Computer Practicum 1

Introduction to C

Vida Groznik

Source: cprogramming.com, tutorialspoint.com, learn-c.org, fresh2refresh.com

Functions

C functions aspects	Syntax
Function definition	`return_type function_name (arguments list) { body of function; }`
Function call	`function_name (arguments list);`
Function declaration	`return_type function_name (argument list);`

Functions should be declared and defined **before** calling in a C program.

Functions - example

```c
#include<stdio.h>
// function prototype, also called function declaration
float square ( float x );
// main function, programme starts from here

int main( )
{
    float m, n ;
    printf ( "\nEnter some number for finding square \n");
    scanf ( "%f", &m ) ;
    // function call
    n = square ( m ) ;
    printf ( "\nSquare of the given number %f is %f",m,n );
}

float square ( float x )   // function definition
{
    float p ;
    p = x * x ;
    return ( p ) ;
}
```

Function arguments and return values

All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function.

C functions aspects	Syntax

With argumets and return values	**function declaration:**
	`int function ( int );`
	function call: `function ( a );`
	function definition:
	`int function( int a )`
	`{`
	`    statements;`
	`    return a;`
	`}`

With argumets and without return values	**function declaration:**
	`void function ( int );`
	function call: `function( a );`
	function definition:
	`void function( int a )`
	`{`
	`    statements;`
	`}`

Without arguments and without return values	**function declaration:**
	`void function();`
	function call: `function();`
	function definition:
	`void function()`
	`{`
	`    statements;`
	`}`

Without arguments and with return values	**function declaration:**
	`int function ( );`
	function call: `function ( );`
	function definition:
	`int function( )`
	`{`
	`    statements;`
	`    return a;`
	`}`

Function return values

- Only **one** value can be returned from a function.
- If you try to return more than one value from a function, only the value that appears at the **right most place** of the return statement will be returned.
- For example, if you use “`return a,b,c`” in your function, only the value for `c` will be returned and values `a, b` won’t be returned to the program.
- In case, if you want to return more than one value, pointers can be used to directly change the values in address instead of returning those values to the function.

Pointers

- Pointer in C language is a variable that **stores/points the address** of another variable.
- A pointer in C is used to **allocate memory dynamically** i.e. at run time. The pointer variable might be belonging to any of the data type such as `int, float, char, double, short` etc.
- Pointer Syntax : `data_type *var_name;`

  Example: `int *p;` `char *p;`
- Where, * is used to denote that "p" is a pointer variable and not a normal variable.
- The content of the C pointer is a whole number i.e. address.
- C pointer is always initialized to null (`int *p = null`).
- The value of null pointer is `0`.
- & symbol is used to **get the address** of the variable.
- * symbol is used to **get the value** of the variable that the pointer is pointing to.
- If a pointer in C is assigned to NULL, it means it is pointing to nothing.
- Two pointers can be subtracted to know how many elements are available between these two pointers. Pointer addition, multiplication, division are not allowed.
- The size of any pointer is 2 byte (for 16 bit compiler).

Pointers - example

```c
#include <stdio.h>
int main()
{
   int *ptr, q;
   q = 50;
   /* address of q is assigned to ptr */
   ptr = &q;
   /* display q's value using ptr variable */
   printf("%d", *ptr);
   return 0;
}
```

Output:

```
50
```

Exercise 1

Write a program in C to add two numbers using pointers.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
The sum of the entered numbers is : 11
```

Exercise 1 - solution

Write a program in C to add two numbers using pointers.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
The sum of the entered numbers is : 11
```

```c
#include <stdio.h>
int main()
{
    int fno, sno, *ptr, *qtr, sum;
    printf("\n\n Pointer : Add two numbers :\n");
    printf("--------------------------------\n");
    printf(" Input the first number : ");
    scanf("%d", &fno);
    printf(" Input the second number : ");
    scanf("%d", &sno);
    ptr = &fno;
    qtr = &sno;
    sum = *ptr + *qtr;
    printf(" The sum of the entered numbers is : %d\n\n",sum);
    return 0;
}
```

Exercise 2

Write a program in C to add numbers using call by reference.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
The sum of 5 and 6  is 11
```

Exercise 2 - solution

Write a program in C to add numbers using call by reference.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
The sum of 5 and 6  is 11
```

```c
#include <stdio.h>
long addTwoNumbers(long *, long *);
int main()
{
    long fno, sno, *ptr, *qtr, sum;
    printf("\n\n Pointer : Add two numbers using
    call by reference:\n");

    printf("--------------------------------\n");
    printf(" Input the first number : ");
    scanf("%ld", &fno);
    printf(" Input the second number : ");
    scanf("%ld", &sno);

    sum = addTwoNumbers(&fno, &sno);
    printf(" The sum of %ld and %ld is %ld\n\n",
    fno, sno, sum);
    return 0;
}

long addTwoNumbers(long *n1, long *n2)
{
    long sum;
    sum = *n1 + *n2;
    return sum;
}
```

