STRUCTURE

It is a user defined data type.

It is a complex data type.

It is collection of heterogeneous variables.

Structure is a user defined, complex data type where we can store and manage more than one variable of different data types under one name.

Structure allows to store both primitive and derived data types (arrays, pointers) at one place, under one name.

In real time applications, data is stored in the form of objects. In this situation, we need structures. Structures are the foundation for object oriented.

Primitive and derived data types are designed to work with basic data types like int, float and char.

Primitive and derived data types doesn’t supports real time requirements. Hence we have to use the user defined data type structure.

Structure allows to carry different types of variables at a time.

When structure address is available, automatically all the variables address also available. Due to this the search time is reduced.

Structure allows to store information in the form of records.

In data files we are using structures very much.

Syntax:

```
struct   [ <structure-tag-name> ]
{
datatype   variable;
datatype   variable;      structure members
}
[structure_variables] ;
```

Here struct is a keyword.

The structure tag name is used to identify the structure and it is optional, but required when the structure variables are declared in other places of the program.

The variables that are declared inside the structure are called **structure members**.

Structure size is sum of all the structure members datatype size.

Without structure members [empty] structure size is **1 byte.**

Structure variables are the **instances** [**copies**] of the structure.

Structure is a blue-print [original copy] to create the structure variables.

Structure variable is the physical representation of a structure.

When structure variables are declared then only memory is allocated for structure members.

Every structure should be end with ;

To access the structure members we should have to use the following syntax.

```
structurevariable.structuremember;
```

It is called calling / accessing / invoking the structure members.

Here **.** (dot) operator is called

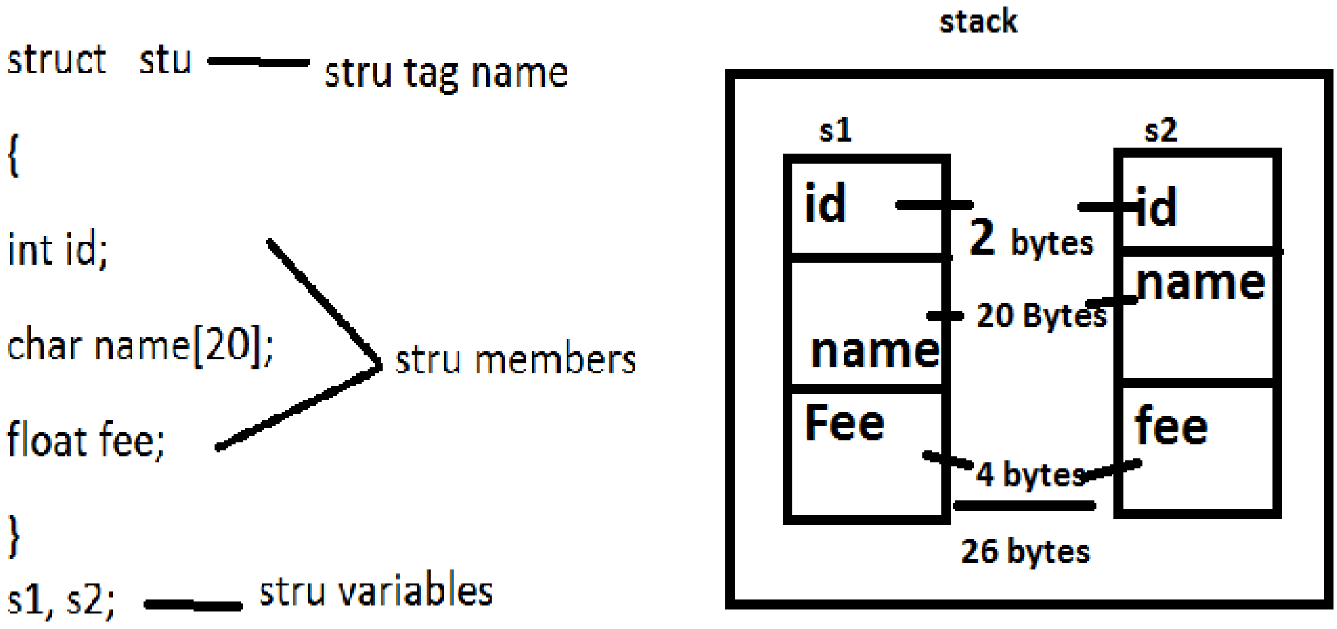
- Member access operator
- Field access operator
- Member of operator
- Membership operator
- Belongs to operator

