**Array**

1. **Write what do you know about an array.**

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.

1. **Write two different programs on one dimensional array**

*FIRST*

#include<stdio.h>

main()

{

int i,arr[3];

printf("Enter 3 different numbers \n");

for(i=0;i<3;i++)

{

printf("Enter %d number",i+1);

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

}

printf("THe numbers that you entered are:\n");

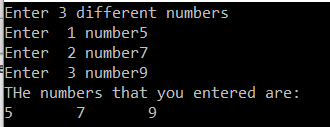
for(i=0;i<3;i++)

{

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

}

}



*SECOND*

#include<stdio.h>

main()

{

int i;

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

printf("The list of 1st five natural number is :\n");

for(i=0;i<5;i++)

{

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

}

}

****

1. **Write a program on m\*n matrix multiplication**

#include<stdio.h>

main()

{

int m, n,i,j,k,p,q;

printf("INput the number of rows and column of 1st matrix\n\n");

scanf("%d%d",&m,&n);

printf("INput the number of rows and column of 2nd matrix\n\n");

scanf("%d%d",&p,&q);

int mat1[m][n],mat2[p][q],mult[m][q],sum=0;

printf("Enter 1st matrix elements \n\n");

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

{

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

{

scanf("%d",&mat1[i][j]);

}

}

printf("Enter 2nd matrix elements \n\n");

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

{

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

{

scanf("%d",&mat2[i][j]);

}

}

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

{

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

{

for(k=0;k<p;k++)

{

sum=sum+mat1[i][k]\*mat2[k][j];

}

mult[i][j]=sum;

sum=0;

}

}

printf("Matrix after multiplication is \n\n");

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

{

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

{

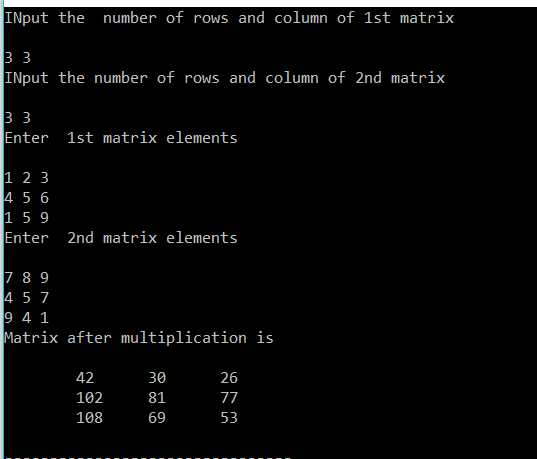
printf("\t%d",mult[i][j]);

}

printf("\n");

}

}



**STRUCTURE**

1. **Write what do you know about structure and elaborate some of its advantages.**

Structure is a user defined data type available in C that allows to combine data items of different kinds.

Structures are used to represent a record. Suppose if we want to keep track of our books in a library. We might want to track the following attributes about each book −

**•** Title

• Author

• Subject

• Book ID

1. **Write two different programs on structure**

*FIRST*

#include<stdio.h>

main()

{

struct student

{

char name[20];

int roll;

float marks;

char remarks;

};

struct student s;

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

gets(s.name);

printf("\n Enter the roll number");

scanf("%d",&s.roll);

printf("\nEnter the marks:\t");

scanf("%f",&s.marks);

printf("Enter remark p for pass or f for fail:\t");

s.remarks=getche();

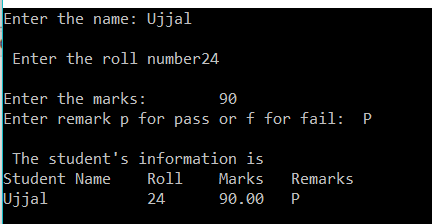
printf("\n\n The student's information is\n");

printf("Student Name\tRoll\tMarks\tRemarks\n");

printf("%s\t\t%d\t%.2f\t%c",s.name,s.roll,s.marks,s.remarks);

getch();

}



*SECOND*

#include<stdio.h>

struct employee

{

char fname[20];

char lname[20];

float height;

struct dates

{

int date;

int month;

int year;

}doj;

}emp={"Crissy","Costanza",5.1,{12,12,1689}};

void main()

{

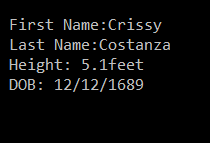
printf("\n First Name:%s",emp.fname);

printf("\n Last Name:%s",emp.lname);

printf("\n Height: %.1ffeet",emp.height);

printf("\n DOB: %d/%d/%d",emp.doj.date,emp.doj.month,emp.doj.year);

getch();

}

**UNIONS**

1. **List some of the difference between structure and unions**

|  |  |  |
| --- | --- | --- |
| **S.N** | **Structure** | **Union** |
| **1.** | The keyword struct is used to define a structure | The keyword union is used to define a union. |
| **2.** | When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members. The smaller members may end with unused slack bytes. | When a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member. |
| **3.** | Each member within a structure is assigned unique storage area of location. | Memory allocated is shared by individual members of union. |
| **4.** | Individual member can be accessed at a time | Only one member can be accessed at a time. |

1. **How can unions be useful?**

A union is a special data type available in C that allows to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple-purpose.

**POINTERS**

1. **Define a structure of employee having data members name, address, age and salary. Take data for n employee in an array dynamically and find the average salary.**

#include <stdio.h>

#include<stdlib.h>

struct name {

char s\_name[30];

char add[30];

int age;

float salary;

};

void main(){

struct name \*ptr;

int i,n,tot,avg;

printf("Enter the number : ");

scanf("%d",&n);

ptr=(struct name\*)malloc(n\*sizeof(struct name));

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

printf("Enter name,address,age and salary respectively :\n");

scanf("%s %s %d %f",&(ptr+i)->s\_name, &(ptr+i)->add, &(ptr+i)->age, &(ptr+i)->salary);

tot=tot+(ptr+i)->salary;

}

printf("Displaying Infromation:\n");

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

printf("%s\t%s\t%d\t%f\n",(ptr+i)->s\_name,(ptr+i)->add,(ptr+i)->age,(ptr+i)->salary);

}

printf("The average salary is:\n");

avg=tot/n;

printf("%d",avg);

}

