| Ousmane Toure & Jianning Chen  EECE2160 | Embedded Design: Enabling Robotics  Lab Assignment 5 |
| --- | --- |

Lab Assignment 5

Ousmane Toure & Jianning Chen

Toure.o@northeastern.edu

chen.jiann@northeastern.edu

Submit date: 6/6/2022

Due Date: 6/6/2022

**5.0 Code/Schematics**

**PRELAB CODE:**

#include <iostream>

#include <string>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

double \*v;

int count;

int size;

void Shrink();

void InsertElement();

void RemoveElement();

void PrintVector();

void Grow();

void Initialize();

void Finalize();

void Shrink() {

}

void InsertElement() {

}

void RemoveElement() {

}

void AddElement() {

}

void PrintVector() {

}

void Grow() {

}

void Initialize() {

size = 2;

v = new double[size];

v[0] = 0;

v[1] = 1;

}

void Finalize() {

delete[] v;

}

int main() {

Initialize();

int condition;

bool plays = true;

while(plays){

cout<<"Select an option: "<<endl;

std::cout << "1. Print the Array" << std::endl;

std::cout << "2. Append element at the end "<< std::endl;

std::cout << "3. Remove last element" << std::endl;

std::cout << "4. Insert one element" << std::endl;

std::cout << "5. Exit" << std::endl;

std::cin >> condition;

switch(condition){

case 1: {

cout<<"You selected \" Print the Array \""<< endl;

break;

}

case 2: {

cout<<"You selected \" Append element at the end \""<< endl;

break;

}

case 3: {

cout<<"You selected \" Remove last element \""<< endl;

break;

}

case 4: {

cout<<"You selected \" Insert one element \""<< endl;

break;

}

case 5: {

cout<<"You selected \" Exit \""<< endl;

plays = false;

break;

}

default:

cout<<"Please select a proper selection"<< endl;

}

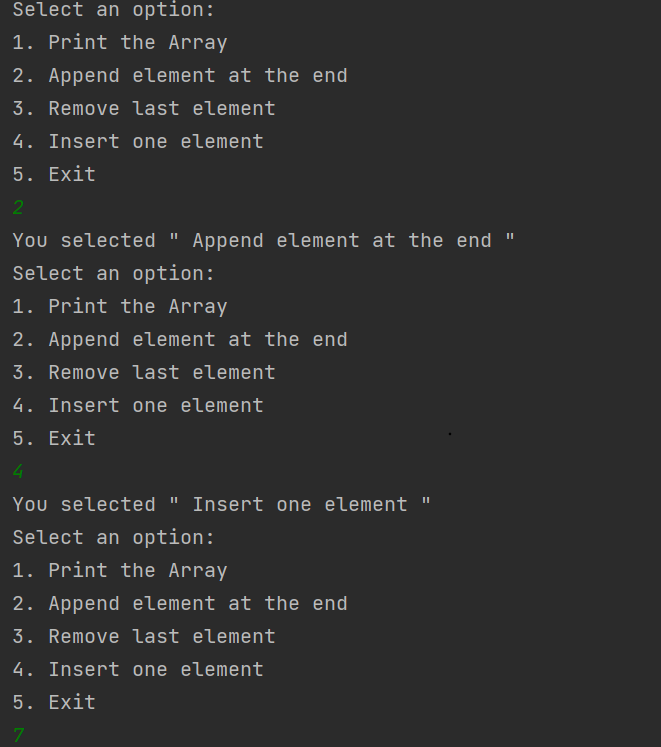
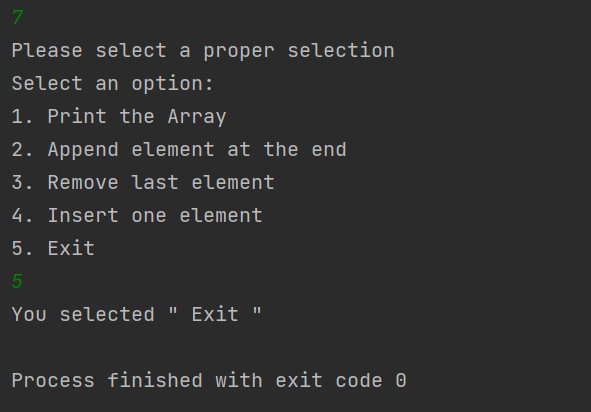
}

