CECS 326 - Operating Systems

Assignment 1 - Programming Review

Name: Eric Do
Professor: Ratana Ngo
Date: February 2nd, 2019
Email: ericdo62497@gmail.com

Program Overview

In addition to helping us refresh our memory on the C++ language, this assignment helps us have a better understanding of memory management and memory allocation, which are some of the main tasks of Operating Systems.

In this implementation, we first initialized the structure of two arrays: an array of integers that represent the sizes of the char arrays, and the pointers to the char arrays. After initializing the structure, the user has four options: accessing a pointer, listing deallocated memory, deallocating all memory, and exiting the program.

Source Code

```
#include <iostream>
#include <string>
using namespace std;

void printMenu();
void printPointerMenu();
int validInput();
int initializeIntegers();
void initializeArrays(struct MyArrays &a);

const string letters = "ABCDEFGHJIKLMNOPQRSTUVXYZ";
const int arraySize = 20;
struct MyArrays
{
  char *ptrChar[arraySize];
  int intArray[arraySize];
};

int main()
{
  //Initialize variables and struct
  bool program = true;

  //Start Program
  cout << "~Assignment 1~" << endl;
  cout << endl;
  //Initialize Array Struct
  cout << "Initializing Arrays..." << endl;
  MyArrays a;
  initializeArrays(a);
  while (program)
  {
    printMenu();
    int userInput = validInput();

    if (userInput == 1)
    {
      //Prompts user for index of array
      int userArrayIndex = 0;
      bool pointerIsInRange = false;
      while (!pointerIsInRange)
      {
        cout << "Enter index (1-20): ";
```

```
        userArrayIndex = validInput();
        if (userArrayIndex >= 1 && userArrayIndex <= 20)
        {
          //Check if chars exist at this pointer
          if (a.ptrChar[userArrayIndex - 1] == NULL)
          {
            // Notify user of missing chars
            cout << "The index you chose is missing chars" << endl;
            cout << "Reallocating memory and reinitializing chars..." << endl;
            cout << endl;

            //Reallocate memory and reinitialize char
            a.ptrChar[userArrayIndex - 1] = new char[a.intArray[userArrayIndex
- 1]];
            for (int j = 0; j < a.intArray[userArrayIndex - 1]; j++)
            {
              a.ptrChar[userArrayIndex - 1][j] = letters[rand() % 26];
            }
          }
          //Menu to manipulate array
          pointerIsInRange = true;
          bool pointerMenuActive = true;
          while (pointerMenuActive)
          {
            printPointerMenu();
            int userPointerInput = validInput();
            if (userPointerInput == 1)
            {
              //Print the first 10 chars in the chosen array
              for (int i = 0; i < 10; i++)
              {
                cout << a.ptrChar[userArrayIndex - 1][i] << endl;
              }
            }
            else if (userPointerInput == 2)
            {
              //Delete all the chars associated with this pointer
              cout << "Deleting all chars at index: " << userArrayIndex <<
endl;
              delete a.ptrChar[userArrayIndex - 1];
              a.ptrChar[userArrayIndex - 1] = NULL;
              cout << "Going back to main menu..." << endl;
              pointerMenuActive = false;
            }
            else if (userPointerInput == 3)
            {
              //Return to main menu
              cout << "Returning to main menu" << endl;
```

```
          pointerMenuActive = false;
        }
        else
        {
          cout << "Input not in range. Try Again." << endl;
        }
        cout << endl;
      }
    }
    else
    {
      cout << "Index is not in range. Try Again." << endl;
    }
  }
}
else if (userInput == 2)
{
  //List deallocated memory
  cout << "Listing Indexes with Deallocated memory: " << endl;

  for (int i = 0; i < arraySize; i++)
  {
    if (a.ptrChar[i] == NULL)
    {
      cout << i + 1 << " ";
      cout << endl;
    }
  }
}
else if (userInput == 3)
{
  //Deallocate all memory
  cout << "Deallocating all memory..." << endl;
  for (int i = 0; i < arraySize; i++)
  {
    if (a.ptrChar[i] != NULL)
    {
      delete a.ptrChar[i];
      a.ptrChar[i] = NULL;
    }
  }
}
else if (userInput == 4)
{
  //Deallocate all memory and Exit
  cout << "Deallocating all memory..." << endl;
  for (int i = 0; i < arraySize; i++)
  {
```

```
        if (a.ptrChar[i] != NULL)
        {
          delete a.ptrChar[i];
          a.ptrChar[i] = NULL;
        }
      }
      cout << "Program will now exit" << endl;
      program = false;
    }
    else
    {
      cout << "Input not in range. Try Again." << endl;
    }
    cout << endl;
  }
  return 0;
}

/* Prints Main Menu */
void printMenu()
{
  cout << "Menu:" << endl;
  cout << " 1) Access a Pointer" << endl;
  cout << " 2) List deallocated memory" << endl;
  cout << " 3) Deallocate all memory" << endl;
  cout << " 4) Exit Program" << endl;
  cout << "Enter: ";
}

/* Prints Pointer Menu */
void printPointerMenu()
{
  cout << "Options:" << endl;
  cout << " 1) Print the 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;
  cout << "Enter: ";
}

/* Checks for valid user integer input */
int validInput()
{
  int num = 0;
  while (!(cin >> num))
  {
    cout << "Invalid Input. Try Again: ";
    cin.clear();
    cin.ignore(100, '\n');
```

```
  }
  return num;
}

/* Function to initialize the array of integers */
int initializeIntegers(int i)
{
  if (i == 0)
  {
    return 2700;
  }
  else
  {
    return initializeIntegers(i - 1) * 2;
  }
}

/* Function to initialize the Array Struct */
void initializeArrays(struct MyArrays &a)
{
  for (int i = 0; i < arraySize; i++)
  {
    a.intArray[i] = initializeIntegers(i);
    a.ptrChar[i] = new char[a.intArray[i]];
    for (int j = 0; j < a.intArray[i]; j++)
    {
      a.ptrChar[i][j] = letters[rand() % 26];
    }
  }
}
```