Data Structures & Algorithms Lab CSL-221

*A Project Report On*

“COLOR SORTER”

*Submitted By*

1. Abdullah Khan (02-134221-023) (BSCS-3A)
2. Jahanzaib Ali (02-134221-057) (BSCS-3A)
3. Rayyan Sajid (02-134221-043) (BSCS-3A)

*Under The Supervision Of*

**Ms. Lubna Siddiqui**

**(Course Lecturer)**

**Ms. Tooba**

**(Lab Instructor)**



Bahria University Karachi Campus

(Spring 2023)

**Table of Contents**

*Pg. No*

**Chapter 1: Introduction**

1.1 Theory 3

1.2 Project Objectives 3

**Chapter 2: System Requirements**

2.1 Hardware Requirements

2.2 Software Requirements

**Chapter 3: Design & Working**

3.1 Program Description 4-5

3.2 Code 11-17

3.3 Output Screenshots 18-21

**Chapter 4: Conclusion**  22

**CHAPTER 1: INTRODUCTION**

* 1. **Theory:**

The color sorter program using arrays and test tubes employs an array-based data structure to represent test tubes and organizes objects based on their colors. The program assigns color categories to objects and applies a sorting algorithm to group objects of the same color together within the array. Once sorted, the program directs the objects into their respective test tubes using a mechanical system or automated gates. This simple yet effective theory simplifies the sorting process, allowing for efficient and customizable color-based sorting operations.

* 1. **Project Objectives:**

Specifically, objectives of this project will consist of:

1. Develop an automated system that can accurately sort objects based on their colors.
2. Utilize data structures to organize and process color information efficiently.
3. Implement sorting algorithms to optimize the speed and accuracy of the sorting process.
4. Minimize human intervention and streamline industrial sorting operations.
5. Demonstrate the practical application of data structures in real-world scenarios
6. To be played as game in order to gain better thinking

**CHAPTER 2: SYSTEM REQUIREMENTS**

**2.1) Hardware Specifications:**

- Dual Core x86 Processor or above.

- 2GB RAM minimum or above.

- Minimum 20GB Hardware required.

**2.1) Software Specification:**

- Visual Studio 2013 - 2022

- Microsoft Windows 7 or above required

**CHAPTER 3: DESIGN & WORKING**

* 1. **Program Description:**

The color sorter program utilizes arrays as test tubes as sorting containers involves the following program description:

The color sorter program is designed to automate the process of sorting colors using arrays as test tubes. The colors are to be sorted in each text tube. It has multiple levels which are based on their difficulty.

The color data is then stored in an array-based data structure. Each array element represents a test tube and can hold objects of a specific color category. The program assigns an appropriate color category to each object based on its color attributes.

To sort the objects, the program implements a sorting algorithm tailored for array-based sorting. Push and Pop functions of stack are used to sort them out. The sorting algorithm rearranges the elements within the array, grouping objects of the same color category together.

Once the sorting process is complete, the program directs the objects into their respective test tubes based on their color categories. Each Level has a certain amount of moves the user can use to sort the colors.

The user who has the least moves is at the top of the scoreboard and the list carries so so on depending on the number of moves

The program provides a user-friendly interface for users to interact with the system. Users can configure sorting parameters, monitor the sorting process, and view the arrangement of objects in the test tubes in real-time.

The color sorter program utilizing arrays and Stacks offers a customizable and scalable solution for sorting objects based on their colors. It simplifies the sorting process by organizing objects into distinct test tubes, providing an efficient and reliable method for color-based sorting operations.

* 1. **Code:**

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

struct player {

int moves = 0;

string name, levels;

