**Batch: A1 Roll No.: 1911004**

**Experiment / assignment / tutorial No. 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE: Passing array and element to function.** |

**AIM: Accept one dimensional unsorted array and an element to be inserted from user. Pass array and element to a function which sorts the array in the ascending or descending order and then inserts this element in the array in its correct place**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO1. Formulate a problem statement and develop the logic (algorithm/flowchart) for its solution.**

**CO2. Apply basic concepts of C programming for problem solving.**

**CO3. Illustrate the use of derived and structured data types such as arrays, strings, structures and unions.**

**CO4. Demonstrate the concepts of modular programming through functions and dynamic memory allocation through use of pointers.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

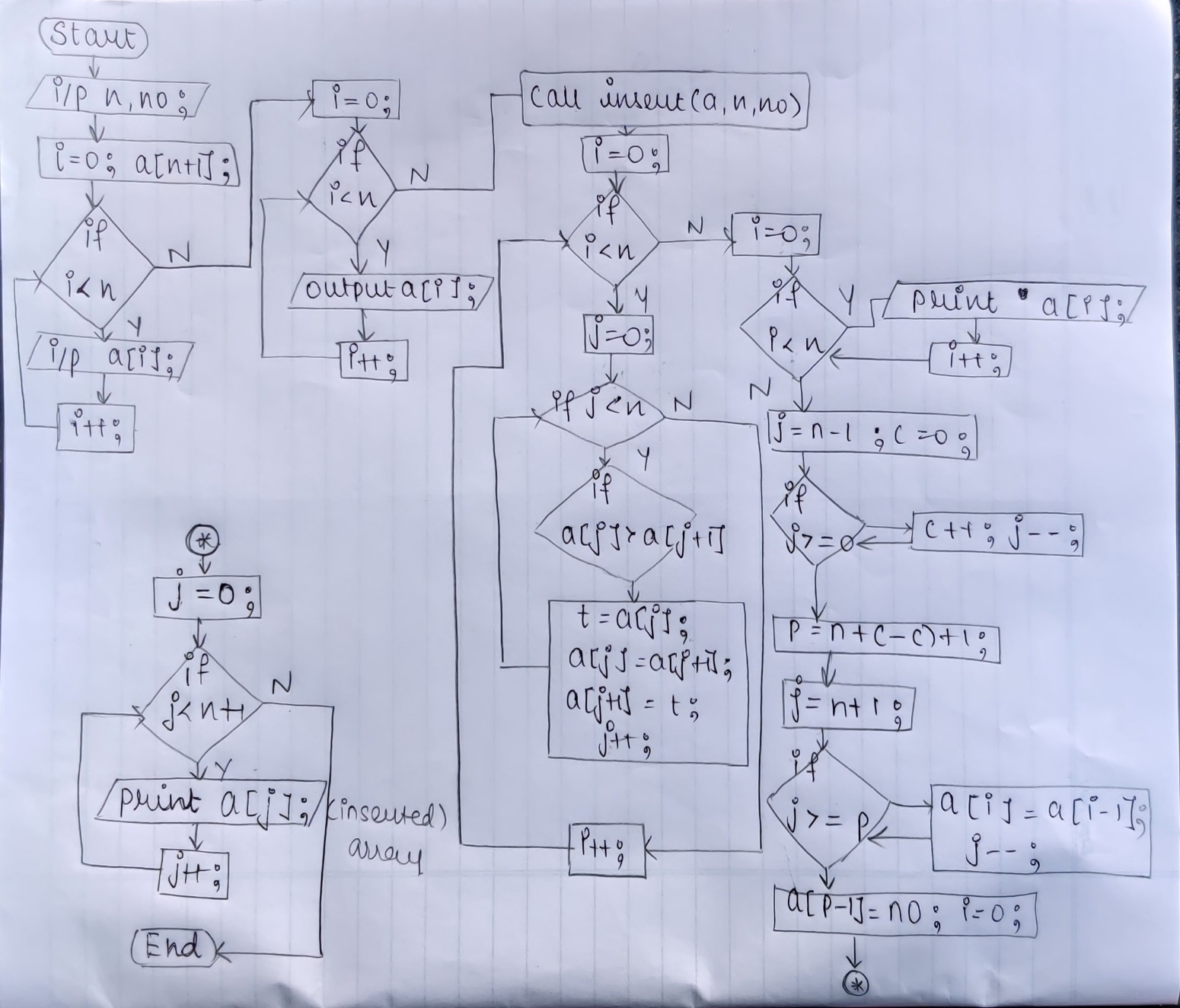
1. **Programming in C, second edition, PradeepDey and ManasGhosh, Oxford University Press.**
2. **Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.**
3. **Introduction to programming and problem solving, G. Michael Schneider ,Wiley India edition.**
4. **Let’s C by YashwantKanetkar**
5. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

