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

#define INT_SIZE sizeof(int)
#define FLOAT_SIZE sizeof(float)
#define DOUBLE_SIZE sizeof(double)
#define CHAR_SIZE sizeof(char)

int main(void)
{
    //variable declarations
    int *ptr_iArray = NULL;
    unsigned int intArrayLength = 0;

    float *ptr_fArray = NULL;
    unsigned int floatArrayLength = 0;

    double *ptr_dArray = NULL;
    unsigned int doubleArrayLength = 0;

    char *ptr_cArray = NULL;
    unsigned int charArrayLength = 0;

    int i;

    //code

    // ****** INTEGER ARRAY *******
    printf("\n\n");
    printf("Enter The Number Of Elements You Want In The Integer Array : ");
    scanf("%u", &intArrayLength);

    ptr_iArray = (int *)malloc(INT_SIZE * intArrayLength);
    if (ptr_iArray == NULL)
    {
        printf("\n\n");
        printf("MEMORY ALLOCATION FOR INTEGER ARRAY FAILED !!! EXITTING NOW...\n\n");
        exit(0);
    }
    else
    {
        printf("\n\n");
        printf("MEMORY ALLOCATION FOR INTEGER ARRAY SUCCEEDED !!!\n\n");
    }

    printf("\n\n");
    printf("Enter The %d Integer Elements To Fill Up The Integer Array : \n\n", intArrayLength);
    for (i = 0; i < intArrayLength; i++)
        scanf("%d", (ptr_iArray + i));

    // ****** FLOAT ARRAY *******
    printf("\n\n");
    printf("Enter The Number Of Elements You Want In The 'float' Array : ");
    scanf("%u", &floatArrayLength);
```

```
55
56      ptr_fArray = (float *)malloc(FLOAT_SIZE * floatArrayLength);
57      if (ptr_fArray == NULL)
58      {
59          printf("\n\n");
60          printf("MEMORY ALLOCATION FOR FLOATING-POINT ARRAY FAILED !!! EXITTING
               NOW...\n\n");
61          exit(0);
62      }
63      else
64      {
65          printf("\n\n");
66          printf("MEMORY ALLOCATION FOR FLOATING-POINT ARRAY SUCCEEDED !!!\n
              \n");
67      }
68
69      printf("\n\n");
70      printf("Enter The %d Floating-Point Elements To Fill Up The 'float'
          Array : \n\n", floatArrayLength);
71      for (i = 0; i < floatArrayLength; i++)
72          scanf("%f", (ptr_fArray + i));
73
74      // ****** DOUBLE ARRAY *******
75      printf("\n\n");
76      printf("Enter The Number Of Elements You Want In The 'double' Array : ");
77      scanf("%u", &doubleArrayLength);
78
79      ptr_dArray = (double *)malloc(DOUBLE_SIZE * doubleArrayLength);
80      if (ptr_dArray == NULL)
81      {
82          printf("\n\n");
83          printf("MEMORY ALLOCATION FOR 'DOUBLE' ARRAY FAILED !!! EXITTING
              NOW...\n\n");
84          exit(0);
85      }
86      else
87      {
88          printf("\n\n");
89          printf("MEMORY ALLOCATION FOR 'DOUBLE' ARRAY SUCCEEDED !!!\n\n");
90      }
91
92      printf("\n\n");
93      printf("Enter The %d Double Elements To Fill Up The 'double' Array : \n
          \n", doubleArrayLength);
94      for (i = 0; i < doubleArrayLength; i++)
95          scanf("%lf", (ptr_dArray + i));
96
97      // ****** CHAR ARRAY *******
98      printf("\n\n");
99      printf("Enter The Number Of Elements You Want In The Character Array : ");
100     scanf("%u", &charArrayLength);
101
102     ptr_cArray = (char *)malloc(CHAR_SIZE * charArrayLength);
103     if (ptr_cArray == NULL)
104     {
105         printf("\n\n");
```

```
106         printf("MEMORY ALLOCATION FOR CHARACTER ARRAY FAILED !!! EXITTING
              NOW...\n\n");
107         exit(0);
108     }
109     else
110     {
111         printf("\n\n");
112         printf("MEMORY ALLOCATION FOR CHARACTER ARRAY SUCCEEDED !!!\n\n");
113     }
114
115     printf("\n\n");
116     printf("Enter The %d Character Elements To Fill Up The Character Array :
          \n\n", charArrayLength);
117     for (i = 0; i < charArrayLength; i++)
118     {
119         *(ptr_cArray + i) = getch();
120         printf("%c\n", *(ptr_cArray + i));
121     }
122
123
124     // ********** DISPLAY OF ARRAYS **********
125
126     // ****** INTEGER ARRAY ******
127     printf("\n\n");
128     printf("The Integer Array Entered By You And Consisting Of %d Elements Is
          As Follows : \n\n", intArrayLength);
129     for (i = 0; i < intArrayLength; i++)
130         printf(" %d \t \t At Address : %p\n", *(ptr_iArray + i), (ptr_iArray +
              i));
131
132     // ****** FLOAT ARRAY ******
133     printf("\n\n");
134     printf("The Float Array Entered By You And Consisting Of %d Elements Is As
           Follows : \n\n", floatArrayLength);
135     for (i = 0; i < floatArrayLength; i++)
136         printf(" %f \t \t At Address : %p\n", *(ptr_fArray + i), (ptr_fArray +
              i));
137
138     // ****** DOUBLE ARRAY ******
139     printf("\n\n");
140     printf("The Double Array Entered By You And Consisting Of %d Elements Is
          As Follows : \n\n", doubleArrayLength);
141     for (i = 0; i < doubleArrayLength; i++)
142         printf(" %lf \t \t At Address : %p\n", *(ptr_dArray + i), (ptr_dArray
              + i));
143
144     // ****** CHARACTER ARRAY ******
145     printf("\n\n");
146     printf("The Character Array Entered By You And Consisting Of %d Elements
          Is As Follows : \n\n", charArrayLength);
147     for (i = 0; i < charArrayLength; i++)
148         printf(" %c \t \t At Address : %p\n", *(ptr_cArray + i), (ptr_cArray +
              i));
149
150
151     // ****** FREEING MEMORY OCCUPIED BY ARRAYS (IN REVERSE ORDER OF
```

```
          ALLOCATION) ******
    if (ptr_cArray)
    {
        free(ptr_cArray);
        ptr_cArray = NULL;

        printf("\n\n");
        printf("MEMORY OCCUPIED BY CHARACTER ARRAY HAS BEEN SUCCESSFULLY
          FREED !!!\n\n");
    }

    if (ptr_dArray)
    {
        free(ptr_dArray);
        ptr_dArray = NULL;

        printf("\n\n");
        printf("MEMORY OCCUPIED BY 'DOUBLE' ARRAY HAS BEEN SUCCESSFULLY
          FREED !!!\n\n");
    }

    if (ptr_fArray)
    {
        free(ptr_fArray);
        ptr_fArray = NULL;

        printf("\n\n");
        printf("MEMORY OCCUPIED BY FLOATING-POINT ARRAY HAS BEEN SUCCESSFULLY
          FREED !!!\n\n");
    }

    if (ptr_iArray)
    {
        free(ptr_iArray);
        ptr_iArray = NULL;

        printf("\n\n");
        printf("MEMORY OCCUPIED BY INTEGER ARRAY HAS BEEN SUCCESSFULLY
          FREED !!!\n\n");
    }

    return(0);
}

```