}players;

bool boolean = true;

string level1\_testTube\_1[4]= {"R","B","R","B"};

string level1\_testTube\_2[4]= {"R","R","B","B"};

string level1\_testTube\_3[4]= {};

int top1 = 3;

int top2 = 3;

int top3 = -1;

string level2\_testTube\_1[4] = { "O","B","R","O" };

string level2\_testTube\_2[4] = { "O","B","O","B" };

string level2\_testTube\_3[4] = { "R","R","R","B" };

string level2\_testTube\_4[4] = {};

string level2\_testTube\_5[4] = {};

int topp1 = 3;

int topp2 = 3;

int topp3 = 3;

int topp4 = -1;

int topp5 = -1;

string level3\_testTube\_1[4] = { "G","P","B","O" };

string level3\_testTube\_2[4] = { "P","G","B","O" };

string level3\_testTube\_3[4] = { "O","R","P","R" };

string level3\_testTube\_4[4] = { "P","O","R","B" };

string level3\_testTube\_5[4] = { "G","G","R","B" };

string level3\_testTube\_6[4] = {};

string level3\_testTube\_7[4] = {};

int toppp1 = 3;

int toppp2 = 3;

int toppp3 = 3;

int toppp4 = 3;

int toppp5 = 3;

int toppp6 = -1;

int toppp7 = -1;

string test1[4] = {"R","R","R","R"};

string test2[4] = {"B","B","B","B"};

string test3[4] = {"O","O","O","O"};

string test4[4] = {"P","P","P","P"};

string test5[4] = {"G","G","G","G"};

bool push\_tubesFunction(string num,int& top,string testTube[])

{

bool b = true;

if (top == 3)

{

cout << "Not Enough space in the test tube\n";

b = false;

}

else

{

top = top + 1;

testTube[top] = num;

b = true;

}

return b;

}

string popTubesFunction(int& top,string testTubes[])

{

boolean = true;

string c;

if (top == -1)

{

cout << "\nStack Underflow\n";

boolean = false;

}

else

{

c = testTubes[top];

top = top - 1;

}

return c;

}

void displayAllTubes\_level1() {

int maxTop = max(top1, max(top2, top3));

for (int i = 3; i >= 0; --i) {

if (i <= top1) {

cout << "|"<<level1\_testTube\_1[i] << "|" << "\t";

}

else {

cout<<"|" << "0|\t";

}

if (i <= top2) {

cout << "|" << level1\_testTube\_2[i] << "|" << "\t";

}

else {

cout << "|0|\t";

}

if (i <= top3) {

cout << "|" << level1\_testTube\_3[i]<<"|" ;

}

else {

cout<<"|"<< "0""|" ;

}

cout << endl;

}

}

void displayAllTubes\_level3() {

int maxTop = max(toppp1, max(toppp2, max(toppp3, max(toppp4, max(toppp5, max(toppp6, toppp7))))));

for (int i = 3; i >= 0; --i) {

if (i <= toppp1) {

cout << "|" << level3\_testTube\_1[i] << "|";

}

else {

cout << "|0|";

}

if (i <= toppp2) {

cout << "\t|" << level3\_testTube\_2[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= toppp3) {

cout << "\t|" << level3\_testTube\_3[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= toppp4) {

cout << "\t|" << level3\_testTube\_4[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= toppp5) {

cout << "\t|" << level3\_testTube\_5[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= toppp6) {

cout << "\t|" << level3\_testTube\_6[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= toppp7) {

cout << "\t|" << level3\_testTube\_7[i] << "|";

}

else {

cout << "\t|0|";

}

cout << std::endl;

}

}

void displayAllTubes\_level2() {

int maxTop = max(topp1, max(topp2, max(topp3, max(topp4, topp5))));

for (int i = 3; i >= 0; --i) {

if (i <= topp1) {

cout << "|" << level2\_testTube\_1[i] << "|";

}

else {

cout << "|0|";

}

if (i <= topp2) {

cout << "\t|" << level2\_testTube\_2[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= topp3) {

cout << "\t|" << level2\_testTube\_3[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= topp4) {

cout << "\t|" << level2\_testTube\_4[i] << "|";

}

else {

cout << "\t|0|";

}

if (i <= topp5) {

cout << "\t|" << level2\_testTube\_5[i] << "|";

}

else {

cout << "\t|0|";

}

cout << std::endl;

}

}

bool checkAndPrintMessage() {

bool match = true;

bool done = false;

for (int i = 0; i < 4; i++) {

if (level1\_testTube\_2[i] != test1[i] || level1\_testTube\_3[i] != test2[i] ) {

match = false;

break;

}

}

int i;

if (match == true) {

cout << " ------------------------------------"<<endl;

cout << "| Congratulations! Level 1 completed | " << endl;

cout << " ------------------------------------"<<endl;

done = true;

cout << "Press any key to go to the main menu" << endl;

cin >> i;

