1) Write a C program to sort given list of n integers into ascending order using selection sort. Use function to sort.

**Description:** Assume we have an array ‘A’ with ‘N’ number of elements. This algorithm arranges elements in ascending order. ‘Pass’ is an index variable, which indicates the number of passes. The variable ‘min\_index’ denotes the position of the smallest element encountered in that pass. ‘I’ is another index variable. The array and the size are passed to the function.

**Algorithm:** Selection Sort

Step 1: Using Pass index variable repeat the steps from first record to last – 1 records and perform all the steps 1to 4.

Step 2: Initialize the minimum index as

min\_index = pass.

Step3: Obtain the element with the smallest value. for(i= pass+1; i < N; i++)

{

if( A[i] < A [min\_index])

min\_index = i;

}

Step4: Exchange the elements

if(min\_index != pass)

{

temp = A[pass];

A[pass] = A[min\_index]; A[min\_index] = temp;

}

Step5: Stop

### Trace of Selection Sort:

0 42 11 11 11 11

11

1 23 23 23 23

23

2 74 74 42 42

74

42

3 42 65

74

65

4 65 65 65 74

### File Name: selection\_sort\_fun.h

// function to swap array elements at positions x and y void Swap(int arr[], int x, int y)

{

int temp = arr[x]; arr[x] = arr[y]; arr[y] = temp;

}

*/\* function Selection Sort \*/*

void SelectionSort(int arr[], int n)

{

int pass, j, min\_indx;

// One by one move boundary of unsorted subarray for (pass = 0; pass < n-1; pass++)

{

// Find the minimum element in unsorted array min\_indx = pass;

for (j = pass+1; j < n; j++) if (arr[j] < arr[min\_indx]) min\_indx = j;

// Swap the found minimum element with the first element if (min\_indx != pass)

Swap(arr, min\_indx, pass);

}

}

### File Name: selection\_sort.c

#include <stdio.h>

#include "selection\_sort\_fun.h" void main()

{

int array[10]; int i, j, n, temp;

printf("Enter the value of n \n"); scanf("%d", &n);

printf("Enter the elements \n"); for (i = 0; i < n; i++)

scanf("%d", &array[i]);

*/\* Selection sorting begins \*/*

SelectionSort(array, n);

printf("The sorted list is (using selection sort): \n"); for (i = 0; i < n; i++)

printf("%d\t", array[i]);

}

### Sample Input and output:

Enter the total no of elements: 5 Enter the elements: 99 20 -12 43 34

The sorted list is (using selection sort): -12 20 34 43 99

## LAB EXERCISE:

### Write C programs to implement the following.

1. Given an array of distinct elements. The task is to find triplets in an array whose sum is zero. Take the array as input. (Triplets Game)

Sample Input 0 -1 2 -3 1

Sample output 0 -1 1

2 -3 1

1. Check whether the given matrix is sparse matrix or not using functions.
2. Implement an iterative Lsearch(….) function to search for an element in an 1D array of type integer using linear search technique.
3. Implement a C program to read, display and to find the product of two matrices using functions with suitable parameters. Check for the compatibility of the input matrices before multiplication.

## ADDITIONAL EXERCISES:

1. **Random number generation and finding the frequency of occurrence**: Generate a large number of random numbers (say around 10K Samples). Each sample value should be between -100 to 100 (integers only). After generating the samples find the frequency of each distinct sample. Repeat the above steps with unknown size (hint: user will decide at run time). The purpose is to realize the advantages and disadvantages of using array.
2. **Addition of polynomials with two terms:** To perform different operations on polynomial with two terms x, y using 2-D array representations. Operations like addition and multiplication have to be implemented [ref: J.P Trembly]. If the 2D array representation is sparse then optimize the memory usage by using suitable alternative representation.
3. Bishu went to fight for Coding club. There were N soldiers with various powers. There will be Q rounds to fight and in each round Bishu’s power will be varied. With power M, Bishu can kill all the soldiers whose power is less than or equal to M(<=M). After each round, all the soldiers who are dead in previous round will reborn. Such that in each round there will be N soldiers to fight. As Bishu is weak in mathematics, help him to count the number of soldiers that he can kill in each round and total sum of their powers. Explanation: In first round bishu power is 3 so there are 3 soldiers whose power is <=3 and the sum of their power is 1+2+3=6 therefore ans = 3 6 same for the next round.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 7  1 2 3 4 5 6 7  3  3  10  2 | 3 6  7 28  2 3 |

**LAB NO: 2 Date:**

## SOLVING PROBLEMS USING RECURSION

### Objectives:

In this lab, student will be able to:

* + Formulate a recursive solution to a given problem
  + Familiarize with recursion concept in C programs

## SOLVED EXERCISE:

* 1. Write a C program to implement binary search

### Title: IMPLEMENT BINARY SEARCH

1. Program to perform binary search on a set of keys.

**Aim:** To understand the working recursive function call and also binary search technique.

**Description** Binary search method employs the process of searching for a record only in half of the list, depending on the comparison between the element to be searched and the central element in the list. It requires the list to be sorted to performsuch a comparison. It reduces the size of the portion to be searched by half after eachiteration.

