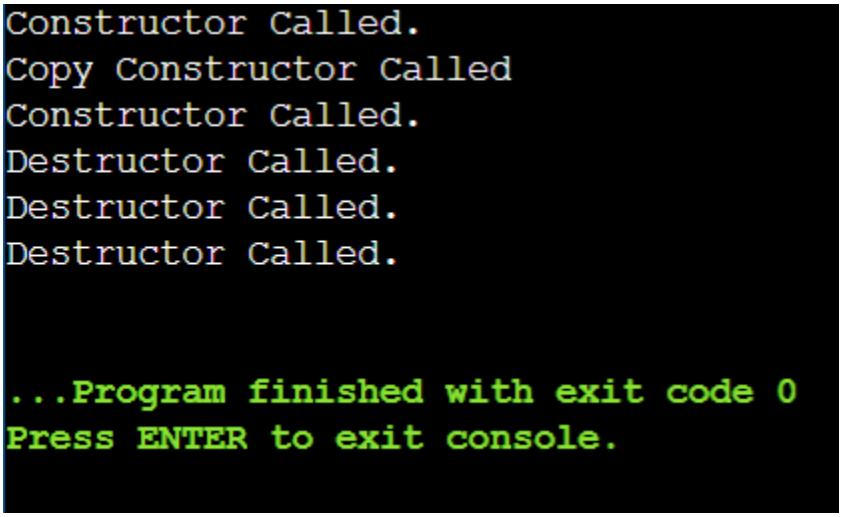
Activity No. <2>	
<Hands-on Activity 2.1 Arrays, Pointers and Dynamic Memory Allocation>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: Sept 11, 2024**
Section: CPE21S4	**Date Submitted: Sept 13, 2024**
Name(s): Kenn Jie Valleser	**Instructor: Ma'am Ma. Rizette Sayo**

6. Output

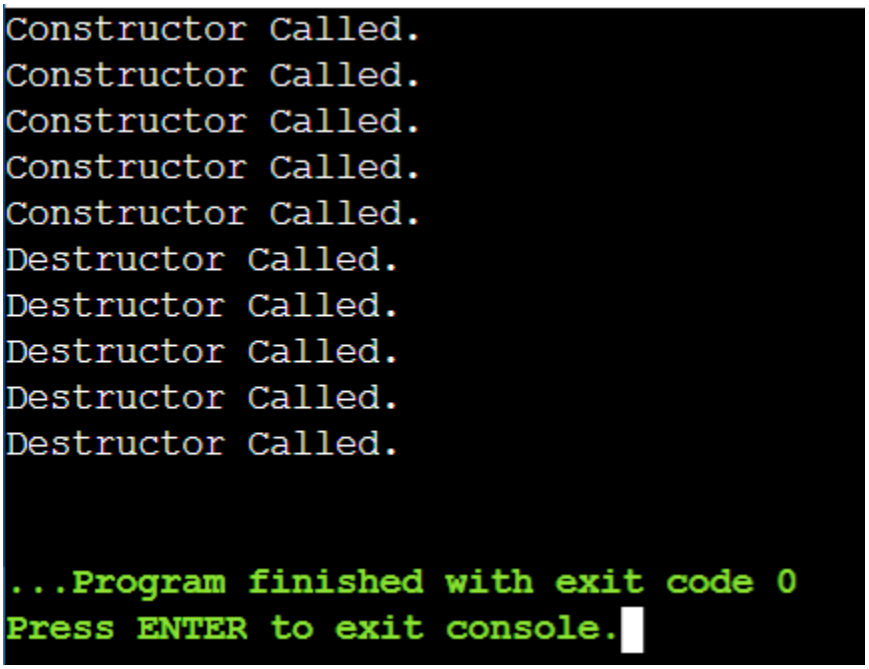
Screenshot



Observation it shows that the constructor was used 2 times and 1 for copy constructor and the destructor was called after everything was done

Table 2-1. Initial Driver Program

Screenshot



Observation The constructor was used 5 times because there were 5 values for name and age.

Table 2-2. Modified Driver Program with Student Lists

Loop A

```
33 int main() {
34 const size_t j = 5;
35
36 Student studentList[j] = {};
37 std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
38 int ageList[j] = {15, 16, 18, 19, 16};
39
40 for(int i = 0; i < j; i++){ //Loop A
41 Student *ptr = new Student(namesList[i], ageList[i]);
42 studentList[i] = *ptr;
43 }
44
45
46 return 0;
47 }
48
```

input

```
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

...Program finished with exit code 0
Press ENTER to exit console.
```

Observation the loop is used to create objects in the array of students

Loop B

```
33 int main() {
34 const size_t j = 5;
35
36 Student studentList[j] = {};
37 std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};
38 int ageList[j] = {15, 16, 18, 19, 16};
39
40
41 for(int i = 0; i < j; i++){ //Loop B
42 studentList[i].printDetails();
43 }
44
45 return 0;
46 }
```

input

```
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
John Doe 18
John Doe 18
John Doe 18
John Doe 18
John Doe 18
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.

...Program finished with exit code 0
Press ENTER to exit console.
```