}

return done;

}

bool compareTestTube(string\* testTube, string\* test) {

for (int i = 0; i < 4; i++) {

if (testTube[i] != test[i]) {

return false;

}

}

return true;

}

bool checkAndPrintMessageLevel3() {

int matchCount = 0;

bool done = false;

if (compareTestTube(level3\_testTube\_1, test1) || compareTestTube(level3\_testTube\_2, test1) || compareTestTube(level3\_testTube\_3, test1) || compareTestTube(level3\_testTube\_4, test1) || compareTestTube(level3\_testTube\_5, test1) || compareTestTube(level3\_testTube\_6, test1) || compareTestTube(level3\_testTube\_7, test1))

matchCount++;

if (compareTestTube(level3\_testTube\_1, test2) || compareTestTube(level3\_testTube\_2, test2) || compareTestTube(level3\_testTube\_3, test2) || compareTestTube(level3\_testTube\_4, test2) || compareTestTube(level3\_testTube\_5, test2) || compareTestTube(level3\_testTube\_6, test2) || compareTestTube(level3\_testTube\_7, test2))

matchCount++;

if (compareTestTube(level3\_testTube\_1, test3) || compareTestTube(level3\_testTube\_2, test3) || compareTestTube(level3\_testTube\_3, test3) || compareTestTube(level3\_testTube\_4, test3) || compareTestTube(level3\_testTube\_5, test3) || compareTestTube(level3\_testTube\_6, test3) || compareTestTube(level3\_testTube\_7, test3))

matchCount++;

if (compareTestTube(level3\_testTube\_1, test4) || compareTestTube(level3\_testTube\_2, test4 )|| compareTestTube(level3\_testTube\_3, test4) || compareTestTube(level3\_testTube\_4, test4) || compareTestTube(level3\_testTube\_5, test4) || compareTestTube(level3\_testTube\_6, test4) || compareTestTube(level3\_testTube\_7, test4))

matchCount++;

if (compareTestTube(level3\_testTube\_1, test5) || compareTestTube(level3\_testTube\_2, test5) || compareTestTube(level3\_testTube\_3, test5) || compareTestTube(level3\_testTube\_4, test5) || compareTestTube(level3\_testTube\_5, test5) || compareTestTube(level3\_testTube\_6, test5) || compareTestTube(level3\_testTube\_7, test5))

matchCount++;

int i;

if (matchCount >= 5) {

cout << " -------------------------------------" << endl;

cout << "| Congratulations! Level 3 completed! |" << endl;

cout << " -------------------------------------" << endl;

cout << "Press any key to go to the main menu" << endl;

cin >> i;

done = true;

}

return done;

}

bool level\_3() {

players.moves = 0;

x:

bool completion = false;

system("CLS");

int moves = 0;

int a;

int b;

int c;

int k;

int choi;

cout << R"(

\_\_\_\_\_\_ \_\_ \_\_\_\_\_\_\_

/ \_\_/ / \_\_\_ \_\_\_/ /\_\_ / \_\_/ \_ \

\_\ \/ \_ \/ -\_) \_ / \_ \\_\ \/ \_\_\_/

\_\_\_/\_//\_/\\_\_/\\_,\_/ .\_\_/\_\_\_/\_/

/\_/

Welcome to the Level 3!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

)" << endl;

cout << "| Maximum moves limit for level 2 is 25 moves |\n";

cout << "| You have to sort the balls by colors | " << endl;

cout << "| press 1 to remove the ball from test tube 1 |" << endl;

cout << "| press 2 to remove the ball from test tube 2 |" << endl;

cout << "| press 3 to remove the ball from test tube 3 |" << endl;

cout << "| press 4 to remove the ball from test tube 4 |" << endl;

cout << "| press 5 to remove the ball from test tube 5 |" << endl;

cout << "| Press 0 to exit |" << endl;

string s;

do {

if (moves > 25)

{

cout << "Moves Limit exceeded!\n";

cout << "Press 1 to add 25 more moves or 0 to exit" << endl;

cin >> k;

if (k == 1) {

goto x;

