Lab 2

Thomas Bischoff

Original

Date: 2 / 12 / 2018

Problem:

Code –

//######################## // pointer.cpp //######################## #include <iostream> using namespace std; int main() { // Part 1: int v\_int = 8; short v\_short = 2; cout << “The Size of the Integer is ” << sizeof(int) << “B” << endl; cout << “The Address of the Integer is ” << &v\_int << endl; cout << “The Size of the Short is ” << sizeof(short) << “B” << endl; cout << “The Address of the Short is ” << &v\_short << endl; // Part 2: char v\_char = ‘C’ cout << “The Size of the Character is ” << sizeof(char) << “B” << endl; cout << “The Address of the Character is ” << (void\*)&v\_char << endl; // Part 3: short \*ptr\_short; char \*ptr\_char; ptr\_short = &v\_short; ptr\_char = &v\_char; cout << “The Value of the Pointer for the Short is ” << ptr\_short << endl; cout << “The Address of the Pointer for the Short is ” << &ptr\_short << endl; cout << “The Value Pointed to by the Pointer of the Short is ” << \*ptr\_short << endl; cout << “The Size of the Pointer for the Short is ” << sizeof(ptr\_short) << “B” << endl; cout << “The Value of the Pointer for the Character is ” << (void\*)ptr\_char << endl;

Code (Continued) –

cout << “The Address of the Pointer for the Character is ” << (void\*)&ptr\_char << endl; cout << “The Value Pointed to by the Pointer of the Character is ” << \*ptr\_char << endl; cout << “The Size of the Pointer for the Character is ” << sizeof(ptr\_char) << “B” << endl; // Part 4: int \*ptr\_int; ptr\_int = &v\_int; ptr\_int = new int; \*ptr\_int = -5; cout << “The Value of the Pointer for the Integer is ” << ptr\_int << endl; cout << “The Address of the Pointer for the Integer is ” << &ptr\_int << endl; cout << “The Value Pointed to by the Pointer of the Integer is ” << \*ptr\_int << endl; \*ptr\_int = -10; cout << “The New Value Pointed to by the Pointer of the Integer is ” << \*ptr\_int << endl; ptr\_int = (int\*)ptr\_short; cout << “The Value Pointed to by the Pointer of the Integer and the Short is ” << \*ptr\_int << endl;

Sample Output –

The Size of the Integer is 4B The Address of the Integer is 0x72fe34 The Size of the Short 2B The Address of the Short 0x72fe32 The Size of the Character is 1B The Address of the Character is 072fe31 The Value of the Pointer for the Short is 0x72fe32 The Address of the Pointer for the Short is 0x72fe28 The Value Pointed to by the Pointer of the Short is 2 The Size of the Pointer for the Short is 8B The Value of the Pointer for the Character is 0x72fe31 The Address of the Pointer for the Character is 0x72fe31 The Value Pointed to by the Pointer of the Character is C The Size of the Pointer for the Character is 8B The Value of the Pointer for the Integer is 0x2b014d0 The Address of the Pointer for the Integer is 0x72fe20

Sample Output (Continued) –

The Value Pointed to by the Pointer of the Integer is -5 The New Value Pointed to by the Pointer of the Integer is -10 The Value Pointed to by the Pointer of the Integer and the Short is 5

Part 1: Comments / Answers –

1. **What is the address assigned to v\_int? To v\_short?**

The address assigned to v\_int is 0x72fe34 while the address for v\_short is 0x72fe32.

1. **What is the difference (in Bytes) between them?**

The size of v\_int in Bytes is 4 and the size of v\_short in Bytes is 2, meaning that the difference between them is 2B.

1. **Read the hexadecimal notes to figure out how to subtract or add in hexadecimal.**

B

Part 2: Comments / Answers –

1. **What is the address of v\_char?**

The address assigned to v\_char is 0x72fe31.

1. **What is the difference (in Bytes) between the address of v\_char and the addresses of v\_int and that of v\_short?**

Difference Between v\_int and v\_char

B

Difference Between v\_short and v\_char

B

Part 3: Comments / Answers –

1. **Compare the size ptr\_char and ptr\_short to the values they point to. Explain the differences.**

The sizes of ptr\_char and ptr\_short are 8B each meaning the difference between ptr\_char and v\_char is 7B while the difference between ptr\_short and v\_short is 6B. The reason why difference exists is because the pointer is storing the address of the value it is supposed to be pointing at and not the value itself.

Part 3: Comments / Answers (Continued) –

1. **Memory Locations and Memory Addresses:**

|  |  |
| --- | --- |
| Memory Address | Memory Location |
| 0x72fe3c | v\_int = 8 (4B) |
| 0x72fe3a | v\_short = 2 (2B) |
| 0x72fe39 | v\_char = ‘C’ (1B) |
| 0x72fe30 | ptr\_short = 0x72fe3a (8B) |
| 0x72fe28 | ptr\_char = 0x72fe39 (8B) |

Part 4: Comments / Answers –

1. **Is the value output for ptr\_int related to any other memory address you found before? Explain what you observe.**

The value of the ptr\_int is 0x2851d0, and this address does not have any relation to the ones previously gathered.

1. **At what address is the ptr\_int stored? What is its relation to the addresses where ptr\_short, ptr\_char, v\_char, v\_int and v\_short are stored.**

The address where ptr\_int is stored is 0x72fe20. The relation between ptr\_int and the rest of the values is that they all begin with 0x72fe…

1. **What happened to -10. Can you still find the memory location where -10 is stored? If not, explain why?**

-10 is now the value which is being contained at the address which is being pointed to by ptr\_int. The location will be the same memory address initially pointed to by ptr\_int.

1. **Memory Locations and Memory Addresses:**

|  |  |
| --- | --- |
| Memory Addresses | Memory Locations |
| 0x72fe3c | v\_int = 8 (4B) |
| 0x72fe3a | v\_short = 2 (2B) |
| 0x72fe39 | v\_char = ‘C’ (1B) |
| 0x72fe30 | ptr\_short = 0x72fe3a (8B) |
| 0x72fe28 | ptr\_char = 0x72fe39 (8B) |
| 0x28514d0 | v\_int = -5 (4B) |
| 0x72fe20 | ptr\_int = 0x28514d0 |