Lab: 8

Statement: Accessing Array Elements and Displaying them.

Q.1 Explain how to access array elements and to display those elements under two topics:

- Accessing array elements.
- Displaying array elements.

Program:

```c
#include <stdio.h>

#define MAX_SIZE 100 // Maximum array size

int main()
{
    int arr[MAX_SIZE];
    int N, i;
    int * ptr = arr;     // Pointer to arr[0]

    printf("Enter size of array: ");
    scanf("%d", &N);

    printf("Enter elements in array:\n");
    for (i = 0; i < N; i++)
    {
        // &ptr[i] is equivalent to &arr[i]
        scanf("%d", &ptr[i]);
    }

    printf("Array elements: ");
    for (i = 0; i < N; i++)
    {
        // i[ptr] is equivalent to arr[i], i[arr] and ptr[i]
        // I have used i[ptr] syntax for knowledge. You can also use ptr[i]
        printf("%d, ", i[ptr]);
    }

    return 0;
}
```

Output:

```
Enter size of array: 6
Enter elements in array:
1 2 3 4 5 6
Array elements: 1, 2, 3, 4, 5, 6,
--------------------------------
Process exited after 11.62 seconds with return value 0
Press any key to continue . . .
```

Lab 14:

Statement: Call by value and Call by reference.

Questions: Write a program to illustrate call by value and call by reference method of calling a function for swapping values of two variables.

Program:

```c
#include<stdio.h>

void swap(int *,int *);

int main( )
{
    int n1,n2;
    printf("Enter the two numbers to be swapped\n");
    scanf("%d%d",&n1,&n2);
    printf("\nThe values of n1 and n2 in the main function before calling the swap function are n1=%d n2=%d",n1,n2);
    swap(&n1,&n2);
    printf("\nThe values of n1 and n2 in the main function after calling the swap function are n1=%d n2=%d",n1,n2);
}

void swap(int *n1,int *n2)
{
    int temp;
    temp=*n1;
    *n1=*n2;
    *n2=temp;
    printf("\nThe values of n1 and n2 in the swap function after swapping are n1=%d n2=%d",*n1,*n2);
}
```

Output:

```
Enter the two numbers to be swapped
12
34

The values of n1 and n2 in the main function before calling the swap function are n1=12 n2=34
The values of n1 and n2 in the swap function after swapping are n1=34 n2=12
The values of n1 and n2 in the main function after calling the swap function are n1=34 n2=12
--------------------------------
Process exited after 7.437 seconds with return value 0
Press any key to continue . . .
```

Lab 15: Pointer.

Q. 1 A program to illustrate pointer declaration.

Program:

```
Address of c: 000000000065FE14
Value of c: 22

Address of pointer pc: 000000000065FE14
Content of pointer pc: 22

Address of pointer pc: 000000000065FE14
Content of pointer pc: 11

Address of c: 000000000065FE14
Value of c: 2


--------------------------------
Process exited after 0.0359 seconds with return value 0
Press any key to continue . . .
```

Output:

```c
#include <stdio.h>
int main()
{
    int* pc, c;

    c = 22;
    printf("Address of c: %p\n", &c);
    printf("Value of c: %d\n\n", c);  // 22

    pc = &c;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 22

    c = 11;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 11

    *pc = 2;
    printf("Address of c: %p\n", &c);
    printf("Value of c: %d\n\n", c); // 2
    return 0;
}
```

Q.2 Program to display the contents of the variable their address using pointer variable.

Program:

```c
/* Simple Print address of Variable Using Pointer in C*/
/* Print Pointer Address Program,C Pointer Examples */

#include <stdio.h>

int main() {
  int a;
  int *pt;

  printf("Pointer Example Program : Print Pointer Address\n")
  a = 10;
  pt = &a;

  printf("\n[a  ]:Value of A = %d", a);
  printf("\n[*pt]:Value of A = %d", *pt);
  printf("\n[&a ]:Address of A = %p", &a);
  printf("\n[pt ]:Address of A = %p", pt);
  printf("\n[&pt]:Address of pt = %p", &pt);
  printf("\n[pt ]:Value of pt = %p", pt);

  return 0;
}
```

Output:

```
Pointer Example Program : Print Pointer Address

[a  ]:Value of A = 10
[*pt]:Value of A = 10
[&a ]:Address of A = 000000000065FE1C
[pt ]:Address of A = 000000000065FE1C
[&pt]:Address of pt = 000000000065FE10
[pt ]:Value of pt = 000000000065FE1C
--------------------------------
Process exited after 0.03764 seconds with return value 0
Press any key to continue . . .
```

Q.3 Program to illustrate the pointer expression and pointer arithmetic.