}

else if (k == 0)

{

break;

}

else {

cout << "Invalid choice!" << endl;

}

}

bool done;

done = checkAndPrintMessageLevel3();

if (done == true) {

completion = true;

break;

}

displayAllTubes\_level3();

cout << "Moves done:" << moves << endl;

cout << endl;

cout << "Enter your choice ";

cin >> choi;

switch (choi) {

case 1:

s = popTubesFunction(toppp1, level3\_testTube\_1);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

bool b, c;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp1, level3\_testTube\_1);

}

break;

}

else

break;

case 2:

s = popTubesFunction(toppp2, level3\_testTube\_2);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp2, level3\_testTube\_2);

}

break;

}

else

break;

case 3:

s = popTubesFunction(toppp3, level3\_testTube\_3);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp3, level3\_testTube\_3);

}

break;

}

else

break;

case 4:

s = popTubesFunction(toppp4, level3\_testTube\_4);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp4, level3\_testTube\_4);

}

break;

}

else

break;

case 5:

s = popTubesFunction(toppp5, level3\_testTube\_5);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp5, level3\_testTube\_5);

}

break;

}

else

break;

case 6:

s = popTubesFunction(toppp6, level3\_testTube\_6);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (b == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp6, level3\_testTube\_6);

}

break;

}

else

break;

case 7:

s = popTubesFunction(toppp7, level3\_testTube\_7);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, toppp1, level3\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, toppp2, level3\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, toppp3, level3\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, toppp4, level3\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, toppp5, level3\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 6) {

c = push\_tubesFunction(s, toppp6, level3\_testTube\_6);

if (c == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else if (a == 7) {

c = push\_tubesFunction(s, toppp7, level3\_testTube\_7);

if (c == false) {

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, toppp7, level3\_testTube\_7);

}

break;

}

else

break;

default:

cout << "Invalid Choice, please try again";

}

cout << endl;

moves++;

} while (choi != 0);

players.moves = players.moves + moves;

players.levels = "3";

return completion;

}

bool checkAndPrintMessageLevel2() {

int matchCount = 0;

bool done = false;

if (compareTestTube(level2\_testTube\_1, test1) || compareTestTube(level2\_testTube\_2, test1) || compareTestTube(level2\_testTube\_3, test1)|| compareTestTube(level2\_testTube\_4, test1)|| compareTestTube(level2\_testTube\_5, test1))

matchCount++;

if (compareTestTube(level2\_testTube\_1, test2) || compareTestTube(level2\_testTube\_2, test2) || compareTestTube(level2\_testTube\_3, test2) || compareTestTube(level2\_testTube\_4, test2) || compareTestTube(level2\_testTube\_5, test2))

matchCount++;

if (compareTestTube(level2\_testTube\_1, test3) || compareTestTube(level2\_testTube\_3, test3) || compareTestTube(level2\_testTube\_3, test3) || compareTestTube(level2\_testTube\_4, test3) || compareTestTube(level2\_testTube\_5, test3))

matchCount++;

int i;

if (matchCount >= 3) {

cout << " -------------------------------------" << endl;

cout << "| Congratulations! Level 2 completed! |" << endl;

cout << " -------------------------------------" << endl;

cout << "Press any key to go to the main menu" << endl;

cin >> i;

done= true;

}

return done;

}

bool level\_2() {

players.moves = 0;

x:

bool completion = false;

system("CLS");

int moves = 0;

int a;

int b;

int c;

int k;

int choi;

string s;

do {

system("CLS");

cout << R"(

\_\_\_\_\_\_ \_\_ \_\_\_\_\_\_\_

/ \_\_/ / \_\_\_ \_\_\_/ /\_\_ / \_\_/ \_ \

\_\ \/ \_ \/ -\_) \_ / \_ \\_\ \/ \_\_\_/

\_\_\_/\_//\_/\\_\_/\\_,\_/ .\_\_/\_\_\_/\_/

/\_/

Welcome to the Level 2!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

)" << endl;

cout << "| Maximum moves limit for level 2 is 15 moves |\n";

cout << "| You have to sort the balls by colors | " << endl;

cout << "| press 1 to remove the ball from test tube 1 |" << endl;

cout << "| press 2 to remove the ball from test tube 2 |" << endl;

cout << "| press 3 to remove the ball from test tube 3 |" << endl;

cout << "| press 4 to remove the ball from test tube 4 |" << endl;

cout << "| press 5 to remove the ball from test tube 5 |" << endl;

cout << "| Press 0 to exit |" << endl;

if (moves > 15)

