# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BESE 1****3AB**

**Lab 2:****Dynamic Memory Allocation**

**Name:** Aimen Munawar

**Class:** BESE-13-A

**CMS Id:**415867

**Date: 22nd September, 2023**

**Time: 10:00 am – 12:50 am & 2:00 pm – 5:00pm**

# Lab Engineer: Anum Asif

# Lab 2: Dynamic Memory Allocation

**Introduction**

This lab is about dynamic memory allocation. Dynamic memory allocation in C/C++ refers to performing memory allocation manually by programmer. Dynamically allocated memory is allocated on **Heap** and non-static and local variables get memory allocated on **Stack**

**Objectives**

This lab will revise the old concepts taught to the students in the previous semesters.

**Tools/Software Requirement**

Visual Studio C++

**Description : Static vs Dynamic Arrays**

Consider two variants of declaring arrays below. Memory for the first variant gets allocated on the Stack. The lifetime of an array created using the method A depends on its scope. If it is defined globally, its life is equal to the lifetime of the application. If it is declared in a function, memory for it gets allocated on the stack when the function gets called. It gets deallocated when the function call terminates. All the data related to the function call including the array gets removed from the stack. On the other hand, memory for the array created using new operator gets allocated on the heap at runtime. The lifetime of such an array is at max equal to the execution time of the application. If the array is no more required, the memory allocated for it can be freed using **delete []** command.

**Method A:**

const int size=5;

int x[size];

for (int i = 0; i < size; i++)

{

//cout << "x[" << i << "] = ";

x[i] = i + 1;

}

**Method B**

int size; // Note that size variable is const in variant A whereas it isn’t in variant B. Find out the logic behind it.

cout << "Enter size of array: ";

cin >> size;

int \*x = new int[size];

for (int i = 0; i < size; i++)

{

//cout << "x[" << i << "] = ";

x[i] = i + 1;

}

**Lab Tasks**

**Task 1**

Rewrite this program using pointers in place of arrays (use dynamic memory allocation operators new and delete). The syntax is int \*salArray= new int[size]; and for deletion delete [] salArray.

include<iostream>

using namespace std;

int main(void)

{

int salary[20];

inti;

for (i=0;i<20;++i)

{

cout<<"Enter Salary: ";

cin>>salary[i];

}

for (i=0;i<20;++i)

salary[i]=salary[i]+salary[i]/(i+1);

return 0;

}

**Code:**

// Task 1:Rewrite this program using pointers in place of arrays (use dynamic memory allocation operators new and delete). The syntax is int \*salArray= new int[size]; and for deletion delete [] salArray.

#include<iostream>

using namespace std;

int main(void)

{

int size,i;

cout<<"Enter the size of array:\n";

cin>>size;

int \*salArray= new int[size]; // Dynamic Memory Allocation

for (i=0;i<size;++i)

{

cout<<"Enter Salary:\n";

cin>>salArray[i];

}

cout<<"New salaries are:";

for (i=0;i<size;++i){

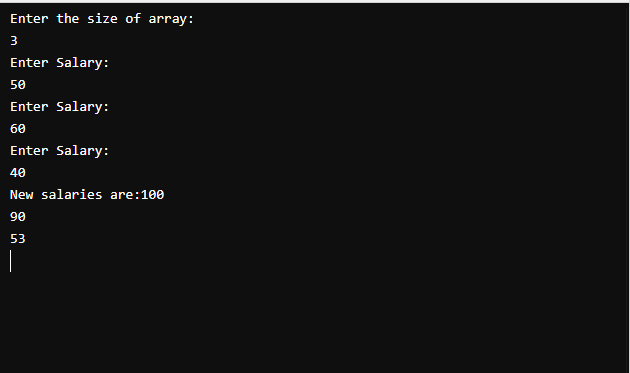
salArray[i]=salArray[i]+salArray[i]/(i+1);

cout<<salArray[i]<<endl;

}

return 0;

}



**Task 2**

Complete the two parts for analyze pointer problem.

**Part 1**: Write a function void analyze\_pointer(int \*ptr) that does two things:

* Write the memory location pointed by the pointer to the console.
* Write the value of the integer (which the pointer points to) to the console.

**Code:**

/\* Task 2: Complete the two parts for analyze pointer problem.

Part 1: Write a function void analyze\_pointer(int \*ptr) that does two things:

 Write the memory location pointed by the pointer to the console.

 Write the value of the integer (which the pointer points to) to the console.

