Junho Choi

Prof. Smallberg

CS 31

24 May 2020

Project 6

**1A**

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 //put brackets around ptr +1

ptr += 2;

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

while (ptr >= arr)

{

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

ptr--; // changed location of ptr--

}

cout << endl;

}

**1B :** There is an error in the findDisorder function. A reference to the pointer must be created, because the pointer does not originally point to any address. Therefore, p is not changed outside of the scope of the findDisorder function. This can be fixed by making the parameter p of findDisorder a call by reference to the pointer instead of just a pointer.

void findDisorder(int arr[], int n, int\*& p) //changed int\* p ==> int\*& p

{

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

{

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

{

p = arr + k;

return;

}

}

p = nullptr;

}

int main()

{

int nums[6] = { 10, 20, 20, 40, 30, 50 };

int\* ptr;

findDisorder(nums, 6, ptr);

if (ptr == nullptr)

cout << "The array is ordered" << endl;

else

{

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

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

cout << "The item's value is " << \*ptr << endl;

}

}

**1C :** There is an error in the main function. Because the third parameter of the hypotenuse function is a pointer, an address must be passed when calling the function in order to modify the value at that address. This can be done by changing p into a double variable, and passing the reference to the address p when calling the function, and printing p without dereferencing it (because it is already dereferenced). Another option is to create a separate double variable, and assign its address to p before calling the function (not shown).

#include <iostream>

#include <cmath>

using namespace std;

void hypotenuse(double leg1, double leg2, double\* resultPtr)

{

\*resultPtr = sqrt(leg1\*leg1 + leg2\*leg2);

}

int main()

{

double p; //changed double\* p ==> double p

hypotenuse(1.5, 2.0, &p); //changed p ==> &p

cout << "The hypotenuse is " << p << endl; //changed \*p ==> p

}

**1D :** There is an error in the match function. When the corresponding characters are compared, the value at the address pointed to by str1 and str2 must be compared, not the address themselves. To fix this, the dereferenced values of the address str1 and str2 must be compared by using the“\*ptr” syntax.

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

{

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

{

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

return false;

str1++; // advance to the next character

str2++;

}

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

}

int main()

{

char a[10] = "pointy";

char b[10] = "pointless";

if (match(a,b))

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

}

**1E :** There is an error in the computeSquares function. The computeSquares function does not work because it returns the address of a variable that is outside the scope of the main function. Therefore, the address of arr is returned correctly, but the actual array is "gone" and calling the values pointed to by ptr will be whatever values are stored at that memory location.

/\*function not modified\*/

using namespace std;

int\* computeSquares(int& n)

{

int arr[10];

n = 10;

for (int k = 0; k < n; k++)

arr[k] = (k+1) \* (k+1);

return arr;

}

void f()

{

int junk[100];

for (int k = 0; k < 100; k++)

junk[k] = 123400000 + k;

}

int main()

{

int m;

int\* ptr = computeSquares(m);

f();

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

cout << ptr[i] << ' ';

}

**2**

int main(){

string\* fp; //a

string fish[5]; //b

fp=&fish[4]; //c (also fp = fish+4 would also work)

\*fp = "yellowtail"; //d

\*(fish+3) = "salmon"; //e

fp-=3; //f

fp[1]= "basa"; //g

fp[0] = "sole"; //h

bool d = (fp==fish); //i

bool b = (\*fp == \*(fp+1)); //j

}

**3A**

double computeAverage(const double\* scores, int nScores)

{

int i = 0;

double tot = 0;

while (i != nScores)

{

tot += \*(scores+i);

i++;

}

return tot/nScores;

}

**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){

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

if(\*str==chr)

return str;

return nullptr;

}

**4 :**

dif=1

4

79

5

9

-1

19

using namespace std;

int\* minimart(int\* a, int\* b) //

{

if (\*a < \*b) //5<4 is not true, go to else

return a;

else

return b; //returns pointer to index 2 of array

}

void swap1(int\* a, int \*b)

{

int\* temp = a;

a = b;

b = temp;

}

void swap2(int\* a, int \*b)

{

int temp = \*a; //temp = 5

\*a = \*b; //\*a = 4

\*b = temp; //\*b = 5

}

int main()

{

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

int\* ptr = minimart(array, &array[2]); //ptr points to index 2 of array, array is {5,3,4,17,22,19}

ptr[1] = 9; //value of array at index (2 + 1)= 3 becomes 9, array is {5,3,4,9,22,19}

ptr += 2; //pointer now points to index 4 of array, array is {5,3,4,9,22,19}

\*ptr = -1; //value of array at index 4 becomes -1, array is {5,3,4,9,-1,19}

\*(array+1) = 79; //value at index 1 of array (0+1)=1 becomes 79, array is {5,79,4,9,-1,19}

cout << "diff=" << &array[5] - ptr << endl; //prints difference between in location of index 5 of array and where pointer currently points which is index 4; 5-4 = 1

swap1(&array[0], &array[1]); //swaps memory location of index 0 and index 1 of array, array is {5,79,4,9,-1,19}

swap2(array, &array[2]); //switches value of array pointing at index 1 and index 2 (switches value of index 0 and 2 instead because memory location of index 0 and 1 are now switched)., array is {4,79,5,9,-1,19}

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

cout << array[i] << endl; //prints entire array on each row

}

**5**

void deleteG(char \*message){

for(;\*message!=0;message++)

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

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

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

}

}

int main()

{

char msg[100] = "I recall the glass gate next to Gus in Lagos, near the gold bridge.";

deleteG(msg);

cout << msg; // prints I recall the lass ate next to us in Laos, near the old bride.

}