printf("Provide the size of Array 1 \n ");
scanf("%d",&N1);
for(k=0;k<N1;k++){
    printf("tab1[%d]: ",k);
    scanf("%d",&tab1[k]);}
    //Read tab2
printf(" Provide the size of Array 2 \n ");
scanf("%d",&N2);
for(k=0;k<N2;k++){
    printf("tab2[%d]: ",k);
    scanf("%d",&tab2[k]);}
Display_Array(tab1,N1);
Display_Array(tab2,N2);
return 0 ;}
```

Global Variable vs. Local Variable

	Tab1	Tab2	N1	N2	K	T	N	i
Local								
Global								
main								
Display_Array								

	Tab1	Tab2	N1	N2	K	T	N	i
Local						x	x	x
Global	x	x	x	x	x			
main	x	x	x	x	x			
Display_Array	x	x	x	x	x	x	x	x

Predefined Functions

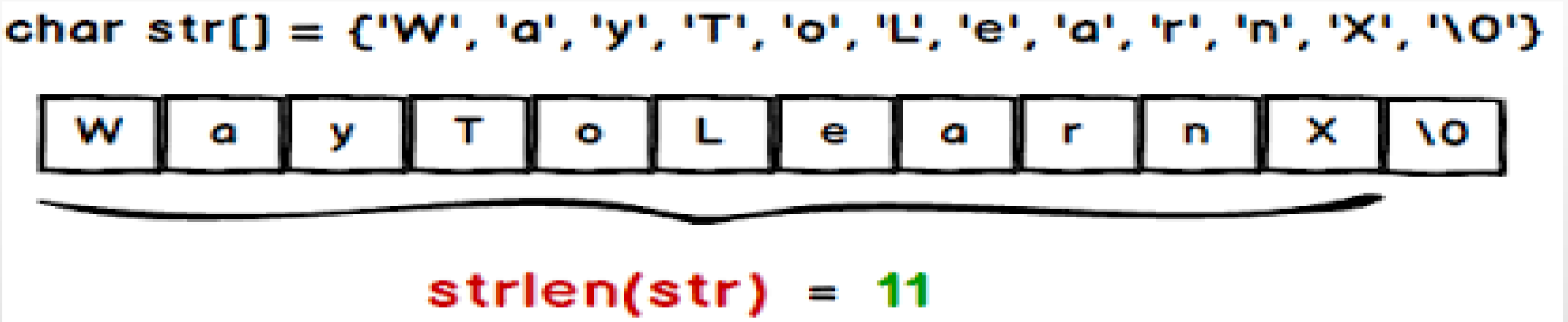
