**Subject Code:** KCS351

**Subject Name:** Data Structure using C Lab

**Semester:** III

**Session:** 2020-21

**LAB-3**

**List Of Programs**

1. Write a simple program on pointer to print address.
2. WAP for Dynamic Memory Allocation using malloc().

## WAP for Accessing array elements(Traversing array) by incrementing a Pointer.

**Program Implementation in C:**

1. Write a simple program on pointer to print address.

Few important points to remember:

* \* is used to access the value stored in the pointer variable.
* & is used to store the address of a given variable.

Below is a simple program on pointer.

int \*p; is a pointer variable declaration where p is a pointer to an int variable i.e. it stores the location of an integer.

%x is a format specifier to print hexadecimal value. It is usually used to print the location.

#include <stdio.h>

int main()

{

int var = 24; // actual variable declaration

int \*p;

p = &var; // storing address of int variable var in pointer p

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

// address stored in pointer variable

printf("\n\nAddress stored in pointer variable p is: %x", p);

// access the value using the pointer variable

printf("\n\nValue of var variable or the value stored at address p is %d ", \*p);

return 0;

}

1. Program for Dynamic Memory Allocation using malloc()

Below is a program on dynamic memory allocation using malloc() and clearing out memory space using free().

sizeof() returns the number of bytes occupied by any datatype, in this case by an integer.

#include <stdio.h>

int main()

{

int n, i, \*ptr, sum = 0;

printf("\n\nEnter number of elements: ");

scanf("%d", &n);

// dynamic memory allocation using malloc()

ptr = (int \*) malloc(n\*sizeof(int));

if(ptr == NULL) // if empty array

{

printf("\n\nError! Memory not allocated\n");

return 0; // end of program

}

printf("\n\nEnter elements of array: \n\n");

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

{

// storing elements at contiguous memory locations

scanf("%d", ptr+i);

sum = sum + \*(ptr + i);

}

// printing the array elements using pointer to the location

printf("\n\nThe elements of the array are: ");

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

{

printf("%d ",ptr[i]); // ptr[i] is same as \*(ptr + i)

}

/\*

freeing memory of ptr allocated by malloc

using the free() method

\*/

free(ptr);

return 0;

}

1. Accessing array elements(Traversing array) by incrementing a Pointer

Name of the array refers to the base address of the array.

Below is a program to access elements of an array using pointer increment.

#include <stdio.h>

const int MAX = 3; // Global declaration

int main()

{

int var[] = {100, 200, 300};

int i, \*ptr;

/\*

storing address of the first element

of the array in pointer variable

\*/

ptr = var;

for(i = 0; i < MAX; i++)

{

printf("\n\n\nAddress of var[%d] = %x ", i, ptr);

printf("\nValue of var[%d] = %d ", i, \*ptr);

// move to the next location

ptr++;

}

return 0;

}

