Firat Taxpulat

Mr. Smallberg

CS31

27 November, 2017

Project 6 Homework

1a. int main()

{

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

int\* ptr = arr;

\*ptr = 30;

\*(ptr + 1) = 20; // I added parentheses

ptr += 2;

ptr[0] = 10;

ptr = arr; // I changed the starting location of the pointer

while (ptr <= arr + 2)

{

cout << \*ptr << endl;

ptr++;

}

}

1b. The findMax function does not complete its purpose because the function only copies the value of the pointer from the main function. By only copying the value of the pointer from the main function, the findMax function only changes the value of the copied pointer and does not change the value of the original pointer from the main function. As a result, the main function is unable to output any values of ptr, the original pointer from the main function, because ptr is never initialized. The findMax function is supposed to copy the reference of the pointer from the main function. In order to do this, I added an ampersand symbol next to the pToMax parameter.

void findMax(int arr[], int n, int\*& pToMax) // \*\*\*\* I added an ampersand symbol \*\*\*\*\*

{

if (n <= 0)

return; // no items, no maximum!

pToMax = arr;

for (int i = 1; i < n; i++)

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

int main()

{

int nums[4] = { 5, 3, 15, 6 };

int\* ptr;

findMax(nums, 4, ptr);

cout << "The maximum is at address " << ptr << endl;

cout << "It's at position " << ptr - nums << endl;

cout << "Its value is " << \*ptr << endl;

}

1c. The main function’s problem is that the pointer ptr is never initialized. In order to fix this problem, I created an empty int variable, x, and set its address equal to the pointer ptr.

void computeCube(int n, int\* ncubed)

{

\*ncubed = n \* n \* n;

}

int main()

{

int a; //I created an int variable

int\* ptr = &a; //I initialized ptr to the newly created int variable

computeCube(5, ptr);

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

}

1d. The strequal function is trying to compare each corresponding character between the arrays str1 and str2. However, the strequal function is only comparing the addresses of str1 and str2. This can be seen when the strequal function is attempting to recognize when both str1 and str2 have null bytes that have corresponding positions. Furthermore, the comparison of str1 and str2’s addresses can be seen when the function is trying to return false when one character in str1 that corresponds to str2’s character does not equal to str2’s character. Lastly, another example of this comparison is when strequal attempts to return true when str1 is equal to str2, but this statement is only comparing the beginning addresses of the two arrays. In order to fix this problem, I added a \* operator to str1 and str2 within the three examples I detailed earlier. The \* operators can be seen below.

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

{

while (\*str1 != 0 && \*str2 != 0) // \*\*\*\*\*\*I added an \* operator next to str1 and str2 in order to dereference the value of the two arrays \*\*\*\*\*\*\*

{

if (\*str1 != \*str2) // compare corresponding characters // \*\*\*\*\*\*I added an \* operator next to str1 and str2 in order to dereference the value of the two arrays \*\*\*\*\*\*\*

return false;

str1++; // advance to the next character

str2++;

}

return \*str1 == \*str2; // both ended at same time? // \*\*\*\*\*\*I added an \* operator next to str1 and str2 in order to dereference the value of the two arrays \*\*\*\*\*\*\*

}

int main()

{

char a[15] = "Zhou";

char b[15] = "Zhu";

if (strequal(a, b))

cout << "They're the same person!\n";

}

1e.

There is no array declared in the main routine that ptr can be initialized to. The anArray in the getPtrToArray function cannot be accessed in the main function, so there is undefined behavior when ptr is initialized to anArray.

2a. double\* cat;

2b. double mouse[5];

2c. cat = &mouse[4];

2d. \*cat = 25;

2e. \*(mouse + 3) = 42;

2f. cat -= 3;

2g. cat[1] = 54;

2h. cat[0] = 27;

2i. bool b = (\*cat == \*(cat + 1));

2j. bool d = (cat == mouse);

3a. double mean(const double\* scores, int numScores)

{

const double\* ptr = scores;

double tot = 0;

int i = 0;

while (i < numScores)

{

tot += \*(ptr + i);

i++;

}

return tot / numScores;

}

3b. // This function searches through str for the character chr.

// If the chr is found, it returns a pointer into str where

// the character was first found, otherwise nullptr (not found).

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

{

while (\*str != 0)

{

if (\*str == chr)

return str;

str++;

}

return nullptr;

}

4.

First Output: 3

* ptr points to the 3rd element in the array. &array[5] is the position of the 6th element. The difference between the two is three

Second Output: 4

* swap1 does nothing. swap2 switches the 1st and 3rd elements of the array so the 1st element is now 4

Third Output: 79

* \*(array+1) sets the 2nd element of the array to 79

Fourth Output: -1

* swap 2 switches the 3rd element of the array, 4, with the 1st element of the array, -1

Fifth Output: 9

* Since ptr was at the 3rd position of the array, ptr[1] = 9 sets the 4th element of the array to 9

Sixth Output: 22

* The 5th element does not change

Seventh Output: 19

* The 6th element does not change

5.

void removeS(char\* ptr)

{

while (\*ptr != 0)

{

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

for (char\* var = ptr; \*var != 0; var++)

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

else

ptr++;

}

}