We can declare structure variables in other places of the program by using below syntax.

struct structure-tag-name structure-variables;

Eg: struct stu s1, s2;

Memory allocation for structure variables



```
struct   stu
{
int id;
char name[20];
float fee;
}
s1, s2;
```

s1.id=100;/* calling structure member*/

s2.id=200;

Eg:Direct initialization of structure members:

It is the process of passing values for structure members, without using scanf() at design time using =.

Note: In direct initialization of structure members, the passing values datatype and structure members datatype should be matched.

When all the structure members are not initialized, they will store the default values as follows.

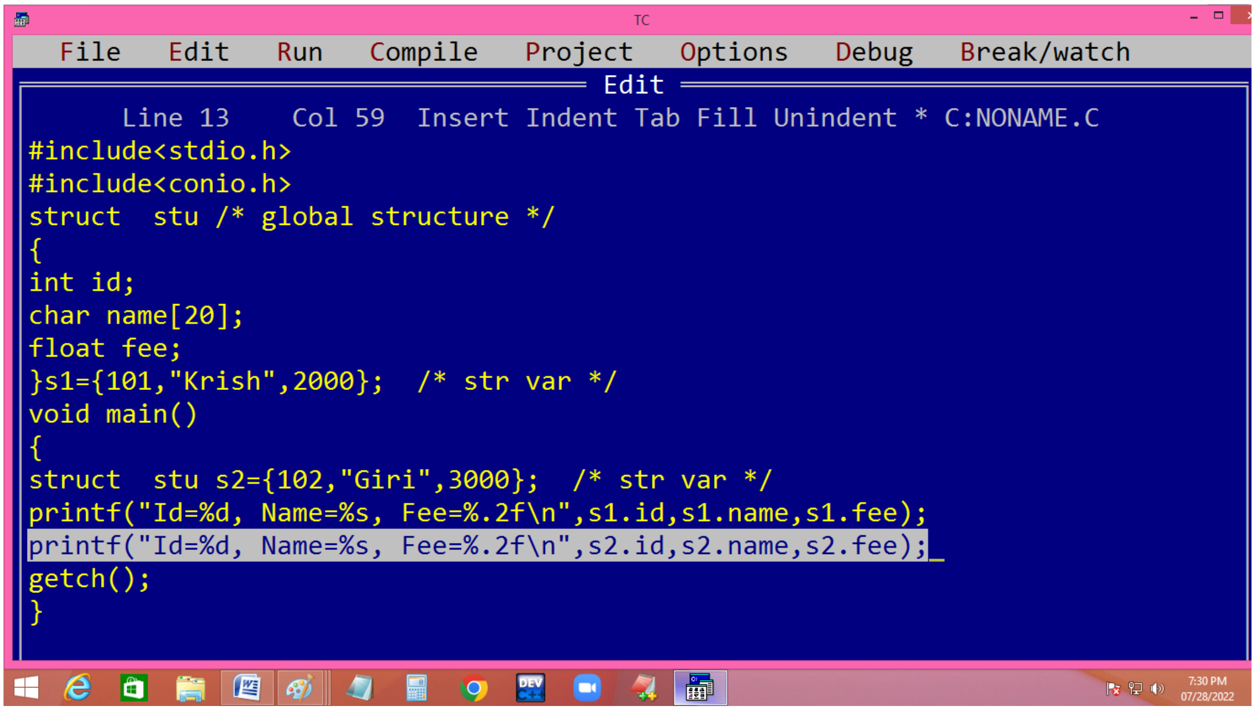
Int – 0

Float – 0.000000

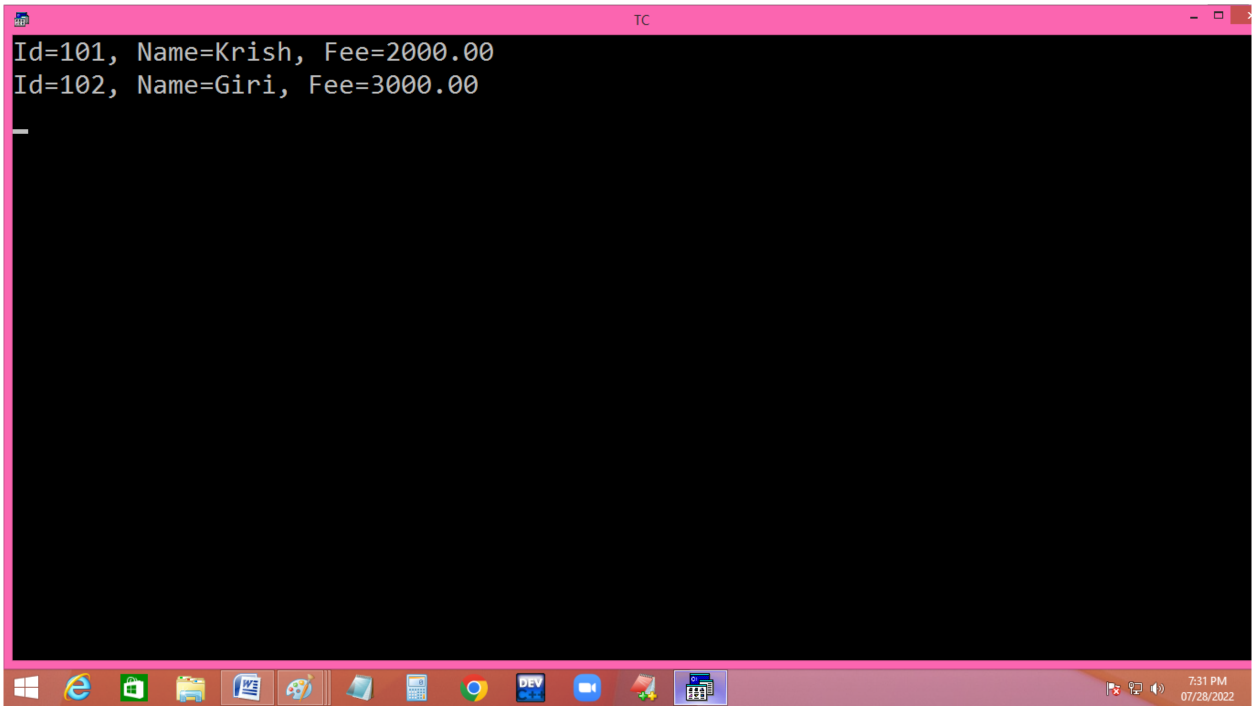
Char – blank space

structures stores 0, 0.00 and blank in int, float and char.

