Review-Activity3-Pointers

Tuesday, January 28, 2020 11:52 AM

Review Activity 3

Using Pointers to Access Memory Locations

1) What will be the output of the following code fragment? Explain.

```
int* p = new int(32);
int* q = new int(24);
*p = *q;
*q = 42;
cout << *p << " and " << *q <<endl;
```

Output
24 and 42

//the answer will be "24 and 42"
//first, *p , the value p is pointing to is changed to what q is pointing to, 24
//then the value being pointed to by q is changed to 42
//hence the output will be 24 and 42

2) If "`cout << *p1 << endl;`" outputs `5`, and
"`cout << *p2 << endl;`" outputs `6`,
what will be the output of "`cout << p1 + p2 << endl;`"? Explain.
`p1` and `p2` are of `int*` type.

//you can't add 2 pointers together; they aren't to be used as regular operands
//the program may crash or return a random value that isn't zero

3) If "`cout << p3 << endl;`" outputs `0x596834`,
and "`cout << *p4 << endl;`" outputs `1`,
what will be the output of
"`cout << p3 + *p4 << endl; cout << p3 - *p4 << endl;`"? Explain.
`p3` and `p4` are of `int*` type.

//the first one will output 0x596838 - since *p4 is 1 (an int) and ints take up 4 bytes of space, the printed address will be p3 + 4 more units
//the second one will output 0x596830 - since we are now doing the opposite operation of the first output

4) `int* foo()` that is given below returns the address of a local variable. Can this address be used by the caller to store other values reliably? Explain.

```
int* foo() {
     int a = 5;
     return &a;
}
```

Example use:

```
int *p = foo();
*p = 7;
```

//no, it can't be used reliably since whatever p points to can changed somewhere else i.e. *p = 7

```
*p = 7;
```

//no, it can't be used reliably since whatever p points to can changed somewhere else i.e. *p = 7

5) What will be the output of the following code fragment? Explain.

```
int *p = new int(56);
int *q = new int(56);
cout << *p << endl;
cout << *q << endl;
delete p;
delete q;
cout << *p << endl;
cout << *q << endl;
```

Output
56
56
unknown
unknown
//we no longer have control of what the pointer is pointing to
//so we might get garbage data if someone has already written something in that slot

6) Something "bad" will happen if this code is run. Explain.

```
int i = 66;
int *ip = &i;
delete ip;
```

//can't delete a pointer to the stack memory
//3rd line will upset the compiler

7) In the following code fragment, where is `t->i` located? Is it on the stack or heap?

```
class Test {
   public:
   int i;
};

int main() {
   Test* t = new Test();
   cout << t->i;
   delete t;
}
```

//t-> i is located on the heap. t was initialized with the 'new' operator meaning it was created in the heap memory. Since i is inside the t object, it will also be on the heap.