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) = 10; // set arr[2] to 10

while (ptr <= arr+2)

{

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

ptr++;

}

}

The pointer passed to the function *findMax* will not be modified by the function; instead, we should pass a reference of the pointer to the function

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 pointer variable in the main function is not initialized before passing to the function *computeCube*

int main()

{

int cube = 5;

int\* ptr = &cube;

computeCube(5, ptr);

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

}

1. The quit condition in the while loop is not correct because it compares a pointer to 0, and the return statement at the end of the function compares two pointers, which will never be 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] = "Shen";

char b[15] = "Shen";

if (strequal(a, b))

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

}

1. The array in the function getPtrToArray is a local variable, which means that the array created is temporary, and any pointer that points to this local array will points to some random memory address after the local array is cleaned.
2. double\* cat;
3. double mouse[5];
4. cat = mouse+4;
5. \*cat = 25;
6. \*(mouse + 3) = 54;
7. cat = cat - 3;
8. cat[1] = 42;
9. cat[0]=27;
10. bool b = (\*cat == \*(cat + 1));
11. bool d = (mouse[0] == \*cat);

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

{

const double\* ptr = scores;

double tot = 0;

int i = 0;

while (i< numScores)

{

tot += \*(ptr+i);

i++;

}

return tot / numScores;

}

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)

{

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

if (\*str == chr)

return str;

return nullptr;

}

The program prints 3, 4, 79, -1, 9, 22, each is a separate line

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

int\* ptr = maxwell(array, &array[2]);

//the pointer ptr points to the larger value between array[0] and array[2], which is //the address of array[0]

\*ptr = -1; //The value ptr ponits to, which is array[0], is set to -1

ptr += 2;//ptr now points to array[2]

ptr[1] = 9; //The value of array[3] is set to 9

\*(array + 1) = 79; //The value of array[1] is set to 79

cout << &array[5] - ptr << endl;

//the program prints the address of array[5] minus the address of array[2], which is 3

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

the address of array[0] and array[1] is swaped in local variable of swap1, but the //address in main is not swaped

swap2(array, &array[2]);

//The value of &array[0] points to and &array[2] points to is swaped, now

//array[0] is 4 and array[2] is 79

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

cout << array[i] << endl;

// the value of the elements in the array is 4, 79, -1, 9, 22, 19

void removeS(char\* str)

{

char\* i = str;

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

{

if(\*str!='s' && \*str!='S')

{

\*i = \*str;

i++;

}

}

\*i = '\0';

}