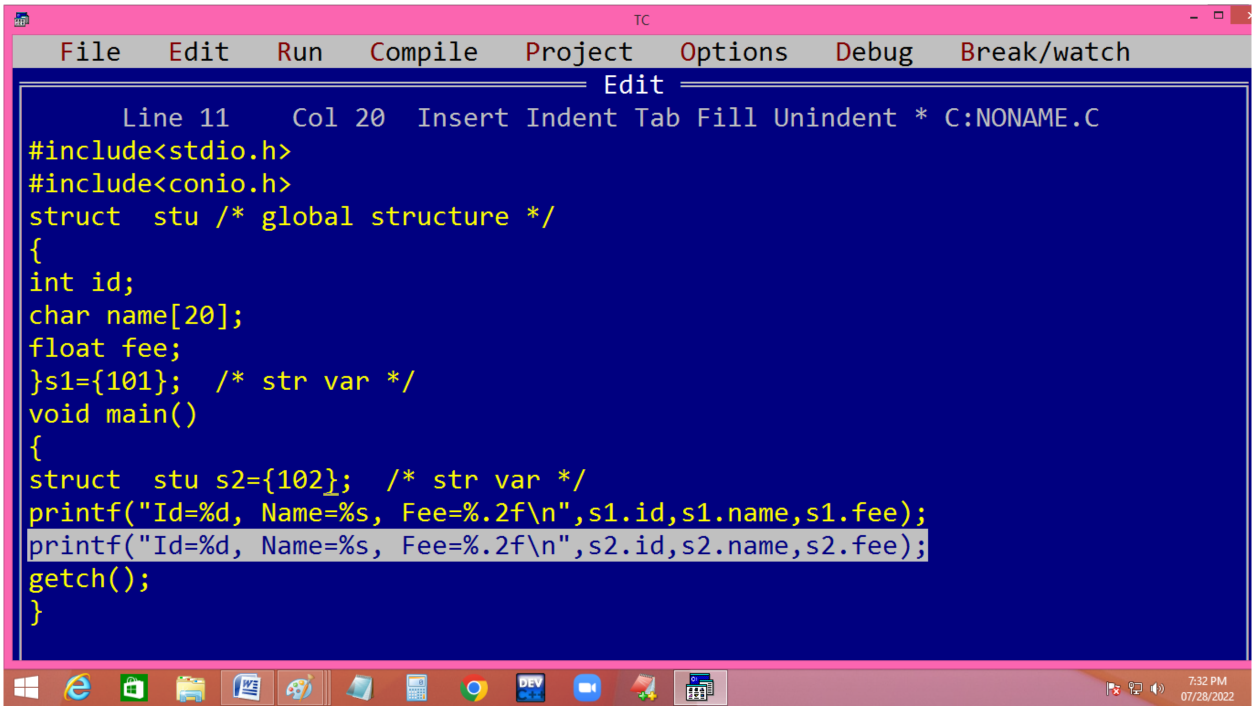
when it is a local structure, without initialization, stores garbage values.



```
#include<stdio.h>
#include<conio.h>
struct  stu /* global structure */
{
int id;
char name[20];
float fee;
}s1={101,"Krish",2000};  /* str var */
void main()
{
struct  stu s2={102,"Giri",3000};  /* str var */
printf("Id=%d, Name=%s, Fee=%.2f\n",s1.id,s1.name,s1.fee);
printf("Id=%d, Name=%s, Fee=%.2f\n",s2.id,s2.name,s2.fee);
getch();
}
```

