Day\_12 Classwork

Single character declarations are different from strings.

Char a[] ={‘2’} è size of a is 1 as it is a single char

Char a[] ={“2”} èsize of a is 2 as it is a string the null value is considered by the compiler

2 table without printf using sprintf or using functions

The named address are the address that are given by compiler in stack memory

Strings got by strtok stored in structures. We need to give indices

To store string to char we can use indexing[][]

**Dynamic memory allocation of structures;**

testEmp = (EMP \*) malloc(sizeof(EMP));

#include<stdio.h>

typedef struct Employee

{

Int id;

Int sal;

Int phno;

Char Name [20];

Char gender;

} EMP;

//Int printEmp(Emp \*);

//Int getEmp(Emp \*);

Int main ()

{

EMP e1;

EMP \*e=NULL;

e = &e1;

scanf(“%d%d%d%s%c”,&e1.id,&e1.sal,&e1.phno,e1.name,&e1.gender);

printf(“\n Id : %d”,e->id);

printf(“\n Name: %s”,e->Name);

printf(“\n Salary:%d”,e->sal);

.

.

.

Return 0;                                                  /// named address

}

Int main()                                                                     //unnamed address

{

EMP \*e=NULL;

e = (EMP \*)malloc(sizeof(EMP));

scanf(“%d%d%d%s%c”,&e1->id,&e1->sal,&e1->phno,e1->name,&e1->gender);

printf(“\n Id : %d”,e->id);

printf(“\n Name: %s”,e->Name);

printf(“\n Salary:%d”,e->sal);

.

.

Free(e);

Return 0;

}

Generally in pointers while scanning we do not give address. We only specify the  ptr. But for structures we need to give the &ptr i.e the address.

Int main()                                                                     //unnamed address with multiple records

{

EMP \*e=NULL;

Int n;

Printf(“Enter the number of employees”);

Scanf(“%d”,&n);

e = (EMP \*)malloc(n\*sizeof(EMP));

temp=e;            // We store the address of e  to get back to the base address

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

scanf(“%d%d%d%s%c”,&e->id,&e->sal,&e->phno,e->name,&e->gender);

e=temp;

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

        {

printf(“\n Id : %d”,e->id);

printf(“\n Name: %s”,e->Name);

printf(“\n Salary:%d”,e->sal);

.

.

         }

Free(e);

Return 0;

}

Error !: Free(): invalid pointer

Aborted (core dumped)

Here we got the address because the e value is incremented that means the e address is shifted to some other address and we are pointing. So we need to get back to the base address and then free

E=temp;

Free(e);

We can avoid array type by using the loop in main and use only single