Brian Pham

February 13, 2018

CECS 326

brianthongpham@gmail.com

Software Description:

In this software, we were instructed to make a C++ program that contains two types of arrays: one array contains a list of twenty pointers pointing to a char variable and another array contains the number of char variables that will be displayed. The number of the char variable created is made by a recursive equation that was given:

$$f(0) = 2700$$
$$f(n) = 2 * f(n-1)$$

Setting up the program I used the ASCII to get me the char variables and added randomly into the array. Afterwards, I asked the user to select from a menu to either access a specific pointer, show a list of deallocated index of the pointer, deallocate all the memory, or exit the program. Inside the access of a specific pointer I requested the user to pick a specific pointers from a list of one through twenty. Once that is done I displayed a new menu telling them to either print the first ten characters, delete all char variables specific to that pointer, or return to the main menu.

If a allocated memory index is deleted and is recalled again, the program initializes that specific pointer or refills that pointer with random characters per basis

```
// Brian Pham
// main.cpp

#include <iostream>
#include <cstdlib>
#include <time.h>
using namespace std;

int recursiveFunction(int);
bool validateInput(string);

int main()
{
  // Setup variables
  int arraySize = 20;             //set array size to 20
  bool menuStatus = true;         //main menu ends or not
  bool subMenuStatus;             //sub menu ends or not
  bool pointerAccessMenu;         //inside the sub menu option ends or not
  string decision;                //user description
  int arrayInteger[arraySize];    //setting up the array for numbers of
  char* ptrChar[arraySize];       //setting up the character array
  int rand();                     //random
  srand (time(NULL));             //time set


  // setting up the arrays
  for (int i = 0; i < arraySize; i++)
  {
     arrayInteger[i] = recursiveFunction(i);
     ptrChar[i] = new char[arrayInteger[i]];
     for (int j = 0; j < arrayInteger[i]; j++)
     {
        ptrChar[i][j] = char(rand()%26 + 65);
     }
  }

  //start of the menu
  while (menuStatus)
  {
```

```
    string mainMenu = "(1) Access a pointer \n(2) List deallocated memory (index) \n(3) Deallocate all memory \n(4) Exit Program \nPlease enter your choice: ";
    cout << mainMenu;
    cin >> decision;
    cout << "\n";
    subMenuStatus = validateInput(decision);
    if (subMenuStatus)
    {
       int decisionValue = stoi(decision);
       switch (decisionValue)
       {
          case 1: //accessing specific pointers
          {
             while (subMenuStatus)
             {
                cout << "Which pointers to access?\n\t1\t2\t3\t4\t5\n\t6\t7\t8\t9\t10\n\t11\t12\t13\t14\t15\n\t16\t17\t18\t19\t20\nYour Input: ";
                cin >> decision;
                cout << "\n";
                if (validateInput(decision))
                {
                   decisionValue = stoi(decision);
                   if (decisionValue > 0 && decisionValue < 21)
                   {
                      if (ptrChar[decisionValue-1] != NULL)
                      {
                         pointerAccessMenu = true;
                         while (pointerAccessMenu)
                         {
                            cout << "What would you like to do with pointer " << decisionValue << "?\n\t1. Print the first 10 Char\n\t2. Delete all the Char in pointer and return to Main Menu\n\t3. Return to Main Menu\nYour Input: ";
                            cin >> decision;
                            cout << "\n";
                            if (validateInput(decision))
                            {
                               switch (stoi(decision))
                               {
                                  case 1: //prints out the first 10 chars of the pointer
```

```
                    {
                        cout << "First 10 chars at pointer " << decisionValue << " are: '";
                        for (int k=0; k < 9; k++)
                        {
                            cout << ptrChar[decisionValue-1][k] << " - ";
                        }
                        cout << ptrChar[decisionValue-1][10];
                        cout <<"'\n\n";
                        break;
                    }
                    case 2: //deletes all chars at the pointer and returns user to the main page
                    {
                        cout << "Deleting all char at pointer "<<decisionValue<<"\n..... Going
back to the main menu ....\n"<<endl;
                        ptrChar[decisionValue-1] = NULL;
                        pointerAccessMenu = false;
                        subMenuStatus = false;
                        break;
                    }
                    case 3: //goes back to the main page
                    {
                        cout << "\n";
                        pointerAccessMenu = false;
                        subMenuStatus = false;
                        break;
                    }
                    default: //for all other values
                    {
                        cout << "Invalid Input!!! Please enter 1 - 3"<<endl;
                        break;
                    }
                }
            }
            else
            {
                cout << "Invalid Input!! Please select a number, no other characters
allow!"<<endl;
            }
        }
```

```
            }
            else
            {
                cout << "Currently there is nothing in index value " << decisionValue << endl;
                cout << "Starting to refill all chars now ..... \n" << endl;
                ptrChar[decisionValue-1] = new char[arrayInteger[decisionValue-1]];
                for (int j = 0; j < arrayInteger[decisionValue-1]; j++)
                {
                    ptrChar[decisionValue-1][j] = char(rand()%26 + 97);
                }
            }
        }
        else
        {
            cout << "Invalid Input!!! Input must be a valid pointer number."<<endl;
        }
    }
    else
    {
        cout << "Invalid Input!!! Input must contain all number values."<<endl;
    }
  }
  break;
}
case 2: //printing a list of memories that has null as memory
{
  cout << "Deallocated Memory:";
  for (int i=0; i < arraySize; i++)
  {
    cout << "\n\t" << i+1 << ". " << &ptrChar[i];
  }
  cout << "\n\n";
  break;
}
case 3: //deleting all memory
{
  cout << "Deallocating All Memory......\n" << endl;
  for (int i=0; i < arraySize; i++)
  {
```

```
                ptrChar[i] = NULL;
            }
            break;
        }
        case 4: //ends the program
        {
            cout << "Exiting Program...." << endl;
            for (int i = 0; i < (sizeof(arrayInteger)/sizeof(arrayInteger[0])); ++i)
            {
                delete ptrChar[i];
            }
            menuStatus = false;
            break;
        }
        default: //for all other outputs
        {
            cout << "Invalid input!!! please choose one of the 4 options!" << endl;
            break;
        }

      }
    }
    else
    {
      cout << "Invalid input!!! Input must be a number!" << endl;
      menuStatus = true;
    }
  }
}


// using the given equation provided
int recursiveFunction(int i)
{
  if (i == 0)
  {
    return 2700;
  }
  else
```

```
  {
     return recursiveFunction(i-1) * 2;
  }
}

// Determines if the choice is a digit or letter
bool validateInput(string input)
{
  for (int i = 0; i < input.length(); i++)
  {
     char c = input[i];
     if (!isdigit(c))
     {
        return false;
     }
  }
  return true;
}
```