**Project 6**

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

    ptr = arr;

    while (ptr <= arr + 2)

    {

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

        ptr++;

    }

}

* 1. The problem with the original code was that the pointer was not passed by reference.

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;

    }

}

* 1. The problem with the original code was that the pointer was not initialized.

int main()

{

    int x;

    int\* ptr = &x;

    computeCube(5, ptr);

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

}

* 1. The problem with the original code was that the function was comparing the memory locations of the two strings, which will always compute false, instead of the content of the strings.

// return true if two C strings are equal

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

{

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

    {

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

            return false;

        str1++;            // advance to the next character

        str2++;

    }

    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";

}

* 1. The problem with this code is that ptr is only initialized to the first element of anArray. This is because getPtrToArray is supposed to return a pointer to an int, but the function is actually returning the whole array of elements. Since ptr only points to the first element, the rest of the values in the array are ignored and those memory addresses are allocated to other values, like values in the array junk.
  2. double\* cat;
  3. double mouse[5];
  4. cat = &mouse[4]
  5. \*cat = 25;
  6. \*(mouse + 3) = 42;
  7. cat -= 3;
  8. cat[1] = 54;
  9. cat[0] = 27;
  10. bool b = (\*cat == \*(cat + 1));
  11. bool d = (cat == mouse);

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

{

    const double\* ptr = scores;

    double tot = 0;

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

        tot += \*(ptr + i);

    }

    return tot / numScores;

}

// 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;

}

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

{

    while (\*str != '\0') {

        if (\*str == chr)

            return str;

        else

            str++;

    }

    return nullptr;

}

1. Output is:

3

4

79

-1

9

22

19

3: ptr points to the first element and then increments by 2 to point to the third element. &array[5] points to the sixth element. The difference between the two is 3.

4: swap2 switches the first and third element of the array. Before this function, the third element was 4, so now the first element is 4.

79: \*(array + 1) changes the value of the second element to 79.

-1: swap2 switches the first and third element of the array. Before this function, the first element was -1 because int\* ptr = maxwell(array, &array[2]) points ptr to the first element, and \*ptr = -1 sets the first element to -1. So now the third element is -1.

9: ptr[1] = 9 sets the fourth element to 9 because ptr was at the third position.

22: The fifth element of the array is unchanged.

19: The sixth element of the array is unchanged.

void removeS(char\* ptr) {

    while (\*ptr != 0) {

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

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

                \*var = \*(var + 1);

        } else {

            ptr++;

        }

    }

}