Assignment 1

Kyle Pamintuan

CECS 326

Professor Ratana Ngo

9/13/17

Overview:

The purpose of this programming assignment is largely to help us review C++ but its also to simulate memory management and to help us get a better understanding of one of the main jobs of the operating system: resource allocation.

We created a single struct that contains an array of pointers that points to other arrays. We also created a UI that prompts the user to let the program know what memory they want to allocate, deallocate, and access. We can say that the UI would represent a user app and the memory allocation would represent the OS.

**Assignment1.cpp**

//#include "stdafx.h"

#include <iostream>

using namespace std;

char alphabet[26] = { 'A' , 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

// Function for filling in the 'sizes' array

long int myFunction(int n)

{

if (n == 0)

return 2900;

else

return(2 \* myFunction(n - 1));

}

// Random letter generator

char randLetter()

{

int randIndex = rand() % sizeof(alphabet);

char randChoice = alphabet[randIndex];

return randChoice;

}

// Stuct to facilitate memory management simulation

struct Arrays

{

long int sizes[20];

char \*pointers[20];

};

// Initialize members of the Struct

void initArrays(struct Arrays &a)

{

// Inserting values into the 'sizes' array

for (int i = 0; i < 20; i++)

{

a.sizes[i] = myFunction(i);

}

// Allocate memory for each element in this array. The sizes of memory comes from the 'sizes' array.

for (int i = 0; i < 20; i++)

{

a.pointers[i] = new char[a.sizes[i]];

// Insert letters into the arrays INSIDE the 'pointers' array

for (long int j = 0; j < a.sizes[i]; j++)

{

a.pointers[i][j] = randLetter();

}

}

}

int main()

{

Arrays a1;

initArrays(a1);

int userInput = 0;

int userInput2 = 0;

int accessIndex = 0;

cout << "=== Main Menu ===" << endl;

cout << "1. Access a pointer" << endl;

cout << "2. List deallocated memory (index)" << endl;

cout << "3. Deallocate all memory" << endl;

cout << "4. Exit program" << endl;

cin >> userInput;

while (userInput != 4)

{

if (userInput == 1)

{

cout << endl;

cout << "Which index of the array would you like to access?" << endl;

cin >> accessIndex;

cout << endl;

cout << "=== Options ===" << endl;

cout << "1. Print first 10 chars in the chosen array" << endl;

cout << "2. Delete all the chars associated with this pointer" << endl;

cout << "3. Return to Main Menu" << endl;

cin >> userInput2;

while (userInput2 != 3)

{

if (userInput2 == 1)

{

if (a1.pointers[accessIndex] != NULL) // If accessed memory has NOT been deallocated...

{

// Show the first 10 (or less) characters of memory.

cout << endl;

cout << "Acessing Index: " << accessIndex << endl;

long int arrayLength = a1.sizes[accessIndex];

if (arrayLength < 10)

{

for (long int i = 0; i < arrayLength; i++)

{

cout << a1.pointers[accessIndex][i] << " ";

}

}

else if (arrayLength > 10)

{

for (long int i = 0; i < 10; i++)

{

cout << a1.pointers[accessIndex][i] << " ";

}

}

cout << endl;

}

else if (a1.pointers[accessIndex] == NULL) // If accessed memory has been deallocated...

{

// Notify user

cout << endl;

cout << "This index has been deallocated... " << endl;

cout << endl;

cout << "Reallocating memory now..." << endl;

// Reallocate memory

a1.pointers[accessIndex] = new char[a1.sizes[accessIndex]];

for (long int j = 0; j < a1.sizes[accessIndex]; j++)

{

a1.pointers[accessIndex][j] = randLetter();

}

cout << endl;

}

}

else if (userInput2 == 2)

{

cout << endl;

cout << "Deallocating Index: " << accessIndex << endl;

// Deallocate memory

delete a1.pointers[accessIndex];

a1.pointers[accessIndex] = NULL;

}

cout << endl;

cout << "=== Options ===" << endl;

cout << "1. Print first 10 chars in the chosen array" << endl;

cout << "2. Delete all the chars associated with this pointer" << endl;

cout << "3. Return to Main Menu" << endl;

cin >> userInput2;

}

}

else if (userInput == 2)

{

cout << endl;

cout << "Listing all indexes with deallocated memory..." << endl;

// Output all deallocated memory

for (int i = 0; i < 20; i++)

{

if (a1.pointers[i] == NULL)

cout << i << " ";

}

cout << endl;

}

else if (userInput == 3)

{

cout << endl;

cout << "Deallocating all memory..." << endl;

// Deallocate all memory

for (int i = 0; i < 20; i++)

{

if (a1.pointers[i] != NULL)

{

delete a1.pointers[i];

a1.pointers[i] = NULL;

}

}

}

else

{

// Deallocate all memory & End program

for (int i = 0; i < 20; i++)

{

if (a1.pointers[i] != NULL)

{

delete a1.pointers[i];

a1.pointers[i] = NULL;

}

}

return 0;

}

cout << endl;

cout << "=== Main Menu ===" << endl;

cout << "1. Access a pointer" << endl;

cout << "2. List deallocated memory (index)" << endl;

cout << "3. Deallocate all memory" << endl;

cout << "4. Exit program" << endl;

cin >> userInput;

}

return 0;

}