Introduction to Computer Science I

Winter 2019 - COM SCI31-1 - STAHL

Project 6

Programming Assignment 6
Get The Point!

Time due: 9:00 PM ~~Wednesday, March 6th~~ Thursday, March 7th

Introduction

This project is designed to help you master pointers. To that end, you'll get the most out of it by working through the problems by hand. Only after that should you resort to running the programs (and stepping through them with the debugger) to check your understanding. Remember, on the final exam you'll have to be able to do problems like this by hand.

Your task

This "project" is more like one of the accompanying homeworks. There are ~~five~~ four problems. In the problems that ask you to change code, make the few changes necessary to fix the code without changing its overall approach. For example, don't fix the program in problem 1a by changing it to

```
int main()
{
     cout << " 1000 100 10 1" <<  endl;
}
```

Problem 1: This problem involves errors related to the use of pointers.

a. This program is supposed to write `1000 100 10 1` , but it doesn't. Find all of the bugs and show a fixed version of the program:

```
    #include <iostream>
    using namespace std;

    int main()
    {
        int arr[4] = { 0, 1, 2, 3 };
        int* ptr = arr;

        *ptr = arr[ 1 ];                   // set arr[0] to 1
        *(ptr + 1) = arr[ 0 ] * 10;        // set arr[1] to 10
        ptr += 2;
        ptr[0] = arr[ 1 ] * 10;            // set arr[2] to 100
        ptr[1] = 1000;                     // set arr[3] to 1000

        while (ptr >= arr)
        {
            ptr--;
            cout << " " << *ptr;       // print values
        }
        cout << endl;
        return( 0 );
    }
```

b. The `findLastZero` function is supposed to find the last element in an array whose value is zero, and sets the parameter `p`to point to that element, so the caller can know the location of that element holding the value zero. Explain why this function won't do that, and show how to fix it. Your fix must be to the function only; you must not change the `main` routine below in any way. As a result of your fixing the function, the `main` routine below needs to work correctly.

```
    #include <iostream>
    using namespace std;

    void findLastZero(int arr[], int n, int* p)
    {
        p = nullptr;    /// default value if there isn't a 0 in the array at
all
        for (int k = n - 1; k >= 0; k--)
        {
            if (arr[k] == 0)      // found an element whose value is 0
            {
                 p = arr + k;     // change the value of p
                 break;           // stop looping and return
            }
        }
    }

    int main()
    {
        int nums[6] = { 10, 20, 0, 40, 30, 50 };
        int* ptr;

        findLastZero(nums, 6, ptr);
        if (ptr == nullptr)
        {
            cout << "The array doesn't have any zeros inside it." << endl;
        }
        else
        {
            cout << "The last zero is at address " << ptr <<  endl;
            cout << "It's at index " << ptr - nums << endl;
            cout << "The item's value is " << *ptr << " which is zero!"
<< endl;
        }
        return( 0 );
    }
```

c. The `biggest` function is correct, but the main function has a problem. Explain why it may not work, and show a way to fix it. Your fix must be to the main function only; you must not change the `biggest` function in any way.

```
    #include <iostream>
    using namespace std;

    void biggest(int value1, int value2, int * resultPtr)
    {
        if( value1 > value2 )
        {
               *resultPtr = value1;
        }
        else
        {
               *resultPtr = value2;
        }
    }

    int main()
```

```
{
    int* p;
    biggest(15, 20, p);
    cout << "The biggest value is " << *p << endl;
    return( 0 );
}
```

d. The `match` function is supposed to return `true` if and only if its two C-string arguments have exactly same text. Explain what the problems with the implementation of the function are, and show a way to fix them.

```
// return true if two C strings are equal
bool match(const char str1[], const char str2[])
{
    bool result = true;
    while (str1 != 0  && str2 != 0)  // zero bytes at ends
    {
        if (str1 != str2)  // compare corresponding characters
        {
            result = false;
            break;
        }
        str1++;              // advance to the next character
        str2++;
    }
    if (result)
    {
        result = (str1 == str2);    // both ended at same time?
    }
    return( result );
}

int main()
 {
    char a[10] = "pointy";
    char b[10] = "pointless";

    if (match(a,b))
    {
        cout << "They're the same!" << endl;
    }
}
```

