Structures

A structure is a derived data type. It is constructed using objects of other types.

A structure is a collection of related variables under one name. It may contain variables of many different data types.

Structures are used to define objects and their attributes.

Here is the syntax for the structure definition:

```
struct structure_tag {
        data_type var_name;  }
                             }  Data members
        …                    }
};
```

Where:

Keyword **struct** – introduces the structure definition.

Structure tag – names the structured definition and is used with the keyword **struct** to declare variables of the structure type.

Structure Members – the variables declared within braces of the structure definition. These members are the attributes of the structure.

For example,

Define the structure definition of a student.

First: Identify the possible attributes of a student.

Id number, name, age, tuition fee

Second: Define the structure.

```
struct Student{
      int idnumber;
      char name[30];
      int age;
      float tuitionfee;
};
```

By convention, the name of the structure always starts with a big letter.

Note: Members of the same structures must have unique names, but two different structures may contain members of the same name without conflict.

For example,

Student has attributes id number, name, age and tuition fee.

Employee has attributes id number, name, age and salary.

Based on the comparison, both objects have the same attributes (i.e. id number, age and name) and only differ on the fourth attribute that is tuition fee for student and salary for employee.

For the structure definitions:

```
struct Student{
    int idnumber;
    char name[30];
    int age;
    float tuitionfee;
};


struct Employee{
    int idnumber;
    char name[30];
    int age;
    float salary;
};
```

These two structures will work without conflict since to refer to a structure member, you need to refer first to the structure where it belongs to.

Declaration of Structure Variable

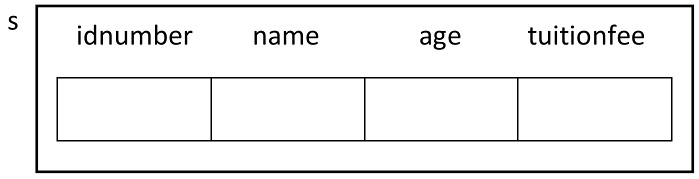
Here is the syntax:

```
struct structure_tag var_name;
```

For example,

```
struct Student s;
```

Variable s is the structure variable. To declare, always specify the name of the structure with the struct keyword. With the above example, s is allocated memory with the size of struct Student. Below is an illustration of s If using the previous definition of struct Student.



The Use of typedef

Since to declare a structure variable you always need to have the struct keyword and the structure tag, there would always be two words before the variable. So to simplify, we can use typedef.

The keyword **typedef** provides a mechanism for creating synonyms (or aliases) for previously defined data types. There are three ways to use typedef for structures. Here are the examples:

1. `typedef struct Student Stude;`
2.
```
typedef struct Student{

        int idnumber;

        char name[30];

        int age;

        float tuitionfee;

}Stude;
```
3.
```
typedef struct {

        int idnumber;

        char name[30];

        int age;

        float tuitionfee;

}Stude;
```

When using the third example, the structure tag becomes optional. In this way, there is only one name to use in declaring a structure variable, that is Stude.

From the above examples, Stude is not a variable name but an alias for struct Student. So instead of declaring,

```
struct Student s;
```

We can use,

```
Stude s;
```

Note: Creating a new name with typedef does not create a new data type; typedef simply creates a new type name, which may be used as an alias for an existing type names.

Initializing Structures

Structures can be initialized using initializer lists as with arrays. To initialize a structure, follow the variable name in the structure declaration with an equal sign and a brace-enclosed, comma-separated list of initializers.

For example,

```
Stude s = {101, “Jose Rizal”, 18, 24500.25};
```

To illustrate:

idnumber	name	age	tuitionfee
101	Jose Rizal	18	24500.25

However, remember to initialize with values that correctly match the data type. Otherwise, an error may occur.

Accessing Structures Members

To access the data members, two operators can be used:

1. dot operator (.)

   The dot operator is used to directly access the structure member.

   Here is the syntax:

   structureVariable.memberName

   For example,

   ```
   s.age = 20;
   strcpy(s.name,“Marlowe”);
   printf(“%d”,s.age);
   printf(“%s”,s.name);
   ```

2. structure pointer operator / arrow operator (->)

   The arrow operator is used when using a pointer to indirectly access the structure members.

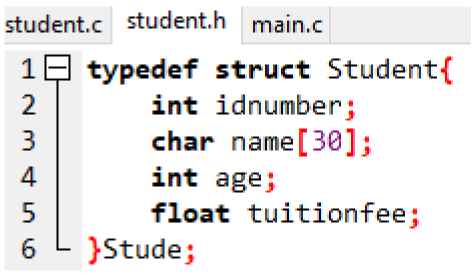
   Here is the syntax:

   structurePointer->memberName

   For example:

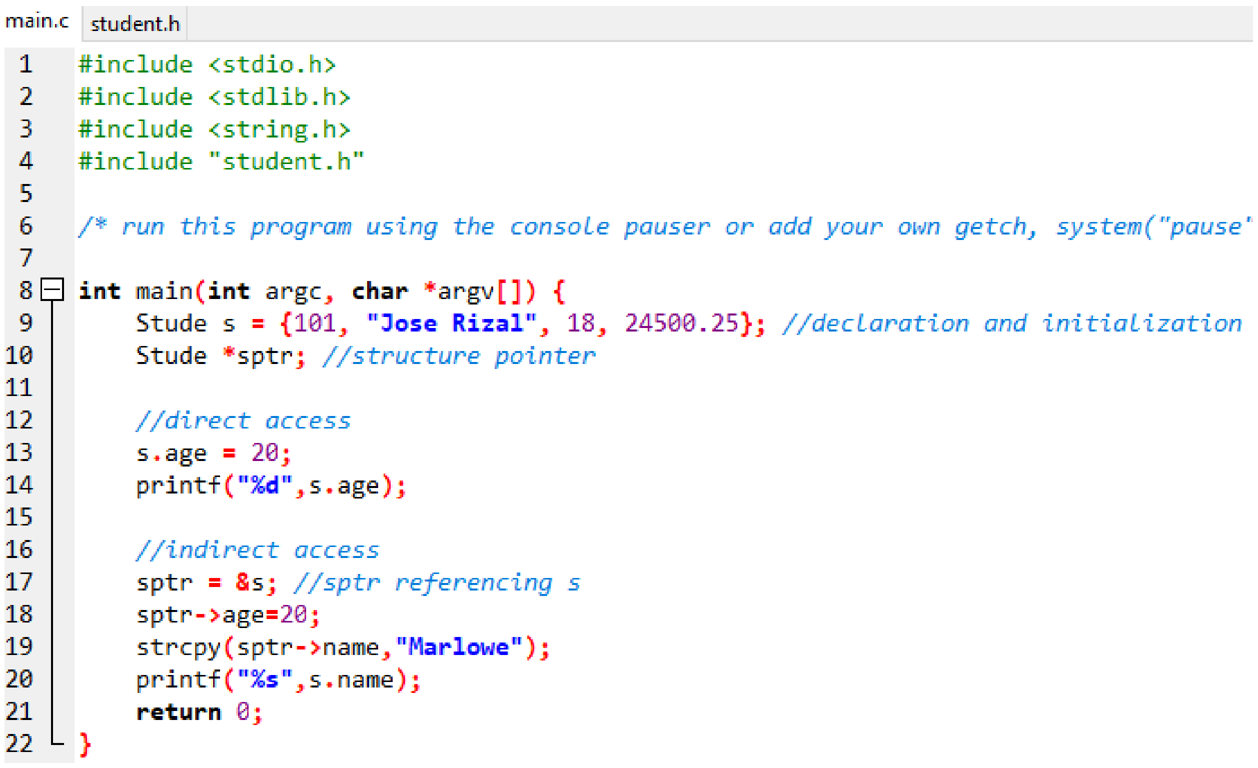
   ```
   struct Student s, *sptr;
   sptr=&s;
   sptr->age=20;
   strcpy(sptr->name,”Marlowe”);
   ```

Here is the summary on the previous discussion:



student.c | student.h | main.c

```
typedef struct Student{
	int idnumber;
	char name[30];
	int age;
	float tuitionfee;
}Stude;
```



main.c | student.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "student.h"

/* run this program using the console pauser or add your own getch, system("pause'

int main(int argc, char *argv[]) {
	Stude s = {101, "Jose Rizal", 18, 24500.25}; //declaration and initialization
	Stude *sptr; //structure pointer

	//direct access
	s.age = 20;
	printf("%d",s.age);

	//indirect access
	sptr = &s; //sptr referencing s
	sptr->age=20;
	strcpy(sptr->name,"Marlowe");
	printf("%s",s.name);
	return 0;
}
```

Nested Structures

It is possible to place structures inside another structure.

For example,

```
struct Birth{

      int month, day, year;

};

struct Stud{

      char name[20];

      int age;
```

```
        float grade;

        struct Birth bday;

    };

    struct Stud s;
```

Structure variable bday is of type struct Birth. How can we access an inner structure variable?

For example, to access day,

```
    s.bday.day = 6;

    printf("%d", s.bday.day);
```

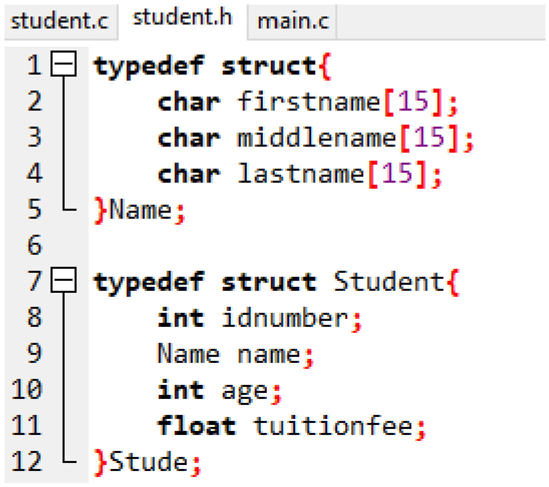
Here is a sample exercise. Consider the following structures:

```
    struct name{

        char fname[30], lname[30], mi[2];

    };

    struct birth{

        int month, day, year;

    };

    struct bio{

        struct name Nm;

        struct birth Bd;

        int idno;

    };