Program:

```c
// Program showing pointer expressions
// during Arithmetic Operations
#include <stdio.h>

int main()
{
    // Integer variables
    int a = 20, b = 10;

    // Variables for storing arithmetic
    // operations solution
    int add, sub, div, mul, mod;

    // Pointer variables for variables
    // a and b
    int *ptr_a, *ptr_b;

    // Initialization of pointers
    ptr_a = &a;
    ptr_b = &b;

    // Performing arithmetic Operations
    // on pointers
    add = *ptr_a + *ptr_b;
    sub = *ptr_a - *ptr_b;
    mul = *ptr_a * *ptr_b;
    div = *ptr_a / *ptr_b;
    mod = *ptr_a % *ptr_b;

    // Printing values
    printf("Addition = %d\n", add);
    printf("Subtraction = %d\n", sub);
    printf("Multiplication = %d\n", mul);
    printf("Division = %d\n", div);
    printf("Modulo = %d\n", mod);
    return 0;
}
```

Output:

```
Addition = 30
Subtraction = 10
Multiplication = 200
Division = 2
Modulo = 0

--------------------------------
Process exited after 0.05642 seconds with return value 0
Press any key to continue . . .
```

Q. 4 Program to illustrate call by value

Program:

```c
#include<stdio.h>
void change(int num) {
    printf("Before adding value inside function num=%d \n",num);
    num=num+100;
    printf("After adding value inside function num=%d \n", num);
}
int main() {
    int x=100;
    printf("Before function call x=%d \n", x);
    change(x);//passing value in function
    printf("After function call x=%d \n", x);
return 0;
}
```

Output:

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=100

--------------------------------
Process exited after 0.04153 seconds with return value 0
Press any key to continue . . .
```

Q.5 Program to illustrate call by reference.

Program:

```c
#include<stdio.h>
void change(int *num) {
    printf("Before adding value inside function num=%d \n",*num);
    (*num) += 100;
    printf("After adding value inside function num=%d \n", *num);
}
int main() {
    int x=100;
    printf("Before function call x=%d \n", x);
    change(&x);//passing reference in function
    printf("After function call x=%d \n", x);
return 0;
}
```

Output:

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=200

--------------------------------
Process exited after 0.04658 seconds with return value 0
Press any key to continue . . .
```

Lab 16:
Dynamic memory allocation.

Q. 1 Write a program illustrating malloc.

Program:

```c
// Program to calculate the sum of n numbers entered by the user

#include <stdio.h>
#include <stdlib.h>

int main() {
  int n, i, *ptr, sum = 0;

  printf("Enter number of elements: ");
  scanf("%d", &n);

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

  // if memory cannot be allocated
  if(ptr == NULL) {
    printf("Error! memory not allocated.");
    exit(0);
  }

  printf("Enter elements: ");
  for(i = 0; i < n; ++i) {
    scanf("%d", ptr + i);
    sum += *(ptr + i);
  }

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

  // deallocating the memory
  free(ptr);

  return 0;
}
```

Output:

```
Enter number of elements: 5
Enter elements: 1
2
3
4
5
Sum = 15
--------------------------------
Process exited after 9.212 seconds with return value 0
Press any key to continue . . .
```

Write a program illustrating calloc.

