M03S01 [Recursive Functions and Pointers] Practice Drill

CS110: Computing Lab
Department of CSE, IIT Guwahati
Jan-May 2018

Objective of the Stage

Primary objective of this drill is to make you practice with recursive functions and pointers. In the last stage of the Module 3, we have focused on recursive functions. In the first stage of the Module 4, we have focused on pointers. It also contains practice and assessment problems on recursive functions and pointers.

Reference Lecture Slides

- 6th Feb and 7th Feb Lectures [Pointers]
- 13th March and 14th March Lectures [Pointers]

Learning Goal

After doing this practice drill and working on assessment problems of this stage, we expect you to be comfortable with

- Basic understanding of recursive functions
- Writing recursive functions
- Basic understanding of pointers
- Declaration and arithmetic of pointers
- Accessing individual array elements and performing some basic operations on arrays and strings using pointers
- Applying sorting techniques on data stored in an array using pointers

Recursive Function

A recursive function is a function that calls itself.

```
int func()
{
    .........
    func(..);
    return 0;
}
```

Every recursive function comprises of two essential steps known as base case and recursive step. At some point, the recursive function must encounters a subtask that it can perform without calling itself. This case, where the function does not recur, is called the base case. A recursive method solves a problem by calling a copy of itself to work on a smaller problem. This step is called recursion step. General format of a recursive function is shown below.

```
int func()
{
    if (testfor a basecase)
      return basecase value
    else
      some work
      recursive step
      some work
}
```

Examples

Now consider the following code which use recursion to print numbers from n to 1.

```
#include <stdio.h>
int print(int k)
{
    if (k==0)
      return 0;
    else
    {
      printf("%d",k);
      return print(k-1);
     }
}
int main()
{
    int n;
    printf("Enter a number");
    scanf("%d",&n);
```

```
    print(n);
    return 0;
}
```

Computing factorial of a number using recursion.

```
#include <stdio.h>
int factorial(int k)
{
    if (k==1 || k==0)
      return 1;
    else
      return k*factorial(k-1);
}
int main()
{
    int n,f;
    printf("Enter a number");
    scanf("%d",&n);
    f=factorial(n);
    printf("Factorial of number %d is %d \n",n,f);
    return 0;
}
```

Important things to remember:

- If we get infinite recursion, the program may run out of memory.
- It is important to ensure that the recursive function must terminates.
- Terminates when a base case is reached.

Exercise problems

1. Write a program in C to compute n^{th} Fibonacci number. Your program should read the input (n) from user during runtime.
2. Write a program in C to compute GCD (a,b). Your program should read the inputs (a,b) from user during runtime.
3. Write a program in C to compute n^{th} power of a number using recursion. Your program should read the inputs (number , n) from user during runtime.
4. Write a program in C to to find whether a Number is Prime or Not, using Recursion.
5. Write a program in C to to check whether a string is palindrome or not, using recursion.
6. Write a program in C to to to find sum of all digits using recursion.

7. Write a program in C to to to compute Ackerman's function using recursion. The Ackermann function A(x,y) is defined for integer x and y by

$$A(x,y) = \begin{cases} y+1, & \text{if } x = 0 \\ A(x-1,1), & \text{if } y = 0 \\ A(x-1, A(x,y-1)), & \text{otherwise} \end{cases} \quad (1)$$

Pointers

Pointer is a variable which contains memory address of another variable. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

& and * operators

To access address of a variable, we use the unary operator & (ampersand) that returns the address of that variable. For example, & a in the following example gives us address of variable a.

```
#include<stdio.h>
int main ( )
{
   int a = 5;
   printf("value of a = %d\n",a);
   printf("address of a = %p\n",&a);
   return 0;
}
```

The indirection or dereference operator * gives the value at the specified address.

```
#include<stdio.h>
int main ( )
{

   int a = 5;
   printf("value of a = %d\n",a);
   printf("address of a = %p\n",&a);
   printf("value at address %p = %d\n",&a,*(&a));
   return 0;
}
```

Declaring a pointer

When a pointer variable is declared then an asterisk (*) symbol should precede the variable name. Consider the following three statements:

int a=5;

int *b;
b=&a;

We are declaring a normal variable in first step. In step two, we are declaring a pointer variable of type int. In step three, we are assigning the address of variable 'a' in the pointer variable 'b'. We must associate a pointer to a particular type: You can't assign the address of a short int to a long int.

