C programming file:

Contents:

Pointer

String Literal

Array of Strings

Structures

Array of Structures

Unions

Data Structure

Structure and Functions

Pointer Programs

**Program - 1**

**OBJECTIVE:-**  Simple Pointer program which gets the value of integer x and its address.

**PROGRAM:-**

#include <stdio.h>

int main()

{

int x=5;

int \*a; // Pointer Declaration

a = &x; // & is used to make the pointer a, store the

address of x

printf("%d\n",\*a); // \* operator fetches the value of x from pointer a.

printf("%d\n",a); // address of x contained in pointer a.

printf("%d",&x); // & operator here prints the address of x.

}

Output:



**Program - 2**

**OBJECTIVE:-**  Pointer program

**PROGRAM:-**

#include <stdio.h>

int main()

{

int i = 10;

int \*p, \*q;

p = &i;

q = p;

printf("%d %d", \*p,\*q);

}

Output:



**Program - 3**

**OBJECTIVE:-**  Pointer program

**PROGRAM:-**

#include <stdio.h>

int main()

{

int i = 10, j = 20;

int \*p, \*q;

p = &i;

q = &j;

\*p = \*q;

printf("%d %d", \*p,\*q);

}



**Program - 4**

**OBJECTIVE:-**  Pointer program to get the mid value of array

**PROGRAM:-**

#include <stdio.h>

int main()

{

int a[ ] = {1,2,3,4,5};

int n = sizeof(a) / sizeof(a[0]); //size of whole array / size of one array element

int \*mid = findMid(a,n);

printf("%d", \*mid);

return 0;

}

int \*findMid(int a[], int n) // here findMid is a pointer as well as name of function

{

return &a[n/2];

}

Output:

3

**Program - 5**

**OBJECTIVE:-**  Pointer program

**PROGRAM:-**

#include <stdio.h>

int main()

{

int i = 10; // let address of i=1000

int \*p = &i; // let address of p=2000. Value contained in p = 1000

printf("%d\n", \*p); // Value of i

printf("%d\n", \*&p); // \*&p = \*(&p) = \*(2000)= 1000

printf("%d\n", &p); // address of p = 2000

printf("%d", \*&i); // \*&i = \*(&i) = \*(1000)= 10

}

**Program - 5**

**OBJECTIVE:-**  Pointer program to print sum of array.

**PROGRAM:-**

#include <stdio.h>

int main()

{

int a[] = {11,22,33,45,78};

int sum = 0, \*p;

for (p= &a[0]; p<= &a[4]; p++) //Pointer p pointing to the address in a loop

sum += \*p;

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

}

Output:



Alternate:

#include <stdio.h>

int main()

{

int a[] = {11,22,33,45,78};

int sum = 0, \*p;

for (p= a; p<= a+4; p++) // a Points to first element of array

sum += \*p; // a+4 points to fifth element.

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

}



**Program - 6**

**OBJECTIVE:-**  Pointer program to print array elements in Reverse order.

**PROGRAM:-**

#include <stdio.h>

#define N 5

int main()

{

int a[N], \*p;

printf("Enter %d elements of Array: ", N);

for (p=a; p<=a+N-1; p++)

scanf("%d",p);

printf("Elements in Reverse Order: \n");

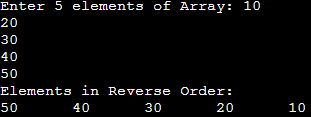
for (p=a+N-1; p>=a; p--)

printf("%d\t", \*p);

return 0;

}

Output:



Character: A single element

Eg. (A-Z), (a-z), symbols

Note: For character, “%c” is used

String: A set of characters

Eg. Words, sentences.

Note: For string, “%s” is used

String Programs

**Program - 1**

**OBJECTIVE:-**  Write simple String Programs.

**PROGRAM:-**

#include <stdio.h>

int main()

{

printf("%s\n", "Hello World");

printf("%s\n","Quantum Computing is the\

new era of programming.");

printf("%s\n","Quantum Computing is the "

"new era of programming.");

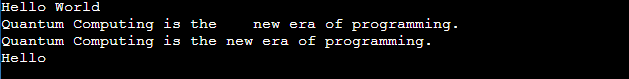
char \*x = "Hello World";

printf("%.5s", x); // Prints upto 5 places of string

return 0;

}

Output:



**Program - 2**

**OBJECTIVE:-**  Write program for Character array.

**PROGRAM:-**

#include <stdio.h>

int main()