    struct bio x;
```

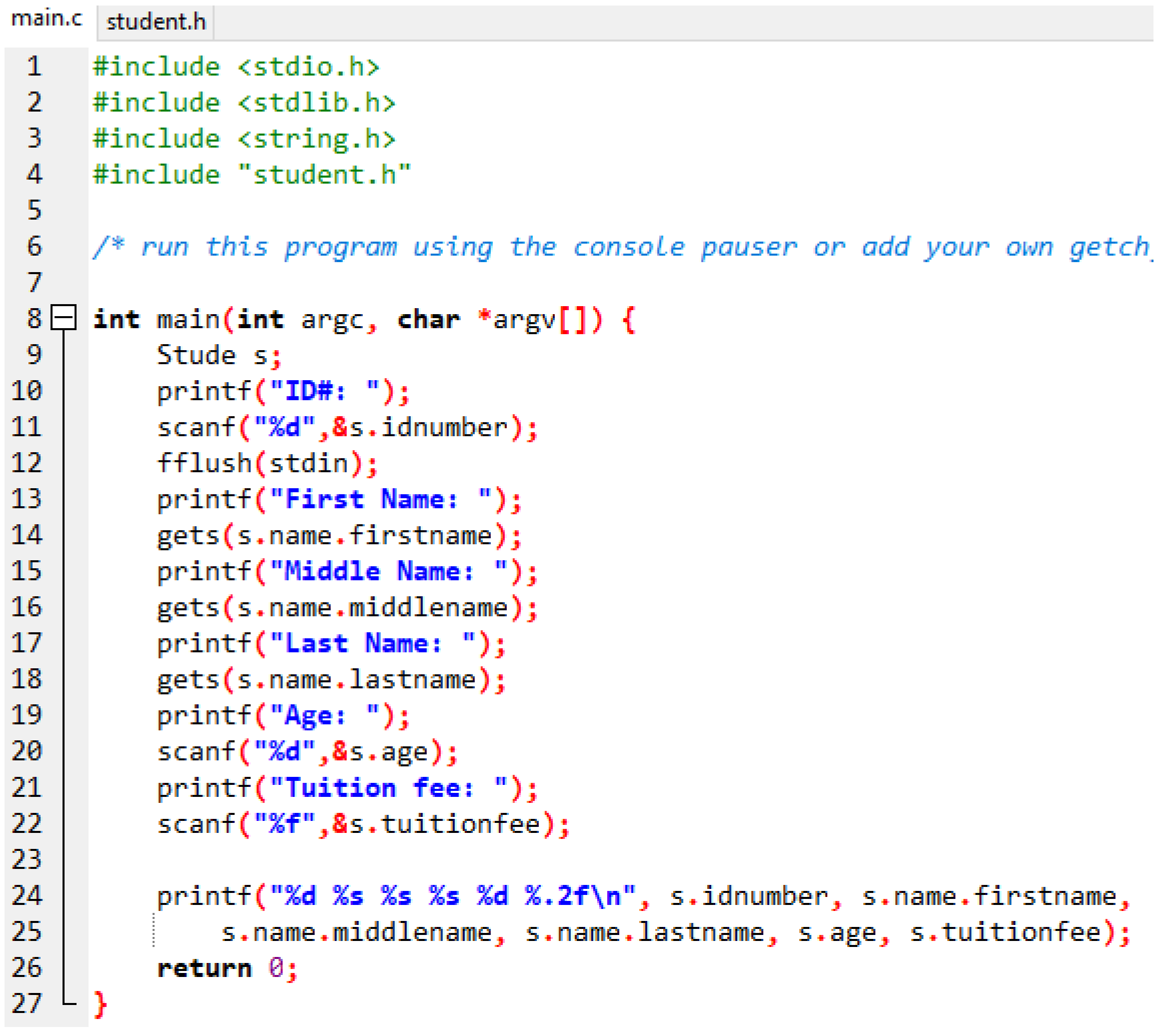
Identify if the following statements are valid or not:

1. x.bio.idno=1012;	//invalid
2. strcpy(x.Nm.fname, "Christopher");	//valid
3. strcpy(x.name.lname, "Smith");	//invalid
4. x.Bd.month=6;	//valid
5. bio.birth.year=1975;	//invalid

Here is a sample program on the use of nested structures:



```c
typedef struct{
    char firstname[15];
    char middlename[15];
    char lastname[15];
}Name;

typedef struct Student{
    int idnumber;
    Name name;
    int age;
    float tuitionfee;
}Stude;
```

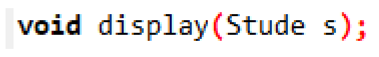


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "student.h"

/* run this program using the console pauser or add your own getch

int main(int argc, char *argv[]) {
    Stude s;
    printf("ID#: ");
    scanf("%d",&s.idnumber);
    fflush(stdin);
    printf("First Name: ");
    gets(s.name.firstname);
    printf("Middle Name: ");
    gets(s.name.middlename);
    printf("Last Name: ");
    gets(s.name.lastname);
    printf("Age: ");
    scanf("%d",&s.age);
    printf("Tuition fee: ");
    scanf("%f",&s.tuitionfee);

    printf("%d %s %s %s %d %.2f\n", s.idnumber, s.name.firstname,
        s.name.middlename, s.name.lastname, s.age, s.tuitionfee);
    return 0;
}
```

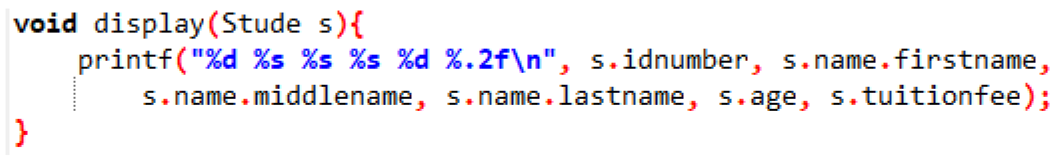
Using Structures with Functions

When structures or individual structure members are passed to functions, they are passed by value.