**Algorithm:** Binary Search

**Assumption:** The input array is in sorted order.

Step1: Given array A[low, high], find the value of mid location as mid =(low+high)/2 Step2: if Low > high return a Failure status and terminate the search; go to step 4.

Step3: Else Compare the key (element to be searched) with the mid element.

If key matches with middle element, we return the mid index; go to step 4.

Else If key is greater than the mid element, then key can only lie in right half sub-array after the mid element. So we recur for right half.

Else (x is smaller) recur for the left half until there are no more elements left in the array.

Step4: stop

### Program:

**File Name : binary\_search\_function.h**

int bin\_search(int low,int high,int item,int a[])

{

int mid; if(low>high)

return(-1);

else

{

mid=(low+high)/2; if(item==a[mid])

return(mid); else if(item<a[mid])

return(bin\_search(low,mid-1,item,a));

else

}

}

return(bin\_search(mid+1,high,item,a));

### File Name : binary\_search.c

#include <stdio.h>

#include "binary\_search\_function.h" void main()

{

int i, pos, a[30],n, item; printf("Enter number of items:"); scanf("%d",&n);

printf("Enter the elements in ascending order:\n"); for(i=0;i<n;i++)

scanf("%d",&a[i]);

printf("Enter element to be searched:"); scanf("%d",&item); pos=bin\_search(0,n-1,item,a);

if(pos!=-1)

printf("Item found at location %d",pos+1);

else

printf("Item not found");

}

**Sample Input and Output:**

Enter number of items: 6

Enter the elements in ascending order: 12 23 54 65 88 99

Enter element to be searched:99 Item found at location 6

## LAB EXERCISES:

### Write C programs using recursion. Also, write the call tree indicating call order.

* 1. To copy one string to another using Recursion.
  2. To check whether a given String is Palindrome or not, using Recursion
  3. Simulate the working of Tower of Hanoi for n disks. Print the number of moves.
  4. To implement selection sort using recursion

## ADDITIONAL EXERCISES:

* 1. To find the first capital letter in a string using Recursion.
  2. To read a line of text and write it out backwards using Recursion

**LAB NO: 3 Date:**

## SOLVING PROBLEMS USING ARRAYS, POINTERS AND DYNAMIC MEMORY ALLOCATION FUNCTIONS

### Objectives:

In this lab, student will be able to:

* + 1. Familiarize with syntax and usage of pointers and pointer to arrays and dynamic memory management functions
    2. Write C programs making use of pointer concepts and dynamic memory allocation functions

## SOLVED EXERCISE:

* + - * 1. Write a program to read n names of different sports and store them using array pointers. Use dynamic memory allocation and deallocation functions. The program should display all the names and deallocate the dynamic memory at the end of the program.

**Description:** The following figure illustrates the use of array of character pointers to store the names of different sports. First, an array of character pointers is created. Then, the name of different sports is read from the user and memory is allocated as per the input string length. This representation is memory efficient in comparison to the storage of multiple strings using 2D array of characters.

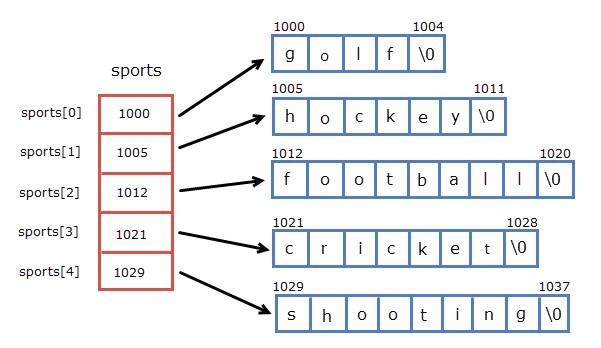


Fig. Memory representation of array of pointers

Program:

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

int i,n;

char \*sports[10]; char str[100];

printf("\n enter the number of sports \n");

scanf("%d", &n);

printf("\nenter the names of sports:\n"); for (i = 0; i < n; i++)

{

scanf("%s", str);

//allocating memory equal to the length of string + 1

//Last 1 byte to accommodate the ‘\0’

sports[i] = (char\*) calloc(strlen(str)+1, sizeof(char)); strcpy(sports[i],str);

}

printf("\nGiven list of sports: \n"); for (i = 0; i < n; i++)

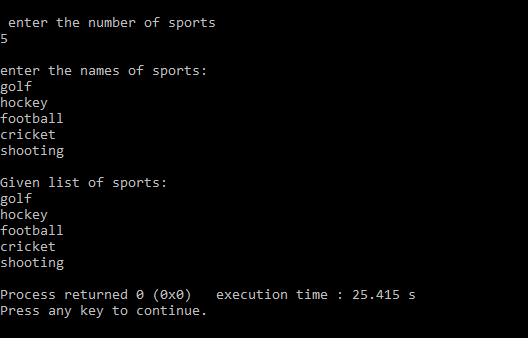
printf("%s\n", sports[i]);

//Deallocate the dynamic memory for (i = 0; i < n; i++)

free(sports[i]);

return 0;

}



**Sample input and output:**

## LAB EXERCISES :

### Note: Pass parameters using pointer to all the following functions.

* + - * 1. Write a function Reverse to reverse the elements of an integer array using pointer. Access the elements of the array using dereference operator. Read the values from the keyboard in main function. Display the result in the main function.
        2. Write a function Smallest to find the smallest element in an array using pointer. Create a dynamically allocated array and read the values from keyboard in main. Display the result in the main function.
        3. Write a C program to

Demonstrate passing pointers to a function.