{

char s[6] = "Hello"; //array can be initialised

char a[16] = {'H','E','L','L','O'}; //by any of the 2 methods

printf("%s\n", s); // here s is treated as pointer to first element of array

printf("%s\t", a); // here a is treated as pointer to first element of array

char x[10]; // mentioning size of array is mandatory

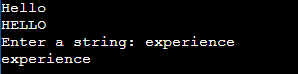
printf("Enter a string: ");

scanf("%s", &x);

printf("%s", x);

}

Output:



**Program - 3**

**OBJECTIVE:-**  Write program to show difference between scanf and gets function in string

**PROGRAM:-**

With scanf():

#include <stdio.h>

int main() {

char x[30];

printf("Enter a string: ");

scanf("%s", &x);

printf("%s", x);

}

Output:



With gets():

#include <stdio.h>

int main() {

char y[30];

printf("Enter a string: ");

gets(y);

printf("Output: %s",y);

}

Output:



**Program - 4**

**OBJECTIVE:-**  Write program to showcase various string functions (strcpy( ), strlen( ), strcat( ))

**PROGRAM:-**

1. Using strcpy( ):

#include <stdio.h>

#include <string.h>

int main()

{

char str1[10]= "Hello";

char str2[10]; // Format : strcpy(target,source);

strcpy(str2, str1); //copies the string from source to target destination

printf("%s", str2);

return 0;

}

Output:



Using strlen( ):

#include <stdio.h>

#include <string.h>

int main()

{

char \*str= "Hello World"; //pointer str pointing to H.

printf("Size of string: %d", strlen(str));

return 0;

}

Output:



Using strcat( ): to append one string to another

#include <stdio.h>

#include <string.h>

int main()

{

char str1[15], str2[20];

strcpy(str1, "I want to ");

strcpy(str2, "become Successful.");

strcat(str1, str2); //appended string 2 at the end of string 1.

printf("%s", str1);

return 0;

}

Output:



**Program - 5**

**OBJECTIVE:-**  Write program to showcasing Array of Strings.

**PROGRAM:-**

Pending

Structures Programs

**Program - 1**

**OBJECTIVE:-**  Write simple Structure Program.

**PROGRAM:-**

#include <stdio.h>

struct{

char \*engine;

}car1, car2; // car1 and car2 get the access to char \*engine inside the struct

int main(){

car1.engine = "DDis 190 Engine";

car2.engine = "1.2 L Kappa Dual VTVT";

printf("%s\n",car1.engine);

printf("%s\n",car2.engine);

return 0;

}

Output:



**Program - 2**

**OBJECTIVE:-**  Write Structure Program using Structure Tag

**PROGRAM:-**

#include <stdio.h>

struct employee{ // Here employee acts as name of the structure as well -

char \*name; // -as structure tag.

int age;

int salary;

};

int manager()

{

struct employee manager; //here employee - structure tag

manager.age = 27; // manager - name of structure.

if (manager.age > 30)

manager.salary = 65000;

else

manager.salary = 55000;

return manager.salary;

}

int main(){

struct employee emp1; //structure tag used again.

struct employee emp2;

printf("Enter salary of employee 1: ");

scanf("%d", &emp1.salary);

printf("Enter salary of employee 2: ");

scanf("%d", &emp2.salary);

printf("Employee 1 salary is: %d\n", emp1.salary);

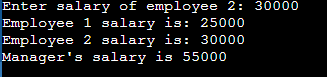
printf("Employee 2 salary is: %d\n", emp2.salary);

printf("Manager's salary is %d", manager());

return 0;

}

Output:



**Program - 3**

**OBJECTIVE:-**  Write Structure Program using Designated Initialization.

**PROGRAM:-**

#include <stdio.h>

struct abc{

int x;

int y;

int z;

};

int main()

{

struct abc a = {.y=20, .x=10, .z=30};

printf("%d %d %d", a.x, a.y, a.z);

return 0;

}

Output:



**Program - 4**

**OBJECTIVE:-**  Write an Array of Structures and fetch the output.

**PROGRAM:-**

#include <stdio.h>

struct car{

int fuel\_tank\_cap;

int seating\_cap;

float city\_mileage;

};

int main(){

struct car c[2];

int i;

for (i=0; i<2; i++){

printf("Enter car %d fuel tank capacity: ", i+1);

scanf("%d", &c[i].fuel\_tank\_cap);

printf("Enter car %d seating capacity: ", i+1);

scanf("%d", &c[i].seating\_cap);

printf("Enter car %d mileage: ", i+1);

scanf("%f", &c[i].city\_mileage);

