| 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:** 09/11/24 |
| **Section:** CPE21S4 | **Date Submitted:** 09/12/24 |
| **Name(s):** Rodriguez, Kyle Franz P. | **Instructor:** Maria Rizette Sayo |

**6. Output**

Table 2-1. Initial Driver Program

Screenshot

main.cpp

```cpp
#include <iostream>
#include <string.h>

class Student {
    private:
    std::string studentName;
    int studentAge;

    public:
    //constructor
    Student (std::string newName = "John Doe", int newAge = 18) {
        studentName = std::move(newName);
        std::cout<< "Constructor Called." << std::endl;
    };

    //deconstructor
    ~Student() {
        std::cout<< "Deconstructor Called." << std::endl;
    }

    //Copy constructor
    Student(const Student &copyStudent) {
        std::cout <<"Copy Constructor Called." << std::endl;
        studentName = copyStudent.studentName;
        studentAge = copyStudent.studentAge;
    }

    //Display Attributes
    void printDetails() {
        std::cout<< this->studentName << " " << this->studentAge << std::endl;
    }
};

int main() {
    Student student1("Roman", 28);
    Student student2(student1);
    Student student3;
    student3 = student2;
return 0;
```

Output

```
/tmp/ZHvcyHZ6U3.o
Constructor Called.
Copy Constructor Called.
Constructor Called.
Deconstructor Called.
Deconstructor Called.
Deconstructor Called.


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

Observation

In what I can see, placing a new student name and age will always call the constructor in the class. This is what happened in the line “Student student1(“Roman”,28), as the string Roman and the integer 28 was constructed as the student1, 2 “Constructor Called” was printed. Next, the “Copy constructor called” output was printed because of the next line in the main driver, “Student student2(student1). This is because the constructed name and age for the student1 was copied in student2’s place. In the line located in the deconstructor function, a deconstructor (~) was in place with “Student()”. This is the reason why the line “Student student3” in the main driver caused a “deconstructor called” output.

| Table 2-2. Modified Driver Program with Student Lists | |
|---|---|
| Screenshot | main.cpp<br>Share Run<br>Output<br><br>```cpp<br>#include <iostream><br>#include <string.h><br><br>class Student {<br>private:<br>std::string studentName;<br>int studentAge;<br><br>public:<br>//constructor<br>Student (std::string newName = "John Doe", int newAge = 18) {<br>studentName = std::move(newName);<br>std::cout<< "Constructor Called." << std::endl;<br>};<br><br>//deconstructor<br>~Student() {<br>std::cout<< "Deconstructor Called." << std::endl;<br>}<br><br>//Copy constructor<br>Student(const Student &copyStudent) {<br>std::cout <<"Copy Constructor Called." << std::endl;<br>studentName = copyStudent.studentName;<br>studentAge = copyStudent.studentAge;<br>}<br><br>//Display Attributes<br>void printDetails() {<br>std::cout<< this->studentName << " " << this->studentAge << std::endl;<br>}<br>};<br><br>int main() {<br>const size_t j = 5;<br>Student studentList[j] = {};<br>std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};<br>int ageList[j] = {15, 16, 18, 19, 16};<br>return 0;<br>}<br>```<br><br>```<br>/tmp/kBIY2AYhYP.o<br>Constructor Called.<br>Constructor Called.<br>Constructor Called.<br>Constructor Called.<br>Constructor Called.<br>Deconstructor Called.<br>Deconstructor Called.<br>Deconstructor Called.<br>Deconstructor Called.<br>Deconstructor Called.<br><br>=== Code Execution Successful ===<br>```<br><br>Share<br>Run<br>Output |
| Observation | In this case, an array was given in the driver. The output has 5 “Constructor called” as the array of studentList contains five strings for the name of the student. I believe the 5 “deconstructor called” was because of the ageList array that contains an integer for the student age. This is because the strings allocated in the memory were deleted so that there will be room for the new list. |