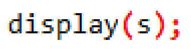
Here is an example:



```c
void display(Stude s);
```



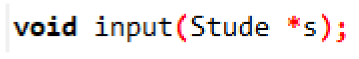
```c
void display(Stude s){
    printf("%d %s %s %s %d %.2f\n", s.idnumber, s.name.firstname,
        s.name.middlename, s.name.lastname, s.age, s.tuitionfee);
}
```



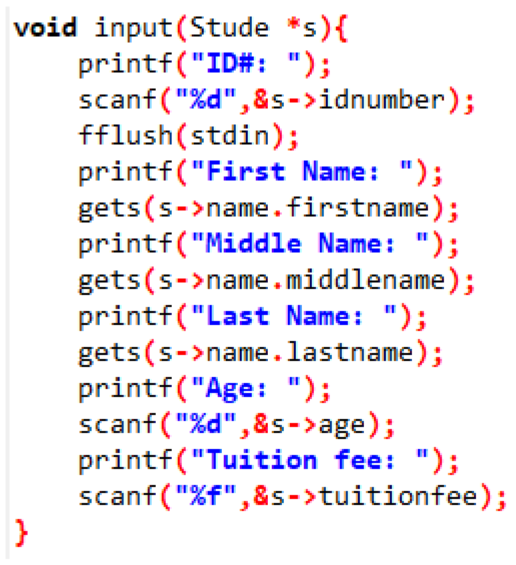
```c
display(s);
```

To pass a structure call by reference, pass the address of the structure variable.

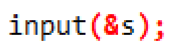
Here is an example:



```c
void input(Stude *s);
```



```c
void input(Stude *s){
    printf("ID#: ");
    scanf("%d",&s->idnumber);
    fflush(stdin);
    printf("First Name: ");
    gets(s->name.firstname);
    printf("Middle Name: ");
    gets(s->name.middlename);
    printf("Last Name: ");
    gets(s->name.lastname);
    printf("Age: ");
    scanf("%d",&s->age);
    printf("Tuition fee: ");
    scanf("%f",&s->tuitionfee);
}
```



```c
input(&s);
```

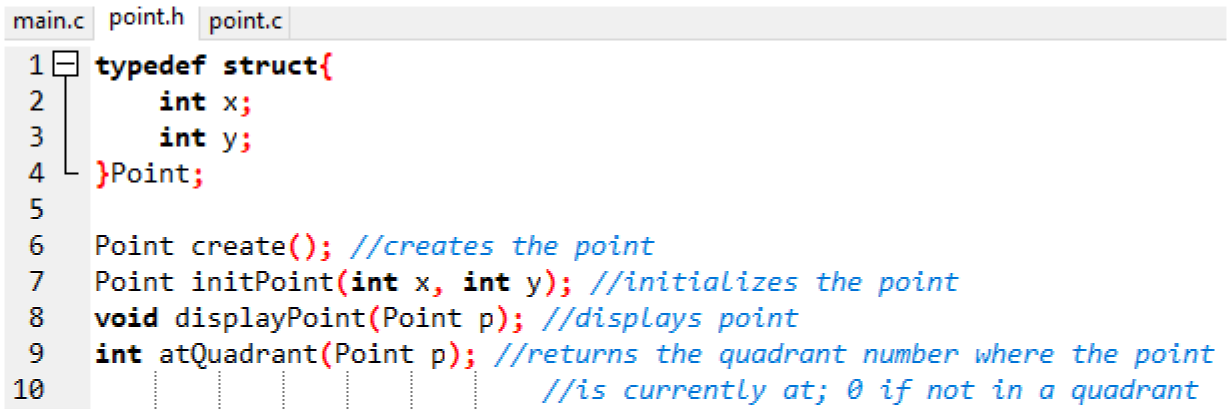
From the above example, since input() has to accept new values for s then it needs to pass s by reference. Thus, the function uses a pointer as its parameter and the arrow operator to access the elements.

More Examples

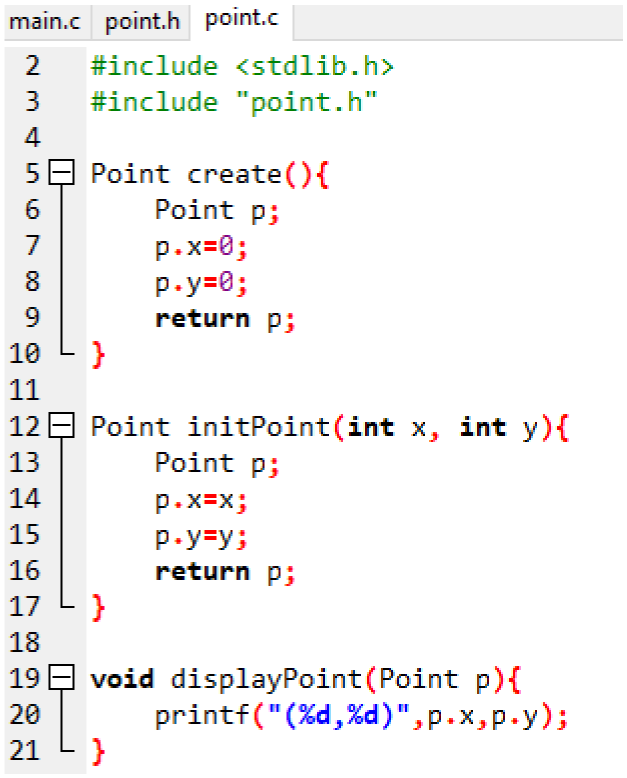
A point in a plane can be represented by its two coordinates, x and y. Therefore, we can represent a point in a plane by a structure having two fields as shown below.

```c
typedef struct{
        int x;
        int y;
    }Point;
```

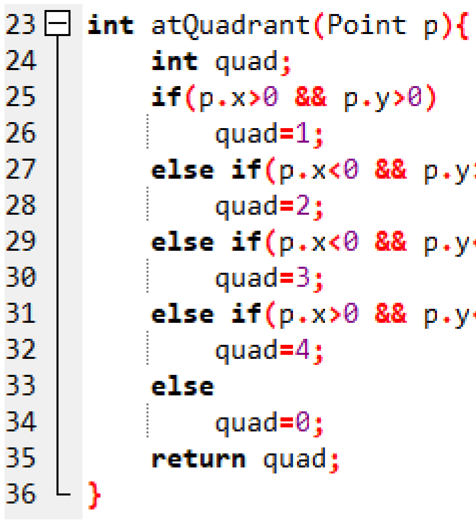
Define structure Point.