\*/

#include<iostream>

using namespace std;

void analyze\_pointer(int \*ptr){

cout<<"Memory location pointed to by the pointer:"<<ptr<<endl;

cout<<"Value of Integer(which the pointer points to): "<<\*ptr<<endl;

}

int main(void)

{

int a=10;

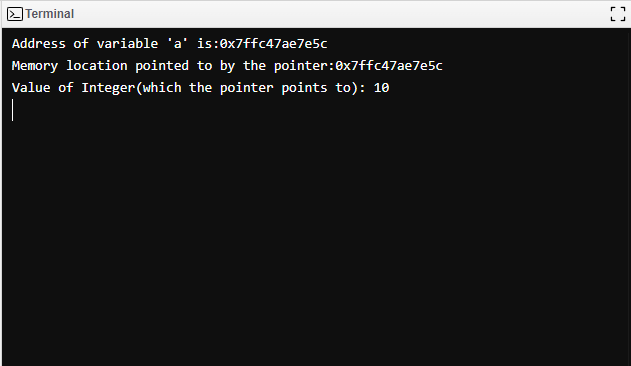
int \*ptr=&a;

cout<<"Address of variable 'a' is:"<<&a<<endl;

analyze\_pointer(ptr);

return 0;

}



**Part 2**: Use the function to complete two tasks:

* Allocate an int on the stack (e.g., “int iValue;"), assign a value to it, and get its memory location (with the reference operator—&) to pass this value to analyze\_pointer.
* Allocate an int on the heap (with the new operator). Assign a value to it, and pass it to analyze\_pointer.

For Task 2 Refer to the help file for more details.

**Code:**

/\* Task 2: Complete the two parts for analyze pointer problem.

Part 2: Use the function to complete two tasks:

 Allocate an int on the stack (e.g., “int iValue;"), assign a value to it, and get its memory location (with the reference operator—&) to pass this value to analyze\_pointer.

 Allocate an int on the heap (with the new operator). Assign a value to it, and pass it to analyze\_pointer.

\*/

#include<iostream>

using namespace std;

void analyze\_pointer(int \*ptr){

cout<<"Memory location pointed to by the pointer:"<<ptr<<endl;

cout<<"Value of Integer(which the pointer points to): "<<\*ptr<<endl;

}

int main(void)

{

int iValue=10;// variable in Stack

int\* ptr = new int;// variable in Heap

\*ptr=60;

cout<<"Address of Stack variable is:"<<&iValue<<endl;

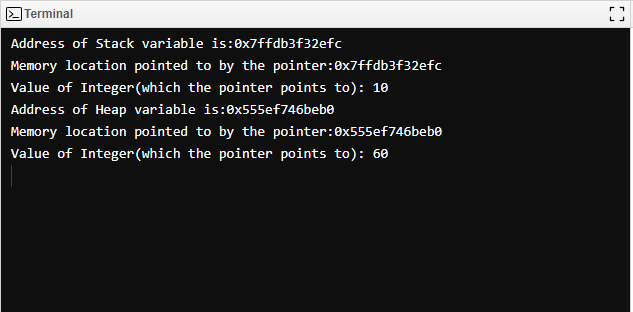
analyze\_pointer(&iValue);

cout<<"Address of Heap variable is:"<<ptr<<endl;

analyze\_pointer(ptr);

return 0;

}



**Task 3**

Define a struct Area that has two private variable members; units of type string and area\_value of type float. Modify the Lab\_3\_-\_Problem\_3.cpp program to create a dynamic variable of type Area.

* Input from the keyboard the area\_value and its units. Compute one-half and one quarter of the area and display the results
* Destroy the dynamic variable at the end
* For Task 3 Refer to the help file for more details.

**Code:**

/\*Task 3: Define a struct Area that has two private variable members; units of type string and area\_value of type float. Modify the Lab\_3\_-\_Problem\_3.cpp program to create a dynamic variable of type Area.

 Input from the keyboard the area\_value and its units. Compute one-half and one quarter of the area and display the results

 Destroy the dynamic variable at the end

 For Task 3 Refer to the help file for more details.

\*/

#include<iostream>

using namespace std;

struct Area{

private:

string units;

float area\_value;

public:

void set\_units(string x){

units=x;

}

void set\_area\_value(float a){

area\_value=a;

}

void cal\_area(){

// Computing one-half and one quarter of the area and displaying the results

cout<<"one half of area:"<<area\_value/2.0<<endl;

cout<<"one quarter of area:"<<area\_value/4.0<<endl;

}

};

int main(void)

{

// creating a dynamic varaible of type area

Area \*areaptr =new Area;

string units;

float area\_value;

// Taking units and area\_value as input from the user

cout<<"Enter units:";

cin>>units;

cout<<"Enter area value:";

cin>>area\_value;

areaptr->set\_units(units);

areaptr->set\_area\_value(area\_value);

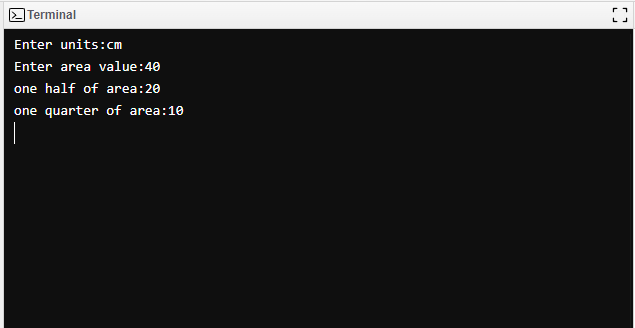
areaptr->cal\_area();

// Destroy the dynamic variable at the end

delete areaptr;

return 0;

}



**Deliverables**

Compile a single word document by filling in the solution part and submit this Word file on LMS. The name of word document should follow this format. i.e. **YourFullName(reg)\_Lab#.** You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.

**Note:** Students are required to upload the lab on LMS before deadline.