

```

1  /**
2   * file: pointer.c
3   *
4   * Created by hengxin on 11/24/23.
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 int main() {
11     /***** On radius *****/
12     int radius = 100;
13
14     printf("radius = %d\n", radius);
15
16     // every variable has an address
17     // &:amp; address-of operator ("&")
18     printf("The address of radius is %p\n", &radius);
19     // we have already used the address of a variable before
20     // scanf("%d", &radius);
21
22     // radius as a left value; refer to its address (the storage space)
23     radius = 200;
24     // radius as a right value; refer to its value
25     double circumference = 2 * 3.14 * radius;
26     printf("radius = %d; circumference = %f\n", radius, circumference);
27     /***** On radius *****/
28
29     /***** On ptr_radius1 *****/
30     // ptr_radius1 is a variable of type "pointer to int"
31     int *ptr_radius1 = &radius;
32     // ptr_radius1 is a variable: has its value
33     printf("ptr_radius1 = %p\n", ptr_radius1);
34     // ptr_radius1 is a variable: has its address
35     printf("The address of ptr_radius1 is %p\n", &ptr_radius1);
36     /***** On ptr_radius1 *****/
37
38     /***** On *ptr_radius1 *****/
39     // IMPORTANT:
40     // *ptr_radius1: behaves just like radius
41     // type: int; value: the value of radius; address: the address of
radius
42     // *: indirection/dereference operator ("&"/"")
43     printf("radius = %d\n", *ptr_radius1);
44     // *ptr_radius1 as a right value
45     circumference = 2 * 3.14 * (*ptr_radius1);
46     // take the address of *ptr_radius1
47     // &*ptr_radius1 is the same as ptr_radius1
48     printf("The address of *ptr_radius1 is %p\n", &*ptr_radius1);
49     // *ptr_radius1 as a left value
50     *ptr_radius1 = 100;
51     printf("radius = %d\n", *ptr_radius1);
52     /***** On *ptr_radius1 *****/

```

```

53
54  /***** On ptr_radius1 again *****/
55  // ptr_radius1 as a left value
56  int radius2 = 200;
57  int *ptr_radius2 = &radius2;
58
59  ptr_radius1 = ptr_radius2;
60  printf("radius = %d\n", *ptr_radius1);
61
62  // ptr_radius1 as a right value
63  ptr_radius2 = ptr_radius1;
64  printf("radius = %d\n", *ptr_radius2);
65  /***** On ptr_radius1 again *****/
66
67  /***** On array names *****/
68  int numbers[5] = {0};
69  // vs. numbers[2] = {2};
70  // numbers++;
71  // numbers = &radius;
72  int *ptr_array = numbers;
73  ptr_array++;
74  /***** On array names *****/
75
76  /***** On malloc/free *****/
77  // undefined behavior
78  // free(numbers);
79  /***** On malloc/free *****/
80
81  /***** On const *****/
82  // const int * and int const *
83  // You cannot modify the value pointed to by ptr_radius3
84  // through the pointer (without casting the constness away).
85  const int *ptr_radius3 = &radius;
86  // *ptr_radius is read-only
87  // *ptr_radius3 = 300;
88  // You are allowed to do this, but you should not do it!
89  int *ptr_radius4 = ptr_radius3;
90  *ptr_radius4 = 400;
91  printf("radius = %d\n", radius);
92
93  // int * const
94  int *const ptr_radius5 = &radius;
95  // ptr_radius5 = ptr_radius3;
96  *ptr_radius5 = 500;
97  printf("radius = %d\n", radius);
98
99  // const int * const
100 const int *const ptr_radius6 = &radius;
101 // ptr_radius6 = ptr_radius3;
102 // *ptr_radius6 = 600;
103 /***** On const *****/
104 }

```