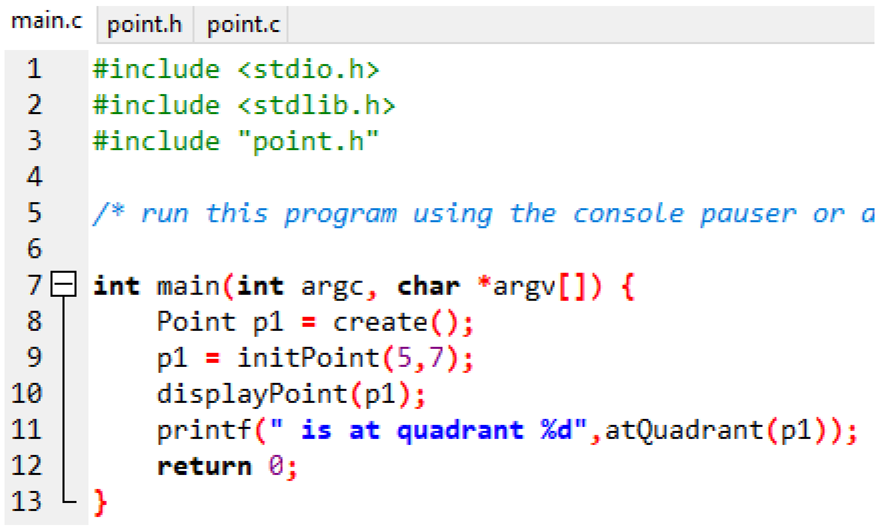
```c
typedef struct{
    int x;
    int y;
}Point;

Point create(); //creates the point
Point initPoint(int x, int y); //initializes the point
void displayPoint(Point p); //displays point
int atQuadrant(Point p); //returns the quadrant number where the point
                           //is currently at; 0 if not in a quadrant
```



```c
#include <stdlib.h>
#include "point.h"

Point create(){
    Point p;
    p.x=0;
    p.y=0;
    return p;
}

Point initPoint(int x, int y){
    Point p;
    p.x=x;
    p.y=y;
    return p;
}

void displayPoint(Point p){
    printf("(%d,%d)",p.x,p.y);
}
```



```c
int atQuadrant(Point p){
	int quad;
	if(p.x>0 && p.y>0)
		quad=1;
	else if(p.x<0 && p.y>0)
		quad=2;
	else if(p.x<0 && p.y<0)
		quad=3;
	else if(p.x>0 && p.y<0)
		quad=4;
	else
		quad=0;
	return quad;
}
```



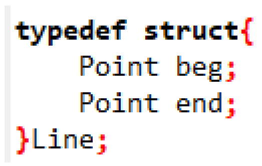
main.c point.h point.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "point.h"

/* run this program using the console pauser or a

int main(int argc, char *argv[]) {
	Point p1 = create();
	p1 = initPoint(5,7);
	displayPoint(p1);
	printf(" is at quadrant %d",atQuadrant(p1));
	return 0;
}
```

A straight line is an object connecting two points. Therefore, a line can be represented by a nested structure having two structures of the type POINT.

```
typedef struct{
	Point beg;
	Point end;
}Line;
```

As a continuation, define structure Line.



```c
typedef struct{
	Point beg;
	Point end;
}Line;
```

```c
Line createLine();  //creates line
Line initLine(Point p1, Point p2); //initializes line
void displayLine(Line l1); //displays line
int lineType(Line l1); //determines whether the line is 1-horizontal,
                         //2-vertical, 3-oblique

Line createLine(){
    Point p1=create(), p2=create();
    Line l;
    l.beg=p1;
    l.end=p2;
    return l;
}

Line initLine(Point p1, Point p2){
    Line l;
    l.beg=p1;
    l.end=p2;
    return l;
}

void displayLine(Line l1){
    printf("[");
    displayPoint(l1.beg);
    displayPoint(l1.end);
    printf("]");
}

int lineType(Line l1){
    int type;
    if(l1.beg.y==l1.end.y)
        type=1;
    else if(l1.beg.x==l1.end.x)
        type=2;
    else
        type=3;
    return type;
}
```

```
int main(int argc, char *argv[]) {
    Line l1 = createLine();
    Point p1 = initPoint(3,10);
    Point p2 = initPoint(2,6);
    l1 = initLine(p1, p2);
    displayLine(l1);
    printf(" = %d",lineType(l1));
    return 0;
}
```

Practice Exercise (Ungraded)

1. The circle has two data members, a Point representing the center of the circle and a float value representing the radius as shown below.

