Assignment VI

1. **What is pointer and explain its applications. Write a program that uses pointer to copy an array of double.**

Ans: Pointer is the variable that holds address of another variable of same data type.

The application of pointer are explained below:-

* Pointer allow C to support dynamic memory management.
* Pointer reduce length and complexity of the programs.
* Pointer are more efficient in handling arrays and data tables.
* They increase the execution speed and thus reduce the program execution time.
* Pointer can be used to return multiple value from a function via function arguments.
* The use of pointer arrays to character string results in saving of data storage space in memory.
* Pointer permit reference to functions and thereby facilitating passing of functions as arguments to other functions.
* Pointer provide an efficient tool for manipulating dynamic data structure such as structures, linked list, queues, stacks and trees.

**Program that uses pointer to copy an array of double.**

#include<stdio.h>

#include<conio.h>

int main ()

{

double a[10],\*b[10];

int i,m;

printf("enter the limit of the array:");

scanf("%d",&m);

for(i<m;i=0;i++){

printf("enter the element[%d]",i);

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

}

printf("\n before copying array:\n");

for(i=0;i<m;i++){

printf("a[%d]=%lf\n",i,a[i]);

}

for(i=0;i<m;i++){

b[i]=&a[i];

}

printf("\n\nAfter copying array:\n");

for(i=0;i<m;i++){

printf("b[%d]=%lf\n",i,\*b[i]);

}

return 0;

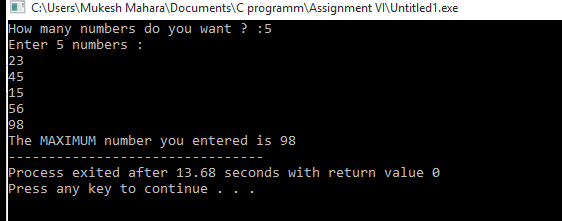
}

1. **Write a function that is passed with an array of n pointer which return the maximum.**

Ans:- A program with a function that is passed with an array of n pointers which returns the maximum is given below:

#include<stdio.h>

**Output:-**

****int max(int \*parr, int n){

int i;

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

if (\*parr < \*(parr + i)){

\*parr = \*(parr+i);

}

}

return \*parr;

}

int main()

{

int n,i,maximum;

printf("How many numbers do you want ? :");

scanf("%d",&n);

int arr[n];

printf("Enter %d numbers : \n",n);

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

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

}

maximum = max(arr,n);

printf ("The MAXIMUM number you entered is %d",maximum);

return 0;

}

1. **Justify that pointer is jewel in C language.Write a function that is passed an array of n pointers to float and returns a newly created array that contains those n float value in reverse order.Assume any necessary data**.

Ans:- A pointer is a derived data type in C. Pointer are undoubtedly one of the most distinct and exciting features of C language . It has added power and flexibility to the language.

Pointer are more efficient in handling arrays and tables. Pointer can be used to support dynamic memory management .Pointer reduce length and complexity of programs.

It increases the execution speed and thus reduce the program execution time by following characters. So we conclude that the real power of C lies in a proper use of pointers.

Thus, pointer is jewel in C language.

***Program***

#include <stdio.h>

#include<conio.h>

#define MAX 30

int main(){

int size,i;

int \*ptr arr[MAX];

ptr=&arr[0];

printf("\nEnter the size of array : ");

scanf("%d", &size);

printf("\nEnter %d element into array: ", size);

for (i = 0; i < size; i++) {

scanf("%d", ptr);

ptr++;

}

ptr = &arr[size - 1];

printf("\nElements of array in reverse order are : \n");

for (i = size ; i >= 1; i--) {

printf("\ %d\t ", i, \*ptr);

ptr--;

}

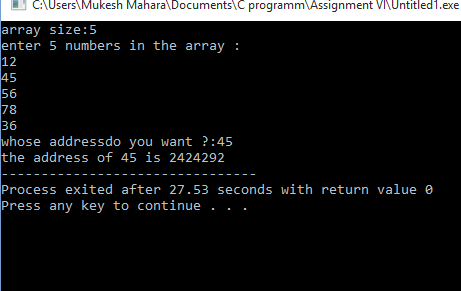
getch();