{

cout << "Moves Limit exceeded!\n";

cout << "Press 1 to add 15 more moves or 0 to exit" << endl;

cin >> k;

if (k == 1) {

goto x;

}

else if (k == 0)

{

break;

}

else {

cout << "Invalid choice!" << endl;

}

}

bool done = checkAndPrintMessageLevel2();

if (done == true) {

completion = true;

break;

}

displayAllTubes\_level2();

cout << "Moves done:" << moves << endl;

cout << endl;

cout << "Enter your choice ";

cin >> choi;

switch (choi) {

case 1:

s = popTubesFunction(topp1, level2\_testTube\_1);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

bool b, c;

if (a == 1) {

b = push\_tubesFunction(s, topp1, level2\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, topp2, level2\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, topp3, level2\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, topp4, level2\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, topp5, level2\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, topp1, level2\_testTube\_1);

}

break;

}

else

break;

case 2:

s = popTubesFunction(topp2, level2\_testTube\_2);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, topp1, level2\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, topp2, level2\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, topp3, level2\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, topp4, level2\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, topp5, level2\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, topp2, level2\_testTube\_2);

}

break;

}

else

break;

case 3:

s = popTubesFunction(topp3, level2\_testTube\_3);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, topp1, level2\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, topp2, level2\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, topp3, level2\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, topp4, level2\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, topp5, level2\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

break;

}

else

break;

case 4:

s = popTubesFunction(topp4, level2\_testTube\_4);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, topp1, level2\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, topp4, level2\_testTube\_4);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, topp2, level2\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, topp4, level2\_testTube\_4);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, topp3, level2\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, topp4, level2\_testTube\_4);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, topp4, level2\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, topp3, level2\_testTube\_3);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, topp5, level2\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, topp4, level2\_testTube\_4);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, topp4, level2\_testTube\_4);

}

break;

}

else

break;

case 5:

s = popTubesFunction(topp5, level2\_testTube\_5);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1) {

b = push\_tubesFunction(s, topp1, level2\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

}

else if (a == 2) {

b = push\_tubesFunction(s, topp2, level2\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, topp3, level2\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

}

else if (a == 4) {

c = push\_tubesFunction(s, topp4, level2\_testTube\_4);

if (c == false) {

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

}

else if (a == 5) {

c = push\_tubesFunction(s, topp5, level2\_testTube\_5);

if (c == false) {

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, topp5, level2\_testTube\_5);

}

break;

}

else

break;

default:

cout << "Invalid Choice, please try again";

}

cout << endl;

moves++;

} while (choi != 0);

players.moves = players.moves+ moves;

players.levels = "2";

return completion;

}

void checkLevel1Completion() {

// Check if level1\_testTube\_1 and level1\_testTube\_2 have the same colors

if (std::equal(std::begin(level1\_testTube\_1), std::end(level1\_testTube\_1), std::begin(level1\_testTube\_2))) {

// Check if level1\_testTube\_1 or level1\_testTube\_2 matches level1\_testTube\_3

if (std::equal(std::begin(level1\_testTube\_1), std::end(level1\_testTube\_1), std::begin(level1\_testTube\_3)) ||

std::equal(std::begin(level1\_testTube\_2), std::end(level1\_testTube\_2), std::begin(level1\_testTube\_3))) {

std::cout << "Congratulations! Level 1 completed.\n";