Pointer arithmetic

Postfix/prefix increment or decrement operations on a normal variable either increases or decrease it by 1. But the postfix/prefix increment or decrement operations on a pointer variable means addition or subtraction of bytes that pointer data type holds, with the value that the pointer variable contains.

int a = 5;
int *b;
b = &a;

Suppose the address of variable 'a' and 'b' are 5000 and 6000 respectively.

Then, b++ equal to (value stored in b + number of bytes occupied by the data type of pointer variable =5000+4=5004).

Similarly, b– equal to 5000-4=4996.

Accessing array elements using pointers

Let the array 'A' contains the following contents.

int A[5] = {15,20,34,7,43};

A[0] refers to the first member of the array A, A[1] refers to the 2nd member and so on. i^{th} member of the array A can be accessed by A[i-1].

When we declare an array, consecutive memory locations are allocated to array elements. The base address of array is the address of 0th element of array. Name of the array also gives the base address of array.

Hence, A=&A[0]
A+1=&A[1]
A+i=&A[i].

Similarly, *A=A[0]
*(A+1)=A[1]
*(A+i)=A[i].

Some practice problems

Test the output of the following programs?

```c
#include<stdio.h>
int main ( )
{
   int a = 5;
   int *b;
   int **c;
   b = &a;
   c = &b;
   printf("value of a = %d\n",a);
   printf(" value of a = %d\n",*(&a));
   printf(" value of a = %d\n",*b);
   printf(" value of a = %d\n",**c);
   printf(" value of b = address of a = %p\n",b);
   printf(" value of c = address of b = %p\n",c);
   printf(" address of a = %p\n",&a);
   printf(" address of a = %p\n",b);
   printf(" address of a = %p\n",*c);
   printf(" address of b = %p\n",&b);
   printf(" address of b = %p\n",c);
   printf(" address c = %p\n",&c);
   return 0;
}
```

```c
/* program to understand the postfix, prefix, increment, decrement in the
    pointer variable */
#include<stdio.h>
int main ( )
{
   int a = 5;
   int *a1;
   char b = 'x';
   char *b1;
   float c = 5.5;
   float *c1;
   a1 = &a;
   b1 = &b;
   c1 = &c;
   printf ( " address of a=value of a1= %p\n",a1);
   printf ( " address of b=value of b1 = %p\n",b1);
   printf ( " address of c =value of c1= %p\n",c1);
   a1++;
   b1++;
   c1++;
   printf ( " Now value of a1=%p\n",a1);
   printf ( " Now value of b1 = %p\n",b1);
   printf ( " Now value of c1 = %p\n",c1);
```

```
   return 0;
}
```

```
/*Use pointer to print the value and address of array elements*/
#include<stdio.h>
int main ( )
{
   int arr[4] = { 5 , 10 , 15 , 20 };
   int i = 0;
   int *b;
   b = arr; /*we can also write b = &arr[0] */
   for ( i = 0;i<=4;i++)
   {
      printf ( "value of arr[%d]=%d\n",i,*b);
      printf ( "address of arr[%d]=%p\n",i,b);
      b=b+1;
   }
   return 0;
}
```

```
/* program to print the value of the array element */
#include<stdio.h>
int main ( )
{
   int arr[4] = { 5 , 10 , 15 , 20 };
   int i = 0;

   for ( i=0;i<4;i++)
   {
      printf ( " value of arr[%d]=",i);
      printf ( "%d-----",arr[i] );
      printf ( " %d-----",*(arr+i));
      printf ( "%d-----",*(i+arr));
      printf ( "%d\n",i[arr] );
      printf ( " address of arr[%d]=%p\n",i,&arr[ i] );
   }
   return 0;
}
```

```
/*Program to print the elements of the array*/
#include<stdio.h>
int main ( )
{
   int arr[3][2] = {
        { 10 , 100 },
        { 20 , 200 },
```

```
        { 30 , 300 }
          };
   int i , j;
   for ( i = 0 ; i < 3 ; i++)
   {
      printf ( " \n" );
      for ( j=0;j< 2;j++)
         printf ( "value = %d\t", *(*(arr+i)+j));
   }
   return 0;
}
```

```
/*program to print the elements of 3-D array */
#include<stdio.h>
int main ( )
{
