**STRLEN(): Returns the size of a c string BEFORE the ‘\0’, BUT THE SIZE OF THE CHAR ARRAY WHEN DECLARED HAS TO INCLUDE THE 0 BYTE.**

**Looping through a c string:**

For(int i=0; i!=strlen(cstring); i++){

}

We can’t += to C STRINGS, only assign values to array elements!

**If you’re comparing the current index to the next index remember to have the condition LESS THAN the SIZE – 1 !!!**

**Strcpy(a, b) – copies b into a IF a has enough room**

**Strcat(a, b) – adds b onto a if there’s enough room**

Bool containsReverse(char first[], char second[]){

If(strlen(second)>strlen(first)){

Return false;

// Go through all possible substrings of length strlen(second) within first

for (int k1 = 0; k1 < strlen(first) - strlen(second) + 1; k1++) {

int char\_count = 0; // Counter keeps track of what char we're looking at

bool match = true; // Assume they match until proved otherwise

// Go through second in reverse order to try to find a match

for (int k2 = strlen(second) -1 ; k2 >= 0; k2--) {

// If we find a mismatch, go to next possible substring

if (first[k1+char\_count] != second[k2]){

match = false;

break;

}

char\_count++;

}

if (match)

return true;

}

return false;

}

#include <iostream>

using namespace std;

int main () {

// local variable declaration:

char grade = 'D';

switch(grade) {

case 'A' :

cout << "Excellent!" << endl;

break;

case 'B' :

case 'C' :

cout << "Well done" << endl;

break;

case 'D' :

cout << "You passed" << endl;

break;

case 'F' :

cout << "Better try again" << endl; break;

default :

cout << "Invalid grade" << endl;

}

cout << "Your grade is " << grade << endl; return 0; }

OUTPUT: YOU PASSED

YOUR GRADE IS D

**Strcmp(a ,b) – compares a to b**

**If(strcmp(a,b) <0) – Checks if a<b**

**“ “>0) – Checks if a>b**

void strfly(char a[]) {

int i = 0;

int d = 1;

while (true) {

a[i] = toupper(a[i]);

if (i > 0) {

// start check a[i-1] = tolower(a[i-1]); }

if (a[i] != '\0' && a[i+1] == '\0') {

// end check a[i+1] = tolower(a[i+1]); }

if (i == 0) {

d = 1;

}

else if (a[i] == '\0') {

d = -1;

}

i += d;

cout << a << endl;

usleep(100000); // 0.1 second } }

**DECLARING ARRAYS**

double d[5]; // 5 undefined double values.

double d[3] = {1,2,3}; // 3 double values: 1, 2, 3.

double d[] = { 4, 5 }; // 2 double values: 4 and 5.

double d[10] = { 0 }; // 10 double values equal to 0.

* **Pass by REFERENCE**
  + Used to return multiple variables/params and to SAVE MEMORY AND TIME INSTEAD OF COPYING
  + Can only take VARIABLES as its parameters, NO LITERALS!
  + Parameters are changed INSIDE and OUTSIDE the function
  + The SWAP function is always PASS BY REF
  + An arrays name is a REFERENCE to the first element‘s ([0]) position in MEMORY.
  + Use **CONST** to get the reference benefit of not copying and your param won’t get MODIFIED
* **Pass by VALUE**
  + A COPY OF THE PARAM PASSED IN
  + Can take in literals
  + Will only change the input within the function and can return it