Program:

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main()
{
    int n, *ptr, *p, i, sum = 0;
    /* n = number of elements, *ptr = store base address of the dynamic memory,
    *p store temporary address of the *ptr */
    printf (" Enter the number of elements: ");
    scanf (" %d", &n); // it takes number of elements

    // use calloc syntax to create memory block of int data type
    ptr = (int *) calloc (n, sizeof(int));
    p = ptr; // assign the address of ptr
    if (ptr == NULL) // it checks whether the memory is allocated
    {
        printf (" Memory is not allocated. ");
        exit(0); // exit from the program
    }
    printf (" Enter %d numbers \n", n);
    for ( i = 1; i <= n; i++)
    {
        scanf ( "%d", ptr);
        sum = sum + *ptr;
        ptr++;
    }

    printf (" Elements are: ");
    for (i = 1;  i <= n; i++)
    {
        printf (" %d", *p);
    p++;
    }
    printf (" \n The addition of the elements is: %d ", sum);
    getch();
}
```

Output:

```
 Enter the number of elements: 3
 Enter 3 numbers
1
1
2
 Elements are:  1 1 2
 The addition of the elements is: 4
```

Write a program illustrating realloc.

Program:

```c
#include<stdio.h>
#include<stdlib.h>     //required for using realloc() in C
int main()
{
    int *ptr;
    int i;
    // typecasting pointer to integer
    ptr= (int *)calloc(4,sizeof(int));


    if(ptr!=NULL)
    {
        for(i=0;i<4;i++)
        {
            printf("Enter number number %d: ", i+1);
            scanf("%d",(ptr+i));
            }
        }
       //reallocation of 6 elements
        ptr= (int *)realloc(ptr,6*sizeof(int));
        if(ptr!=NULL)
        {
            printf("\nNew memory allocated!\n");
            for(;i<6;i++)
            {
                printf("Enter new number %d: ",i);
                scanf("%d",(ptr+i));
            }
        }
        printf("\n\nThe numbers are:\n");
        for(i=0;i<6;i++)
        {
            printf("%d \n",ptr[i]);
        }
        free(ptr);
       return 0;
}
```

Output:

```
Enter number number 1: 4
Enter number number 2: 6
Enter number number 3: 8
Enter number number 4: 5

New memory allocated!
Enter new number 4:
```

Lab 17: File handling.

Q.1 Write a simple c program to open a file and store information.

Program:

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int num;
    FILE *fptr;
    // use appropriate location if you are using MacOS or Linux
    fptr = fopen("C:\\program.txt","w");
    if(fptr == NULL)
    {
        printf("Error!");
        exit(1);
    }
    printf("Enter num: ");
    scanf("%d", &num);

    fprintf(fptr,"%d",num);
    fclose(fptr);
    return 0;
}
```

Output:

```
Error!
--------------------------------
Process exited after 0.03866 seconds with return value 1
Press any key to continue . . .
```

Q. 2 Write a program illustrating getw and putw.

Program:

```c
#include<stdio.h>
int main( )
{
   FILE *fp;
   int i;
   fp = fopen ("num.txt", "w");
   for (i =1; i<= 10; i++){
      putw (i, fp);
   }
   fclose (fp);
   fp =fopen ("num.txt", "r");
   printf ("file content is\n");
   for (i =1; i<= 10; i++){
      i= getw(fp);
      printf ("%d",i);
      printf("\n");
   }
   fclose (fp);
   return 0;
}
```

Output:

```
file content is
1
2
3
4
5
6
7
8
9
10

--------------------------------
Process exited after 0.05173 seconds with return value 0
Press any key to continue . . .
```

Q. 3 Write a c program illustrating fprintf and fscanf.

Program:

```c
#include <stdio.h>
int main()
{
    FILE *fptr;
    int id;
    char name[30];
    float salary;
    fptr = fopen("emp.txt", "w+");/*  open for writing */
    if (fptr == NULL)
    {
        printf("File does not exists \n");
        return 0;
    }
    printf("Enter the id\n");
    scanf("%d", &id);
    fprintf(fptr, "Id= %d\n", id);
    printf("Enter the name \n");
    scanf("%s", name);
    fprintf(fptr, "Name= %s\n", name);
    printf("Enter the salary\n");
    scanf("%f", &salary);
    fprintf(fptr, "Salary= %.2f\n", salary);
    fclose(fptr);
}
```

Output:

```
Enter the id
2735
Enter the name
suren
Enter the salary
50000

--------------------------------
Process exited after 22.6 seconds with return value 0
Press any key to continue . . .
```