CS 16 5/28

# Strings

Another Sample File: Recursion and removing spaces from a string

#include <iostream>

#include <string>

using namespace **std**;

string **recursivelyRemoveSpaces**(string *a*){

 int len = a.**length**();

 string recurse;

 string space = " ";

 if(len == 1 && a[0] != ' '){

 return a.**substr**(0,1);

 }

 recurse = a.**substr**(0, len - 1); *// This is the rest of the string except the last element*

 if(a[len-1] != ' '){

 return **recursivelyRemoveSpaces**(recurse) + a[len-1];

 }

 else{

 return **recursivelyRemoveSpaces**(recurse);

 }

}

string **recursivelyRemoveSpacesAndReverse**(string *a*){

 int len = a.**length**();

 string recurse;

 string space = " ";

 if(len == 1 && a[0] != ' '){

 return a.**substr**(0,1);

 }

 recurse = a.**substr**(0, len - 1); *// This is the rest of the string except the last element*

 if(a[len-1] != ' '){

 return a[len-1] + **recursivelyRemoveSpacesAndReverse**(recurse);

 }

 else{

 return **recursivelyRemoveSpacesAndReverse**(recurse);

 }

}

int **main**()

{

 string m = "H el lo \_W orl d! ";

 string reversedWithoutSpaces, noSpaces;

 cout << "Before removing spaces the string is: " << m <<'\n';

 noSpaces = **recursivelyRemoveSpaces**(m);

 cout << "After removing spaces the string is: " << noSpaces <<'\n';

 cout << "\n\n";

 cout << "Before removing spaces and reversing, the string is: " << m <<'\n';

 reversedWithoutSpaces = **recursivelyRemoveSpacesAndReverse**(m);

 cout << "After removing spaces and reversing, the string is: " << reversedWithoutSpaces <<'\n';

}

Output:

Before removing spaces the string is: H el lo \_W orl d!

After removing spaces the string is: Hello\_World!

Before removing spaces and reversing, the string is: H el lo \_W orl d!

After removing spaces and reversing, the string is: !dlroW\_olleH

* Notice the subtle difference between the two functions is in the return statement

# Fibonacci Sequence

* The Fibonacci Sequence is a mathematical series that explains some of our natural environment.
	+ The sequence: 1,1,2,3,5,8,13,21,34,55,89…
	+ Notice that the third element is a sum of the prior two, the fourth is the sum of the prior two, the fifth is the sum of the prior two…

Sample File:

#include <iostream>

#include <string>

using namespace **std**;

int **Fibbonaci**(int *n*){

 if(n == 0 || n == 1){

 if(n == 0) return 0;

 else return 1; *// ASSUME FIBONACCI STARTS AS 0,1*

 }

 return **Fibbonaci**(n-1) + **Fibbonaci**(n-2);

}

int **main**()

{

 int n = 25;

 cout << "We want to find the " << n << "th element in the Fibonacci Sequence.\n\n";

 cout << "The " << n << "th element of the Fibonacci Sequence: " << **Fibbonaci**(n) << "\n";

}

Output:

We want to find the 25th element in the Fibonacci Sequence.

The 25th element of the Fibonacci Sequence: 75025

# Binary Trees

* Sum of the Tree is the sum of the subtrees + the root

Sample File:

#include <iostream>

#include <string>

using namespace **std**;

struct Tree{

 int data;

 Tree\* left;

 Tree\* right;

};

int **sumTree**(Tree\* *t*){

 if(t == NULL) return 0;

 return **sumTree**(t->left) + t->data + **sumTree**(t->right);

}

int **main**()

{

 Tree\* myTree = new Tree;

 Tree\* a = new Tree;

 Tree\* b = new Tree;

 Tree\* c = new Tree;

 Tree\* d = new Tree;

 Tree\* e = new Tree;

 Tree\* f = new Tree;

 Tree\* g = new Tree;

 myTree->data = 10;

 myTree->left = a;

 myTree->right = b;

 a->data = 45;

 b->data = 55;

 a->left = c;

 a->right = d;

 c->data = 100;

 d->data = 900;

 b->left = e;

 b->right = f;

 e->data = 1000;

 f->data = 9000;

 c->left = g;

 c->right = NULL;

 g->data = 100000;

 d->left = NULL;

 d->right = NULL;

 e->left = NULL;

 e->right = NULL;

 f->left = NULL;

 f->right = NULL;

 g->left = NULL;

 g->right = NULL;

 cout << "The sum of the tree is : " << **sumTree**(myTree) << "\n\n";

}

Output:

The sum of the tree is : 111110