Submitted by: Anup Dahal

1. Write what do you know about array.

Arrays are useful critters that often show up when it would be convenient to have one name for a group of variables of the same type that can be accessed by a numerical index. For example, a tic-tac-toe board can be held in an array and each element of the tic-tac-toe board can easily be accessed by its position (the upper left might be position 0 and the lower right position 8). At heart, arrays are essentially a way to store many values under the same name. You can make an array out of any data-type including structures and classes.

2. 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,hold;

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]){

hold=a[i];

a[i]=a[j];

a[j]=hold;

}

}

}

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

return 0;

}

3. Write a program on m\*n multiplication.

#include <stdio.h>

int main()

{

int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k;

printf("Enter rows and column for first matrix: ");

scanf("%d%d", &r1, &c1);

printf("Enter rows and column for second matrix: ");

scanf("%d%d",&r2, &c2);

/\* If colum of first matrix in not equal to row of second matrix, asking user to enter the size of matrix again. \*/

while (c1!=r2)

{

printf("Error! column of first matrix not equal to row of second.\n\n");

printf("Enter rows and column for first matrix: ");

scanf("%d%d", &r1, &c1);

printf("Enter rows and column for second matrix: ");

scanf("%d%d",&r2, &c2);

}

/\* Storing elements of first matrix. \*/

printf("\nEnter elements of matrix 1:\n");

for(i=0; i<r1; ++i)

for(j=0; j<c1; ++j)

{

printf("Enter elements a%d%d: ",i+1,j+1);

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

}

/\* Storing elements of second matrix. \*/

printf("\nEnter elements of matrix 2:\n");

for(i=0; i<r2; ++i)

for(j=0; j<c2; ++j)

{

printf("Enter elements b%d%d: ",i+1,j+1);

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

}

/\* Initializing elements of matrix mult to 0.\*/

for(i=0; i<r1; ++i)

for(j=0; j<c2; ++j)

{

mult[i][j]=0;

}

/\* Multiplying matrix a and b and storing in array mult. \*/

for(i=0; i<r1; ++i)

for(j=0; j<c2; ++j)

for(k=0; k<c1; ++k)

{

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

}

/\* Displaying the multiplication of two matrix. \*/

printf("\nOutput Matrix:\n");

for(i=0; i<r1; ++i)

for(j=0; j<c2; ++j)

{

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

if(j==c2-1)

printf("\n\n");

}

return 0;

}

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

Structures provide a way of storing many different values in variables of potentially different types under the same name. This makes it a more modular program, which is easier to modify because its design makes things more compact. Structs are generally useful whenever a lot of data needs to be grouped together--for instance, they can be used to hold records from a database or to store information about contacts in an address book. In the contacts example, a struct could be used that would hold all of the information about a single contact--name, address, phone number, and so forth.

5. 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;

}

WAP to store information dynamically

#include <stdio.h>

#include<stdlib.h>

struct name {

int a;

char c[30];

};

int main(){

struct name \*ptr;

int i,n;

printf("Enter n: ");

scanf("%d",&n);

/\* Allocates the memory for n structures with pointer ptr pointing to the base address. \*/

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

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

printf("Enter string and integer respectively:\n");

scanf("%s%d",&(ptr+i)->c, &(ptr+i)->a);

}

printf("Displaying Infromation:\n");

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

printf("%s\t%d\t\n",(ptr+i)->c,(ptr+i)->a);

return 0;

}

7. 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

8. 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.