**OBE IMPLEMENTATION:UNIVERSITY SETTING**

***by***

##### Tanmayee Chanda[AP23110011035] Anurag[AP23110011041]

##### Vishwanadh[AP23110011047] Komal[AP23110011050] Vishnu[AP23110011049]

##### Tatparya Rudra[Ap23110011030]

#### A report for the CS204:Design and Analysis of Algorithm project



###### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**SRM UNIVERSITY AP::AMARAVATI**

**INDEX**

[Introduction 2](#_TOC_250011)

[Project Modules: 2](#_TOC_250010)

Architecture Diagram 3

[Module Description 4](#_TOC_250009)

[Programming Details naming conventions to be used: 4](#_TOC_250008)

Field/table details:(Course Outcome Setting] 5

[Algorithm Details: 5](#_TOC_250007)

1. [Sorting 5](#_TOC_250006)
2. [Searching 5](#_TOC_250005)
3. [Storing the details in a text file 6](#_TOC_250004)

[Source Code 6](#_TOC_250003)

[Comparison of Sorting Algorithms 7](#_TOC_250002)

[Comparison of Searching Algorithms 8](#_TOC_250001)

Screen Shots 9

# Introduction

Our University (herewith considered as SRM-AP) is going to implement OBE(Outcome Based Education) in their university and you assigned in the project to develop an application with any programming Language you are well versed and you were supposed to do searching and sorting using learned algorithms,comparing your sorting algorithm with any one of existing algorithm,displaying the time complexity of both algorithms and explaining advantages and disadvantages of the algorithm.

## Project Modules:

Various Modules available in the project are

1. Blooms Level setting
2. Program Level Objective Setting 3.University

4.Schools 5.Department 6.Programs 7.Courses

1. Course objective setting
2. Course Outcome Setting
3. Course Articulation matrix Setting
4. Course Utilization Setting
5. Course Reference Setting.

# Module Description

**Module Name:**Eg)Course Outcome Setting

Module Description:

This module is used to create,Update,Retrieve,Delete(hereafter known as CURD) details of the module and storing the details in the text file.you have to provide option for searching and sorting of fields mentioned below according to algorithms given for you

## Programming Details naming conventions to be used:

* + **File name:AP23110011035\_ Course Outcome Setting**
  + Function/method name
* **Create: AP23110011035**\_ **Course Outcome Setting** \_create
* **Update: AP23110011035**\_ **Course Outcome Setting** \_update
* **Retrieve: AP23110011035**\_ **Course Outcome Setting** \_retrive
* **Delete: AP23110011035**\_ **Course Outcome Setting** \_delete
* **Sorting: AP23110011035**\_ **Course Outcome Setting** \_Quick
* **Searching: AP23110011035**\_ **Course Outcome Setting** \_Linear
* **Storing: AP23110011035**\_ **Course Outcome Setting** \_storing

**○**

* + - Comparison(both searching and Sorting):
* For Searching- **AP23110011035**\_ **Course Outcome Setting**

\_Compare\_Search\_Merge

* For Sorting- **AP23110011035**\_ **Course Outcome Setting**

\_Compare\_sorting\_Binary

■

* + - Time Complexity(both searching and Sorting):
* For Searching- **AP23110011035**\_ **Course Outcome Setting**

\_O(n)\_Linear

* For Sorting- **AP23110011035**\_ **Course Outcome Setting** \_O(n log n)

\_Quick Sort

■

* + - Algorithm Details(pseudocode or steps)(both searching and Sorting):
* For Searching- **AP23110011035**\_ **Course Outcome Setting**

\_Linear\_details

* For Sorting- **AP23110011035**\_ **Course Outcome Setting**

\_Quick\_sort\_details

■

* File name(for storing the details)
  + File name to be used is:-Course Outcome Setting.txt

Field/table details:(eg university)[you consider you module ]

|  |  |
| --- | --- |
| **Field Name** | **Data type** |
| outcome\_id | integer |
| outcome\_code | String |
| outcome\_name | String |
| outcome\_address | String |
| outcome\_email | String |
| outcome\_website | String |

## Algorithm Details:

### Sorting

* + You have to provide sorting based on **Outcome code ,Outcome\_name , Outcome\_email.**
  + Compare the algorithm you have used with any of the other sorting algorithm
  + Display the time complexity of both algorithms.
  + Display the pseudocode/algorithm of the sorting algorithm used by you

### Searching

* + You have provide sorting based on **Outcome code ,Outcome\_name , Outcome\_email.**
  + Compare the algorithm used with any of the other algorithm you have learned
  + Display the time complexity of both algorithms.
  + Display the pseudocode/algorithm of the searching algorithm used by you.