}

1. **Write a function that uses pointer to search for address of the given integer in a given array. It the given integer is found , the function return its address; otherwise return null.**

Ans:- #include<stdio.h>

#include<conio.h>

****int\*search (int \*parr,int tar,int n){ **Output:-**

int i;

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

if (\*(parr+i) ==tar){

return (parr+i);

}

}

return NULL;

}

int main(){

int i,n,tar;

printf("array size:");

scanf("%d",&n);

int arr[n];

printf("enter %d numbers in the array :\n",n);

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

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

}

printf("whose addressdo you want ?:");

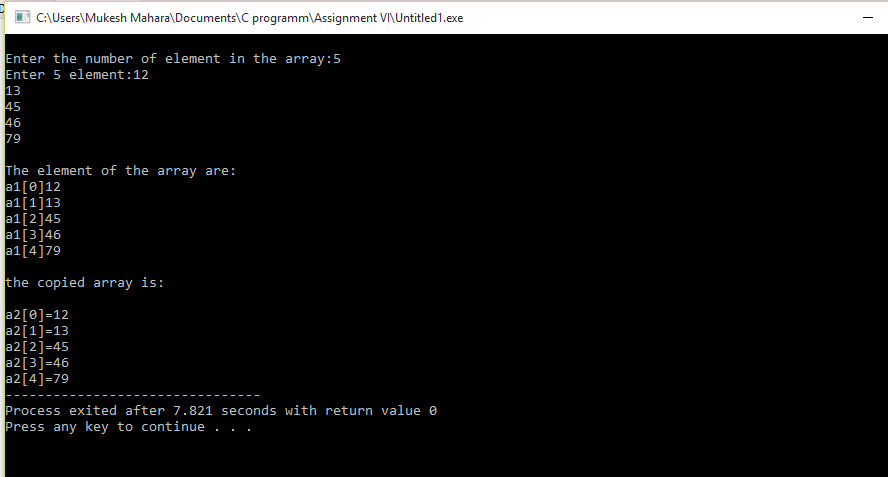
scanf("%d",&tar);

printf("the address of %d is %u",tar,search(arr,tar,n));

return 0;

}

1. Write a program that uses pointer to copy an array of integer.

****Ans:-

#include<stdio.h> **Output:-**

#include<conio.h>

int main()

{

int n;

int i;

printf("\nEnter the number of element in the array:") ;

scanf("%d",&n);

int \*p,a1[n];

p=&a1[n];

printf("Enter %d element:",n);

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

scanf("%d",&a1[i]);

}

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

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

printf("\na1[%d]%d\t",i,\*(a1+i));

}

printf("\n\nthe copied array is:\n");

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

printf("\na2[%d]=%d\t",i,\*(a1+i));

}

return 0;

}

1. **Write a function that is passed through the array of n pointers to float and return a newly created array that contain those n float values.**

Ans:- #include<stdio.h>

int main()

{

float\*a[10],\*b[10];

int I,m;

printf("Enter the limit of the array:");

scanf("%d",&m);

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

{

printf("Enter the element [%d]",i);

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

}

printf("\nBefore array : \n");

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

{

printf("a[%d]=%f\n",i,a[i]);

}

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

{

b[i]=&a[i];

}

printf("\n\nAfter array : \n");

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

{

printf("b[%d]=%f\n",i,\*b[i]);

}

return 0;

}

1. **Explain the pointer to structure with example**.

Ans:- A Structure type pointer variable can be declared as:

Struct book

{

char name[20];

int pages;

float price;

};

Struct book \*bptr;

However, this declaration for a pointer to structure does not allocate any memory for a structure but allocates only for a pointer, so that to access structure’s members through pointer bptr, we must allocate the memory using malloc() function.

Now, individual structure member are accessed as :

Bptr->name bptr->pages bptr->price

(\*bptr).name(\*bptr).pages(\*bptr).price

Here, -> is called arrow operator and there must be a pointer to the structure on the left side of this operator.

