Daniel Avila

CPSC 121

Chapter 10 Questions || Page 697

#2-10, 13, 15, 17, 18, 20, 25, 26, 28, 29, 31, 38(a-g)

1. The **&** operator can be used to determine a variable’s address.
2. **Pointer** variables are designed to hold addresses.
3. The **\*** operator can be used to work with the variable a pointer points to.
4. Array names can be used as **pointers** and vice versa.
5. Creating variables while a program is running is called **memory allocation**.
6. The **new** operator is used to dynamically allocate memory.
7. If the *new* operator cannot allocate the amount of memory requested, it throws **bad allocation exception**.
8. A pointer that contains the address 0 is called a(n) **null** pointer.
9. When a program is finished with a chunk of dynamically allocated memory, it should free it with the **delete** operator.
10. Look at the following code.

int x = 7;

int \*ptr = &x;

What will be displayed if you send the expression *\*iptr* to *cout*? What happens if you send the expression *ptr* to *cout*.

**7**

**Address of x**

1. Which arithmetic operations can be applied to pointers?

**++ Increment, -- Decrement; Int values only can + Add, or - Subtract**

1. Look at the following array definition.

int numbers [] = {2, 4, 6, 8, 10};

What will the following statement display?

cout << \*(numbers + 3) << endl;

**8; the same as numbers[3], \*(numbers) is 0 in the array index**

1. What is the purpose of the new operator?

**Allocate memory dynamically**

1. Under what circumstances can you successfully return a pointer from a function?

**When it is passed as an argument**

1. How do smart pointers differ from regular pointers?

**They have the ability to delete the dynamically allocated memory automatically**

1. Name the header file that needs to be included in a program that uses smart pointers.

**#include <memory>**

1. What does the *get()* method of the *unique\_ptr* class do?

**Returns the raw pointer**

1. What is the name of the class of smart pointer that allows more than one pointer to own the same object?

**shared\_ptr**

1. Why should programmers prefer the use of the *make\_shared* function when creating shared pointers?

**To make shared function when creating shared pointers; 2 lines becomes 1 line**

1. Each of the following declarations and program segments has errors. Locate as many as you can.
2. A)int ptr\*;

**The \* should be before the variable name, so \*ptr**

1. B) int x, \*ptr;

&x = ptr;

**Variable cannot hold the address, only the pointer, so ptr = &x;**

1. C)int x, \*ptr;

\*ptr = &x;

**The value is being set to the same value, so it needs to be ptr = &x;**

1. D) int x, \*ptr;

ptr = &x;

ptr = 100; //Store 100 in x

cout << x << endl;

**The address is being set to 100, so it should be \*ptr = 100**

1. E) int numbers [] = {10, 20, 30, 40, 50};

cout << “The third element in the array is “;

cout << \*numbers + 3 << endl;

**This is adding 3 to the 0th array index, so \*(numbers + 3) will be 4th index**

1. F) int values[20], \*iptr;

iptr = values;

iptr \*= 2; **The equal sign shouldn’t have the \*, so \*iptr = 2;**

1. G) double level;

int dPtr = &level;

**The dPtr should point to it with an \* and it should be a double to keep it the same value, so it should be double \*dPtr = &level;**

Chapter 17 Questions || Page 1080

#1, 2, 3, 6, 7, 8, 14, 17(a,b,c)

1. The **head pointer** points to the first node in a linked list.
2. A data structure that points to an object of the same type as itself is known as a(n) **self-referential** data structure.
3. To indicate that a linked list is empty, you should set the pointer to its head to the value **null or 0**.
4. **Traversing** a list means traveling through the list.
5. In a(n) **circular** list, the last node has a pointer to the first node.
6. In a(n) **double-linked** list, each node has a pointer to the one before it and the one after it.
7. For each of the following program fragments, predict what the output will be.

ListNode \*p = new ListNode(56.4);

p = new ListNode(34.2, p);

cout << (\*p).value << endl << p->value;

**56.4**

**34.2**

1. Each of the following member functions for performing an operation on a linked list of type *NumberList* has at least one error. Explain what is wrong and how to fix it.
2. A) NumberList::printList() **There is no data type for the function: ex. Int, double, void**

{

while(head)

{

cout << head->value;

head = head->next; **This destroys the list so use an auxiliary pointer initialized to head instead**

}

}

1. B) NumberList::printList() **There is no data type for the function: ex. int, double, void**

{

ListNode \*p =head; **Make value set equal to NULL**

while (p->next)

{

cout << p->value;

p = p->next;

}

}

1. C) NumberList::printList() **There is no data type for the function: ex. int, double, void**

{

ListNode \*p = head;

while(p)

{

cout << p->value; **Make it point to the NULL value**

p++; **p = p->next;**

}

}