CECS 326 Operating Systems

Ryan Hennes

Ryan.c.hennes@gmail.com

014603743

9/12/2018

Description:

This program is built to allocate memory based on the sizes of an integer array determined by this function, f(0)=2700 & f(n+1)=2\*f(n). Once the memory is allocated by the program it is then filled with random uppercase letters. The user of the program can access any of the twenty pointers in the program where they are able to either print the first 10 characters of the pointer or deallocate all the characters from this pointer. The user can also choose to deallocate all the memory from the pointers as well as list the index of the pointers who have been deallocated. In the even that a user accesses a pointer that has been deallocated the program will re allocate the memory and fill it with random characters once again. This index will no longer show up as a deallocated pointer.

#include <array>

#include <iostream>

#include<limits>

using namespace std;

//Struct required by the instructions

struct data

{

char \*charPointer[20];

unsigned int intArray[20];

}obj;

//Generates a random uppercase character then returns it to the function where it is stored into memory

char randChar()

{

char c = (90 - (rand() % 26));

return c;

}

//function used to fill all the character arrays with a random character based on the size dictated by the integer array

int fillArray()

{

//Console output letting the user know that the arrays are being filled.

cout << "Filling the arrays with random Upper case Characters," << endl;

cout << "This may take around 30 seconds." << endl;

//looping through the 20 character arrays to fill

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

{

//Looping through the size of the integer array and adding in the random character

for (int j = 0; j < obj.intArray[i]; j++)

{

char charToAssign = randChar();

\*(obj.charPointer[i]+j) = charToAssign;

}

}

return 0;

}

//Allocates memory based on the size of the integer array

void initCharArray()

{

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

{

obj.charPointer[i] = new char[obj.intArray[i]]();

}

}

//Fills the array using the function specified by the instructions

//n will always start at 1 due to setting the very first array index to 2700

int initArray(int n)

{

obj.intArray[0] = 2700;

if (n < 20)

{

//Setting the current index of the array to double the previous index

obj.intArray[n] = (obj.intArray[n - 1] \* 2);

initArray(n + 1);

}

return 0;

}

//Helper function to print out either the main menu or the sub menu

int printMenu(int a)

{

//Main menu

if (a == 1)

{

cout << "(1) Access a pointer" << endl;

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

cout << "(3) Deallocate all memory" << endl;

cout << "(4) Exit program" << endl;

}

//Sub Menu

if (a == 2)

{

cout << "(1) Print the first 10 char's in the chosen array" << endl;

cout << "(2) Delete all the char's associated with this pointer" << endl;

cout << "(3) Return to main menu" << endl;

}

return 0;

}

//Prints the first 10 characters of the array based on the index of the array that is passed in

void arrayPrint(int index)

{

cout << "Printing the first 10 characters of the array" << endl;

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

{

cout << \*(obj.charPointer[index]+i) << endl;

}

}

//Deletes the pointer at the index that is passed in

void deletePointer(int index)

{

//Made sure to delete the pointer first then set it to null thus preventing a memory leak

delete obj.charPointer[index];

obj.charPointer[index] = NULL;

}

//This is called from the access pointer function

//This handles what to do in the scenario that a user attempts to access

void validateNotNull(int index)

{

//In the scenario that there is a null pointer we know that the memory has been deallocated.

if(obj.charPointer[index] == NULL)

{

cout << "You have attempted to access a deallocated!" << endl;

cout << "Memory will now be allocated and the characters will be initialized." << endl;

obj.charPointer[index] = new char[obj.intArray[index]]();

cout << (obj.intArray[index]) << endl;

for (int j = 0; j < obj.intArray[index]; j++)

{

char charToAssign = randChar();

\*(obj.charPointer[index]+j) = charToAssign;

}

}

}

//Prompts the user for the index of the array to manipulate then displays a sub menu of options for that index

int accessPointer()

{

//collecting user input for the index

int index = 0;

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

cin >> index;

//Validating the input to prevent crashing

while (index < 0 || index > 19 || cin.fail())

{

cout << "That is not a valid input, please enter a number again: " << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> index;

}

//Ensuring that the pointer being access is not a deallocated pointer

validateNotNull(index);

//Collecting user input for the sub menu

int subMenuInput = 0;

printMenu(2);

cin >> subMenuInput;

//validating the input to prevent crashing

while (subMenuInput < 1 || subMenuInput > 3 || cin.fail())

{

cout << "That is not a valid input, please enter a number again: " << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> subMenuInput;

}

//These are the three options when manipulating the data at the index specified

switch (subMenuInput)

{

case 1: arrayPrint(index); break;

case 2: deletePointer(index); break;

case 3: cout << " Return to main menu" << endl; break;

}

return 0;

}

//Lists out all of the memory that has been deallocated

int listDeallocatedMemory()

{

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

{

if (obj.charPointer[i] == NULL)

{

cout << "Deallocated Index: " << i << endl;

}

}

return 0;

}

//physically deallocates memory from the system

int deallocateMemory()

{

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

{

deletePointer(i);

}

return 0;

}

//deallocates all memory and exits the program

int exitProgram()

{

deallocateMemory();

exit(0);

}

int main()

{

//Setting the srand time seed to be the system time ensuring a random sequence on each run

srand (time(0));

//Initializing the integer array by passing in 1 since the first integer will be initialized

initArray(1);

//Allocating the memory for the character array

initCharArray();

//Filling the array with random Characters

fillArray();

//the main is in a loop so that the main menu can be accessed multiple times

int userInput = 0;

while (true)

{

//printing the main menu and collecting input

printMenu(1);

cin >> userInput;

//validating user input

while (userInput < 1 || userInput > 4 || cin.fail())

{

cout << "That is not a valid input, please enter a number again: " << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> userInput;

}

switch (userInput)

{

case 1: accessPointer(); break;

case 2: listDeallocatedMemory(); break;

case 3: deallocateMemory(); break;

case 4: exitProgram();

}

}

return 0;

}