String

#include <string.h>

int strlen (char s1[]);

Return :

- The size of the string



Predefined Functions

String

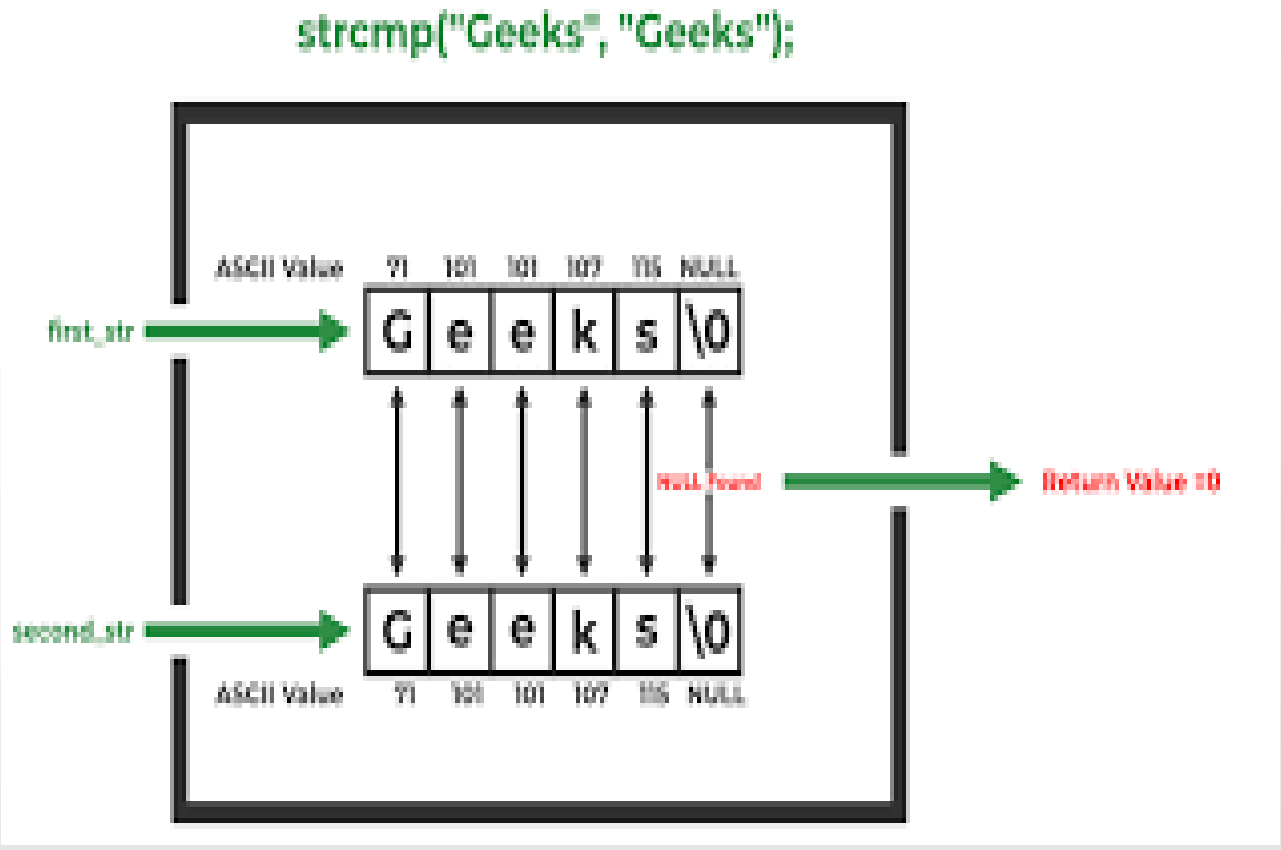
int strcmp (char s1[], char s2[]);

Or

int strcmp(char *s1, char *s2);

Return :

- **Value =0 if s1= s2**
- **Value !=0 if s1!= s2**

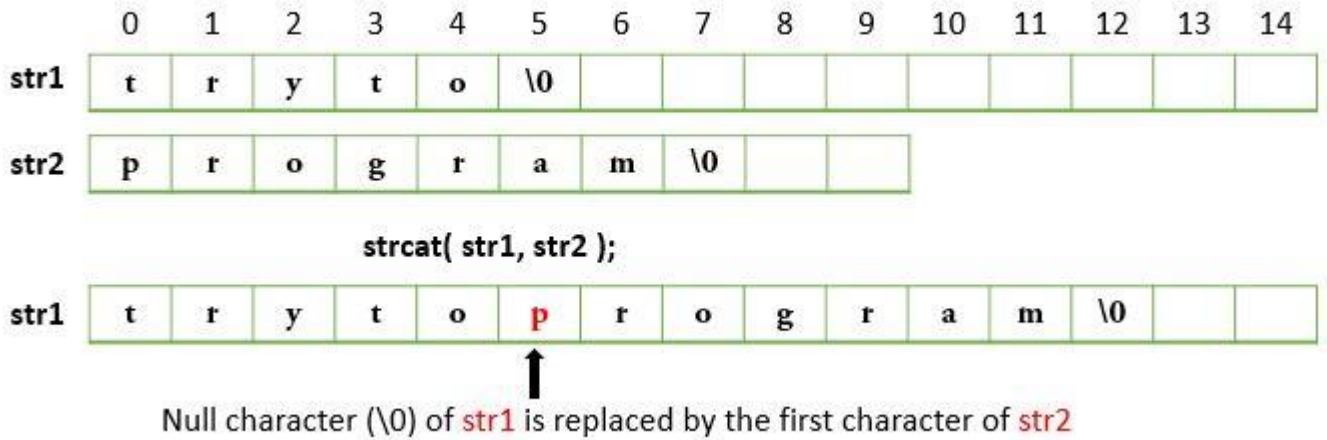


Predefined Functions

String

strcat (char s1[], char s2[]);

- To append **S2** at the end of **S1**

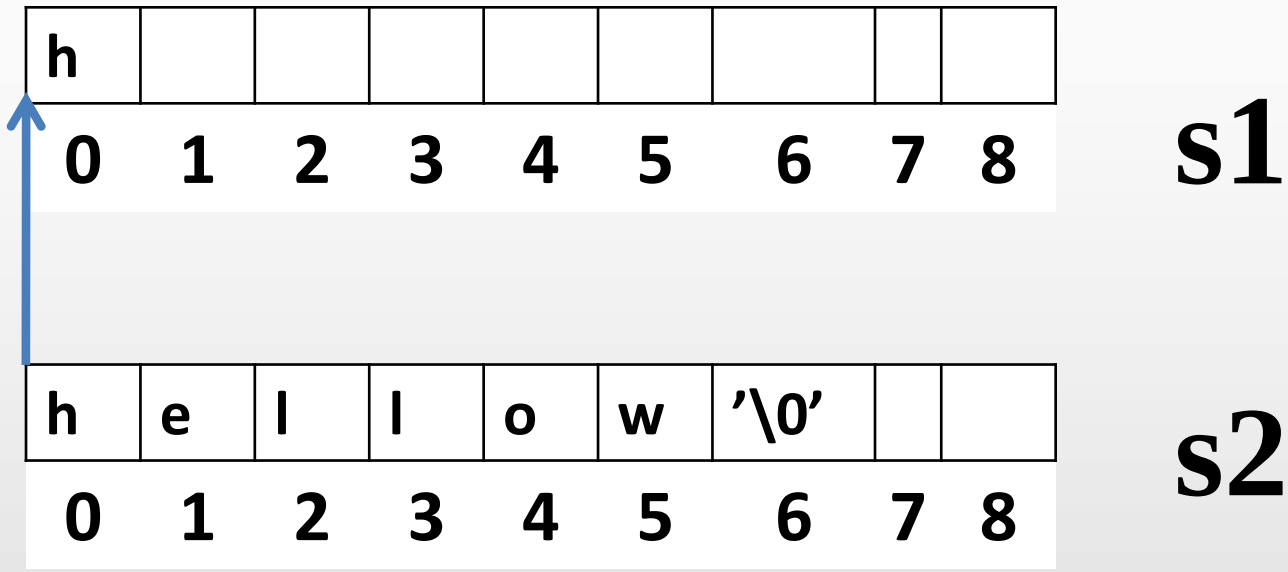


Predefined Functions

String

strcpy (char s1[], **char** s2[]**);**

- Assigns **S1** to **S2**



Predefined Functions

String

