# Task 1 Combining Strings

## Main.cpp

1. // 210CTTask1.cpp : Defines the entry point for the console application.
2. //
4. #include "stdafx.h"
5. #include "combiningStrings.h"
6. **using** **namespace** std;
8. **int** main()
9. {
10. string word1;
11. string word2;
12. getline(cin, word1);
13. getline(cin, word2);
14. string newWord = combineStrings(word1, word2);
15. cout << newWord;
16. cin.get();
17. cin.get();
18. **return** 0;
19. }

## Header Algorithm

1. #pragma once
2. #include <string>
3. #include <iostream>
4. #include <string>
5. **using** **namespace** std;
6. /\*
7. Purpose of this program is to understand how algorithms work
8. A function that takes two inputs and then combines them
9. \*/
10. string combineStrings(string word1, string word2)
11. {
12. //find out the lengths of the word
13. **int** j = word1.length();
14. **int** k = word2.length();
15. string newWord; //initialise new word
16. **if** (j > k || j == k){ //check which word is longer
17. **for** (**int** i = 0; i < k; i++){
18. newWord += word1[i];
19. newWord += word2[i];
20. }
21. **for** (**int** i = k; i < j; i++){
22. newWord += word1[i];
23. }
24. }
25. **if** (k > j){
26. **for** (**int** i = 0; i < j; i++){
27. newWord += word1[i];
28. newWord += word2[i];
29. }
30. **for** (**int** i = j; i < k; i++){
31. newWord += word2[i];
32. }
33. }
34. **return** newWord;
35. }

## Test.cpp

1. #include "stdafx.h"
2. #include "CppUnitTest.h"
3. #include "../210CT\_Task 1/combiningStrings.h"
5. **using** **namespace** Microsoft::VisualStudio::CppUnitTestFramework;
7. **namespace** UnitTest1
8. {
9. TEST\_CLASS(UnitTest1)
10. {
11. **public**:
12. TEST\_METHOD(TestFullWords)
13. {
14. string newWord = "HTehlelroe"; //Hello There
15. string newWord2 = "DTaiyme"; //Day Time DTaiyme
16. string newWord3 = "AGdialmmour"; //Adam Gilmour
17. Assert::AreEqual(combineStrings("Hello", "There"), newWord);
18. Assert::AreEqual(combineStrings("Day", "Time"), newWord2);
19. Assert::AreEqual(combineStrings("Adam", "Gilmour"), newWord3);
20. }
21. TEST\_METHOD(SpacedWords)
22. {
23. string newWord4 = "HAedlalmo Me"; //Hello Me, Adam
24. string newWord5 = "URpisgihdtes iDdoew nUp"; //Upside Down, Rightside Up
25. string newWord6 = "SSkuyn  aanndd  SMeoaon"; //Sky and Sea, Sun and Moon
26. Assert::AreEqual(combineStrings("Hello Me", "Adam"), newWord4);
27. Assert::AreEqual(combineStrings("Upside Down", "Rightside Up"), newWord5);
28. Assert::AreEqual(combineStrings("Sky and Sea", "Sun and Moon"), newWord6);
29. }
30. TEST\_METHOD(SpaceAtBegining)
31. {
32. string newWord7 = " Hello"; // " ", "Hello"
33. Assert::AreEqual(combineStrings(" ", "Hello"), newWord7);
34. }
35. };
36. }

A computer algorithm is a set of rules that a program follows to complete a certain task, in this example the program is told to check which word is the longest, then for each letter of each word, append that to a new word, then print the new word.

# Task 2 Armstrong Numbers

## Main.cpp

1. // ArmstrongsNumber.cpp : Defines the entry point for the console application.
2. //
4. /\*Check if a 3 digit number is an Armstrong number
5. Three digit number where the sum of the cubes is equal to the numbe it self
7. 371 = 3\*\*3+7\*\*3+1\*3 \*/
9. #include "stdafx.h"
10. #include "armstrong.h"
11. **using** **namespace** std;
13. **int** main()
14. {
15. string userNum;
16. cout << "Please enter a positive number ";
17. cin >> userNum;
18. **while** (!checkInt(userNum))
19. {
20. cout << "That is not a positive integer\n";
21. cout << "Please enter a postive integer\n";
22. cin >> userNum;
23. }
25. **int** intUserNumb = stoi(userNum);
26. **int** check = checkArm(intUserNumb, armstrong(intUserNumb));
27. **if** (check == **true**)
28. {
29. cout << "The number " << intUserNumb << " is an Armstrong Number.";
30. cin.get();
31. }
32. **else**
33. {
34. cout << "The number " << intUserNumb << " is not an Armstrong Number";
35. cin.get();
36. }
37. cin.get();
38. **return** 0;
39. }

## Header Algorithm

1. #pragma once
2. #include <string>
3. #include <iostream>
4. #include <cmath>
5. /\* Function to calculate the value of the number when you take their digits
6. and then raise them to the power of 3 and then add them\*/
7. **int** armstrong(**int** integer)
8. {
9. //I used an iterative function because they are much more efficient
10. //If this was going to be a larger number, it would take longer to do a recursive function
11. **int** digit;
12. **int** total = 0;
13. **while** (integer != 0)
14. {
15. digit = integer % 10;
16. total += digit \* digit \* digit;
17. integer /= 10;
18. }
19. **return** total;//base case
20. }
21. //function to compare if the number is an armstrong number
22. **bool** checkArm(**int** number, **int** armstrong)
23. {
24. **if** (number == armstrong) **return** **true**;
25. **else** **return** **false**;
26. }
27. //check if user has entered valid input.
28. **bool** checkInt(std::string testNum)
29. {
30. **int** length = testNum.length();
31. **for** (**int** i = 0; i < length; i++)
32. {
33. **if** (!isdigit(testNum[i])) **return** **false**;
34. }
35. **return** **true**;
36. }

## Test.cpp

1. #include "stdafx.h"
2. #include "CppUnitTest.h"
3. #include "..\210CT\_Task 2\armstrong.h"
5. **using** **namespace** Microsoft::VisualStudio::CppUnitTestFramework;
7. **namespace** UnitTest
8. {
9. TEST\_CLASS(TestArmstrong)
10. {
11. **public**:
13. TEST\_METHOD(TestTrue)
14. {
15. // TODO: Your test code here
16. Assert::AreEqual(armstrong(371), 371);
17. Assert::AreEqual(armstrong(153), 153);
18. Assert::AreEqual(armstrong(370), 370);
19. Assert::AreEqual(armstrong(407), 407);
20. }
21. TEST\_METHOD(TestFalse)
22. {
23. Assert::AreNotEqual(armstrong(1233), 1233);
24. Assert::AreNotEqual(armstrong(1633), 1633);
25. Assert::AreNotEqual(armstrong(372), 371);
26. Assert::AreNotEqual(armstrong(2000), 2000);
27. }
28. TEST\_METHOD(WrongInput)
29. {
30. Assert::IsFalse(checkInt("Hello"));
31. Assert::IsFalse(checkInt("9823o"));
32. Assert::IsFalse(checkInt("-0"));
33. }
34. };
35. }

The purpose of this algorithm is to take a number, split the integer into its digits, times the digits by itself and then add them together until the integer is at 0.

# Task 3