### Storing the details in a text file

* + Storing the details in the text file once details are entered.
  + Delete the detail from the text file once details are deleted.
  + Update the text file once details are updated.

# Source Code

#include <stdio.h> #include <stdlib.h> #include <string.h>

#define MAX 100

typedef struct {

int Outcome\_id;

char Outcome\_code[10]; char Outcome\_name[50];

char Outcome\_description[100]; char Outcome\_outcome[100];

} CourseOutcome;

CourseOutcome courseOutcomes[MAX]; int courseOutcome\_count = 0;

const char\* FILE\_NAME = "course\_outcome\_setting.txt"; void createOutcome();

void updateOutcome();

void retrieveOutcomes(); void deleteOutcome(); void storeOutcomes();

void quickSortOutcomes(int low, int high); int partitionOutcomes(int low, int high); void searchOutcomeLinear();

void load\_from\_file() {

FILE \*file = fopen(FILE\_NAME, "r"); if (file == NULL) {

return;

}

courseOutcome\_count = 0;

while (fscanf(file, "%d %s %s %s %s\n", &courseOutcomes[courseOutcome\_count].Outcome\_id,

courseOutcomes[courseOutcome\_count].Outcome\_code,

courseOutcomes[courseOutcome\_count].Outcome\_name, courseOutcomes[courseOutcome\_count].Outcome\_description, courseOutcomes[courseOutcome\_count].Outcome\_outcome) != EOF) {

courseOutcome\_count++;

}

fclose(file);

}

void storeOutcomes() {

FILE \*file = fopen(FILE\_NAME, "w"); if (file == NULL) {

printf("Error opening file!\n"); return;

}

for (int i = 0; i < courseOutcome\_count; i++) {

fprintf(file, "%d %s %s %s %s\n", courseOutcomes[i].Outcome\_id, courseOutcomes[i].Outcome\_code, courseOutcomes[i].Outcome\_name, courseOutcomes[i].Outcome\_description, courseOutcomes[i].Outcome\_outcome);

}

fclose(file);

}

void createOutcome() {

if (courseOutcome\_count >= MAX) { printf("Course outcome list is full!\n"); return;

}

CourseOutcome c; printf("Enter Outcome ID: "); scanf("%d", &c.Outcome\_id); printf("Enter Outcome Code: "); scanf("%s", c.Outcome\_code);

printf("Enter Outcome Name: "); scanf("%s", c.Outcome\_name); printf("Enter Outcome Description: "); scanf("%s", c.Outcome\_description); printf("Enter Outcome: "); scanf("%s", c.Outcome\_outcome);

courseOutcomes[courseOutcome\_count++] = c; storeOutcomes();

printf("Course outcome created successfully!\n");

}

void updateOutcome() { int id;

printf("Enter Outcome ID to update: "); scanf("%d", &id);

for (int i = 0; i < courseOutcome\_count; i++) { if (courseOutcomes[i].Outcome\_id == id) {

printf("Enter new Outcome Code: ");

scanf("%s", courseOutcomes[i].Outcome\_code); printf("Enter new Outcome Name: ");

scanf("%s", courseOutcomes[i].Outcome\_name); printf("Enter new Outcome Description: ");

scanf("%s", courseOutcomes[i].Outcome\_description); printf("Enter new Outcome: ");

scanf("%s", courseOutcomes[i].Outcome\_outcome);

storeOutcomes();

printf("Course outcome updated successfully!\n"); return;

}

}

printf("Course outcome with ID %d not found.\n", id);

}

void retrieveOutcomes() {

printf("\nList of Course Outcomes:\n");

for (int i = 0; i < courseOutcome\_count; i++) {

printf("ID: %d\nCode: %s\nName: %s\nDescription: %s\nOutcome: %s\n\n", courseOutcomes[i].Outcome\_id, courseOutcomes[i].Outcome\_code, courseOutcomes[i].Outcome\_name, courseOutcomes[i].Outcome\_description, courseOutcomes[i].Outcome\_outcome);

}

}

void deleteOutcome() { int id;

printf("Enter Outcome ID to delete: "); scanf("%d", &id);

for (int i = 0; i < courseOutcome\_count; i++) { if (courseOutcomes[i].Outcome\_id == id) {

for (int j = i; j < courseOutcome\_count - 1; j++) { courseOutcomes[j] = courseOutcomes[j + 1];

