Jason Tran

CS31 Project 6

11/14/22

**1a)**

Original:

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 30; // set arr[0] to 30

\*ptr + 1 = 20; // set arr[1] to 20 Didn’t add the ptr correctly

ptr += 2;

ptr[0] = 10; // set arr[2] to 10

while (ptr >= arr)

{

ptr--;

cout << \*ptr << endl; // print values

}

}

**Fix:**

int main()

{

int arr[3] = { 5, 10, 15 };

int\* ptr = arr;

\*ptr = 30; // set arr[0] to 30

\*(ptr + 1) = 20; // set arr[1] to 20

ptr += 2;

ptr[0] = 10; // set arr[2] to 10

ptr = ptr - 2;

while (ptr <= arr + 2) {

cout << \*ptr << endl; // print values

ptr++;

}

}

**1b)**

The problem is that the program is just printing the data of the 1st position (0th index) of the array and not actually the max value. This is because the function that is trying to find the max value is not actually altering the real pointer. Similar to other data types, passing the pointer is a pass by value, not changing the original variable but making a copy. To fix this you have to pass by reference.

**fix:**

void findMax(int arr[], int n, int\* &pToMax)

**1c)**

This won’t work because ptr is not pointing to the address of anything. So editing it won’t do anything to the pointer that is passed in. The ptr should be initialized to a variable of the same datatype, in which then it will be able to alter the object that the variable is holding and get dereferenced correctly.

**fix:**

int main()

{

int n = 5;

int\* ptr = &n;

computeCube(5, ptr);

cout << "Five cubed is " << \*ptr << endl;

}

**1d)**

Strequal is wrong because if you want to compare the character’s equality you have to dereference the pointer to get the character, not compare the pointers. That would just be comparing addresses which are usually not the same.

**Fix:**

bool strequal(const char str1[], const char str2[])

{

while (\*str1 != '\0' && \*str2 != '\0') // zero bytes at ends

{

if (\*str1 != \*str2) // compare corresponding characters

return false;

str1++; // advance to the next character

str2++;

}

return \*str1 == \*str2; // both ended at same time?

}

**1e)**

One problem occurs in the getPtrToArray function. anArry is locally declared there and seems to want to return a pointer to the first element of that array, however due to variable and function scope, anArray is destroyed after leaving the function. This means that the pointer in main that called that function is now pointing to something random (garbage value) not doing the correct thing the program intended.

**2)**

1. double\* cat;
2. double mouse[5];
3. cat = &mouse[4];
4. \*cat = 25;
5. \*(mouse +3) = 54;
6. cat -= 3;
7. cat[1] = 42;
8. cat[0]= 17;
9. bool d = (cat == mouse);
10. bool b = (\*cat == \*(cat+1));

**3a)**

**Changed:**

double mean(const double\* scores, int numScores)

{

const double\* ptr = scores;

double tot = 0;

for (int i = 0; i < numScores; i++)

{

tot + = \*(scores+i);

}

return tot/numScores;

}

**3b)**

**Changed:**

const char\* findTheChar(const char\* str, char chr)

{

for (int k = 0; \*(str +k) != 0; k++)

if (\*(str + k) == chr)

return &str[k];

return nullptr;

}

**3c)**

**Changed:**

const char\* findTheChar(const char\* str, char chr)

{

while (\*str != '\0'){

if (\*str == chr)

return str;

else

str++;

}

return nullptr;

}

**4)**

**Output:**

3

4

79

-1

9

22

19

**Explanation:**

int main()

{

int array[6] = { 5, 3, 4, 17, 22, 19 };

int\* ptr = maxwell(array, &array[2]); **//returns whichever the address at whichever value is greater than the 2. So it is currently pointing to &array[0].**

\*ptr = -1; **//sets the current object of the pointer to -1, which was previously 5.**

ptr += 2; **//adds 2 to the pointer resulting in the new address of &array[2]**

ptr[1] = 9; **//sets the address one after (&array[3]) the current pointer’s address’object to 9.**

\*(array+1) = 79; **//array without any brackets is the address to the first element, so adding one to it would be the address of the 2nd element. So this sets the second element to 79.**

cout << &array[5] - ptr << endl; **//ptr is currently holding &array[2] so 5-2 is 3. Which is where the first 3 comes from.**

swap1(&array[0], &array[1]); **//the function swap1 seems to just swap the addresses, but doesn’t alter the values of the array nor its elements.**

swap2(array, &array[2]); **//swap2, however swaps the objects that the pointers are pointing 2, changing the 0th (1st element) position to the 2nd (3rd element). Making the resulting array be {4,79,-1,9,22,19}**

for (int i = 0; i < 6; i++)

cout << array[i] << endl; **//prints the elements of the array with new line after each.**

}

**5)**

void removeS(char\* str)

{

while (\*str != '\0')

{

if (\*str == 's' || \*str == 'S')

{

char\* tmp = str;

while (\*tmp != '\0')

{

\*tmp= \*(tmp+1);

tmp++;

}

}

else

{

str++;

}

}

}