//Friday 27 Novemebr 2020
//Example 1.1 | program to store and access details of one student.

```
#include <stdio.h>
#include <string.h> //for string manipulations

struct student
{
	int id;
	char name[20];
	float percentage;
} ;

int main()
{
	struct student record = {0}; //declaring a var record in main Initializing to null //OPTIONAL
	//struct student record;
	record.id=1;
	strcpy(record.name, "Laurence");
	record.percentage = 95.9;

	printf(" Id is: %d \n", record.id);
	printf(" Name is: %s \n", record.name);
	printf(" Percentage is: %f \n", record.percentage);
	return 0;
}
```

//Example 1.2 | Method 2

```
#include <stdio.h>
#include <string.h>

struct student
{
	 int id;
	 char name[20];
	 float percentage;
} record; //structure variable "record" is declared while declaring structure

int main()
{
```

```
        record.id=1;
        strcpy(record.name, "Laurence");
        record.percentage = 95.5;

        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
        return 0;
    }


//Example 2.
#include <stdio.h>
struct book
{

    char name[7] ; //Max 10 char
    float price ;
    int pages ;

}

b1 = { "Basics", 130.00, 550 }, b2 = { "Physics", 150.80, 800 }, b3 = { 0 }; //Ini-1

//OR

// struct book b1 = { "Basic", 130.00, 550 } ; //Ini-2
// struct book b2 = { "Physics", 150.80, 800 } ;
// struct book b3 = { 0 } ;

// OR

// struct book b1 = { "Basics", 130.00, 550 }, b2 = { "Physics", 150.80, 800 }, b3 = {0} ; //Init-3

int main()

{
    struct book;
    printf("%s %f %d \n", b1.name, b1.price, b1.pages); //Accessing str elements using dot
    printf("%u %u %u \n", &b1.name, &b1.price, &b1.pages); //contiguous mem allocation

    printf("%s %f %d \n", b2.name, b2.price, b2.pages);
```

```
    printf("%u %u %u \n", &b2.name, &b2.price, &b2.pages); //contiguous mem allocation
    //printf(b2.price);

    printf("%s %f %d \n", b3.name, b3.price, b3.pages);
    printf("%u %u %u \n", &b3.name, &b3.price, &b3.pages);

    return 0;
}
```

NOTE:

If a structure variable is initialized to a value { 0 } or {}, then all its elements are set to value 0, as in b3 above. This is a handy way of initializing structure variables. In absence of this, we would have been required to initialize each individual element to a value 0.

```
// Example 3.1 | program to store and access details of multiple student
// Method 1
#include <stdio.h>
#include <string.h>

struct student
{
   int id;
   char name[30];
   float percentage;
};

int main()
{
   int i;
   struct student record1 = {1, "Kent", 90.5}; //two structure variables “record1” and “record2”
   struct student record2 = {2, "Tim", 93.5};

   printf("Records of STUDENT1: \n");
   printf("  Id is: %d \n", record1.id);
   printf("  Name is: %s \n", record1.name);
   printf("  Percentage is: %f \n\n", record1.percentage);

   printf("Records of STUDENT2: \n");
   printf("  Id is: %d \n", record2.id);
   printf("  Name is: %s \n", record2.name);
   printf("  Percentage is: %f \n\n", record2.percentage);

   return 0;
```

```
}
```

//Example 3.2| program to store and access details of multiple student.
//Method 2
Structure array is used

```
#include <stdio.h>
#include <string.h>

struct student
{
    int id;
    char name[50];
    float percentage;
};

int main()
{

    struct student record[3];       //for 3 stds       //Structure Array is used

    // 1st student's record
    record[0].id=1;
    strcpy(record[0].name, "Kent");
    record[0].percentage = 91.3;

    // 2nd student's record
    record[1].id=2;
    strcpy(record[1].name, "Timm");
    record[1].percentage = 89.5;

    // 3rd student's record
    record[2].id=3;
    strcpy(record[2].name, "Dan");
    record[2].percentage = 79.5;

    int i;
    for(i=0; i<3; i++)
    {
       printf("     Records of STUDENT : %d \n", i+1);
       printf(" Id is: %d \n", record[i].id);
       printf(" Name is: %s \n", record[i].name);
       printf(" Percentage is: %f\n\n",record[i].percentage);
```

```
   }
   return 0;
}
```

Passing Structurwe as Function Args

```
#include <stdio.h>
#include <string.h>

struct Books {
  char  title[50];
  char  author[50];
  char  subject[100];
  int   book_id;
};

void printBook( struct Books book ) {

  printf( "Book title : %s\n", book.title);
  printf( "Book author : %s\n", book.author);
  printf( "Book subject : %s\n", book.subject);
  printf( "Book book_id : %d\n", book.book_id);
}
int main( ) {

  struct Books Book1;       /* Declare Book1 of type Book */
  struct Books Book2;       /* Declare Book2 of type Book */

  /* book 1 specification */
  strcpy( Book1.title, "C Programming");
  strcpy( Book1.author, "Tim John");
  strcpy( Book1.subject, "C Tutorial");
  Book1.book_id = 12345;

  /* book 2 specification */
  strcpy( Book2.title, "Tourism");
  strcpy( Book2.author, "Hilbert");
  strcpy( Book2.subject, "Tourism Tutorial");
  Book2.book_id = 56789;

  /* print Book1 info */
  printBook( Book1 );
```

```
   /* Print Book2 info */
   printBook( Book2 );

   return 0;
}
```