Exercise 3

Write a program in C to find the maximum number between two numbers using a pointer.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
6 is the maximum number.
```

Exercise 3 - solution

Write a program in C to find the maximum number between two numbers using a pointer.

Test Data :

```
Input the first number : 5
Input the second number : 6
```

Expected Output :

```
6 is the maximum number.
```

```c
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int fno,sno,*ptr1=&fno,*ptr2=&sno;
    printf("\n\n Pointer : Find the maximum number between
    two numbers :\n");
    printf("--------------------------------------------\n");
    printf(" Input the first number : ");
    scanf("%d", ptr1);
    printf(" Input the second number : ");
    scanf("%d", ptr2);

    if(*ptr1>*ptr2)
    {
        printf("\n\n %d is the maximum number.\n\n",*ptr1);
    }
    else
    {
        printf("\n\n %d is the maximum number.\n\n",*ptr2);
    }
}
```

Exercise 4

Write a program in C to store n elements in an array and print the elements using pointer.

Test Data :

```
Input the number of elements to store in the array :5
Input 5 number of elements in the array :
element - 0 : 5
element - 1 : 7
element - 2 : 2
element - 3 : 9
element - 4 : 8
```

Expected Output :

```
 The elements you entered are :
 element - 0 : 5
 element - 1 : 7
 element - 2 : 2
 element - 3 : 9
 element - 4 : 8
```

Exercise 4 - solution

Write a program in C to store n elements in an array and print the elements using pointer.

Test Data :

```
Input the number of elements to store in the
array :5
Input 5 number of elements in the array :
element - 0 : 5
element - 1 : 7
element - 2 : 2
element - 3 : 9
element - 4 : 8
```

Expected Output :

```
 The elements you entered are :
 element - 0 : 5
 element - 1 : 7
 element - 2 : 2
 element - 3 : 9
 element - 4 : 8
```

```c
#include <stdio.h>
int main()
{
    int arr1[25], i,n;
    printf("\n\n Pointer : Store and retrieve elements
    from an array :\n");
    printf("----------------------------------------\n");
    printf(" Input the number of elements to store in the
    array :");
    scanf("%d",&n);
    printf(" Input %d number of elements in the array:
    \n",n);

    for(i=0;i<n;i++)
    {
        printf(" element - %d : ",i);
        scanf("%d",arr1+i);
    }

    printf(" The elements you entered are : \n");

    for(i=0;i<n;i++)
    {
        printf(" element - %d : %d \n",i,*(arr1+i));
    }
    return 0;
}
```

Exercise 5

Write a program in C to find the factorial of a given number using pointers.

Test Data :

```
Input a number : 5
```

Expected Output :

```
The Factorial of 5 is : 120
```

Exercise 5 - solution

Write a program in C to find the factorial of a given number using pointers.

Test Data :

```
Input a number : 5
```

Expected Output :

```
The Factorial of 5 is : 120
```

```c
#include <stdio.h>
void findFact(int,int*);
int main()
{
    int fact;
    int num1;
    printf("\n\n Pointer : Find the factorial of a given number:");
    printf("----------------------------------------------\n");
    printf(" Input a number : ");
    scanf("%d",&num1);

    findFact(num1,&fact);
    printf(" The Factorial of %d is : %d \n\n",num1,fact);
    return 0;
}

void findFact(int n,int *f)
{
    int i; *f =1;
    for(i=1;i<=n;i++)
    {
        *f=*f*i;
    }
}
```

Exercise 6

Write a program in C to compute the sum of all elements in an array using pointers.

Test Data :

```
Input the number of elements to store in the array (max 10) : 5
Input 5 number of elements in the array :
element - 1 : 2
element - 2 : 3
element - 3 : 4
element - 4 : 5
element - 5 : 6
```

Expected Output :

```
The sum of array is : 20
```

Exercise 6 - solution

Write a program in C to compute the sum of all elements in an array using pointers.

Test Data :

```
Input the number of elements to
store in the array (max 10) : 5

Input 5 number of elements in
the array :

element - 1 : 2

element - 2 : 3

element - 3 : 4

element - 4 : 5

element - 5 : 6
```

Expected Output :

```
The sum of array is : 20
```

```c
#include <stdio.h>
void main()
{
    int arr1[10];
    int i,n, sum = 0;
    int *pt;
    printf("\n\n Pointer : Sum of all elements in an array :\n");
    printf("--------------------------------------------\n");
    printf(" Input the no. of elements to store in the array(max 10): ");
    scanf("%d",&n);

    printf(" Input %d number of elements in the array : \n",n);

    for(i=0;i<n;i++)
    {
        printf(" element - %d : ",i+1);
        scanf("%d",&arr1[i]);
    }

    pt = arr1;
    // pt store the base address of array arr1
    for (i = 0; i < n; i++)
    {
        sum = sum + *pt;
        pt++;
    }
    printf(" The sum of array is : %d\n\n", sum);
}
```