Observation it will print the students list if the names and age are initialized

```
for(int i = 0; i < j; i++){ //loop A
Student *ptr = new Student(namesList[i], ageList[i]);
studentList[i] = *ptr;
}
for(int i = 0; i < j; i++){ //loop B
studentList[i].printDetails();
}

return 0;
}
```

input

```
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Constructor Called.
Carly 15
Freddy 16
Sam 18
Zack 19
Cody 16
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.
Destructor Called.


...Program finished with exit code 0
Press ENTER to exit console.
```

Output

Observation after putting the names and age in an array the students were printed out

Table 2-3. Final Driver Program

7. Supplementary Activity

```
#include <iostream>
#include <string>

using namespace std;

class Item {
public:
    virtual ~Item() {}
    virtual double calculateTotalCost() const = 0;
    virtual void printDetails() const = 0;
    virtual const string& getName() const = 0;
};

class Fruit : public Item {
private:
    string name;
    double price;
    int quantity;

public:
    // Constructor
    Fruit(const string& name, double price, int quantity)
        : name(name), price(price), quantity(quantity) {}

    // Calculate total cost
    double calculateTotalCost() const {
        return price * quantity;
    }

    // Print details
    void printDetails() const override {
        cout << "Fruit: " << name << ", Price: PHP " << price
             << ", Quantity: " << quantity << ", Total cost: PHP " << calculateTotalCost() << endl;
    }

    // Accessor for name
    const string& getName() const {
        return name;
    }
};

class Vegetable : public Item {
```

```
private:
    string name;
    double price;
    int quantity;

public:
    // Constructor
    Vegetable(const string& name, double price, int quantity)
        : name(name), price(price), quantity(quantity) {}

    // Calculate total cost
    double calculateTotalCost() const {
        return price * quantity;
    }

    // Print details
    void printDetails() const override {
        cout << "Vegetable: " << name << ", Price: PHP " << price
             << ", Quantity: " << quantity << ", Total cost: PHP " << calculateTotalCost() << endl;
    }

    // Accessor for name
    const string& getName() const  {
        return name;
    }
};


// Function to calculate total sum of all items
double TotalSum(Item* groceryList[], int numItems) {
    double total = 0.0;
    for (int i = 0; i < numItems; ++i) {
        total += groceryList[i]->calculateTotalCost();
    }
    return total;
}

int main() {
    // Create instances of Fruit and Vegetable
    Item* apple = new Fruit("Apple", 10, 7);
    Item* banana = new Fruit("Banana", 10, 8);
    Item* broccoli = new Vegetable("Broccoli", 60, 12);
    Item* lettuce = new Vegetable("Lettuce", 50, 10);

    // Create an array of pointers to Item (Fruit and Vegetable)
    const int maxItems = 4;
    int numItems = 4;
    Item* groceryList[maxItems] = {apple, banana, broccoli, lettuce};

    // Display all details
    for (int i = 0; i < numItems; ++i) {
```

```
        groceryList[i]->printDetails();
    }

    // Calculate and display total cost
    double totalCost = TotalSum(groceryList, numItems);
    cout << "Total cost of groceries: PHP " << totalCost << endl;
    for (int i = 0; i < numItems; ++i) {
        if (groceryList[i]->getName() == "Broccoli") {
            // Delete the item
            delete groceryList[i];

            // Shift remaining items left to fill the gap
            for (int j = i; j < numItems - 1; ++j) {
                groceryList[j] = groceryList[j + 1];
            }
            groceryList[numItems - 1] = nullptr; // Optional: Set last item to nullptr
            --numItems; // Update the number of items
            break; // Exit loop after removing the item
        }
    }

    // Display details after removing Broccoli
    cout << "\nGrocery list after removing Broccoli:" << endl;
    for (int i = 0; i < numItems; ++i) {
        groceryList[i]->printDetails();
    }

    // Calculate and display total cost after removal
    totalCost = TotalSum(groceryList, numItems);
    cout << "Total cost of groceries: PHP " << totalCost << endl;

    // Clean up remaining items
    for (int i = 0; i < numItems; ++i) {
        delete groceryList[i];
    }

    return 0;
}
```

```
Fruit: Apple, Price: PHP 10, Quantity: 7, Total cost: PHP 70
Fruit: Banana, Price: PHP 10, Quantity: 8, Total cost: PHP 80
Vegetable: Broccoli, Price: PHP 60, Quantity: 12, Total cost: PHP 720
Vegetable: Lettuce, Price: PHP 50, Quantity: 10, Total cost: PHP 500
Total cost of groceries: PHP 1370

Grocery list after removing Broccoli:
Fruit: Apple, Price: PHP 10, Quantity: 7, Total cost: PHP 70
Fruit: Banana, Price: PHP 10, Quantity: 8, Total cost: PHP 80
Vegetable: Lettuce, Price: PHP 50, Quantity: 10, Total cost: PHP 500
Total cost of groceries: PHP 650


=== Code Execution Successful ===
```

8. Conclusion

- I learned that Constructor can be used to initialize objects, and destructor are mostly used to delete data in the memory
- I was able to understand the procedure and what constructor and destructor are used in a class.
- This supplementary Activity was hard since it has a lot of code to do and problems to fix
- I think I did okay since I was able to learn something and understand the codes. I should try practicing in creating copy constructors and copy assignment operators.

9. Assessment Rubric

.