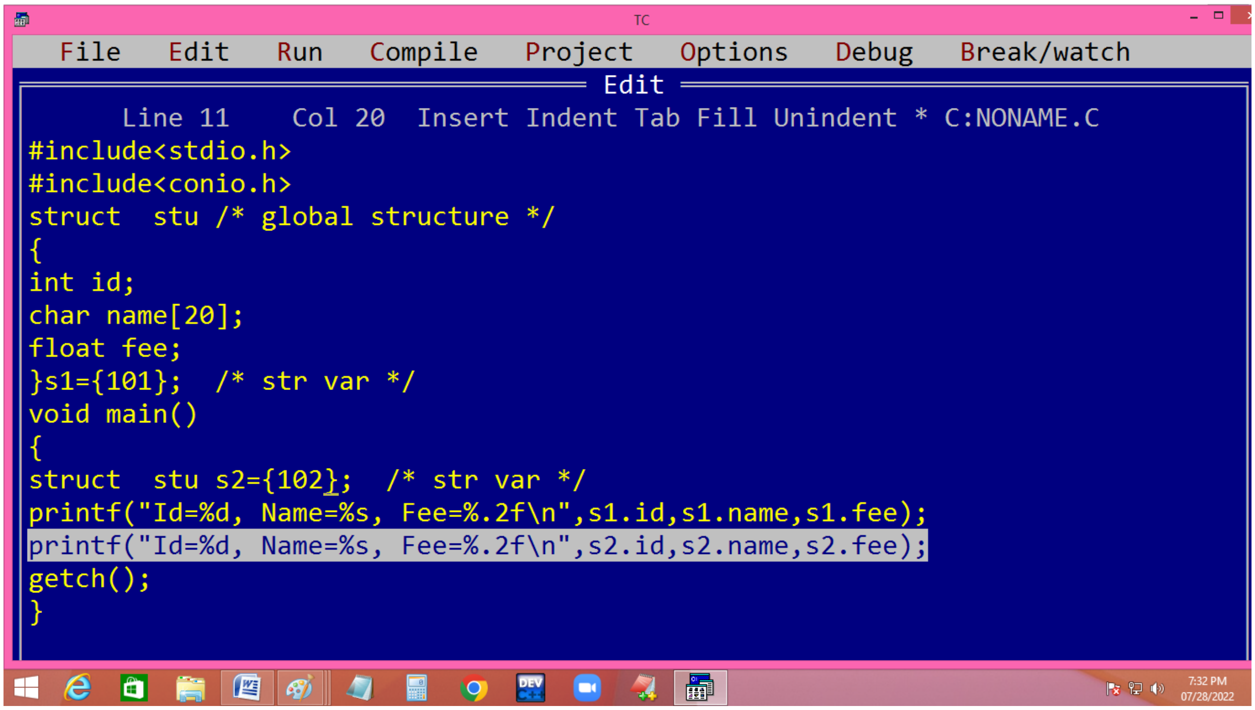


```
Id=101, Name=Krish, Fee=2000.00
Id=102, Name=Giri, Fee=3000.00
```



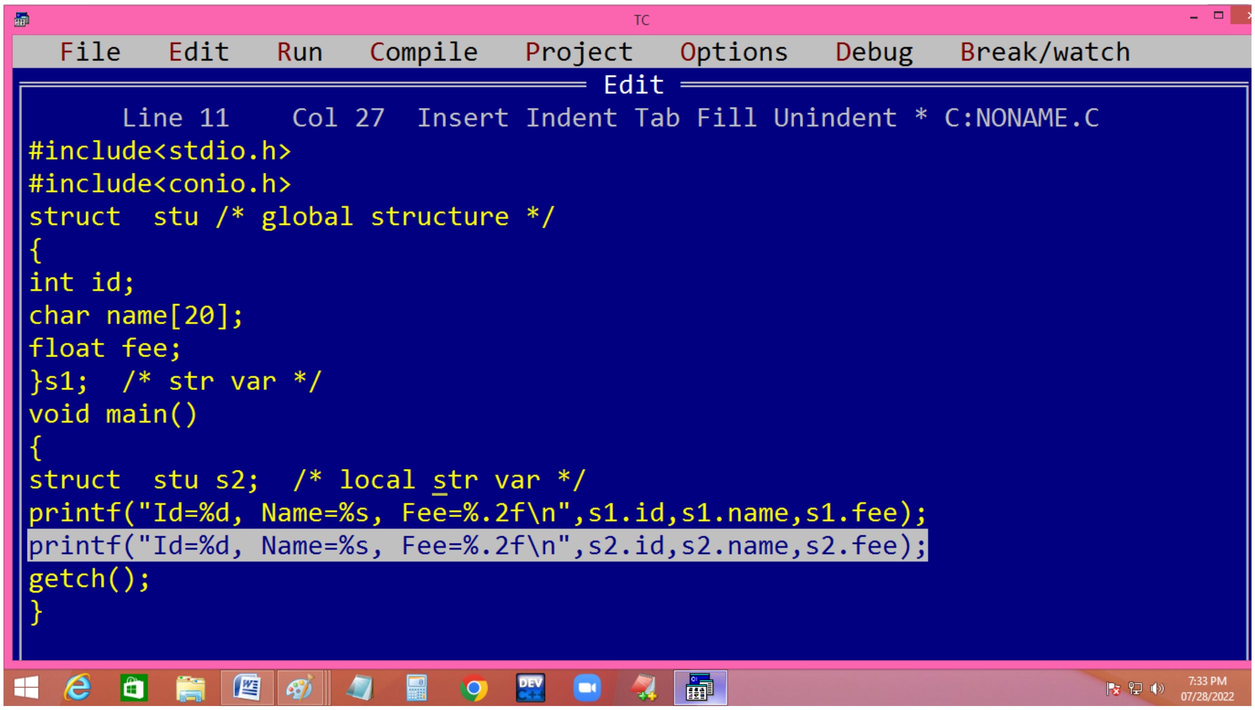
TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 11 Col 20 Insert Indent Tab Fill Unindent * C:NONAME.C
7:32 PM
07/28/2022

```
#include<stdio.h>
#include<conio.h>
struct  stu /* global structure */
{
int id;
char name[20];
float fee;
}s1={101};  /* str var */
void main()
{
struct  stu s2={102};  /* str var */
printf("Id=%d, Name=%s, Fee=%.2f\n",s1.id,s1.name,s1.fee);
printf("Id=%d, Name=%s, Fee=%.2f\n",s2.id,s2.name,s2.fee);
getch();
}
```