}

courseOutcome\_count--; storeOutcomes();

printf("Course outcome deleted successfully!\n"); return;

}

}

printf("Course outcome with ID %d not found.\n", id);

}

int partitionOutcomes(int low, int high) { char pivot[10];

strcpy(pivot, courseOutcomes[high].Outcome\_code); int i = low - 1;

for (int j = low; j < high; j++) {

if (strcmp(courseOutcomes[j].Outcome\_code, pivot) < 0) { i++;

CourseOutcome temp = courseOutcomes[i]; courseOutcomes[i] = courseOutcomes[j]; courseOutcomes[j] = temp;

}

}

CourseOutcome temp = courseOutcomes[i + 1]; courseOutcomes[i + 1] = courseOutcomes[high]; courseOutcomes[high] = temp;

return i + 1;

}

void quickSortOutcomes(int low, int high) { if (low < high) {

int pi = partitionOutcomes(low, high); quickSortOutcomes(low, pi - 1); quickSortOutcomes(pi + 1, high);

}

}

void searchOutcomeLinear() { char code[10];

printf("Enter Outcome Code to search: "); scanf("%s", code);

for (int i = 0; i < courseOutcome\_count; i++) {

if (strcmp(courseOutcomes[i].Outcome\_code, code) == 0) {

printf("ID: %d\nCode: %s\nName: %s\nDescription: %s\nOutcome: %s\n", courseOutcomes[i].Outcome\_id, courseOutcomes[i].Outcome\_code, courseOutcomes[i].Outcome\_name, courseOutcomes[i].Outcome\_description, courseOutcomes[i].Outcome\_outcome);

return;

}

}

printf("Course outcome with code %s not found.\n", code);

}

int main() { load\_from\_file();

int choice; while (1) {

printf("\n1. Create Outcome\n2. Update Outcome\n3. Retrieve Outcomes\n4. Delete Outcome\n5. Search by Code (Linear)\n6. Sort by Code (Quick Sort)\n7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) { case 1:

createOutcome(); break;

case 2:

updateOutcome(); break;

case 3:

retrieveOutcomes(); break;

case 4:

deleteOutcome(); break;

case 5:

searchOutcomeLinear(); break;

case 6:

quickSortOutcomes(0, courseOutcome\_count - 1); printf("Course outcomes sorted by code!\n"); retrieveOutcomes();

break; case 7:

exit(0); default:

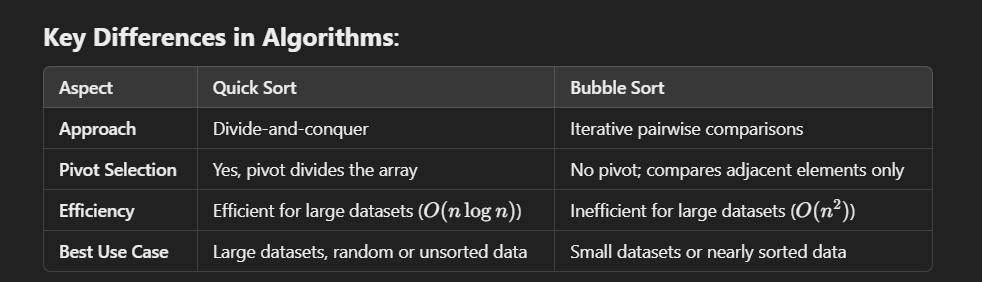
printf("Invalid choice!\n");