int arr[2][3][2] = {
        {
           { 5 , 10 },
           { 6 , 11 },
           { 7 , 12 },
        },
        {
           { 20 , 30 },
           { 21 , 31 },
           { 22 , 32 },
         }
         };
   int i,j,k;
   for ( i=0;i< 2;i++)
   for ( j=0;j < 3 ; j++)
   {
      printf ( " \n");
      for ( k =0;k<2;k++)
         printf ( " %d\t",*(*(*(arr+i)+j)+k));
   }
   return 0;
}
```

```
/* program for understanding the concept of array of pointers */
#include<stdio.h>
int main ( )
{
   int *arr[3];
   int a = 5, b = 10, c = 15,i;
   arr[0] = &a;
```

```
   arr[1] = &b;
   arr[2] = &c;
   for ( i=0;i< 3;i++)
   {
      printf ( "address = %p\t", arr[i] );
      printf ( " value = %d \n",*(arr[i]) );
   }
   return 0;
}
```

```
/*program to print the address and character of the string with using pointer*/
#include<stdio.h>
int main ( )
{
   char arr[ ]= "reena";
   char *a;
   a = arr;
   while ( *a !='\0' )
   {
      printf ( "address= %p\t",a);
      printf ( "character = %c\n",*a);
      a = a+1;
   }
   return 0;
}
```

```
/* program to print the address and string using array of pointer to string*/
#include<stdio.h>
int main ( )
{
   int i;
   char *arr[ ] = {
         "riti",
         "niti",
         "kriti",
         "kittu",
         "nitin"
         };

   for ( i=0;i<5;i++)
   {
      printf ( "address=%p\t",(arr + i) );
      printf ( "string = %s\n", *(arr + i) );
   }
   return 0;
}
```

```c
/*program to understand the sizeof operator */
#include<stdio.h>
int main ( )
{
   struct
   {
       char name[10];
       int age;
       float sal;
   }rec;
   int arr[10];
   printf ( " size of structure = %lu",sizeof ( rec ) );
   printf ( " size of int = %lu",sizeof ( int ) );
   printf ( " size of array = %lu",sizeof ( arr ) );
   return 0;
}
```

```c
/* program to enter the 5 numbers and print them using malloc( )*/
#include<stdio.h>
#include<stdlib.h>
int main ( )
{
   int i;
   int *a;
   a = ( int * ) malloc ( 5 * sizeof ( int ) );
   for ( i=0;i<5;i++)
   {
      printf ( " number %d=", i+1);
      scanf ( "%d",( a+i) );
   }
   for ( i =0;i<5;i++)
   printf ( " %d\n",*(a+i) );
   return 0;
}
```

```c
/*program to understand the use of realloc function */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main ( )
{
   char *ptr;
   ptr = ( char * ) malloc ( 6 );
   ptr = "ankit";
   printf ("%s is in memory block\n",ptr );
   ptr = ( char * ) realloc ( ptr,8);
```

```
   printf ("%s is in memory block\n",ptr );
   strcpy ( ptr , "rishabh" );
   printf ("Now %s is in memory block\n",ptr );
   return 0;
}
```

Important things to remember:

- Normal variable stores the value whereas pointer variable stores the address of the variable.
- 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.

Practice Exercise on pointers

Now, we ask you to do some practice problems given below. This time we are not providing you the code for these problems but ask you to try first by yourself and if you face difficulty, take help of any textbook or TA during the lab session.

1. Write a program in C to swap two numbers using pointers.
2. Write a program in C to print the elements of a array using pointers.
3. Write a program in C to find the sum of elements in array using pointers.
4. Write a program in C to find the length of a string using pointers.
5. Write a program in C to sort elements of an array using pointers.
6. Write a program in C to accept name and marks with the use of structure with pointer. Also print the grade based on the marks.
7. Write a program in C to accept name and marks of 10 students with the use of structure with array of pointer and also print them.