```
1   #include<stdio.h>
2   #include<conio.h>
3   #pragma warning(disable:4996)
4   /*--------------------------------------------------------------------*/
5   int main(void)
6   {
7          int   i1, * i_ptr = NULL;
8          float f1, * f_ptr = NULL;
9
10         i1 = 1;
11         i_ptr = &i1;
12         printf("&i_ptr = %p i_ptr = %p &i1 = %p *i_ptr = %d \n\n",&i_ptr,&i1,i_ptr,*i_ptr);
13
14         f1 = 3.3;
15         f_ptr = &f1;
16         printf("&f_ptr = %p f_ptr = %p &f1 = %p *f_ptr = %0.2f \n\n",&f_ptr,&f1,f_ptr,*f_ptr);
17
18         return 0;
19  }
20  /*--------------------------------------------------------------------*/
21  /*
22  &i_ptr = 0115F828 i_ptr = 0115F834 &i1 = 0115F834 *i_ptr = 1
23
24  &f_ptr = 0115F810 f_ptr = 0115F81C &f1 = 0115F81C *f_ptr = 3.30
25  */
26
```

Commented [KW1]: Declare a variable of type integer i1 and a pointer to an integer *i_ptr. Initialize the pointer to NULL until used.

Commented [KW2]: Declare a variable of type float f1 and a pointer to a float *f_ptr. Initialize the pointer to NULL until used.

Commented [KW3]: Initialize the pointer to variable i1's address.

Commented [KW4]: Show the address of the pointer, address of the variable stored in the pointer, address of the variable, and value stored in the variable.

Commented [KW5]: Do the same for the float data.

```
#include<stdio.h>
#include<conio.h>
#pragma warning(disable:4996)

/*--------------------------------------------------------------------*/
int main(void)
{
  char c1='a',c2='b',c3='c';
  int i1 = 100, i2=200, i3=300;
  float f1=1.1, f2=2.2,f3=3.0;
  static int is4, is5=500, is6=600;

  printf("Variable c1 is at memory address %p and contains %c \n",&c1,c1);
  printf("Variable c2 is at memory address %p and contains %c \n",&c2,c2);
  printf("Variable c3 is at memory address %p and contains %c \n\n",&c3,c3);

  printf("Variable i1 is at memory address %p and contains %d \n",&i1,i1);
  printf("Variable i2 is at memory address %p and contains %d \n",&i2,i2);
  printf("Variable i3 is at memory address %p and contains %d \n\n",&i3,i3);

  printf("Variable f1 is at memory address %p and contains %f \n",&f1,f1);
  printf("Variable f2 is at memory address %p and contains %f \n",&f2,f2);
  printf("Variable f3 is at memory address %p and contains %f \n\n",&f3,f3);

  printf("Variable is4 is at memory address %p and contains %d \n",&is4,is4);
  printf("Variable is5 is at memory address %p and contains %d \n",&is5,is5);
  printf("Variable is6 is at memory address %p and contains %d \n\n",&is6,is6);

  return 0;
}
/*--------------------------------------------------------------------*/
/*
Variable c1 is at memory address 012FFAA7 and contains a
Variable c2 is at memory address 012FFA9B and contains b
Variable c3 is at memory address 012FFA8F and contains c

Variable i1 is at memory address 012FFA80 and contains 100
Variable i2 is at memory address 012FFA74 and contains 200
Variable i3 is at memory address 012FFA68 and contains 300

Variable f1 is at memory address 012FFA5C and contains 1.100000
Variable f2 is at memory address 012FFA50 and contains 2.200000
Variable f3 is at memory address 012FFA44 and contains 3.000000

Variable is4 is at memory address 0021A15C and contains 0
Variable is5 is at memory address 0021A000 and contains 500
Variable is6 is at memory address 0021A004 and contains 600

*/
```

Commented [KW6]: Declare and initialize variables.

Commented [KW7]: Declare variables using the static qualifier. Notice when the program is run, is4 is uninitialized but to set 0 when using the static qualifier. This qualifier also maintains the memory value throughout program execution.

Commented [KW8]: Print the variable data.

Commented [KW9]: Uninitialized variable date is set to 0.

```
#include<stdio.h>
#include<conio.h>
#pragma warning(disable:4996)
#define PI 3.14
void display(float radius, float volume);
void get_min_max(float* min, float* max_ptr);
/*--------------------------------------------------------------------*/
int main(void)
{
       float vol, * vol_ptr, rad, * rad_ptr, min, max;

       get_min_max(&min, &max);

       printf("\n   Radius    Volume\n\n");
       rad_ptr = &rad;
       vol_ptr = &vol;

       for (*rad_ptr = min; (*rad_ptr) <= max; (*rad_ptr)++)
       {
         *vol_ptr = (4.0 / 3.0) * PI * *rad_ptr * *rad_ptr * *rad_ptr;
         display(rad, vol);
       }
       return 0;
}
/*--------------------------------------------------------------------*/
void get_min_max(float* min_ptr, float* max_ptr)
{
       printf("Enter the min radius => ");
       scanf("%f", min_ptr);
       printf("Enter the max radius => ");
       scanf("%f", max_ptr);
}
/*--------------------------------------------------------------------*/
void display(float radius, float volume)
{
       printf("%8.2f %10.2f\n", radius, volume);
}
/*--------------------------------------------------------------------*/
/*
Enter the min radius => 1
Enter the max radius => 5

   Radius    Volume

    1.00       4.19
    2.00      33.49
    3.00     113.04
    4.00     267.95
    5.00     523.33
*/

```

Commented [KW10]: Declare pointer variables that are used inside function get_min_max. This declaration requires an address to be passed in this field as an actual parameter.

Commented [KW11]: Declare float variables and pointers for main.

Commented [KW12]: Pass the address of main's variables where the called function will store its data.

Commented [KW13]: Initialize local variable pointers in main to be used for looping.

Commented [KW14]: Use a for loop with pointers to generate a table of data.

Commented [KW15]: Store data back in main in the min variable. Notice the is no & for this scanf since the &min was passed.

```
1  #include<stdio.h>
2  #include<conio.h>
3  #pragma warning(disable:4996)
4  void increment_num(int *iNum_ptr);
5  int decrement_num(int iNum);
6  /*------------------------------------------------------------------*/
7  int main(void)
8  {
9    int  *iY_ptr, iY;
10
11   iY = decrement_num(5);          //the actual parameter is a number 5
12   printf("iY is now %d \n",iY);
13   iY = decrement_num(iY);         //parameter is a variable containing a number 4
14   printf("iY is now %d \n",iY);
15
16   increment_num(&iY);             //the parameter is the address of iY
17   printf("iY is now %d \n",iY);
18   iY_ptr = &iY;
19   increment_num(iY_ptr);          //parameter is a pointer containing the address of iY
20   printf("iY is now %d \n",iY);
21
22   return 0;
23 }
24 /*------------------------------------------------------------------*/
25 void increment_num(int *iNum_ptr)
26 {
27   (*iNum_ptr)++;
28 }
29 /*------------------------------------------------------------------*/
30 int decrement_num(int iNum)
31 {
32   return  --iNum;
33 }
34 /*------------------------------------------------------------------*/
35
36 /*
37 iY is now 4
38 iY is now 3
39 iY is now 4
40 iY is now 5
41
42 Push a key to return to the editor.
43 */
```

Commented [KW16]: Pass an address.

Commented [KW17]: Pass a value.

Commented [KW18]: See single line comments…

Commented [KW19]: This is how you post increment a variable using pointer notation.

Commented [KW20]: This is how you pre decrement a variable that is passed as a value and then returned back to the calling function.