}

}

}

bool level\_1() {

players.moves = 0;

x:

bool completion = false;

system("CLS");

int moves = 0;

int a;

int b;

int c;

int k;

int choi;

do {

system("CLS");

cout << R"(

\_\_\_\_\_\_ \_\_ \_\_\_\_\_\_\_

/ \_\_/ / \_\_\_ \_\_\_/ /\_\_ / \_\_/ \_ \

\_\ \/ \_ \/ -\_) \_ / \_ \\_\ \/ \_\_\_/

\_\_\_/\_//\_/\\_\_/\\_,\_/ .\_\_/\_\_\_/\_/

/\_/

Welcome to the Level 1!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

)" << endl;

int o;

string s;

cout << "| Maximum moves limit for level 1 is 10 moves |\n";

cout << "| You have to sort the balls by colors | " << endl;

cout << "| press 1 to remove the ball from test tube 1 |" << endl;

cout << "| press 2 to remove the ball from test tube 2 |" << endl;

cout << "| press 3 to remove the ball from test tube 3 |" << endl;

cout << "| Press 0 to exit |" << endl;

cout << endl;

if (moves > 10)

{

cout << "Moves Limit exceeded!\n";

cout << "Press 1 to add 10 more moves or 0 to exit" << endl;

cin >> k;

if (k == 1) {

goto x;

}

else if (k == 0)

{

break;

}

else {

cout << "Invalid choice!" << endl;

}

}

bool done;

done = checkAndPrintMessage();

if (done == true) {

completion = true;

break;

}

displayAllTubes\_level1();

cout << "Moves done:" << moves << endl;

cout << endl;

cout << "Enter your choice ";

cin >> choi;

switch (choi) {

case 1:

s = popTubesFunction(top1, level1\_testTube\_1);

if (boolean == true)

{

displayAllTubes\_level1();

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

bool b, c;

if (a == 2) {

b = push\_tubesFunction(s, top2, level1\_testTube\_2);

if (b == false) {

push\_tubesFunction(s, top1, level1\_testTube\_1);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, top3, level1\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, top1, level1\_testTube\_1);

}

}

else if (a == 1) {

c = push\_tubesFunction(s, top1, level1\_testTube\_1);

if (c == false) {

push\_tubesFunction(s, top1, level1\_testTube\_1);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, top1, level1\_testTube\_1);

}

break;

}

else {

break;

}

case 2:

s = popTubesFunction(top2, level1\_testTube\_2);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1)

{

b = push\_tubesFunction(s, top1, level1\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, top2, level1\_testTube\_2);

}

}

else if (a == 2) {

c = push\_tubesFunction(s, top2, level1\_testTube\_2);

if (c == false) {

push\_tubesFunction(s, top2, level1\_testTube\_2);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, top3, level1\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, top2, level1\_testTube\_2);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, top2, level1\_testTube\_2);

}

break;

}

else

break;

case 3:

s = popTubesFunction(top3, level1\_testTube\_3);

if (boolean == true) {

cout << "Enter number of the test tube to enter the ball" << endl;

cin >> a;

if (a == 1)

{

b = push\_tubesFunction(s, top1, level1\_testTube\_1);

if (b == false) {

push\_tubesFunction(s, top3, level1\_testTube\_3);

}

}

else if (a == 2) {

c = push\_tubesFunction(s, top2, level1\_testTube\_2);

if (c == false) {

push\_tubesFunction(s, top3, level1\_testTube\_3);

}

}

else if (a == 3) {

c = push\_tubesFunction(s, top3, level1\_testTube\_3);

if (c == false) {

push\_tubesFunction(s, top3, level1\_testTube\_3);

}

}

else

{

cout << "Enter the correct number" << endl;

push\_tubesFunction(s, top3, level1\_testTube\_3);

}

}

else

break;

default:

cout << "Invalid Choice, please try again";

}

cout << endl;

moves++;

checkLevel1Completion();

} while (choi != 0);

players.moves = players.moves+ moves;

players.levels = "1";

return completion;

}

void main\_menu() {

cout << R"(

\_\_\_\_\_\_ \_\_ \_\_\_\_\_\_\_

/ \_\_/ / \_\_\_ \_\_\_/ /\_\_ / \_\_/ \_ \

\_\ \/ \_ \/ -\_) \_ / \_ \\_\ \/ \_\_\_/

/\_\_\_/\_//\_/\\_\_/\\_,\_/ .\_\_/\_\_\_/\_/

/\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

)" << endl;

cout << " ----------------------------------------------------------------\n";

cout << " Color Sorter Game \n";

cout << " ----------------------------------------------------------------\n";

cout << " | The rules are:\n" ;

cout << " | 1. You need to arrange balls of different colors into separate test tubes.\n";

cout << " | 2. A single test tube can hold upto 4 balls. \n";

cout << " | 3. You can only remove the balls from the top.\n";

cout << " | 4. There are three different levels of different difficulty.\n";

cout << " | 3. 5. On each level you will have limited moves. \n";

cout << " Enjoy Playing " << endl;

cout << " | 1. Level 1 ------------------ Easy " << endl;

cout << " | 2. Level 2 ------------------ Medium" << endl;

cout << " | 3. Level 3 ------------------ Hard " << endl;

cout << " | 4. Display Scoreboard " << endl;

