```c
#include <stdio.h>

// DEFINING STRUCT
struct MyData
{
    int i;
    float f;
    double d;
};

int main(void)
{
    //function prototypes
    void ChangeValues(struct MyData *);

    //variable declarations
    struct MyData *pData = NULL;

    //code
    printf("\n\n");

    pData = (struct MyData *)malloc(sizeof(struct MyData));
    if (pData == NULL)
    {
        printf("FAILED TO ALLOCATE MEMORY TO 'sturct MyData' !!! EXITTING NOW ... \n\n");
        exit(0);
    }
    else
        printf("SUCCESSFULLY ALLOCATED MEMORY TO 'sturct MyData' !!!\n\n");


    //Assigning Data Values To The Data Members Of 'struct MyData'
    pData->i = 30;
    pData->f = 11.45f;
    pData->d = 1.2995;

    //Displaying Values Of The Data Members Of 'struct MyData'
    printf("\n\n");
    printf("DATA MEMBERS OF 'struct MyData' ARE : \n\n");
    printf("i = %d\n", pData->i);
    printf("f = %f\n", pData->f);
    printf("d = %lf\n", pData->d);

    ChangeValues(pData);

    //Displaying Values Of The Data Members Of 'struct MyData'
    printf("\n\n");
    printf("DATA MEMBERS OF 'struct MyData' ARE : \n\n");
    printf("i = %d\n", pData->i);
    printf("f = %f\n", pData->f);
    printf("d = %lf\n", pData->d);
```

```
52
53      if (pData)
54      {
55          free(pData);
56          pData = NULL;
57          printf("MEMORY ALLOCATED TO 'struct MyData' HAS BEEN SUCCESSFULLY
              FREED !!!\n\n");
58      }
59
60      return(0);
61  }
62
63  void ChangeValues(struct MyData *pParam_Data)
64  {
65      //code
66
67      pParam_Data->i = 9;
68      pParam_Data->f = 8.2f;
69      pParam_Data->d = 6.1998;
70
71      // CAN ALSO DO THIS AS ...
72  /*
73      (*pParam_Data).i = 9;
74      (*pParam_Data).f = 8.2f;
75      (*pParam_Data).d = 6.1998;
76  */
77  }
78
79
80
```