Le’on Norfleet

CS31 Lec 1

Prof. Smallberg

5/21/2022

305771450

Project 6

Errors:

1. Line 4 (\*ptr + 1 = 20;) change to → \*(ptr + 1) = 20;

2. Line 10 (ptr--;) should go under line 11

Fixed:

int main()

{

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

int\* ptr = arr;

\*ptr = 10; // set arr[0] to 10

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

ptr += 2;

ptr[0] = 30; // set arr[2] to 30

while (ptr >= arr)

{

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

ptr--;

}

cout << endl;

}

* 1. The function will not work as intended because the pointer is passed by value, not reference. This means that the address will never change to display what is intended.the way to fix it is change it from passing by value to passing by reference.

Fixed:

void findDisorder(int arr[], int n, int\* &p)

{

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

{

if (arr[k] < arr[k-1])

{

p = arr + k;

return;

}

}

p = nullptr;

}

* 1. p is not pointing to anything. The fix to thus is create a double it can point to

Fixed:

int main()

{

double\* p;

double solution;

p = &solution;

hypotenuse(1.5, 2.0, p);

cout << "The hypotenuse is " << \*p << endl;

}

* 1. The problem with the function is that the if statements and incrementations are reading the full c string because the address is being dereferenced. The solution is to dereference it with asterisks or brackets.

Fixed:

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

{

while (str1[0] != 0 && str2[0] != 0) // zero bytes at ends

{

if (str1[0] != str2[0]) // compare corresponding characters

return false;

str1++; // advance to the next character

str2++;

}

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

}

* 1. arr is a variable local only to the computeSquares function, meaning its not actually returning the address to an accessible array

1. 1. string\* fp;
   2. string fish[5];
   3. fp = fish + 4;
   4. \*fp = “yellowtail”;
   5. \*(fish + 3) = “salmon”;
   6. fp - = 3;
   7. fp[1] = “carp”;
   8. fp[0] = “smelt”;
   9. bool d = (fp == fish);
   10. bool b = (\*fp == \*(fp + 1));
2. 1. double computeAverage(const double\* scores, int nScores)

{

int inc = 0;

double tot = 0;

while (scores + inc != scores + nScores)

{

tot += \*(scores + inc);

inc++;

}

return tot / nScores;

}

* 1. 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;

}

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

{

while (\*str != 0) {

if (\*str == chr) {

return str;

}

str++;

}

return nullptr;

}

1. Will print: diff=1

4

79

5

9

-1

19

Explanation:

1.int\* ptr = minimart(array, &array[2]); → array[2] is less than array[0] so ptr points to array[2]

2.ptr[1]=9 is the same as \*(ptr + 1) = 9 so array[3] becomes 9

3.ptr+=2 moves the pointer forwards by 2 so it now points to array[4]

4.\*ptr = -1; → array[4] becomes -1

5.\*(array+1) = 79; → array[1] becomes 79

6. cout << "diff=" << &array[5] - ptr << endl;

&array[5] - ptr is the same as &array[5] - &array[4] which is 1

So it would cout “diff=1”

7.swap1(&array[0], &array[1]); does nothing because it is not passing by reference

8.swap2(array, &array[2]); → array[0] becomes 4 and array[2] becomes 5

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

cout << array[i] << endl;

\*Print each element then newline

1. void deleteG(char\* c)

{

char\* dp = nullptr;

while (\*c != 0) {

if (\*c == 'g' || \*c == 'G') {

for (dp = c; \*(dp + 1) != 0; dp++) {

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

}

\*dp = '\0';

}

c++;

}

}