Finalize();

return 0;

}

**PreLab 5:**

****

**5.1 Section 1 of Lab 5.**

**void Grow(){**

**std::cout << std::endl;**

**std::cout << "The old array is: ";**

**PrintVector();**

**std::cout << std::endl;**

**prev\_size = size;**

**size \*= 2;**

**double\* new\_arr = new double[size] {0};**

**for(int i = 0; i < count; i++){new\_arr[i] = myarr[i];}**

**delete[] myarr;**

**myarr = new\_arr;**

**std::cout << "Vector Grown" << std::endl;**

**std::cout << "Previous Capacity: " << prev\_size << " elements" << std::endl;**

**std::cout << "New Capacity: " << size << " elements" << std::endl;**

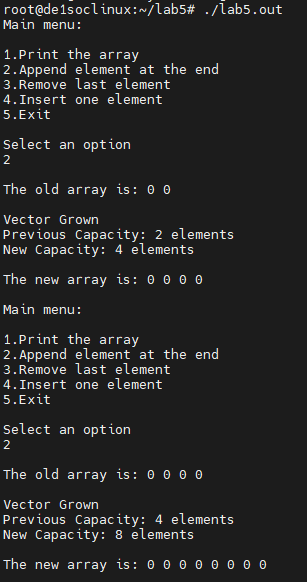
**std::cout << std::endl;**

**std::cout << "The new array is: ";**

**PrintVector();**

**std::cout << std::endl;**

**}**

****

Our code started by initializing some variables and a new pointer double the size of the previous array's size filled with zero. We then used a for loop to copy the contents of the original array to the new grown array. After which we delete the old array and then assign the name of the grown array to the original array, followed by increasing the size parameter, in this case, size is always times 2.

The above code won’t be in the final version of the code because all the cout statements are helper statements to make debugging more convenient. Therefore, in the final version of the code. prev\_size won’t be declared and all the cout statements will be removed.

**5.2 Section 2 of Lab 5.**

**void AddElement() {**

**int val;**

**cout<< "Enter your element:"<<endl;**

**cin>>val;**

**if(count == size){**

**Grow();**

**v[count] = val;**

**}**

**else{**

**v[count] = val;**

**}**

**++count;**

**}**

**void PrintVector() {**

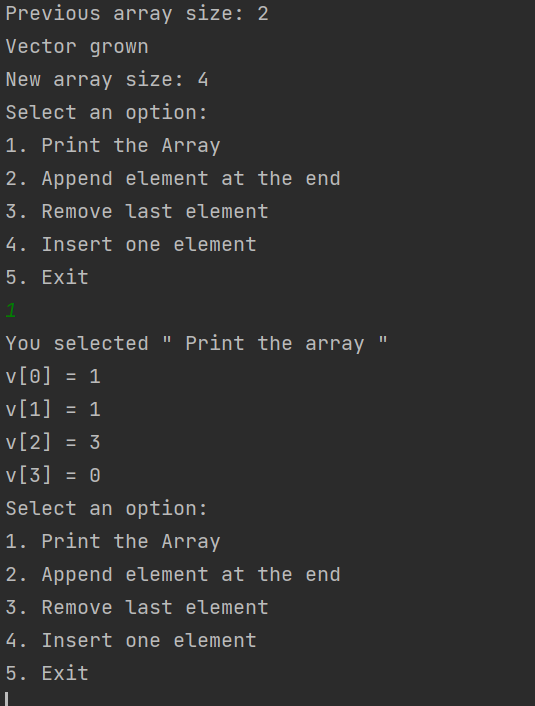
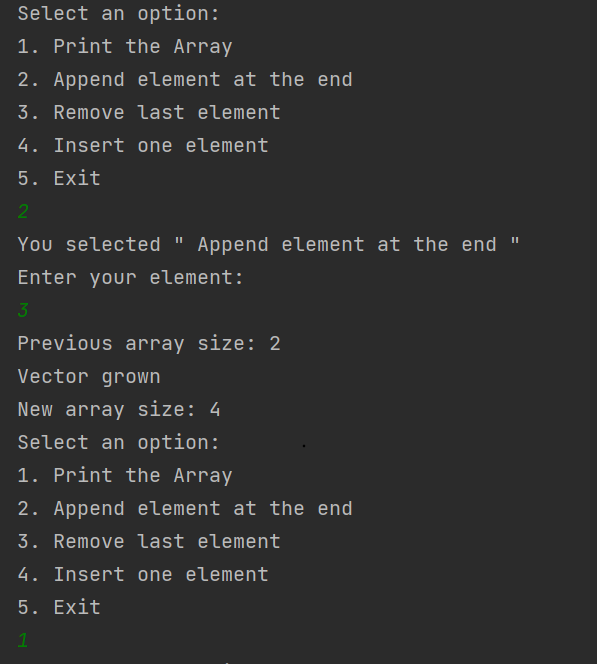
**int x;**