1. **Accept 1D unsorted array and an element from the user.**
2. **Pass this array and element to be inserted to a function named ‘insert’.**
3. **This ‘insert’ function will sort the array (ascending/descending order) and will insert the element inputted by the user at its correct position.**

**Flowchart:**



**Implementation details:**

**#include<stdio.h>**

**int i,j,t,c=0,p;**

**void insert(int a1[],int n1,int no1) //func definition ,declaration**

**{for(i=0;i<n1;i++)**

**{for(j=0;j<n1-1;j++)**

**{if(a1[j]>a1[j+1])**

**{ t=a1[j];**

**a1[j]=a1[j+1];**

**a1[j+1]=t;}}}**

**printf("\n the sorted array is \n");**

**for(i=0;i<n1;i++)**

**printf(" %d",a1[i]);**

**for(i=n1-1;i>=0;i--)**

**if(no1<a1[i])**

**c++;**

**p=n1+1-c;**

**for(i=n1+1;i>=p;i--)**

**a1[i]=a1[i-1];**

**a1[p-1]=no1;**

**printf("\nTHE ARRAY IS= ");**

**for(i=0;i<=n1;i++)**

**printf(" %d ",a1[i]);**

**}//insert() ends**

**void main()**

**{int no,n;**

**printf("Enter the length of the array & the no to be inserted\n");**

**scanf("%d %d",&n,&no); int a[n+1];**

**printf("Enter the elements of the array to sort them \n");**

**for(i=0;i<n;i++)**

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

**printf("\n the array is \n");**

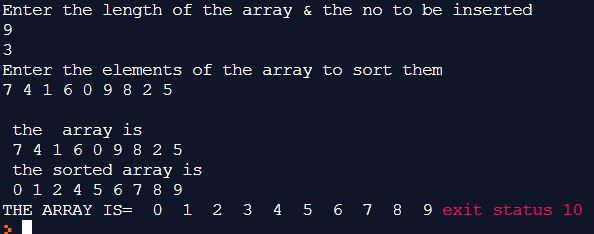
**for(i=0;i<n;i++)**

**printf(" %d",a[i]);**

**insert(a,n,no); //insert() func call**

**}**

**Output(s):**

****

**Conclusion: The unsorted 1D array was sorted and needed element was inserted in it successfully.**

**Post Lab Descriptive Questions:**

1. **Describe two ways of passing parameters to functions. When do you prefer to use each of them?**

**🡺2 Ways of passing parameters to a function are :-**

* **CALL BY VALUE- in this the copy of the values of the actual parameters are created in formal parameters. Any changes made inside that functions don’t affect the real value of actual parameters.**
* **CALL BY REFERENCE- in this the reference(location) of the variables are passed in actual parameter & the formal parameters are the pointer variables. Any changes made in the function are reflected into the real values of the actual variables too.**

**When we want to just use the values passed as actual parameters in function we use *call by value.***

**When we want to change original the values of parameters passed as actual parameters in function we use *call by reference.***

1. **Distinguish between the following:**
2. **Actual and Formal parameters/arguments**

|  |  |
| --- | --- |
| **Actual parameters/arguments** | **Formal parameters/arguments** |
| **The parameters which appear in function call are called as actual parameters/arguments.** | **The parameters which appear in function definition(declaration) are called formal parameters.** |
| **In actual parameters there is no mentioning of datatypes. Only the values is passed.** | **In formal parameters, the data types of the receiving values should be included.** |

**int add(int x,int y) // these are formal parameters in func() definition**

**{ return x+y;}**

**void main(){**

**int a=5,b=1,s;**

**s=add(a,b);//actual parameters in func() call**

**}**

1. **Scope and Visibility of variables**

|  |  |
| --- | --- |
| **Scope** | **Visibility** |
| **It defines the availability and life of the variables in side a particular function/loop or through all function/loops.** | **It defines the accessibility of the data members(variables & functions) to other functions** |
| **The scope of variable determines over what region of the program a variable is actually available for use. The scope of an identifier is a part of the program in which the identifier can be used to access its object.** | **The visibility of variable refers to the accessibility of a variable from the memory. . The visibility of an identifier is a region of the program source code from which an identifier’s associated object can be legally**  **accessed.** |
| **2 types**   * **Global** * **local** | **3 types**   * **public** * **private** * **friendly (default)** |

1. **& operator and \* operator**

|  |  |
| --- | --- |
| **&** | **\*** |
| **this is used for reading the address of the variable along with (bitwise AND)** | **this is used for pointing to value at the read address (along with arithmetic multiplication)** |
| **It appears in function call to pass by reference** | **It appears in the function definition for pointer formal parameter to accept & point at address** |
| **Used for knowing address of variable** | **Used for pointing that address and thus give the value at that address** |

**void main()**

**{**

**int x=5;**

**int \*p; // pointer p**

**p=&x; //pointer p has address of x**

**printf(“x has address =%x and value”, &x ,\*p); // \*p=\*&x=5**

**}**

**Date: 09-04-2020 Signature of faculty in-charge**