e. This program is supposed to write `1 1 2 3 5 8 13 21` but it probably does not. What is the program doing that is incorrect? (We're not asking you explain why the incorrect action leads to the particular outcome it does, and we're not asking you to propose a fix to the problem.)

```
 #include <iostream>
 using namespace std;

 int fibonacci( int n )
 {
     int tmp;
     int a = 1;
     int b = 1;

     for (int i = 0; i < n-2; i ++)
     {
         tmp = a+b;
         a = b;
         b = tmp;
     }
     return b;
}

int* computeFibonacciSequence(int& n)
{
     int arr[8];
     n = 8;
     for (int k = 0; k < n; k++)
     {
         arr[k] = fibonacci( k+1 );
     }
     return arr;
 }

 int main()
 {
     int m;
     int* ptr = computeFibonacciSequence(m);
     for (int i = 0; i < m; i++)
     {
         cout << ptr[i] << ' ';
     }
```

```
        return( 0 );
    }
```

Problem 2: Match each of the following statements with the explanation of what the statement does. (NOTE: One of the descriptions listed below actually matches two statements)

Statement	**Description**
1. `string * fp;`	a. sets a pointer variable to the last element of an array of five strings
2. `string fish[ 5 ];`	b. sets the string pointed to by a pointer variable to the value `"salmon"`
3. `fp = &fish[ 4 ];`	c. sets the fourth element of an array pointed to by the variable `fp` to the value `"salmon"`
4. `*fp = "salmon";`	d. moves the pointer `fp` back three strings in the array it points to
5. `fp -= 3;`	e. initializes a boolean to `true` if the pointer variable `fp` points to the string at the start of the `fish` array, `false` otherwise
6. `*(fp + 3) = "salmon";`	f. declares a pointer variable to point at a variable of type `string`
7. `fp[ 0 ] = "salmon";`	g. declares a five element array of `string`
8. `bool b = (fp == fish);`	h. initializes a boolean to `true` if `fp` points to a `string` whose value matches the string immediately following the string pointed to by `fp`, `false` otherwise
9. `bool b = (*fp == *(fp + 1));`	

Problem 3: What does the following program print and why? Be sure to explain why each line of output prints the way it does to get full credit.

```
    #include <iostream>
    using namespace std;

    int* minimart(int* a, int* b)
      {
        if (*a < *b)
            return a;
        else
            return b;
    }
```

```
void swap1(int* a, int *b)
{
    int* temp = a;
    a = b;
    b = temp;
}

void swap2(int* a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
  }

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

    int* ptr = minimart(array, &array[2]);
    ptr[1] = 9;
    ptr += 2;
    *ptr = -1;
    *(array+1) = 79;

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

    swap1(&array[0], &array[1]);
    swap2(array, &array[2]);

    for (int i = 0; i < 6; i++)
        cout << array[i] << endl;

    return( 0 );
}
```

Problem 4: Write a function named `deleteDigits` that accepts one character pointer as a parameter and returns no value. The parameter must be a C-string. This function must remove all of the digit character letters from the string. The resulting string must be a valid C-string.

Your function must declare no more than one local variable in addition to the parameter; that additional variable must be of a pointer type. Your function must not use any square brackets and must not use the `strlen` or `strcpy` library functions.

```
int main()
```

```
    {
        char msg[100] = "Happy 2019!";
        deleteDigits(msg);
        cout << msg << endl;        // prints:   Happy !
    }
```

What To Turn In

Prepare your solutions to these homework problems in a single Word document named **hw.doc** or **hw.docx**, or a plain text file named **hw.txt**. Put that file in a zip file, and follow the Project 6 link to turn in the zip file. Please be sure to put your name and your UCLA ID number into this homework document.

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Thursday, 7 March 2019, 9:00 PM PST
Time remaining	2 days 10 hours
Last modified	-
Submission comments	Comments (0)

Add submission

◀ DynamicAirplane VS2017

Jump to...

Notes From Class - Monday ▶