***Assignment 1***

***Submitted by: Posh Raj Dahal  
 CSIT,2 Semester***

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

An array is a fixed-size sequence collection of elements of the same data type. It is simply a grouping of like-type data. An array can be used to represent a list of numbers or a list of names(characters).

The main difference between array and ordinary variable is that ordinary variable can only hold a single data at a time.

For example if we want the temperature of Kathmandu for consecutive 24 hours then we need to assign 24 different ordinary variables which becomes lengthy and inefficient.

However, array can store all the 24 variables sequentially in a single step. It allocates the memory space for the consecutive data one after another.

1. **Write two programs on one dimensional array.**  
   Program 1: To arrange the entered data in ascending order  
   #include <stdio.h>

#include<conio.h>

int main()

{

int i, j, a, n, number[30];

printf("how many numbers do you want to input? \n");

scanf("%d", &n);

printf("Enter the numbers \n");

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

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

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

{

for (j = i + 1; j < n; ++j)

{

if (number[i] > number[j])

{

a = number[i];

number[i] = number[j];

number[j] = a;

}

}

}

printf("The numbers arranged in ascending order are given below \n");

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

printf("%d\n", number[i]);

return 0;

}

Program 2: To find maximum and minimum   
#include<stdio.h>

#include<conio.h>

int main()

{

int a[10],i,j,temp,uemp;

printf("Enter the number in the array");

for (i=0;i<10;i++){

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

}

for (i=0;i<=10;i++){

for (j=i+1;j<=10;j++){

if ( a[i]>a[j]){

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

printf("%d is highest and %d is lowest",a[10],a[0]);

return 0;

}

1. **Write a program on m\*n multiplication.**  
   #include<stdio.h>

#include<conio.h>

int main(){

int a[100][100],b[100][100],c[100][100],x,y,p,q,i,j,k,sum;

//for the first matrix

printf("Enter the no of rows and column");

printf("Row:\n");

printf("Column:\n");

scanf("%d%d",&x,&y);

printf("Enter the elements");

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

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

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

}

}

//for the second matrix

printf("Enter the no of rows and columns");

printf("Row:\n");

printf("Column:\n");

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

printf("Enter the elements");

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

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

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

}

}

//display the matrices

printf("The first matrix is: \n");

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

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

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

}

printf("\n\n");

}

printf("\n\n");

printf("The second matrix is: \n");

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

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

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

}

printf("\n\n");

}

//logic

if(p==y){

//matrix multiplication

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

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

for(k=0;k<y;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

//display result

printf("The multiplied matrix is:\n");

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

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

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

}

printf("\n\n");

}}

else{

printf("Matrix multiplication not possible");

}

return 0;

}

1. **Write what you know about structure and point out its advantages.**

Structure is the collection of variables of different types under a single name for better handling. For example: You want to store the information about person about his/her name, citizenship number and salary. You can create these information separately but, better approach will be collection of these information under single name because all these information are related to person.

Keyword struct is used for creating a structure.

Syntax :

struct person

{

char name[50];

int cit\_no;

float salary;

};

Advantages:

Structures are very similar to arrays ,the only difference is that in arrays you can store similar data types whereas in structures you can store two or more different data types including arrays pointers and even pointers of structure type(self refrential structures).Instead of using structures you also have an option to declare these variables globally but that's not a good practice and generally good programmers avoid it as there is a security threat if we do so.

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

To store and display information of students

#include <stdio.h>

struct student{

char name[50];

int roll;

float marks;

};

int main(){

struct student s[10];

int i;

printf("Enter information of students:\n");

for(i=0;i<10;++i)

{

s[i].roll=i+1;

printf("\nFor roll number %d\n",s[i].roll);

printf("Enter name: ");

scanf("%s",s[i].name);

printf("Enter marks: ");

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

printf("\n");

}

printf("Displaying information of students:\n\n");

for(i=0;i<10;++i)

{

printf("\nInformation for roll number %d:\n",i+1);

printf("Name: ");

puts(s[i].name);

printf("Marks: %.1f",s[i].marks);

}

return 0;

}

To calculate difference between two time period

#include <stdio.h>

struct TIME{

int seconds;

int minutes;

int hours;

};

void Difference(struct TIME t1, struct TIME t2, struct TIME \*diff);

int main(){

struct TIME t1,t2,diff;

printf("Enter start time: \n");

printf("Enter hours, minutes and seconds respectively: ");

scanf("%d%d%d",&t1.hours,&t1.minutes,&t1.seconds);

printf("Enter stop time: \n");

printf("Enter hours, minutes and seconds respectively: ");

scanf("%d%d%d",&t2.hours,&t2.minutes,&t2.seconds);

Difference(t1,t2,&diff);

printf("\nTIME DIFFERENCE: %d:%d:%d - ",t1.hours,t1.minutes,t1.seconds);

printf("%d:%d:%d ",t2.hours,t2.minutes,t2.seconds);

printf("= %d:%d:%d\n",diff.hours,diff.minutes,diff.seconds);

return 0;

}

void Difference(struct TIME t1, struct TIME t2, struct TIME \*differ){

if(t2.seconds>t1.seconds){

--t1.minutes;

t1.seconds+=60;

}

differ->seconds=t1.seconds-t2.seconds;

if(t2.minutes>t1.minutes){

--t1.hours;

t1.minutes+=60;

}

differ->minutes=t1.minutes-t2.minutes;

differ->hours=t1.hours-t2.hours;

}

|  |  |  |
| --- | --- | --- |
|  | **Structure** | **Union** |
| 1. | Each member within a structure is assigned its own unique storage. It takes more memory than union | All members within union share the same storage area of computer memory. It takes less memory than structure |
| 2. | The amount of memory required to store a structure is the sum of the sizes of all members | The amount of memory required to store an union is same as member that occupies largest memory |
| 3. | All the structure members can be accessed at any point of time | Only one member of union can be accessed at any given time. |
| 4. | Structure is declared as:  struct student{  int roll;  float marks;  }st; | Union is declared as:  union student{  int roll;  float marks;  }st; |

1. **List some difference between structure and union.**
2. **How can unions be useful?**

Unions can be used when no more than one member need be accessed at a time. That way, you can save some memory instead of using a struct. Union uses less memory and lets you do more dangerous things. It represents one continuous block of memory, which can be interpreted as either an integer, floating point value or a character pointer

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 employee {

char name[30];

char address[30];

int age;

float salary;

};

int main(){

struct employee \*ptr;

int i,n,total=0,avg;

printf("Enter the number : ");

scanf("%d",&n);

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

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

printf("\nEnter name :\n");

scanf("%s",&(ptr+i)->name);

printf("address :\n");

scanf("%s",&(ptr+i)->address);

printf("age :\n");

scanf("%d",&(ptr+i)->age);

printf("salary :\n");

scanf("%g",&(ptr+i)->salary);

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

}

avg=total/n;

printf("The average salary is: %d",avg);

}