# Department of Computing

# School of Electrical Engineering and Computer Science

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

**Class: BSCS 10C**

**Lab 2:  Dynamic Memory Allocation**

**Date: 24th September, 2021**

**Time: 9:00 am – 11:50 am**

# Instructor: Prof. Dr. Faisal Shafait

# Lab Engineer: Mr. Aftab Farooq

# 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];

int i;

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;

}

**Solution:**

|  |
| --- |
| **Code** |
| #include <iostream>  using **namespace** std;  **int** main(**void**)  {  **int** \*salArray = new **int**[20];  **int** i;  for (i = 0; i < 2; ++i)  {  cout << "Enter Salary: ";  cout << endl  << salArray << endl;  cin >> \*salArray;  salArray++;  }  for (i = 0; i < 2; i++)  {  salArray--;  }  for (i = 0; i < 2; ++i){  \*salArray = \*salArray + \*salArray / (i + 1);  salArray++;  }  delete salArray;  return 0;  } |

**Task 2**

Complete the two parts for analyze pointer problem in the Lab\_1\_-\_Problem\_2.cpp file.

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** |
| **void** analyze\_pointer(**int** **\***ptr)  {  *// Add your code here!*  cout << "\nMemory location pointed by the pointer: " << ptr << endl;  cout << "Value of the integer in the location: " << \*ptr << endl;  } |

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.

|  |
| --- |
| **Code** |
| */\**  *\* Kindly fill this information.*  *\* Name: Ahmad Tashfeen*  *\* Student #: 345308*  *\* Date: 24/09/21*  *\*/*  #include <iostream>  using **namespace** std;  */\**  *\* PROBLEM #2: Analyze pointers*  *\*/*  */\**  *\* PART #1: Write a function 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.*  *\*/*  **void** analyze\_pointer(**int** **\***ptr)  {  *// Add your code here!*  cout << "\nMemory location pointed by the pointer: " << ptr << endl;  cout << "Value of the integer in the location: " << \*ptr << endl;  }  **int** main()  {  */\**  *\* PART #2: Use the analyze\_pointer function to complete two TASKS*  *\*/*  */\**  *\* TASK #1*  *\* >> allocate an int on the stack (e.g., "int iValue;")*  *\* >> assign a value to it*  *\* >> get its memory location (with the reference operator---&)*  *\* >> and pass this value to analyze\_pointer.*  *\*/*  cout << "\nPart #2: Memory on stack..." << endl;  *// Add your code here!*  **int** iValue;  cout << "\nEnter an integer value: " << endl;  cin >> iValue;  **int** \*iValueAddress = &iValue;    analyze\_pointer(iValueAddress);  */\**  *\* TASK #2*  *\* >> allocate an int on the heap (with the new operator)*  *\* >> assign a value to it*  *\* >> and pass it to analyze\_pointer.*  *\*/*  cout << "\nMemory on heap..." << endl;  *// Add your code here!*  **int** \*int\_heap = new **int**[1];  cout<< "\nEnter an integer value: " << endl;  cin >> \*int\_heap;  cout << endl;  analyze\_pointer(int\_heap);  return 0;  } |
| **Output** |
| **Text  Description automatically generated** |

Lab#1 (Question #1). What happens?