| Table 2-3. Final Driver Program | |
|---|---|
| Loop A | C/C++<br>```cpp<br>for(int i = 0; i < j; i++){ //loop A<br>Student *ptr = new Student(namesList[i], ageList[i]);<br>studentList[i] = *ptr;<br>}<br>```<br>C/C++ |
| Observation | Based on what I know, this loop is the one that fills the list “StudentList” with the previous lists namely “namesList” and “ageList”. The reason this is necessary is because the StudentList will be directly constructed as the class intended. |
| Loop B | |

| | |
|---|---|
| | C/C++<br>`for(int i = 0; i < j; i++){ //loop B`<br>`studentList[i].printDetails();`<br>`}` |
| Observation | Based on what I see and what I know, this loop functions as an output printer. It means that for every i in the list, the program will print each value. |
| Output | main.cpp<br>`#include <iostream>`<br>`#include <string.h>`<br><br>`class Student {`<br>`private:`<br>`std::string studentName;`<br>`int studentAge;`<br><br>`public:`<br>`//constructor`<br>`Student (std::string newName = "John Doe", int newAge = 18) {`<br>`studentName = std::move(newName);`<br>`std::cout<< "Constructor Called." << std::endl;`<br>`};`<br><br>`//deconstructor`<br>`~Student() {`<br>`std::cout<< "Deconstructor Called." << std::endl;`<br>`}`<br><br>`//Copy constructor`<br>`Student(const Student &copyStudent) {`<br>`std::cout <<"Copy Constructor Called." << std::endl;`<br>`studentName = copyStudent.studentName;`<br>`studentAge = copyStudent.studentAge;`<br>`}`<br><br>`//Display Attributes`<br>`void printDetails() {`<br>`std::cout<< this->studentName << " " << this->studentAge << std::endl;`<br>`}`<br>`};`<br><br>`int main() {`<br>`const size_t j = 5;`<br>`Student studentList[j] = {};`<br>`std::string namesList[j] = {"Carly", "Freddy", "Sam", "Zack", "Cody"};`<br>`int ageList[j] = {15, 16, 18, 19, 16};`<br>`for(int i = 0; i < j; i++){ //loop A`<br>`Student *ptr = new Student(namesList[i], ageList[i]);`<br>`studentList[i] = *ptr;`<br>`}`<br>`for(int i = 0; i < j; i++){ //loop B`<br>`studentList[i].printDetails();`<br>`}`<br>`return 0;`<br>`}`<br><br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Constructor Called.`<br>`Carly 15`<br>`Freddy 16`<br>`Sam 18`<br>`Zack 19`<br>`Cody 16`<br>`Destructor Called.`<br>`Destructor Called.`<br>`Destructor Called.`<br>`Destructor Called.`<br>`Destructor Called.`<br><br>`---------- Process exited after 6`<br>`Press any key to conti` |
| Observation | Overall, the output presents 10 “Constructors called” because of the number of names and ages that was constructed from the list created via the loop 1. Then the output in the middle part is due to loop 2. And as usual, before the program ends, any object allocated will be automatically deallocated. |

| Table 2-4. Modifications/Corrections Necessary | |
|---|---|
| Modification | NONE |
| Observation | I did no modification as I think that this is enough for the code to do its job efficiently. Maybe there are many ways I can modify this to work better, but that is beyond my knowledge for now. However, my instinct tells me that there is a better way to delete the object in a memory if it isn’t needed anymore. |

## 7. Supplementary Activity

Problem 1

```C/C++
#include <iostream>
#include <string>

class GroceryItem {

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

public:
    //constructor
       GroceryItem(std::string name, double price, int quantity) {
        this->name = name;
        this->price = price;
        this->quantity = quantity;
    }

       //Deconstructor
    ~GroceryItem() {
        std::cout << "Deleting " << name << std::endl;
    }

       //Copy constructor
    GroceryItem(const GroceryItem& other) {
       this->name = other.name;
       this->price = other.price;
       this->quantity = other.quantity;
       }

