

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define INT_SIZE sizeof(int)
5 #define FLOAT_SIZE sizeof(float)
6 #define DOUBLE_SIZE sizeof(double)
7 #define CHAR_SIZE sizeof(char)
8
9 int main(void)
10 {
11     //variable declarations
12     int *ptr_iArray = NULL;
13     unsigned int intArrayLength = 0;
14
15     float *ptr_fArray = NULL;
16     unsigned int floatArrayLength = 0;
17
18     double *ptr_dArray = NULL;
19     unsigned int doubleArrayLength = 0;
20
21     char *ptr_cArray = NULL;
22     unsigned int charArrayLength = 0;
23
24     int i;
25
26     //code
27
28     // ***** INTEGER ARRAY *****
29     printf("\n\n");
30     printf("Enter The Number Of Elements You Want In The Integer Array : ");
31     scanf("%u", &intArrayLength);
32
33     ptr_iArray = (int *)malloc(INT_SIZE * intArrayLength);
34     if (ptr_iArray == NULL)
35     {
36         printf("\n\n");
37         printf("MEMORY ALLOCATION FOR INTEGER ARRAY FAILED !!! EXITTING NOW... ?\n\n");
38         exit(0);
39     }
40     else
41     {
42         printf("\n\n");
43         printf("MEMORY ALLOCATION FOR INTEGER ARRAY SUCCEEDED !!!\n\n");
44     }
45
46     printf("\n\n");
47     printf("Enter The %d Integer Elements To Fill Up The Integer Array : \n ?\n", intArrayLength);
48     for (i = 0; i < intArrayLength; i++)
49         scanf("%d", (ptr_iArray + i));
50
51     // ***** FLOAT ARRAY *****
52     printf("\n\n");
53     printf("Enter The Number Of Elements You Want In The 'float' Array : ");
54     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
61     NOW...\n\n");
62     exit(0);
63 }
64 else
65 {
66     printf("\n\n");
67     printf("MEMORY ALLOCATION FOR FLOATING-POINT ARRAY SUCCEEDED !!!\n
68     \n");
69 }
70 printf("\n\n");
71 printf("Enter The %d Floating-Point Elements To Fill Up The 'float'
72 Array : \n\n", floatArrayLength);
73 for (i = 0; i < floatArrayLength; i++)
74     scanf("%f", (ptr_fArray + i));
75
76 // ***** DOUBLE ARRAY *****
77 printf("\n\n");
78 printf("Enter The Number Of Elements You Want In The 'double' Array : ");
79 scanf("%u", &doubleArrayLength);
80
81 ptr_dArray = (double *)malloc(DOUBLE_SIZE * doubleArrayLength);
82 if (ptr_dArray == NULL)
83 {
84     printf("\n\n");
85     printf("MEMORY ALLOCATION FOR 'DOUBLE' ARRAY FAILED !!! EXITTING
86     NOW...\n\n");
87     exit(0);
88 }
89 else
90 {
91     printf("\n\n");
92     printf("MEMORY ALLOCATION FOR 'DOUBLE' ARRAY SUCCEEDED !!!\n\n");
93 }
94
95 printf("\n\n");
96 printf("Enter The %d Double Elements To Fill Up The 'double' Array : \n
97 \n", doubleArrayLength);
98 for (i = 0; i < doubleArrayLength; i++)
99     scanf("%lf", (ptr_dArray + i));
100
101 // ***** CHAR ARRAY *****
102 printf("\n\n");
103 printf("Enter The Number Of Elements You Want In The Character Array : ");
104 scanf("%u", &charArrayLength);
105
106 ptr_cArray = (char *)malloc(CHAR_SIZE * charArrayLength);
107 if (ptr_cArray == NULL)
108 {
109     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) *****
152     if (ptr_cArray)
153     {
154         free(ptr_cArray);
155         ptr_cArray = NULL;
156
157         printf("\n\n");
158         printf("MEMORY OCCUPIED BY CHARACTER ARRAY HAS BEEN SUCCESSFULLY FREED !!!\n\n");
159     }
160
161     if (ptr_dArray)
162     {
163         free(ptr_dArray);
164         ptr_dArray = NULL;
165
166         printf("\n\n");
167         printf("MEMORY OCCUPIED BY 'DOUBLE' ARRAY HAS BEEN SUCCESSFULLY FREED !!!\n\n");
168     }
169
170     if (ptr_fArray)
171     {
172         free(ptr_fArray);
173         ptr_fArray = NULL;
174
175         printf("\n\n");
176         printf("MEMORY OCCUPIED BY FLOATING-POINT ARRAY HAS BEEN SUCCESSFULLY FREED !!!\n\n");
177     }
178
179     if (ptr_iArray)
180     {
181         free(ptr_iArray);
182         ptr_iArray = NULL;
183
184         printf("\n\n");
185         printf("MEMORY OCCUPIED BY INTEGER ARRAY HAS BEEN SUCCESSFULLY FREED !!!\n\n");
186     }
187
188     return(0);
189 }
190
```