Computer Science Project 6

1. \*ptr + 1 = 20; will not set the element in index 1 to 20, as the pointer itself is not being increased, but rather the value that the pointer points to is being increased by 1.

int main()

{

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

int\* ptr = arr;

\*ptr = 30;

\*(ptr + 1) = 20;

// a parenthesis is added so that the pointer is incremented by one

// before accessing the integer value using the \* operator

ptr += 2;

ptr[0] = 10;

while (ptr >= arr)

{

ptr--;

cout << \*ptr << endl;

}

}

1. The pointer is passed by value in the function, void findMax(int arr[], int n, int\* pToMax); The pointer must be passed by reference, so that changes made to the pointer in the function also affect the pointer in the main routine.

void findMax(int arr[], int n, int\* &pToMax)// pointer is passed by

// reference

{

if (n <= 0)

return;

pToMax = arr;

for (int i = 1; i < n; i++)

{

if (arr[i] > \*pToMax)

pToMax = arr + i;

}

}

1. Instead of passing an uninitialized pointer as a parameter, a pointer to a particular variable should be passed by value in the ‘computeCube’ function.

void computeCube(int n, int\* ncubed)

{

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

}

int main()

{

int cubeValue;

computeCube(5, &cubeValue); // a pointer to a particular

// variable is passed by value

cout << "Five cubed is " << cubeValue << endl;

}

1. The function parameters should not be declared as constant, as the value of str1 and str2 is being changed in the while loop

The while loop should check whether the pointer points to a null character, not whether the pointer is a null pointer.

In the If condition inside the while loop, the pointers should not be compared, but rather the values that pointer points to should be compared to test for inequality.

To check whether both strings end at the same time, the last values that the pointer points to should be checked for equality, not the pointers themselves.

bool strequal(char str1[], 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] = "Tang";

char b[15] = "Zhang";

if (strequal(a,b))

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

}

1. The function getPtrToArray returns a pointer to the first element in the array, which was initialized inside the function as a local variable. Hence only the value of the first element (value at index = 0) is returned to the main routine while all other values in the array are discarded.

int anArray[100];// anArray is initialized as a local variable

......

return anArray; // only the pointer to the first element is returned

int main()

{

double\* cat; // (a)

double mouse[5]; // (b)

cat = &mouse[5]; // (c)

\*cat = 42; // (d)

\*(mouse + 3) = 17; // (e)

cat = cat - 3; // (f)

cat[1] = 25; // (g)

cat[0] = 54; // (h)

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

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

}

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

{

const double\* ptr = scores;

double tot = 0;

for (int x = 0; x != numScores; x++)

{

tot += \*(ptr+x);

}

return tot/numScores;

}

1. 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;

}

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

{

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

if (\*(str) == chr)

return str;

return nullptr;

}

1. int main()

{

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

int\* ptr = maxwell(array, &array[2]); // returns pointer to array[0]

\*ptr = -1; // array = { -1, 3, 4, 17, 22, 19 }

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

ptr[1] = 9; // array = { -1, 3, 4, 9, 22, 19 }

\*(array+1) = 79; // array = { -1, 79, 4, 9, 22, 19 }

cout << &array[5] - ptr << endl; // outputs (5 - 2): 3

swap1(&array[0], &array[1]); // no change as pointers are only

// locally altered in the function

swap2(array, &array[2]); // array = { 4, 79, -1, 9, 22, 19 }

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

cout << array[i] << endl;// outputs: 4 79 -1, 9 22 19

}

Output:

3

4

79

-1

9

22

19

1. void removeS(char\* str)

{

while (\*str != 0)

{

if (\*str == ‘s’ || \*str == ‘S’)

{

char\* ptr = str;

while (\*(ptr+1) != 0) // shifts elements one place to the left

{

\*ptr = \*(ptr+1);

ptr++;

}

\*ptr = 0; // last element is set as a null character

str--; // str is decreased by 1 as the elements were shifted left

}

str++;

}

}