Wenjie Mo

Professor Smallberg

CS 31

25 Nov 2019

Project 6

1a.

int main()

{

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

int\* ptr = arr;

\*ptr = 30;

\*(ptr + 1) = 20; // \* has higher priority \*ptr + 1

// refers to int 31.

ptr += 2;

ptr[0] = 10;

for (ptr = arr; ptr < arr + 3; ptr++) // loop begins from the first item

{

cout << \*ptr << endl; // first change the value and then change the position

}

}

1b.

void findMax(int arr[], int n, int\*& pToMax) // pass by reference, so the pointer in the main function can be changed by the function. The original code creates a copy of pointer, so the pointer will not be changed

{

if (n <= 0)

return; // 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; // the reference is required or the pointer will pointe to meaningless values

}

1c.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int k;

int\* ptr = &k; // The pointer needs to be initilized to assign to a position. The use of uninitialized pointer will lead to undefined behavior.

computeCube(5, ptr);

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

}

1d.

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

{

while (\*str1 != 0 && \*str2 != 0) // we need the character the pointer pointed to in the cstring instead of address of a particular element

{ if (\*str1 != \*str2) // compare between values (same problem as above)

return false;

str1++;

str2++;

}

return \*str1 == \*str2;// return the value comparison

}

int main()

{

char a[15] = "Chen";

char b[15] = "Cheng";

if (strequal(a,b))

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

}

1e

The anArray has the scope in the getPtrToArray function, the pointer’s pointed value in the array is limited inside the function. So when the pointer points to a certain value. Exiting the function, the anArray will not be stored in the main function. The pointer will point to some garbage values, which is a undefined behavior.

2

a. double\* cat;

b. double mouse[5];

c. cat = mouse + 4;

d. \*cat = 25;

e. \*(mouse+3) = 54;

f. cat -= 3;

g. cat[1] = 27;

h. cat[0] = 42;

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

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

3a

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

{

double tot = 0;

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

{

tot += \*(scores + i);

}

return tot/numScores;

}

3b

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

This program prints 3 4 79 -1 9 22 19 vertically.

#include <iostream>

using namespace std;

int\* maxwell(int\* a, int\* b) // return the position with larger value

{

if (\*a > \*b)

return a;

else

return b;

} // return a pointer which pointes to a larger value

void swap1(int\* a, int\* b) // swap between the address pointer pointed to

{

int\* temp = a; // create copy of address a pointed to

a = b; // make a pointer to the objected b pointed to

b = temp; // make b pointed to what a formerly pointed to

} // but the swap only happens in the scope of this function, so no effects in the main route

void swap2(int\* a, int\* b) // swap between value

{

int temp = \*a; // created a copy of value a pointed to

\*a = \*b; // the change the value b pointed to to the value of a pointed to

\*b = temp; // change the value of b pointed to to for former a value

}

int main() {

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

int\* ptr = maxwell(array, &array[2]); // compare position 0 and position 2, return position 0(5 is larger)

\*ptr = -1; // position 0's value changes to -1 array is now { -1, 3, 4, 17, 22, 19 }

ptr += 2; // move forward two position (to position 2)

ptr[1] = 9; // change the value at position 2+1(=3) to 9. Now the array is {-1, 3, 4, 9, 22, 19}

\*(array+1) = 79; // change the value of position 1 in array to 79. Now the array is {-1, 79, 4, 9, 22, 19}

cout << &array[5] - ptr << endl; // print the position difference between position 5 and position 2 in array (print 3)

swap1(&array[0], &array[1]); // no change to the original array

swap2(array, &array[2]); // value swap between position 0 and position 2. Now the array is {4, 79, -1, 9, 22, 19}

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

cout << array[i] << endl; (print the elements in the array {4, 79, -1, 9, 22, 19}, each element with one line)

}

//out put

// 3 position difference between pos 5 and pos 2 in array

// 4 value swap between pos 0 and pos 2

// 79 value at pos 1 is assigned to 79

// -1 pos 0's value changes to -1 (its at pos 2 when cout)

// 9 change the value at position 2+1(=3) to 9

// 22 original

// 19 original

5

#include <iostream>

using namespace std;

void removeS(char\* msg)

{

char\* ptr = msg; // create a new copy

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

{

if (\*msg != 's' && \*msg != 'S') //if the s is not detected

{

\*ptr = \*msg; // copy the element into new cstring

ptr++;

}

msg++; // go through next element

}

\*ptr = '\0'; // add 0 pointer to the end

}

int main()

{

char msg[50] = "She'll be a massless princess.";

removeS(msg);

cout << msg; // prints he'll be a male prince.

}