```
typedef struct{
      Point center;
      float radius;
}Circle;
```

Implement the following functions:

a. `float distance(LINE l1);`
- computes and returns the distance between the two points of the line.
  distance = sqrt((x2-x1)^2 + (y2-y1)^2)

b. `float diameter(Circle circ);`
- computes the diameter of a circle.

2. The Rectangle has two data members upperLeft and lowerRight points, that represents upper left and lower right coordinates, respectively. Therefore a rectangle can be represented by a nested structure having two structures of the type POINT (as defined in our example last meeting) as defined below.

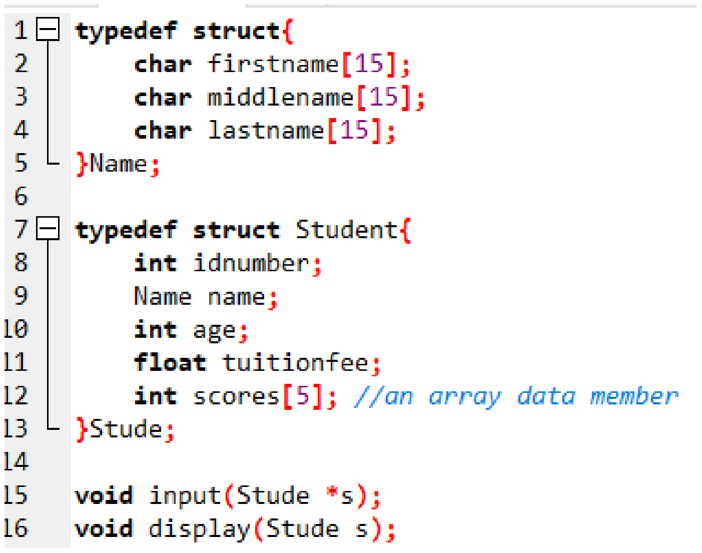
```
typedef struct{
      Point upperLeft;
      Point lowerRight;
}Rectangle;
```

Implement the following functions:

```
a. float length(Rectangle rec);  //returns the length of the rectangle
b. float width(Rectangle rec);  //returns the width of the rectangle
c. float area(Rectangle rec);  //returns the area of the rectangle
d. float perimeter(Rectangle rec);  //returns the area of the rectangle
```

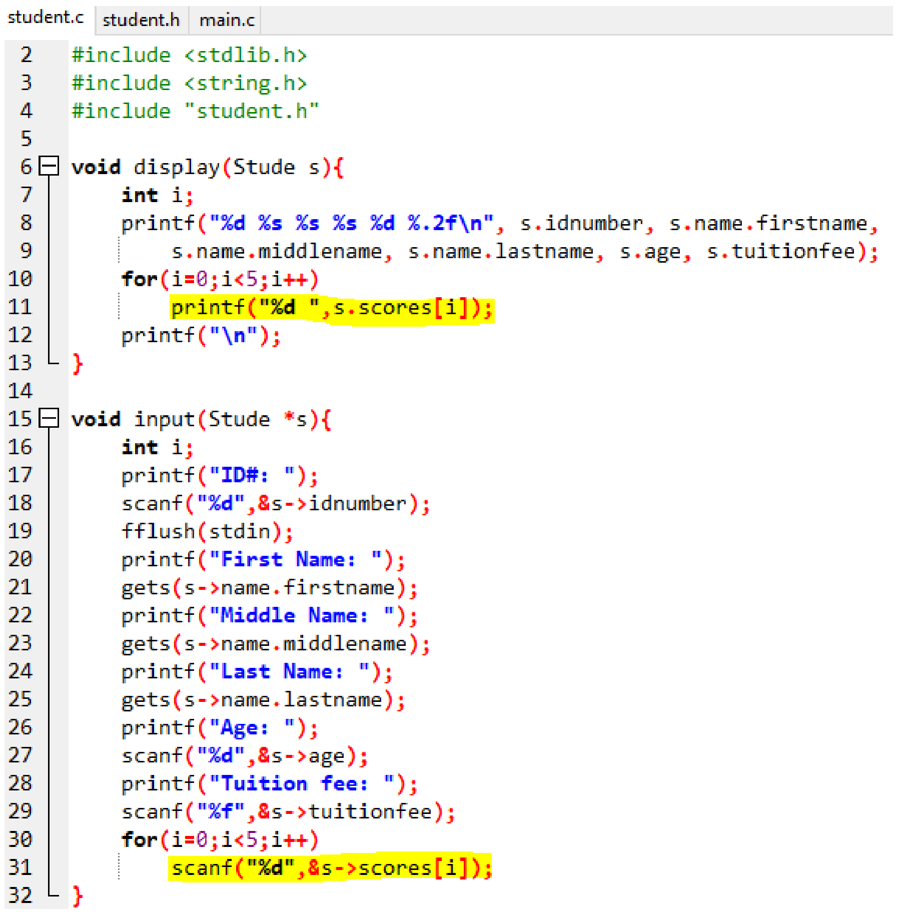
Structure with Array Member

Here is an example on a structure with an array member. Let us modify the previous example on structure Student.



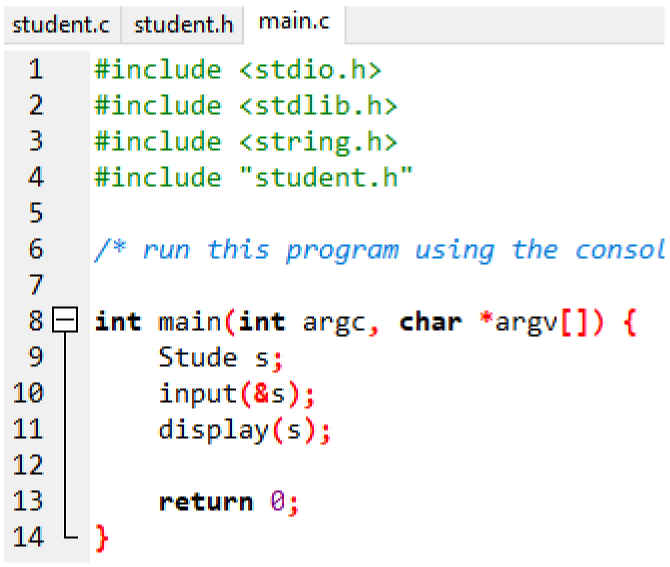
```c
typedef struct{
     char firstname[15];
     char middlename[15];
     char lastname[15];
}Name;

typedef struct Student{
     int idnumber;
     Name name;
     int age;
     float tuitionfee;
     int scores[5]; //an array data member
}Stude;

void input(Stude *s);
void display(Stude s);
```

In the above definition of Student, variable scores is an array.



