Data Structures Lab - 02

Muhammad Irfan Ayub / Mubashra Fayyaz

Topics:

- Debugging
- OOP Programming
    - Using Header Files
    - Classes and Objects
    - Constructors
    - Destructors
- Dynamic Memory
- Rule of 3
- Exercises

→ Debugging

Using the debugger:

- The various features of the debugger are pretty obvious. Click the "Debug" icon to run your program and pause at the current source code cursor location; Click "Next Line" to step through the code; Click "Add Watch" to monitor variables.
- Setting breakpoints is as easy as clicking in the blank space (Line Number) next to the line in the source code.

→ OOP Programming

Using **Header** Files

Header files are used for declaration. In OOP you should use header files to declare classes and functions. It will make your program look more clean and professional.

Header file for a class will include:

- Include guards.
- Class definition.
    - Member variables
    - Function declarations (only prototype)

Implementation File will include:

- Include directive for “header.h”
- Necessary include directives.
- Function definitions for all the functions of the class.

Example:

```
// my_class.h
#ifndef MY_CLASS_H // include guard
#define MY_CLASS_H

namespace N
{
      class my_class
      {
      public:
      void do_something();
      };
}
```

```
// my_class.cpp
#include "my_class.h" // header in
local directory
#include <iostream> // header in
standard library

using namespace N;
using namespace std;

void my_class::do_something()
{
      cout << "Doing something!" <<
endl;
}
```

```
// my_program.cpp
#include "my_class.h"

using namespace N;

int main()
{
      my_class mc;
      mc.do_something();
      return 0;
}
```

Constructors and Destructors

- Default Constructor.
- Parameterized Constructor.
- Initializer List.
- Destructors.

➔ Dynamic Memory

C++ supports three types of memory allocation.

- ➔ **Static memory allocation** happens for static and global variables. Memory for these types of variables is allocated once when your program is run and persists throughout the life of your program.
- ➔ **Automatic memory allocation** happens for function parameters and local variables. Memory for these types of variables is allocated when the relevant block is entered, and freed when the block is exited, as many times as necessary.
- ➔ **Dynamic memory allocation** is a way for running programs to request memory from the operating system when needed.

new Operator

- This operator is used to allocate a memory of a particular type.
- This creates an object using the memory and **returns a pointer** containing the memory address.
- The return value is mostly stored in a **pointer** variable.

```
// new_op.cpp
int main()
{
      int *ptr = new int; // allocate memory
      *ptr = 7; // assign value

      // allocated memory and assign value
      int *ptr2 = new int(5);
}
```

delete Operator

- When we allocate memory dynamically, we need to explicitly tell C++ to deallocate this memory.
- **delete** Operator is used to release / deallocate the memory.

```
// delete_op.cpp

#include <iostream>
int main()
{
    int *ptr = new int; // dynamically allocate an integer
    int *otherPtr = ptr; // otherPtr is now pointed at that same memory
    delete ptr; // ptr and otherPtr are now dangling pointers.
    ptr = 0; // ptr is now a nullptr

    // however, otherPtr is still a dangling pointer!
     return 0; }
```

→ Dynamic Arrays

To allocate an array dynamically we use array form of **new** and **delete**
(new[ ] , delete[ ])

```
dynamic_array.cpp

// dynamic_array.cpp

#include<iostream>
```

```
using namespace std;

int main()
{
    int array[] = {1,2,3};
    cout << array[0];
    cout << endl;

//    int* dArray = new int[] {1,2,3};
    int* dArray = new int[3] {1,2,3};
    cout << *dArray+1;
    cout << endl;
    cout << dArray[2];

    delete[] dArray;
}
```

→ Rule of Three:

If you need to explicitly declare either the **destructor**, **copy constructor** or **copy assignment operator** yourself, you probably need to explicitly declare all three of them.