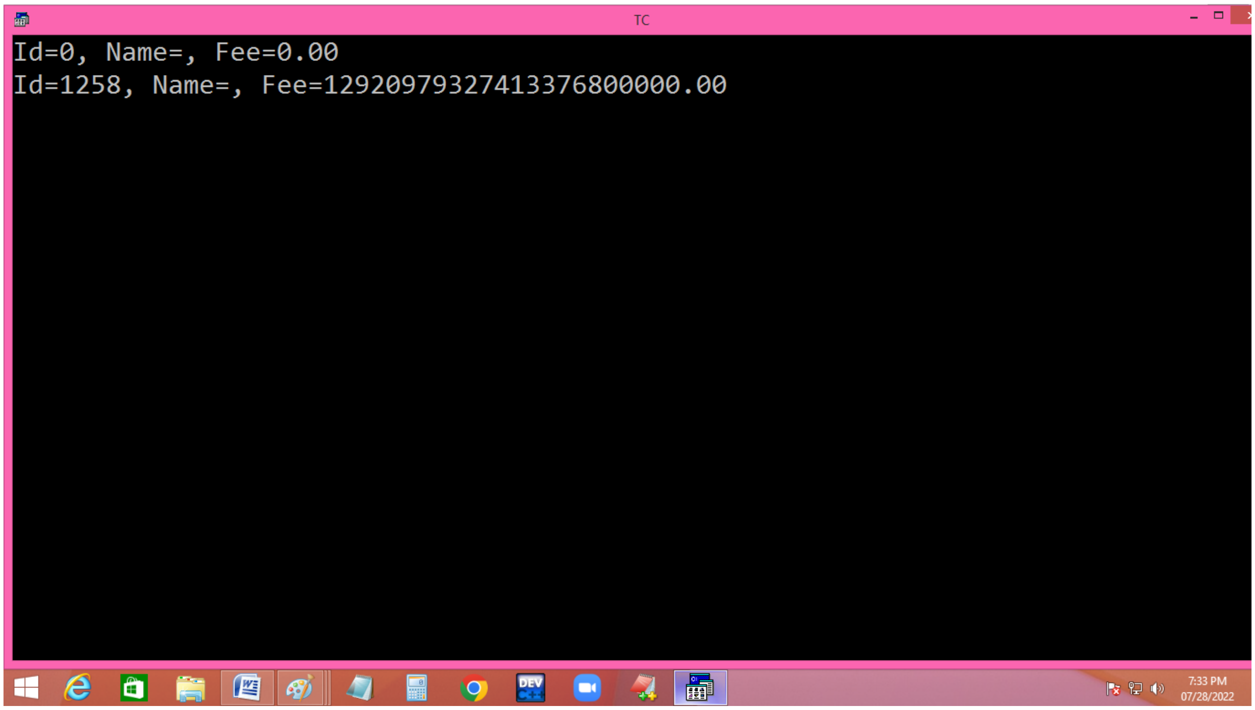


TC
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 11 Col 20 Insert Indent Tab Fill Unindent * C:NONAME.C
7:32 PM
07/28/2022

```
#include<stdio.h>
#include<conio.h>
struct  stu /* global structure */
{
int id;
char name[20];
float fee;
}s1={101};  /* str var */
void main()
{
struct  stu s2={102};  /* str var */
printf("Id=%d, Name=%s, Fee=%.2f\n",s1.id,s1.name,s1.fee);