```c
#include <stdlib.h>
#include <string.h>
#include "student.h"

void display(Stude s){
    int i;
    printf("%d %s %s %s %d %.2f\n", s.idnumber, s.name.firstname,
        s.name.middlename, s.name.lastname, s.age, s.tuitionfee);
    for(i=0;i<5;i++)
        printf("%d ",s.scores[i]);
    printf("\n");
}

void input(Stude *s){
    int i;
    printf("ID#: ");
    scanf("%d",&s->idnumber);
    fflush(stdin);
    printf("First Name: ");
    gets(s->name.firstname);
    printf("Middle Name: ");
    gets(s->name.middlename);
    printf("Last Name: ");
    gets(s->name.lastname);
    printf("Age: ");
    scanf("%d",&s->age);
    printf("Tuition fee: ");
    scanf("%f",&s->tuitionfee);
    for(i=0;i<5;i++)
        scanf("%d",&s->scores[i]);
}
```

The highlighted statements in the above program are examples on how to access an array as a data member.



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "student.h"

/* run this program using the consol

int main(int argc, char *argv[]) {
	Stude s;
	input(&s);
	display(s);

	return 0;
}
```

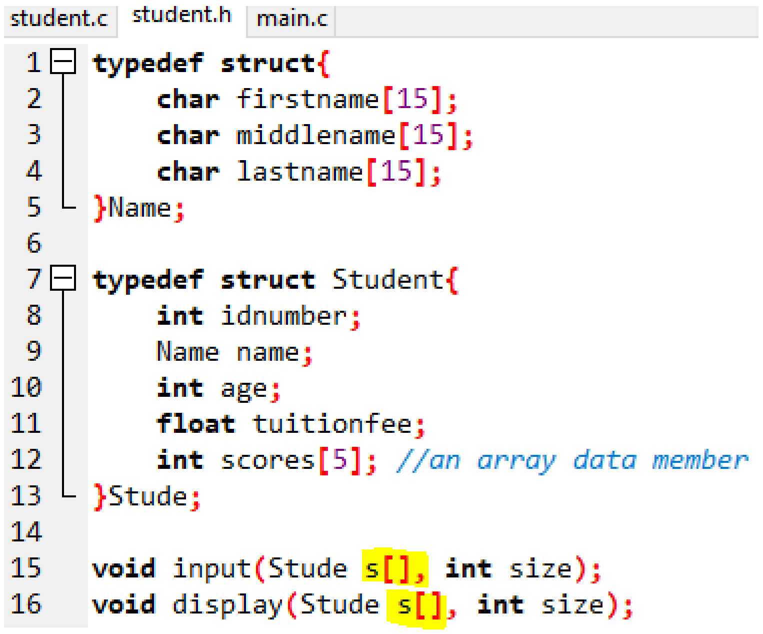
There would be no change in main for the function call if the array is inside the structure since the structure is just one container. Here is an illustration:

idnumber	name	age	tuitionfee	scores				
101	Jose Rizal	18	24500.25	10	8	10	9	9

Array of Structures

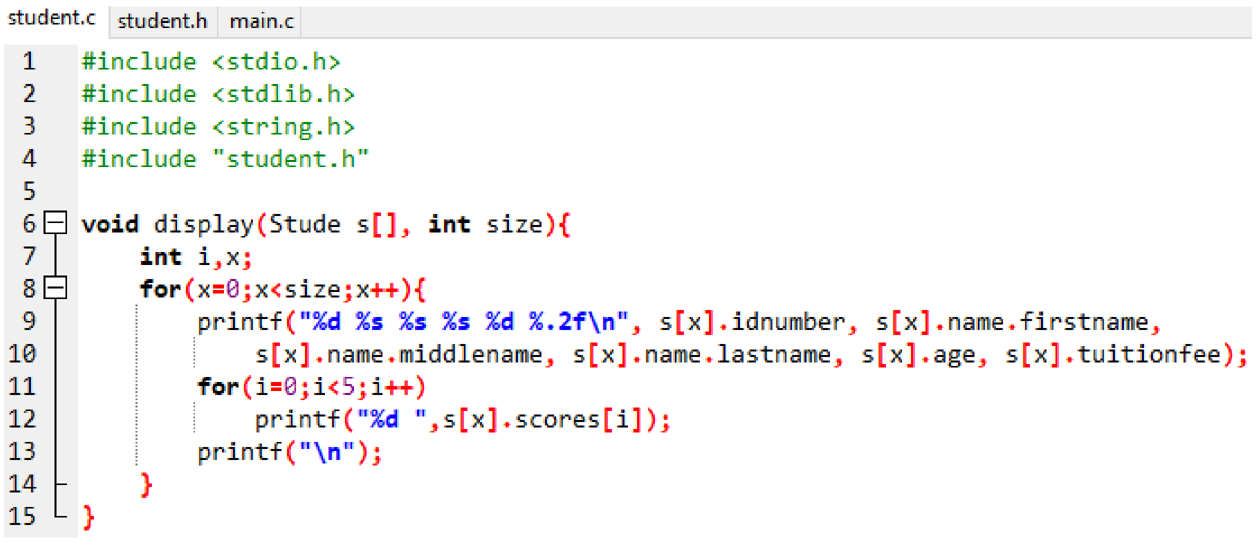
idnumber	name	age	tuitionfee	scores				
101	Jose Rizal	18	24500.25	10	8	10	9	9
idnumber	name	age	tuitionfee	scores				
102	Mari Mar	20	20500.75	5	8	10	9	9
idnumber	name	age	tuitionfee	scores				
103	John Son	19	14506.25	8	8	7	9	9

The above illustration is an example of an array of structures. To create and access an array of structures, here is a sample program, modifying the previous program using the structure Student.