}

printf("\n");

for(i=0; i<2; i++){

printf("\nCar %d details: \n", i+1);

printf("Fuel tank capacity: %d\n", c[i].fuel\_tank\_cap);

printf("Seating capacity: %d\n", c[i].seating\_cap);

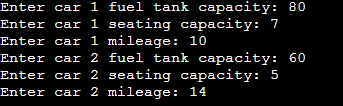
printf("city mileage: %0.2f\n", c[i].city\_mileage);

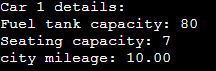
}

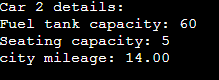
return 0;

}

Output:







**Program - 5**

**OBJECTIVE:-**  Write a program for Pointer to Structure variable.

**PROGRAM:-**

#include <stdio.h>

struct abc{

int x;

int y;

};

int main(){

struct abc a = {0,1};

struct abc \*ptr = &a;

printf("%d %d", ptr -> x, ptr -> y); // ptr -> x is equivalent to (\*ptr).x => (\*&a).x => a.x

return 0; // => a.x => 0

}

Output:



**Program - 6**

**OBJECTIVE:-**  Write a program to claculate Area of Rectangle with the help of Structure.

**PROGRAM:-**

#include <stdio.h>

struct point{ //structure 1

int x;

int y;

};

struct rectangle{ //structure 2

struct point upper\_left; Inside the structure, we declare “upper\_left” as

struct point lower\_right; object of type Struct point. To access the members of

}; Struct point.

int area (struct rectangle r){ //function. Here in “struct rectangle r” we created

int length, breadth; // an object r of type struct rectangle. To access the

length = r.lower\_right.x - r.upper\_left.x; // members of structure rectangle.

breadth = r.upper\_left.y - r.lower\_right.y;

return length\*breadth;

}

int main(){

struct rectangle r;

printf("Enter the upper left coordinates: \n");

scanf("%d %d", &r.upper\_left.x, &r.upper\_left.y);

printf("Enter the lower right coordinates: \n");

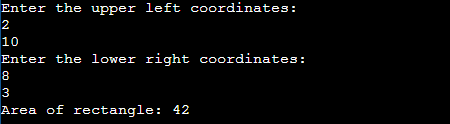
scanf("%d %d", &r.lower\_right.x, &r.lower\_right.y);

printf("Area of rectangle: %d", area(r));

return 0;

}

Output:



**Program – 7 (Structure and Functions)**

**OBJECTIVE:-**  Write a program to pass structure members as arguments.

**PROGRAM:-**

#include <stdio.h>

struct student{

char name[50];

int age;

int roll;

float marks;

};

void record (char name[], int age, int roll, float marks){

printf("%s %d %d %0.2f", name, age, roll, marks);

}

int main(){

struct student s1 = {"Nick", 16, 50, 72.5};

record (s1.name, s1.age, s1.roll, s1.marks);

}

Output:



**Program – 8 (Structure and Functions)**

**OBJECTIVE:-**  Write a program to pass structure variable as arguments.

**PROGRAM:-**

#include <stdio.h>

struct point{

int x;

int y;

};

void record (struct point p){

printf("%d %d \n", p.x, p.y);

}

int main(){

struct point p1 = {23, 45};

struct point p2 = {56, 90};

record (p1);

record (p2);

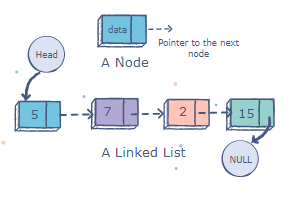
return 0;

}

Output:



***Linked List***



Linked List Programs

**Program - 1**

**OBJECTIVE:-**  Write simple Linked List Program **to create a Node.**

**PROGRAM:-**

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*link; //self refrential structure (because of \*link)

};

int main(){

struct node \*head = NULL; //Head being declared as pointer.

head = malloc(sizeof(struct node)); //malloc allocates memory for Data and Link.

head->data = 45;

head->link = NULL;

printf("%d", head->data);

return 0;

}

**Output:**



**Program - 2**

**OBJECTIVE:-**  Write simple Linked List Program to create a **Single Linked List**.

**PROGRAM:-**

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*link; //self refrential structure

};

int main(){

struct node \*head = NULL; //Head being declared as pointer.

head = malloc(sizeof(struct node)); //malloc allocates memory for Data and Link.

head->data = 45;

head->link = NULL;

struct node \*current = malloc(sizeof(struct node)); //another pointer for another node.

current -> data = 98;

current -> link = NULL;

head -> link = current; //first node is linked to second node

current = malloc(sizeof(struct node));

current -> data = 3;

current -> link -> link = current;

return 0;

}