Huimin Wang

CS 31 Lecture 2

Professor Smallberg

27 November 2017

Homework

1.

a.

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

while (~~ptr >= arr~~)

{

~~ptr--;~~

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

}

}

Fixed version:

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 = 10;        // set arr[2] to 10

    ptr = ptr - 2;

    while (ptr <= &arr[2])

    {

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

        ptr++;

    }

}

b.

The function did not work correctly because the pointer pToMax should be a passed by reference instead of a passed by value. The function only copied ptr and changed the local variable, but left ptr in main routine unchanged.

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

{

    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;

}

c.

The main routine declared a pointer, but did not specify which object it was pointing to. Therefore after the function changed the value of the object pointed by ptr, we can't follow ptr to find out which object it was pointing.

void computeCube(int n, int\* ncubed)

{

    \*ncubed = n \* n \* n;

}

int main()

{

    int result = 0;

    int\* ptr = &result;

    computeCube(5, ptr);

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

}

d.

The function should check the objects following the pointers instead of checking the pointers themselves. Corresponding characters are represented by \*str1 and \*str2, not str1 and str2.

// return true if two C strings are equal

bool strequal(const char\* str1, const char\* str2)

{

    while (\*str1 != 0  &&  \*str2 != 0)

    {

        if (\*str1 == \*str2){

            str1++;

            str2++;

        }  // compare corresponding characters

        else

            return false;

    }

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

}

int main()

{

    char a[15] = "Zhou";

    char b[15] = "Zhu";

    if (strequal(a,b))

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

}

e. The program does not work as intended because anArray is declared and initialized within getPtrToArray() function. anArray is therefore a local variable and does not exist outside the function. When the program tries to use ptr to write the elements that are supposed in anArray, the objects pointed by ptr are no longer the values of anArray, but instead some garbage values from junk.

2.

1. double\* cat;
2. double mouse[5];
3. cat = &mouse[4];
4. \*cat = 25;
5. \*(mouse + 3) = 42;
6. cat -= 3;
7. cat[1] = 54;
8. cat[0] = 27;
9. bool b = (\*cat == \*(cat + 1));
10. bool d = (cat == mouse);

3.

a.

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

{

    int i = 0;

    const double\* ptr = scores;

    double tot = 0;

    while (i != numScores)

    {

        tot += \*(ptr + i);

        i++;

    }

    return tot/numScores;

}

b.

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;

}

c.

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

{

    while (\*str != 0) {

        if (\*str == chr)

            return str;

        str++;

    }

    return nullptr;

}

4.

**3** --> maxwell()initializes ptr to be the pointer to array[0]. ptr+=2 moves the pointer forward by two integers, so ptr now points to array[2]. The difference between &array[5] and &array[2] is 3.

**4** --> swap2(array, &array[2])swaps the first element of the array with the third element of the array. Now array[2], which is 4, becomes the first element. The program writes array[0].

**79** --> \*(array+1) is the object immediately following the first element of the array. We set this integer, or the second element of the array, to 79. The program writes array[1].

**-1** --> After maxwell() initializes ptr to be the pointer to array[0], the program follows ptr and sets its object to -1. Then swap2(array, &array[2]) swaps array[0] with array[2], so array[2] now is -1. The program writes array[2].

**9** --> maxwell()initializes ptr to be the pointer to array[0]. ptr+=2 moves the pointer forward by two integers, so ptr now points to array[2]. The program sets the element after the object pointed by ptr, which is array[3], to 9. The program writes array[3].

**22** --> The program writes array[4].

**19** --> The program writes array[5].

5.

void removeS(char\* str) {

    while (\*str != 0) {

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

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

                \*ptr = \*(ptr + 1);

            }

            str--;

        }

        str++;

    }

    str = 0;

}