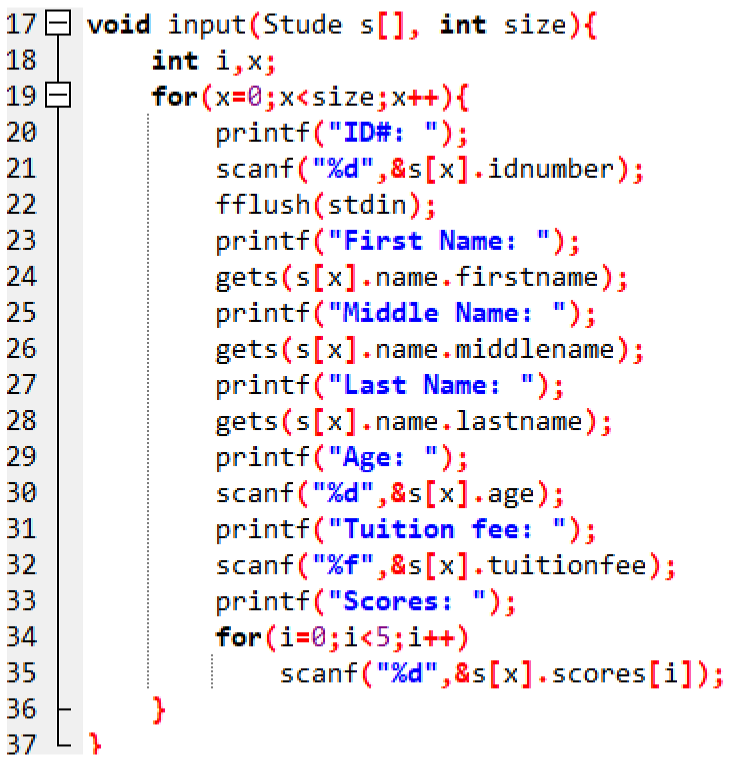


```c
typedef struct{
    char firstname[15];
    char middlename[15];
    char lastname[15];
}Name;

typedef struct Student{
    int idnumber;
    Name name;
    int age;
    float tuitionfee;
    int scores[5]; //an array data member
}Stude;

void input(Stude s[], int size);
void display(Stude s[], int size);
```

In the above function declaration, s is an array of Stude. By convention, when an array is passed to function so as its size.

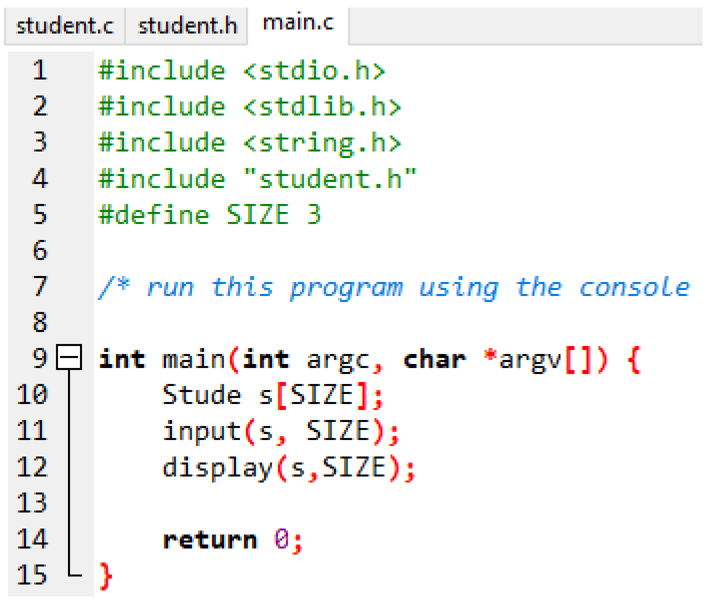


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "student.h"

void display(Stude s[], int size){
    int i,x;
    for(x=0;x<size;x++){
        printf("%d %s %s %s %d %.2f\n", s[x].idnumber, s[x].name.firstname,
            s[x].name.middlename, s[x].name.lastname, s[x].age, s[x].tuitionfee);
        for(i=0;i<5;i++)
            printf("%d ",s[x].scores[i]);
        printf("\n");
    }
}
```



```c
void input(Stude s[], int size){
    int i,x;
    for(x=0;x<size;x++){
        printf("ID#: ");
        scanf("%d",&s[x].idnumber);
        fflush(stdin);
        printf("First Name: ");
        gets(s[x].name.firstname);
        printf("Middle Name: ");
        gets(s[x].name.middlename);
        printf("Last Name: ");
        gets(s[x].name.lastname);
        printf("Age: ");
        scanf("%d",&s[x].age);
        printf("Tuition fee: ");
        scanf("%f",&s[x].tuitionfee);
        printf("Scores: ");
        for(i=0;i<5;i++)
            scanf("%d",&s[x].scores[i]);
    }
}
```

As you can see on the input function, the arrow operator is no longer used.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "student.h"
#define SIZE 3

/* run this program using the console

int main(int argc, char *argv[]) {
    Stude s[SIZE];
    input(s, SIZE);
    display(s,SIZE);

    return 0;
}
```

Remember, since s is an array, then it is implicitly passed by reference.

Graded Exercise

See the attached file Employee Record.