For Example:

#include<stdio.h>

#include<conio.h>

#include<string.h>

struct book

{

char name[20];

int pages;

float price;

};

void main()

{

struct book \*bptr;

float temp;

bptr=(struct book \*)malloc(sizeof(struct book));

printf("\n Enter name:\t");

scanf("%s", bptr->name);

printf("\n Enter no. of pages:\t");

scanf("%d", &bptr->pages);

printf("\n Enter price:\t");

scanf("%f", & bptr->price);

printf("\n Name\t\t No. of Pages\t Price\n");

printf("%s\t\t%d\t\t%f",(\*b ptr).name,(\*bptr).pages,( \*bptr).price);

getch();

}

1. **Explain the pointer arithmetic with example.**

Ans:- A pointer is a variable whose value is the address of another variable, i.e.,

direct address of the memory location. A pointer in C is an address, which is a numeric value. Therefore, we can perform arithmetic operations on a pointer just as we can on a numerical value. There are four arithmetic operators that can be used on pointers: ++, --, + and - .

let see an example for **incrementing a pointer**:

#include<stdio.h>

const in MAX=3;

int main(){

int var[]={10,100,200};

int I,\*ptr;

ptr=var;

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

printf(“Address of var[%d]=%x\n”, i, ptr);

printf(“Value of var[%d]=%d\n”, i, \*ptr);

ptr++;

}

Return 0;

}

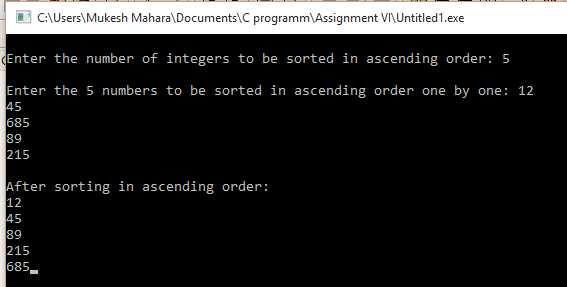
1. **Write a program to sort ‘n’ number in ascending order using dynamic memory.**

Ans:-

#include<stdio.h> **Output:-**

#include<conio.h>

#include<stdlib.h>

****void main()

{

int \*a,n,i,j,t;

printf("\nEnter the number of integers to be sorted in ascending order: ");

scanf("%d",&n);

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

printf("\nEnter the %d numbers to be sorted in ascending order one by one: ",n);

for(i=0;i<=n-1;i++)

{

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

}

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

for(j=0;j<=i;j++){

if(\*(a+i)<\*(a+j))

{

t=\*(a+i);

\*(a+i)=\*(a+j);

\*(a+j)=t;

}

}

}

printf("\nAfter sorting in ascending order: ");

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

printf("\n%d",\*(a+i));

getch();

}

1. **What is dynamic memory allocation (DMA) ? How can you use it in C ?**

Ans: The process of allocating memory at run time is known as dynamic memory

allocation. Library routines known as “memory management function” are used for allocating and freeing memory during execution of a program. These function are defined in stdlib**.**h.

some of the function that are used to allocate dynamic memory are:

1. MALLOC() FUNCTION IN C:

* malloc () function is used to allocate space in memory during the execution of the program.
* malloc () does not initialize the memory allocated during execution. It carries garbage value.
* malloc () function returns null pointer if it couldn’t able to allocate requested amount of memory.

syntax:

malloc(number\*size of (int));

2. CALLOC() FUNCTION IN C:

* calloc () function is also like malloc () function. But calloc () initializes the allocated memory to zero. But, malloc() doesn’t.

syntax:

calloc (number, sizeof(int));

3. REALLOC() FUNCTION IN C:

* realloc () function modifies the allocated memory size by malloc () and calloc () functions to new size.
* If enough space doesn’t exist in memory of current block to extend, new block is allocated for the full size of reallocation, then copies the existing data to new block and then frees the old block.

syntax:

realloc (pointer\_name, number \* sizeof(int));

4. FREE() FUNCTION IN C:

* free () function frees the allocated memory by malloc (), calloc (), realloc () functions and returns the memory to the system.

syntax:

free (pointer\_name);

**NOTES:-**

* malloc():-This function allocates requested size of bytes and return a void

pointer pointing to the first byte of the allocated space

* calloc():- This function allocates space for an array of elements, initialize them to

zero and then return a void pointer to the memory.