```
#include<stdio.h>
#include<string.h>
int main()
{
    char txt1[20],txt2[20];
    int x;
    int y;
    printf("Provide Text 1");
    gets(txt1);
    printf("Provide Text 2\n");
    gets(txt2);
x=strlen(txt1);
 printf(« The size of string 1 is %d\n",x);
y=strcmp(txt1, txt2);
if (y==0)
        printf("txt1=txt2");
    else printf("txt1!=txt2");
 return 0 ;}
```

Predefined Functions

String

```
#include<stdio.h>
#include<string.h>
int main()
{
    char name [20],first_name [20];
    printf("Provide your name \n");
    gets(name);
    printf("Provide your first name\n");
    gets(first_name);
    strcat(name, first_name);
    printf(" Your full name is %s \n",name);
    return 0 ;}
```

Predefined Functions

Mathematical Functions

#include <math.h>

float **sqrt**(**float x**)

float **pow**(**float x, float y**)

float **ceil** (**float x**)

float **floor** (**float x**)

Predefined Functions

Mathematical Functions

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

int main()
{
    printf("%.2f\n",sqrt(64));
    printf("%.2f\n",pow(2,4));
    printf("%.2f\n", ceil(1.4));
    printf("%.2f\n", floor(1.4));
    return 0 ;
}
```

```
8.00
16.00
2.00
1.00
```

Exercices

Exercice 1

Create a function to compute the **factorial** of a number. This function takes the number as an input parameter and returns its factorial.

Exercice 2

Create a function to compute the sum of elements in an integer array. This function accepts the array as an input parameter and returns the sum.

Exercices

Exercice 3

Create a function to concatenate two strings. This function takes both strings as input parameters and displays the concatenated string.