**for(x=0;x<size;x++){**

**cout.precision(0);**

**cout<<"v["<<x<<"] = "<<v[x]<<endl;**

**}**

**}**

Print was straight forward, we initialized a variable that would print out the value of our array at x based on is increasing every iteration until it reaches the size of the array.

Adding element first ensured that the array was big enough to allow another element to be added to it. Next it would assign the user value to into the next open segment of the array and increase the count.

**5.3 Section 3 of Lab 5.**

**void RemoveElement() {**

**v[count-1] = 0;**

**if((double(count-1) < double(size\*.3)) && (size > 2)){**

**Shrink();**

**}**

**else{**

**cout<<"There are no variables to delete."<<endl;**

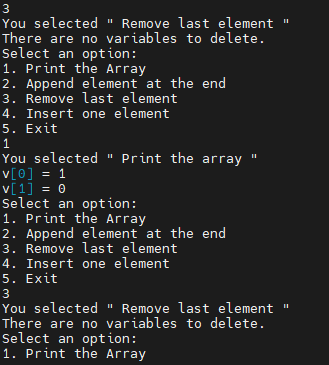
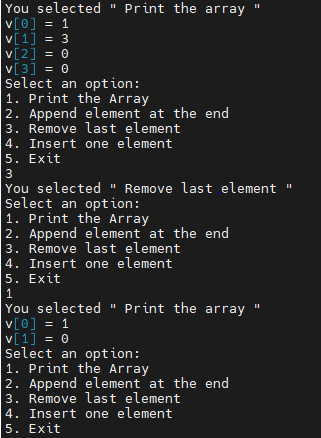
**}**

**if(count-1 >= 0){**

**--count;**

**}**

**}**



The remove function just ensured that the array count was greater than 0, and would assign 0 to the farthest element (allowed according to Note 2) else if it is zero, it would output that there are no variables to delete and decrease the count. after every iteration.

**5.4 Section 4 of Lab 5.**

Ousmane Toure’s Code on InsertElement():

**void InsertElement() {**

**int i,ele,ind;**

**cout<<"Enter Desired Index: "<<endl;**

**cin >>ind;**

**cout<<"Enter Element to Insert: "<<endl;**

**cin >>ele;**

**if(ind >= size){**

**Grow();**

**}**

**for(i=count;i>=ind;i--) {**

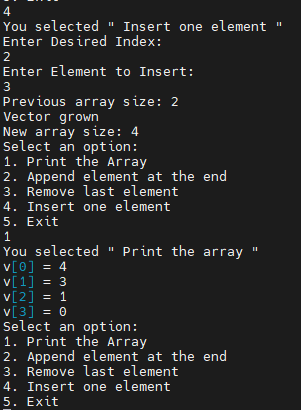
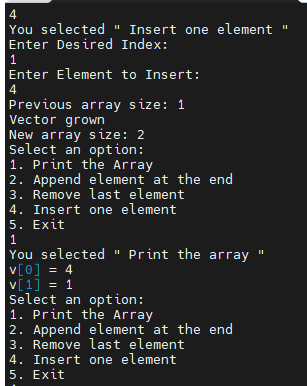
**v[i] = v[i-1];**