printf("Id=%d, Name=%s, Fee=%.2f\n",s2.id,s2.name,s2.fee);
getch();
}
```

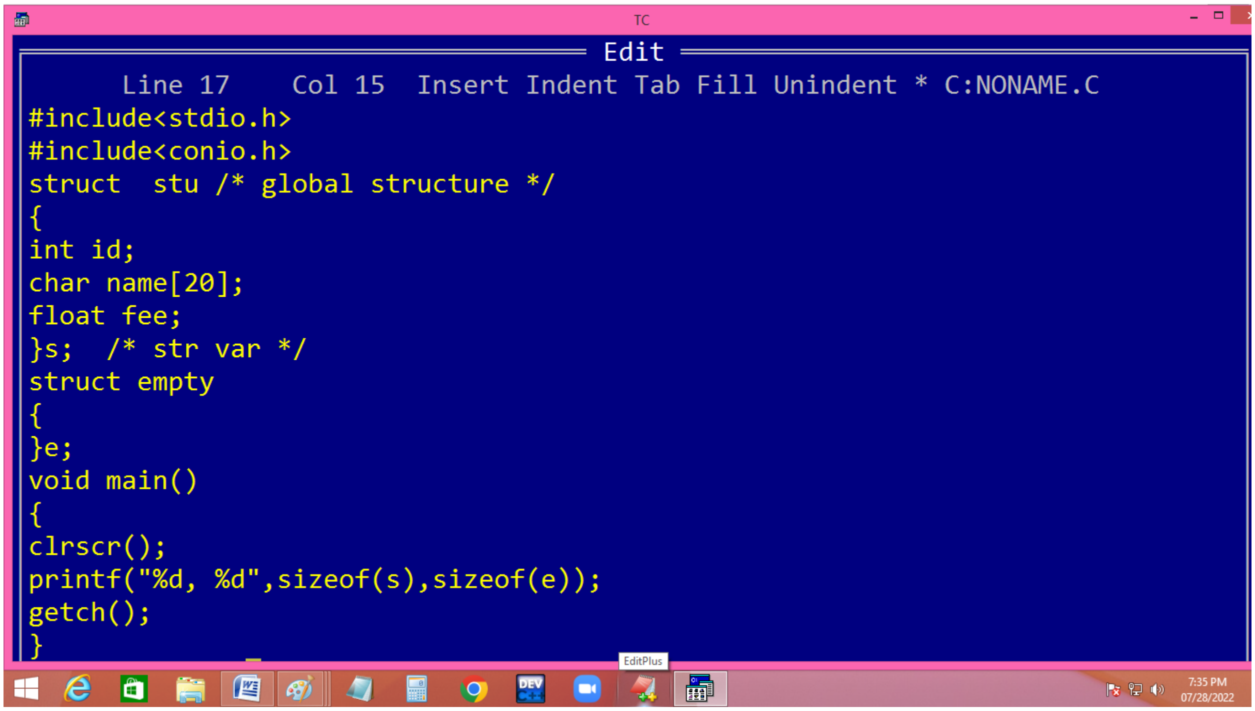


```
#include<stdio.h>
#include<conio.h>
struct  stu /* global structure */
{
int id;
char name[20];
float fee;
}s1;  /* str var */
void main()
{
struct  stu s2;  /* local str var */
printf("Id=%d, Name=%s, Fee=%.2f\n",s1.id,s1.name,s1.fee);
printf("Id=%d, Name=%s, Fee=%.2f\n",s2.id,s2.name,s2.fee);
getch();
}
```