* free():- This function release the previously allocated memory.
* realloc:- This function modift the size of previously allocated space

1. **What are the advantage over dynamic memory allocation over static memory allocation?**

Ans: The advantages over dynamic memory allocation over static memory allocation are:

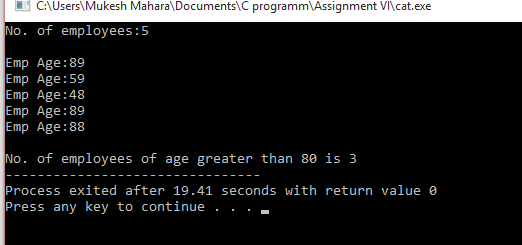
* In static memory allocation, since memory is allocated at compile time , it gets wasted if we tend to use lass that what u have allocated while in dynamic memory allocation, memory is allocated at run time so we can allocate as and when required.
* In dynamic memory allocation, if the memory is being continually allocated but the one allocated for the object that are not in use, is not released, then it can lead to stack overflow condition or memory leak which is big disadvantage.
* In case of dynamically created lists insertions and deletions can be done very easily just by the manipulation of addresses whereas in case of statically allocated memory insertions and deletions lead to more number of movements and wastage of memory.
* In case of statically allocated memory there is every chance of “overflow” during insertions in the lists, whereas in case of dynamically allocated memory it does not come into picture unless otherwise unavailability of main memory.

1. **How is malloc() function different from calloc() function?**

Ans:- The basic difference between malloc() and calloc() function are given below:-

|  |  |
| --- | --- |
| calloc() | malloc() |
| calloc() initializes the allocated memory with 0 value. | malloc() initializes the allocated memory with garbage value. |
| Number of arguments is 2. | Number of argument is 1. |
| **Syntax:**  (cast\_type \*)calloc(block, size\_of\_block); | **Syntax:**  (cast\_type \*)malloc(size\_in\_bytes); |

1. **Write a program to read number of employee, ‘n’ working in a company . Reserve the memory required to store age of ‘n’ employee using malloc() function. Read age of ‘n’ employees from user and count the number of employees of age over 80 years.**

****Ans:- #include<stdio.h> **Output:-**

#include<stdlib.h>

int main(){

int n,\*age ,i,count=0;

printf("No. of employees:");

scanf("%d",&n);

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

printf("\n");

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

printf("Emp Age:");

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

if(\*(age+i)>80){

count++;

}

}

printf("\nNo. of employees of age greater than 80 is %d",count);

return 0;

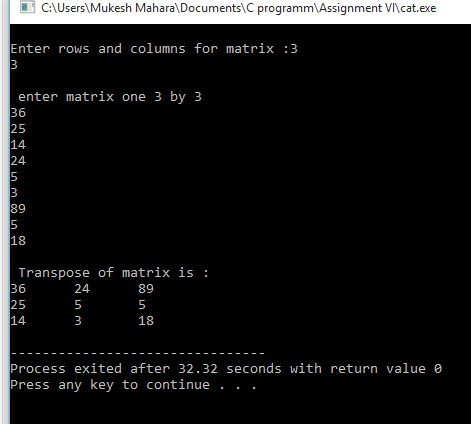
}

1. **Write a program to read a matrix of size m\*n entered by the user and display its transpose of matrix using dynamic memory allocation (DMA).**

Ans:- #include<stdio.h>

#include<stdlib.h> **Output-**

int main()

****{

int \*\*a;

int ar,ac;

int i,j;

printf("\nEnter rows and columns for matrix :");

scanf("%d%d",&ar,&ac);

/\* allocate memory for matrix \*/

a=(int \*\*) malloc(sizeof(int \*)\*ar);

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

{

a[i]=(int \*) malloc(sizeof(int\*)\*ac);

}

// matrix input

printf("\n enter matrix one %d by %d\n",ar,ac);

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

{

for(j=0;j<ac;j++)

{

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

}

}

//output matrix

printf("\n Transpose of matrix is : \n");

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

{

for(j=0;j<ar;j++)

{

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

}

printf("\n");

}

/\*free memory\*/

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

{

free(a[i]);

}

free(a);

}