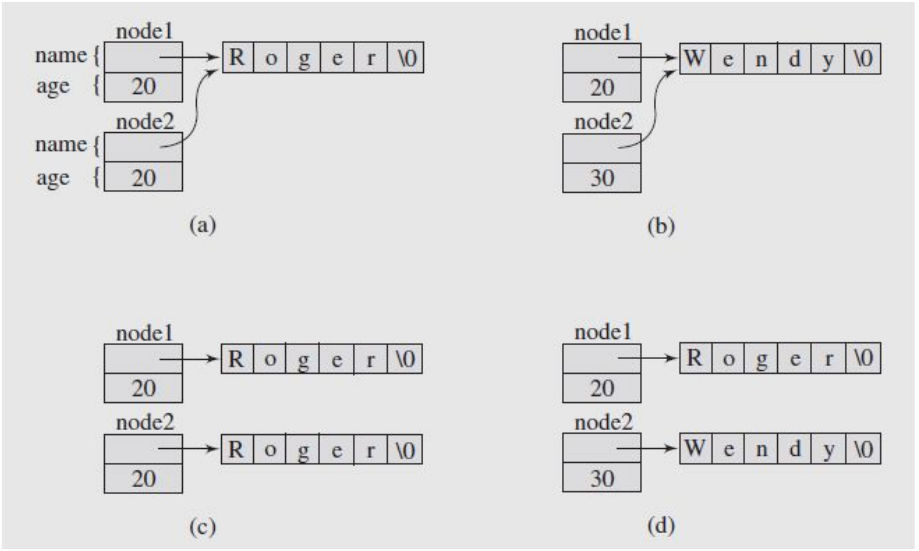
```
struct Node {
     char *name;
     int age;
     Node(char *n = "", int a = 0) {
     name = strdup(n);
     age = a; }
};

Node node1("Roger",20), node2(node1);
strcpy(node2.name,"Wendy");
node2.age = 30;

cout<<node1.name<<' '<<node1.age<<'
'<<node2.name<<' '<<node2.age;
```

```
Node(char *n = 0, int a = 0) {
name = strdup(n);
age = a;
}
Node(const Node& n) { // copy
constructor;
name = strdup(n.name);
age = n.age;
}
```

```
Node& operator=(const Node& n) {
if (this != &n) { // no assignment
to itself;
if (name != 0)
free(name);
name = strdup(n.name);
age = n.age;
}
return *this;
}
```


node1
name
age
20
R o g e r \0
node2
name
age
20
(a)
node1
20
W e n d y \0
node2
30
(b)
node1
20
R o g e r \0
node2
20
R o g e r \0
(c)
node1
20
R o g e r \0
node2
30
W e n d y \0
(d)

Exercises:

1. Create a Program to solve Quadratic Equation and Calculate the roots.
    a. Your program must implement a class Quadratic Equation.
    b. You must use header files for class implementation.
2. Create a program containing information for a **Student**.
    a. A Student can have the following information
        i. ID
        ii. Batch
        iii. Discipline
        iv. Expected Graduation Year
        v. Current Courses (this can be an array of strings)
    b. Use Dynamic Safe Arrays to store the information of multiple students.
3. Using Dynamic Array implement a Grading system for students.
    a. 1st dimension will have the section ID (0 - A, 1 - B etc)
    b. 2nd dimension will have the number of Grades in the section.
        i. 0 - A (value will be the number of A grades).
        ii. 1 - B
        etc.
4. Implement a class for a **Car**. Implement Rule of Three for this class.
    a. A car can have properties of another car. (same category cars).
        i. Using copy constructor.
        ii. Using Assignment Operator.
    b. A car object should be destroyed properly.
        i. Using destructor.

References:

Header Files
https://docs.microsoft.com/en-us/cpp/cpp/header-files-cpp?view=vs-2019
https://www.sitesbay.com/cpp/cpp-header-files

Namespaces
https://docs.microsoft.com/en-us/cpp/cpp/namespaces-cpp?view=vs-2019

Dynamic Memory
https://www.learncpp.com/cpp-tutorial/69-dynamic-memory-allocation-with-new-and-delete/
http://www.cplusplus.com/doc/tutorial/dynamic/
https://stackoverflow.com/questions/12714199/null-pointer-vs-dangling-pointer

Dynamic Arrays
https://www.learncpp.com/cpp-tutorial/6-9a-dynamically-allocating-arrays/

Safe Arrays
https://www.geeksforgeeks.org/overloading-subscript-or-array-index-operator-in-c/

Jagged Array
https://www.experts-exchange.com/questions/28290453/C-jagged-array-array-of-pointers.html

Rule of Three
https://stackoverflow.com/questions/4172722/what-is-the-rule-of-three