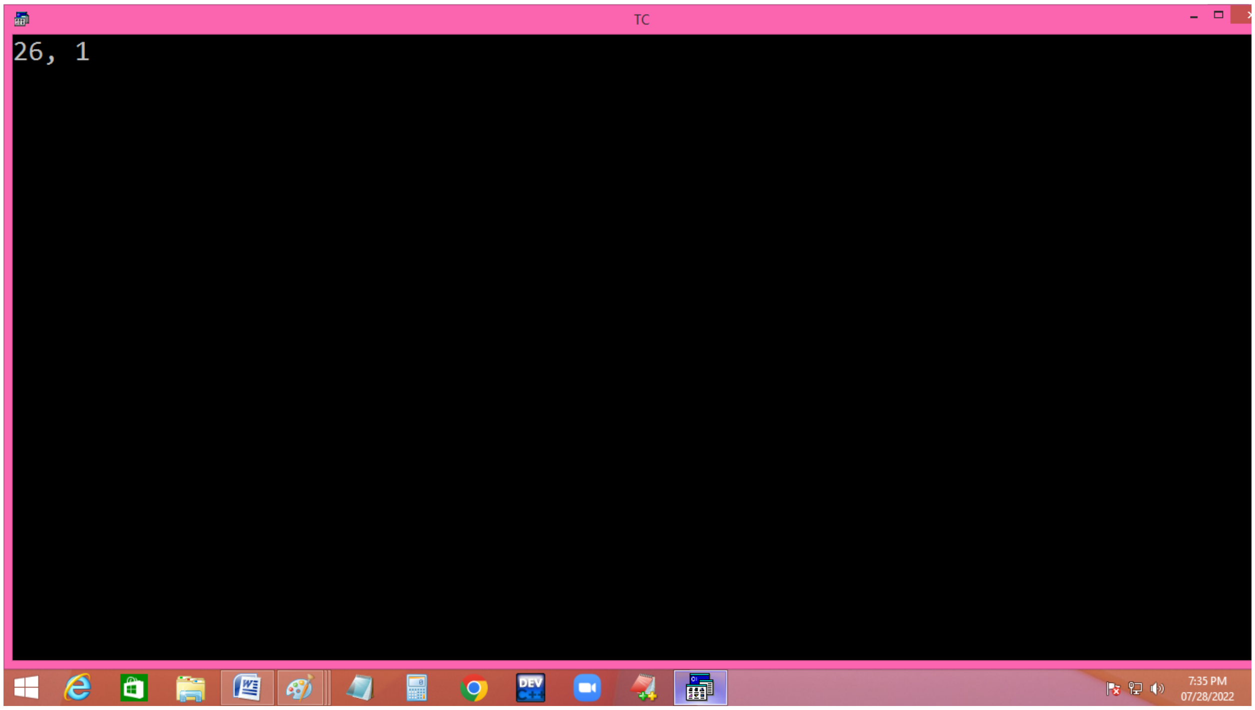


```
Id=0, Name=, Fee=0.00
Id=1258, Name=, Fee=129209793274133768000000.00
```

Finding structure size:



```
      Line 17    Col 15  Insert Indent Tab Fill Unindent * C:NONAME.C
#include<stdio.h>
#include<conio.h>
struct  stu /* global structure */
{
int id;
char name[20];
float fee;
}s;  /* str var */
struct empty
{
}e;
void main()
{
clrscr();
printf("%d, %d",sizeof(s),sizeof(e));
getch();
}
```



```
26, 1
```