Demonstrate Returning pointer from a function.

c) Using pointer to pointer.

* + - * 1. Implement a C program to read, display and to find the product of two matrices using functions with suitable parameters. Note that the matrices should be created using dynamic memory allocation functions and the elements are accessed using array dereferencing.

## ADDITIONAL EXERCISES:

* + - * 1. Write a C program to print an array in forward direction by adding one to pointer and in backward direction by subtracting one from pointer.
        2. Write a function Palindrome to check whether a given string is a palindrome using pointers to array.

**LAB NO: 4 Date:**

## SOLVING PROBLEMS USING RAGGED ARRAYS, STRUCTURES AND POINTERS

### Objectives:

In this lab, student will be able to:

* Familiarize the usage of ragged arrays
* Write programs using structures and pointers
* Familiarize of dynamic memory allocation for structures

## SOLVED EXERCISE:

* 1. Write a C program to implement a ragged array dynamically.

table [rowNum]

rowNum+1 elements

colNum+1 elements

table [1]

- - - - -

colNum

table [0]

[colNum]

table

**Description:** In a ragged array the *table* pointer points to the first pointer in an array of pointers. Each array pointer points to a second array of integers, the first element of which is the number of elements in the list. A sample ragged array structure is shown below.

[0]

**Algorithm:** Construct a ragged array

Step 1: Declare a ragged array as a variable *table*.

Step 2: Ask the user for row size and set a variable – *rowNum*

Step 3: Allocate space for *(rowNum+1)* pointers as row pointers. The last row pointer will hold NULL

Step 4: Ask the user for column size and set a variable – *colNum*

Step 5: Allocate space for *(colNum+1)* data elements. The first element will hold value contained in colNum itself.

Step 6: Repeat step 3 for all rows

Step 7 : Display ragged array contents. Step 8: Stop

**Program:**

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

int rowNum, colNum, i, j; int \*\*table;

printf("\n enter the number of rows \n"); scanf("%d", &rowNum);

table = (int \*\*) calloc(rowNum+1, sizeof(int \*));

for (i = 0; i < rowNum; i++) /\* this will tell which row we are in \*/

{

printf("enter size of %d row", i+1);

scanf("%d", &colNum);

table[i] = (int \*) calloc(colNum+1, sizeof(int)); printf("\n enter %d row elements ", i+1);

for (j = 1; j <= colNum; j++)

{

scanf("%d", &table[i][j]);

}

table[i][0] = colNum;

printf("size of row number [%d] = %d", i+1, table[i][0]);

}

table[i] = NULL;

for (i = 0; i < rowNum; i++) /\* this will tell which row we are in \*/

{

printf("displaying %d row elements\n", i+1); for (j = 0; j <= \*table[i]; j++)

{

printf("%5d", table[i][j]);

}

printf("\n");

}

//freeup the memory

for (i = 0; i < rowNum; i++) { free(table[i]);

}

free(table); return 0;

}

**Sample input and output:** enter the number of rows: 3 enter size of row 1: 4

enter row 1 elements: 10 11 12 13

enter size of row 2: 5

enter row 2 elements: 20 21 22 23 24 enter size of row 3

enter row 3 elements: 30 31 32 displaying

10 11 12 13

20 21 22 23 24

30 31 32

## LAB EXERCISES :

### Note: Use Pointers to structures and dynamic memory management functions in the following programs.

* 1. Implement Complex numbers using structures. Write functions to add, multiply, subtract two complex numbers.
  2. Write a C program to implement the following functions. Use pointers and dynamic memory management functions.
     1. To read one Student object where Student is a structure with name, roll number and CGPA as the data members
     2. To display one Student object
     3. To sort an array of Student structures according to the roll number.
  3. Samuel wants to store the data of his employees, which includes the following fields:

(i) Name of the employee (ii) Date of birth which is a collection of {day, month, year}

(iii) Address which is a collection of {house number, zip code and state}. Write a 'C' program to read and display the data of N employees using pointers to array of structures.

* 1. Create a structure STUDENT consisting of variables of structures:
     1. DOB {day, month (use pointer ), year},
     2. STU\_INFO {reg\_no, name(use pointer), address},
     3. COLLEGE {college\_name (use pointer), university\_name}

where structure types from i to iii are declared outside the STUDENT independently. Show how to read and display member variables of DOB type if pointer variable is created for DOB inside STUDENT and STUDENT variable is also a pointer variable. The program should read and display the values of all members of STUDENT structure.