Pointers Practical Exercises

Answers

Predict Output #1:

Value of ‘V’	Address of variable ‘V’
Address of variable ‘V’ or value of pointer ‘ptr2’	Address of pointer ‘ptr2’
Base address of array ‘Arr1’ or address of ‘Arr1[0]’	Base address of array ‘Arr1’
Base address of array ‘Arr1[2]’	Address of pointer ‘ptr1’
Value of ‘V’	Address of variable ‘V’
Address of variable ‘V’ or value of pointer ‘ptr2’	Address of pointer ‘ptr2’
Base address of array ‘Arr1’ or address of ‘Arr1[0]’	Base address of array ‘Arr1’
Base address of array ‘Arr2’ or address of ‘Arr2[0]’	Address of pointer ‘ptr1’

Predict Output #2

58 58 58

Predict Output #2

K=4

x=950

y=3000

k=14

k=28

Predict Output #4

Lines	Values			
	i	j	ptr	pptr
4	5	10	0x37129	unknown
5	5	10	0x37129	0x1dc825
6	3	10	0x37129	0x1dc825
7	7	10	0x37129	0x1dc825
8	7	10	0x5893a	0x1dc825
9	7	9	0x5893a	0x1dc825
10	7	9	0x5893a	0x1dc825
11	-2	9	0x5893a	0x1dc825

Predict Output #5

5 address of '5'

5 address of '5' (but incremented by 4)

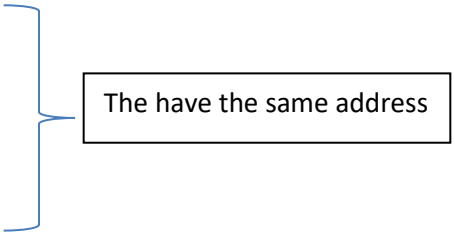
10 address of '10' (but incremented by 4)

10 address of '10'

11 address of '11'

12 address of '12'

12 address of '12'



Predict Output #6

a[0]: 6

a[1]: 5

a[2]: unknown

a[3]: 7

a[4]: unknown

Predict Output #8

Mark[0][2]: 24 Mark[1][3]: 30 Mark[2][2]: 40

Analyze the segment below and identify

- Type of pointers
- Invalid statements

Invalid statements:

*p1 = 20 (assigning an integer (20) to a pointer (p1) is not allowed)

*p2 = 50 (p2 is a pointer to a constant integer so, the value it points to cannot be modified.)

p3 = &y (p3 is constant pointer so, its address cannot be changed)

p4 = &y (p4 is constant pointer so, its address cannot be changed)

*p4 = 90 (p4 is a pointer to a constant integer so, value it points to cannot be modified)

Type of pointers

In the given code, the valid pointers are just the ordinary pointers(don't have a certain type), but we can classify them as:

int *p1: Non-constant pointer to an integer (int*).

const int *p2: Pointer to a constant integer (const int*).

int *const p3: Constant pointer to an integer (int* const).

const int *const p4: Constant pointer to a constant integer (const int* const).

1. Given the 1D – array declarations

Here's the code to print the address of each array elements:

```
#include <iostream>
Using namespace std;
int main() {
   double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
   int arr[3][4] = { { 10, 11, 12, 13 }, { 20, 21, 22, 23 }, { 30, 31, 32, 33 }};

   cout << "Addresses of elements in 'balance' array:" << endl;
   for (int i = 0; i < 5; i++) {
      cout << "&balance[" << i << "] = " << &balance[i] << endl;
   }
   cout << "Addresses of elements in 'arr' array:" << endl;
```

```
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            cout << "&arr[" << i << "][" << j << "] = " << &arr[i][j] << endl;
        }
    }
    return 0;
}
```

Here's a code to print the array elements using pointer instead of the array index

```
#include <iostream>
using namespace std;
int main() {
    double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
    int arr[3][4] = { { 10, 11, 12, 13 }, { 20, 21, 22, 23 }, { 30, 31, 32, 33 }};

    cout << "Elements in 'balance' array (using pointers):" << endl;
    for (int i = 0; i < 5; i++) {
        cout << "*(balance + " << i << ") = " << *(balance + i) << endl;
    }

    cout << "Elements in 'arr' array (using pointers):" <<endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            cout << "*(*(arr + " << i << ") + " << j << ") = " << *(*(arr + i) + j) << endl;
        }
    }
    return 0;
}
```

Here is a code to print the array elements using an other pointer

```
#include <iostream>
Using namespace std;

int main() {
    double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
    int arr[3][4] = { { 10, 11, 12, 13 }, { 20, 21, 22, 23 }, { 30, 31, 32, 33 }};

    cout << "Elements in 'balance' array (using pointer to pointer):" << endl;
    double* ptr = balance;
    for (int i = 0; i < 5; i++) {
        std::cout << "*(ptr + " << i << ") = " << *(ptr + i) << endl;
    }
```

```
    cout << "Elements in 'arr' array (using pointer to pointer):" << endl;
    int** ptr2 = reinterpret_cast<int**>(arr);
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            cout << "*(*(ptr2 + " << i << ") + " << j << ") = " << *(*(ptr2 + i) + j) << endl;
        }
    }
    return 0;
}
```

2. Create a 2D array and string that stores N person *height, weight and BMI* and full name respectively. Write a program to read the *height, weight* and then calculate the *BMI* using pointer. The program should print the record in tabular format using pointer operation.

```
#include <iostream>
#include <iomanip>
#include <cstring>
Using namespace std;

const int MAX_N = 100;
const int MAX_NAME_LENGTH = 50;

struct Person {
   double height;
   double weight;
   double bmi;
   char name[MAX_NAME_LENGTH];};

void calculateBMI(Person* person) {
   person->bmi = person->weight / ((person->height / 100.0) * (person->height / 100.0)); }

void printRecords(const Person* people, int numPeople) {
   cout << "      Name      | Height | Weight |   BMI   " << endl;

   for (int i = 0; i < numPeople; i++) {
      const Person* person = &people[i];
      cout << left << setw(18) << person->name;
```

```
        cout << std::right << setw(10) << person->height;
        cout << std::right << setw(10) << person->weight;
        cout << std::right << setw(10) << person->bmi <<endl;
    }
}

int main() {
    int numPeople;
    cout << "Enter the number of people: ";
    cin >> numPeople;

    Person people[MAX_N];

    for (int i = 0; i < numPeople; i++) {
        Person* person = &people[i];

        cout << "Person #" << i + 1 << endl;
        cout << "Enter full name: ";
        cin.ignore(); // Ignore the newline character from previous input
        cin.getline(person->name, MAX_NAME_LENGTH);

        cout << "Enter height (in cm): ";
        cin >> person->height;

        cout << "Enter weight (in kg): ";
        cin >> person->weight;

        calculateBMI(person);
    }
    printRecords(people, numPeople);
    return 0;
}
```