**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\_1\_-\_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

|  |
| --- |
| Code |
| */\**  *\* Kindly fill this information.*  *\* Name: Ahmad Tashfeen*  *\* Student #: 345308*  *\* Date: 24/09/21*  *\*/*  #include <iostream>  #include <string>  using **namespace** std;  */\**  *\* Define a struct Area that has two private variable members;*  *\* units of type string and area\_value of type float.*  *\*/*  **struct** Area  {  *// Add your code here!*  **private:**  **float** area\_value;  string units;    **public:**  **void** setAreaAndUnits(**float** a, string u){  area\_value = a;  units = u;  }  **float** getArea(){  return area\_value;  }  string getUnits(){  return units;  }  };  **int** main()  {  */\**  *\* Modify program to create a dynamic variable of type Area.*  *\*/*    *// Add your code here!*  Area \*area = new Area;  **float** area\_entered;  string units;  */\**  *\* Input from the keyboard the area\_value and its units.*  *\*/*    *// Add your code here!*  cout << "\nEnter value of Area: " << endl;  cin >> area\_entered;  cout << "\nEnter units of Area: " << endl;  cin >> units;  area->setAreaAndUnits(area\_entered, units);    */\**  *\* Compute one-half and one-quarter of the area*  *\* and display the results*  *\*/*    *// Add your code here!*  **float** oneHalfArea = 0.5 \* area->getArea();  **float** oneQuarterArea = 0.25 \* area->getArea();  cout << "\nOne-half of the Area is: " << oneHalfArea << area->getUnits() << endl;  cout << "One-quarter of the Area is: " << oneQuarterArea << area->getUnits() << endl << endl;  */\**  *\* Destroy the dynamic variable at the end*  *\*/*    *// Add your code here!*  delete area;    return 0;  } |
| Output |
| Text  Description automatically generated |

**Task 4**

To write a C++ program to perform matrix addition and subtraction. Matrices are basically stored in 2D arrays, which are created using dynamic memory allocation.

**Algorithm:**

**Addition:** addition of two matrices is only possible when both matrices are of same dimensions. If this condition is true then corresponding elements are added.

**Requirements:**

* You are required to use dynamically allocated 1D arrays and treat them as 2D arrays i.e. to create an array of 2x3 you will get 6 compartments using new operator.
* You are supposed to write function of addition.
* Your main function should only contain function call to this function
* Your program will ask user about the dimensions of two matrices i.e. rows and columns. Then it will create these matrices dynamically and assign then random values and then perform addition of the two matrices.

|  |
| --- |
| Code |
| #include <iostream>  #include <vector>  #include <time.h>  using **namespace** std;  **int** rows, columns; *//global initialisation of variables*  **void** matrix\_addition(**int** r, **int** c) *//addition function*  {  srand(time(0)); *//starting point for random numbers to create unique seed*  **int** \*matrix1 = new **int**[r\*c]; *//dynamic array of dimensions rows x columns*  cout<<"Matrix 1: \n";  for (**int** i = 0; i < r; i++){  for (**int** j = 0; j < c; j++){  \*matrix1 = rand() % 100; *//values from 0-99 are input in the matrix*  cout << \*matrix1 << "\t";  matrix1++; *//go to next memory location*  }  cout << endl;  }  **int** \*matrix2 = new **int**[r\*c]; *//dynamic array of dimensions rows x columns*  cout<<"\nMatrix 2: \n";  for (**int** i = 0; i < r; i++){  for (**int** j = 0; j < c; j++){  \*matrix2 = rand() % 100; *//values from 0-99 are input*  cout << \*matrix2 << "\t";  matrix2++; *//go to next memory location*  }  cout << endl;  }  for (**int** i = 0; i < r\*c; i++){ *//iterates for the length of the memory allocated to bring the pointer back to the first memory location*  matrix1--;  matrix2--;  }  **int** \*matrixSum = new **int** [r\*c];  cout << "\nSum of Matrices: " << endl;  for (**int** i = 0; i < r; i++){  for (**int** j = 0; j < c; j++){  \*matrixSum = \*matrix1 + \*matrix2; *//adds values from specific memory addresses of matrices*  cout << \*matrixSum << "\t";  matrix1++;  matrix2++;  matrixSum++; *//increment takes pointers to next memory locations*  }  cout << endl;  }  }  **int** main(**void**)  {  cout << "Enter number of rows: " << endl;  cin >> rows;  cout << "\nEnter number of columns: " << endl;  cin >> columns;  cout << "\nPerforming Matrix Addition: (Order of matrices is "<< rows << " x " << columns << ")\n" << endl;  matrix\_addition(rows, columns);  return 0;  } |
| Output |
| Text  Description automatically generated |

**Lab Grading:**

|  |  |
| --- | --- |
| **Task** | **Marks** |
| Lab Viva/Quiz | 5 |
| Comments/ Indentation | 2 |
| Solution Document | 2 |
| Output Screen Shots | 1 |
| -- | -- |
| Total | 10 |

**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#.** This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. 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. In case of any problems discuss it by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).

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

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.