       //Copy assignment
    GroceryItem& operator=(const GroceryItem& other) {
        if (this != &other) {
            name = other.name;
            price = other.price;
            quantity = other.quantity;
        }
        return *this;
    }

    double calculateSum() const {
        return price * quantity;
    }

    std::string getName() const {
        return name;
    }

    double getPrice() const {
        return price;
    }
```

```
    int getQuantity() const {
        return quantity;
    }
};

class Fruit : public GroceryItem {
public:
    Fruit(const std::string& name, double price, int quantity) :
GroceryItem(name, price, quantity) {}
};

class Vegetable : public GroceryItem {
public:
    Vegetable(const std::string& name, double price, int quantity) :
GroceryItem(name, price, quantity) {}
};
```

Problem 2

```
C/C++
int main() {
       //Making GroceryList
    GroceryItem* groceryList[] = {
        new Fruit("Apple", 10.0, 7),
        new Fruit("Banana", 10.0, 8),
        new Vegetable("Broccoli", 60.0, 12),
        new Vegetable("Lettuce", 50.0, 10)
    };
```

Problem 3

```
C/C++
double TotalSum(GroceryItem** groceryList, int size) {
    double total = 0.0;
    for (int i = 0; i < size; ++i) {
        total += groceryList[i]->calculateSum();
    }
    return total;
}
```

Problem 4

```
C/C++
    int groceryListSize = sizeof(groceryList) / sizeof(groceryList[0]);

    // Display items
```

```
    for (int i = 0; i < groceryListSize; ++i) {
        std::cout << groceryList[i]->getName() << ": PHP " <<
groceryList[i]->getPrice() << " x " << groceryList[i]->getQuantity() << " =
PHP " << groceryList[i]->calculateSum() << std::endl;
    }

    // Calculate total sum
    double totalSum = TotalSum(groceryList, groceryListSize);
    std::cout << "Total sum: PHP " << totalSum << std::endl;

    // Delete Lettuce
    delete groceryList[3];

    return 0;
}
```

Output

```
Apple: PHP 10 x 7 = PHP 70
Banana: PHP 10 x 8 = PHP 80
Broccoli: PHP 60 x 12 = PHP 720
Lettuce: PHP 50 x 10 = PHP 500
Total sum: PHP 1370
Deleting Lettuce

--------------------------------
Process exited after 0.103 seconds with return value 0
Press any key to continue . . .
```

**8. Conclusion**

In this topic, I relearned about using classes in C++, and learned new things on memory allocation. I noticed that this is once again similar to how we currently code in object-oriented programming, so I am mostly familiar with how the system works.

In the procedure, in table 2.1, it simply showed what classes could do. In table 2.2, it also showed the same on the surface, however the difference is that instead of using the “John Doe” assigned within the class, the driver instead used its own array. In table 2.3, this one showed a much better usage of class using the loops, and how we can utilize pointers to make things appear easier. Lastly, in table 2.4, I showed no modification, because I do not see any problem at all. I probably will in the future, but for now, I am still learning a lot.

In the supplementary activity, this is where I had a difficult time. Based on the instructions, It was stated to make a class for vegetables and fruit, however I noticed that it is redundant if I do that, so I figured I could make a class that I will use as a parent class. I can use this parent class so that the classes “Vegetable” and “Fruit” can inherit its command, making things a little less redundant. Number 2, I didn’t directly make the array “GroceryList” like a normal array because I think it poses more work if I do that. So instead I just utilized a pointer and made the array directly with the class function and made it create an object. I presented the output that describes the details of the item. The details I went for are price and quantity. For problem 3, I created a function that computes the total, as instructed. Here I used a pointer of a pointer of “GroceryList” and used a for loop in the body. I made it so that the program counts each value one by one. Lastly, for problem 4, I managed to delete the lettuce item with the use of delete operator.

In summary, I think I did just fine with this activity, considering the extensions and all. I probably have to relearn some of these syntax as I was just recently familiar with some of it. Especially with the double pointer and the copy assignment in the class. Overall, I had fun.

| 9. Assessment Rubric |
| --- |
| |