|  |  |
| --- | --- |
| **CS118 Programming Fundamentals** | **LAB 08** Recursion & Pointers |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |

### **Recursion:** A function that calls itself is known as a recursive function. And, this technique is known

### as recursion. **Example: Sum of Natural Numbers Using Recursion**

#include <stdio.h>

int sum(int n);

int main()

{

int number, result;

printf("Enter a positive integer: ");

scanf("%d", &number);

result = sum(number);

printf("sum = %d", result);

return 0;

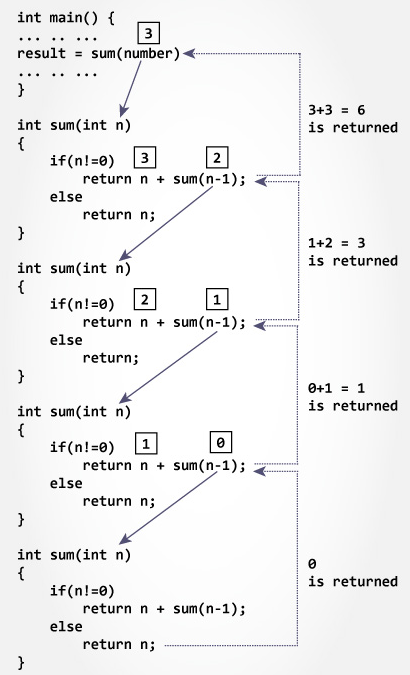
}int sum(int num)

{if (num!=0)

return num + sum(num-1); // sum() function calls itself

else

return num;}



**Pointers in C**

**What Are Pointers?**

A pointer is a variable which holds the address of other variable of the specified data type (like int, float, and char). In programming we basically use pointers to store the other variable’s address.

Some C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers.

Here is a first example of how to declare a pointer:

**Declarations:**

* **int** \*a**;** /\* Declare a pointer to an int \*/
* **float** \*b**;** /\* Declare a pointer to a float \*/
* **char** \*c**;** /\* Declare a pointer to a character \*/
* The asterisk, when used as above in the declaration, tells the compiler that the variable is to be a pointer, and the type of data that the pointer points to, but NOT the name of the variable pointed to.

**Use of & and \*:**

* When is & used?

& -- "address operator" which gives or produces the memory address of a data

variable.

* When is \* used?

\* -- "dereferencing operator" which provides the contents in the memory location

specified by a pointer.

**How Computers Address Memory?**

An address in memory is exactly like an address of a house, or would be if the following conditions were true:

* Every house is numbered in order.
* There are no skipped or duplicated numbers.
* The city consists of one long street.

So, for example, the address of a particular byte of memory might be 0x1000. The next byte after that would be 0x1001. The byte before would be 0x0FFF. Memory addresses are always expressed in hexadecimal.

**Equating pointers:**

Let’s take two different pointers of same data type and see what happens when we assign one pointer to the other.

int b=10;

int c=20;

int \*a=&b;

int \*d=&c;

a=d;

**Example:**

#include <stdio.h>

int main () {

int var = 20; /\* actual variable declaration \*/

int \*ip; /\* pointer variable declaration \*/

ip = &var; /\* store address of var in pointer variable\*/

printf("Address of var variable: %x\n", &var );

/\* address stored in pointer variable \*/

printf("Address stored in ip variable: %x\n", ip );

/\* access the value using the pointer \*/

printf("Value of \*ip variable: %d\n", \*ip );

return 0;

}

When the above code is compiled and executed, it produces the following result –

Address of var variable: bffd8b3c

Address stored in ip variable: bffd8b3c

Value of \*ip variable: 20

* **Null pointer:**

A Null pointer is assigned a reserved value which tells us the pointer doesn’t point to anything.

**int \*ptr=NULL;**

On assigning a pointer to null you are just telling it isn’t pointing to anything.

* **Dangling pointer**:

If the address to which the pointer is pointing to is no longer accessible in the memory then that pointer is called a dangling pointer.

* **Arithmetic and Logical Operations on Pointers:**
  + A pointer may be incremented or decremented
  + An integer may be added to or subtracted from a pointer.
  + Pointer variables may be subtracted from one another.
  + Pointer variables can be used in comparisons, but usually only in a comparison to NULL.
  + When an integer is added to or subtracted from a pointer, the new pointer value is changed by the integer times the number of bytes in the data variable the pointer is pointing to.
  + For example, if the pointer *valptr* contains the address of a double precision variable and that address is 234567870, then the statement:

*valptr = valptr* + 2;

Would change *valptr* to 234567886.

**Pointers with Functions**

* **Pass by reference:**

Normally when passing a variable to a function, the compiler makes a COPY of the variable in the function. Hence changing the value of the argument in the function does not change the original value. This is called pass by value. Sometimes, like in scanf(), we want to change the variable inside the function. To do this, we pass a pointer as input argument to the function this is called pass by reference.

**Example:**

#include <stdio.h>

void swap(int \*n1, int \*n2);

int main()

{

int num1 = 5, num2 = 10;

printf("Before swape \n");

printf("num1 = %d\n", num1);

printf("num2 = %d", num2);

// address of num1 and num2 is passed

swap( &num1, &num2);

printf("After swape \n");

printf("num1 = %d\n", num1);

printf("num2 = %d", num2);

return 0;

}

void swap(int\* n1, int\* n2)

{

int temp;

temp = \*n1;

\*n1 = \*n2;

\*n2 = temp;

}

**Lab Task**

**Question # 01:**

(a) Write a program to find Factorial of a Number Using Recursion

(b)Write a program to count the digits of a given number using recursion function.

**Question # 02:**

Write a program in C to calculate the power of any number using recursion.   
Test Data :  
Input the base value : 2  
Input the value of power : 6  
*Expected Output* :

The value of 2 to the power of 6 is : 64

**Question # 03:**

Write a program to swap two variables by passing the reference of these variables into a function declared as void swap (int \*, int \*).

**Input Data:**

First element: 12

Second element: 3

**Output:** The elements after swapping are:

First element: 3

Second element: 12

**Question # 04:**

1. Write a program in C to find the maximum number between two numbers using a pointer.
2. Write a program in C to add numbers using call by reference.

**Question # 05:**

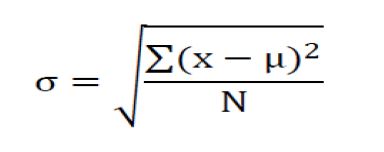
Write a single function to calculate the square, cube and square root of its floating

point argument and make those results available to the calling program.

**Question # 06:**

Write a single function that receives an array of 5 integers and returns the sum, average and standard deviation of these numbers without using return statement.

Call this function from main ( ) and print the results in main ( ).



Where x represents each value in the population, μ is the mean value of the population, Σ is the summation (or total), and N is the number of values in the population.

**Note: Use function type (call by reference)**

**Question # 07:**

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

**Question # 08:**

Suppose you have a compare() function that accepts two integers a and b.

The compare() function returns the following:

* Return 1 if a &gt; b
* Return 0 if a = b
* Return -1 if a &lt; b

Also declare a function pointer \*fp\_func to which assign the address of the

compare( ) function. Write a c program to solve the above scenario.

**Question # 09:**

Write a program to find separately the sum of the even and odd indexed elements of an array of size 10. Pass the even and odd elements to separate functions eg: sumEvenIndex(int\*), sumOddIndex(int\*) to carry out its sum. Also pass this array to a function called sortArray(int[]) and display the array elements into ascending order using pointer.