**}**

**v[i] = ele;**

**++count;**

**}**



Jianning Chen’s Code on InsertElement():

**void ArrayInsert(){**

**std::cout << "You selected \"Insert one element\"" << std::endl;**

**int index;**

**std::cout << "Enter the INDEX of the array you want to insert to: ";**

**std::cin >> index;**

**double num;**

**std::cout << "Enter the NUMBER you want to insert to the array: ";**

**std::cin >> num;**

**if(index < count){**

**if(count >= size) {size \*= 2;}**

**double \*new\_arr = new double[size]{0};**

**for (int i = 0; i < index; i++) {new\_arr[i] = myarr[i]; }**

**new\_arr[index] = num;**

**for (int i = index; i < count; i++) {new\_arr[i+1] = myarr[i]; }**

**delete[] myarr;**

**myarr = new\_arr;**

**}**

**else if(index > count){**

**if(count >= size){ArrayGrow();}**

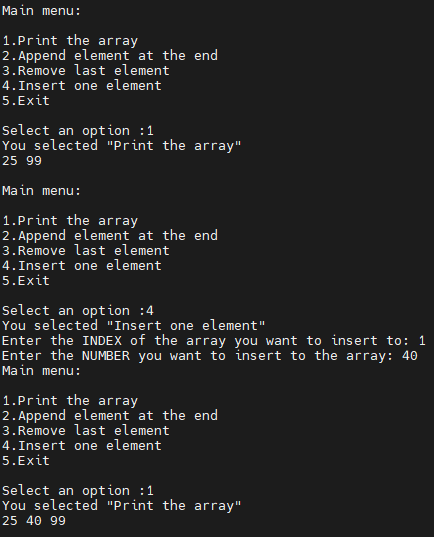
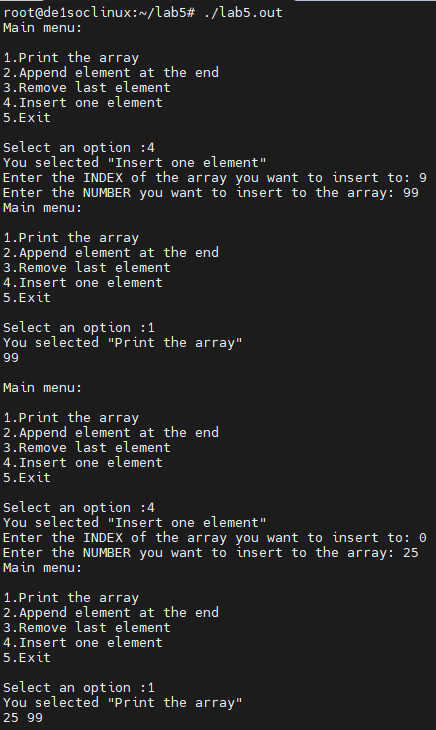
**myarr[count] = num;**

**}**

**else{myarr[index] = num;}**

**count += 1;**

**}**



The insert function handles multiple circumstances. In general, the algorithm of the insert can be split into 3 parts: when the desired index is smaller than count; when the desired index is bigger than count; when the desired index is equal to count. In each circumstance, the vector will only grow when the count is bigger or equal to the size.

When the

**5.5 Section 5 of Lab 5.**

void Shrink() {

size /= 2;

double \*nv = new double[size\*2]{0};

for(int i = 0;i<count;i++){nv[i] = v[i];}

delete[] v;

v = nv;

++count

The Shrink function has been added to the Remove function in 5.3. The Function doesn’t display or output any values but when Remove is run, it will cut the array's size in half. This is calculated in 5.3, that if the array is 30% empty, it will call Shrink.

Reference 5.3 for pictures

# References

If you use any additional texts, papers, websites, etc. and refer to them in the report, then you must include a reference. Note that copying text from other sources is typically considered plagiarism. If you verbatim copy text you will need to put “the copied text” in double quotes and cite the source. In case you are using main ideas from a different paper you need to cite. For example, most of this lab report guide is based by work of Michael Benjamin [1]. Also if you find useful sources, please tell me about them.

1. Prof. Julius Marpaung, “*Lab Report Guide*”, Northeastern University, January 6 2020.