}

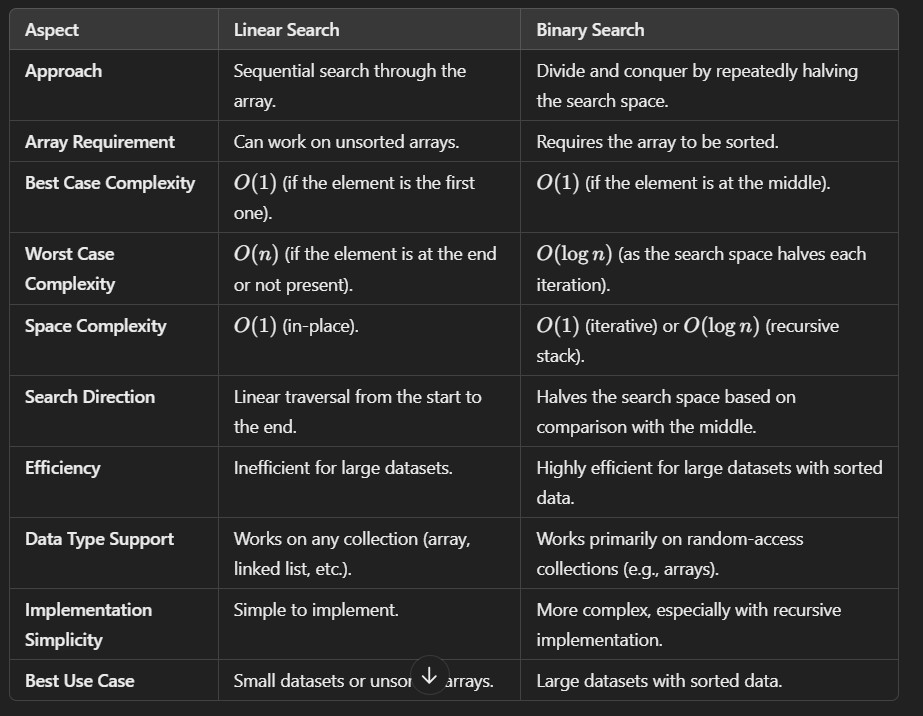
}

return 0;

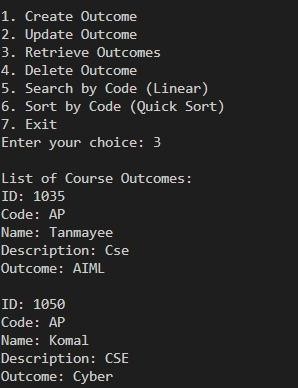
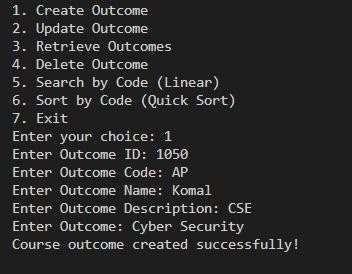
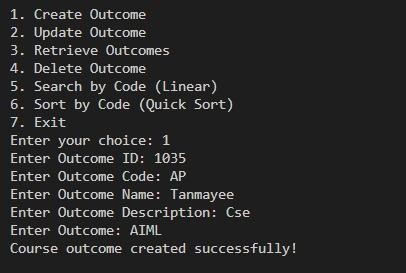
# Comparison of Sorting Algorithms

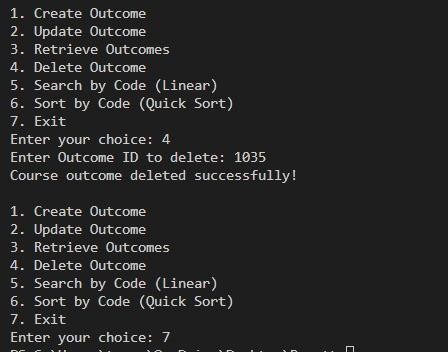


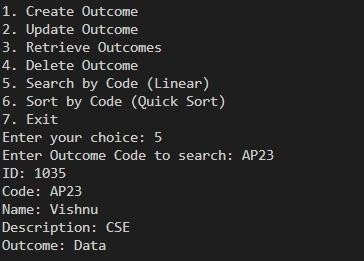
# Comparison of Searching Algorithms

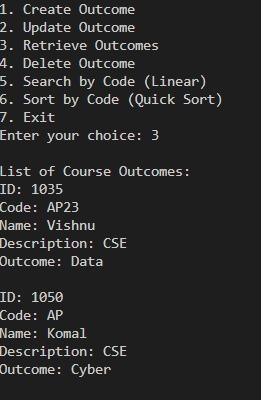


ScreenShots [OUTPUT]









# Conclusion

### In this project, we developed a simple yet efficient system to manage course outcomes, aligning with the principles of Outcome-Based Education (OBE). The system facilitates easy management of course outcomes through CRUD operations, enabling users to add, update, view, and delete outcome details seamlessly.

To enhance functionality, we implemented Linear Search and Binary Search algorithms for efficient retrieval of specific outcomes. These algorithms, especially Binary Search, are particularly beneficial for large datasets, ensuring quick and accurate searches.

Through the integration of sorting algorithms like Quick Sort, we explored the impact of various search and sort methods on speed and efficiency. This understanding underscores the importance of algorithm selection when handling data, especially in educational contexts.

This tool demonstrates how a well-designed system can simplify course outcome tracking and improvement, supporting educators in achieving OBE's emphasis on clear, measurable learning outcomes.

Overall, our project contributes to the effective management of course outcomes, making it easier for institutions to align with OBE objectives and enhance learning experiences.