}

void Scoreboard\_update() {

ofstream fout;

ifstream fin;

fin.open("filename.txt");

fout.open("filename.txt", ios::app);

if (fin.is\_open())

fout << players.name << " " << players.moves << " " << players.levels << "\n";

//cout << "\n Data has been appended to file";

fin.close();

fout.close();

}

void details() {

cout << "Enter PLayers Name:" << endl;

cin >> players.name;

cout << "Enter PLayers Moves:" << endl;

cin >> players.moves;

cout << "Enter PLayers Levels Cleared:" << endl;

cin >> players.levels;

}

void displayScoreboard(const string& filename) {

system("CLS");

int chic;

ifstream inputFile(filename);

if (inputFile.is\_open()) {

string name;

int moves;

int levels;

cout << R"(------------------------------------------------------

Scoreboard

------------------------------------------------------)" << endl;

cout << "Level 1 <= 10 Level 2 <= 15 Level 3 <= 25" << endl;

cout << "Name \t\t" << " Moves " << "\t\t" << " Levels Cleared" << '\n';

while (inputFile >> name >> moves >> levels) {

cout << "";

cout << "" << name << "\t\t " << moves << "\t\t " << levels << '\n';

}

inputFile.close();

}

else {

cout << "Failed to open the file.\n";

}

ofstream outputFile("scoreboard\_output.txt");

if (outputFile.is\_open()) {

ifstream inputFile(filename);

string line;

while (getline(inputFile, line)) {

outputFile << line << '\n';

}

outputFile.close();

}

else {

cout << "Failed to create the output file.\n";

}

cout << "Enter any key for the main menu:";

cin >> chic;

}

void updateFile(const string& filename, const string& wordToUpdate, const string& newLine) {

ifstream inputFile(filename);

if (!inputFile) {

cerr << "Failed to open the file: " << filename << endl;

return;

}

ofstream tempFile("temp.txt");

if (!tempFile) {

cerr << "Failed to create temporary file." <<endl;

inputFile.close();

return;

}

string line;

while (getline(inputFile, line)) {

if (line.find(wordToUpdate) != string::npos) {

line = newLine;

}

tempFile << line << endl;

}

inputFile.close();

tempFile.close();

if (remove(filename.c\_str()) != 0) {

cerr << "Failed to remove the original file." <<endl;

return;

}

if (rename("temp.txt", filename.c\_str()) != 0) {

cerr << "Failed to rename the temporary file." << endl;

return;

}

}

void reset\_array() {

}

void main() {

string filename = "filename.txt";

bool completion1 = false, completion2= false, completion3 = false;

int choic;

cout << "Enter players name(it must be less than 7 cahracters):" << endl;

cin >> players.name;

players.levels = "0";

players.moves = 0;

Scoreboard\_update();

do {

system("CLS");

main\_menu();

cout << "Enter choice" << endl;

cin >> choic;

switch (choic) {

case 1:

completion1 = level\_1();

break;

case 2:

completion2 = level\_2();

break;

case 3:

completion3 = completion3 = level\_3();

break;

case 4:

displayScoreboard(filename);

cout << endl;

break;

default:

cout << "Invalid Choice, please try again!" << endl;

}

if (completion1 == true || completion2 == true || completion3 == true) {

string newLine = players.name + " " + to\_string(players.moves) + " " + players.levels;

updateFile(filename, players.name, newLine);

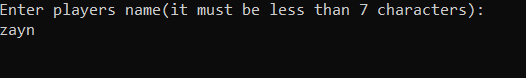
}

} while (choic != 0);

system("pause");

}

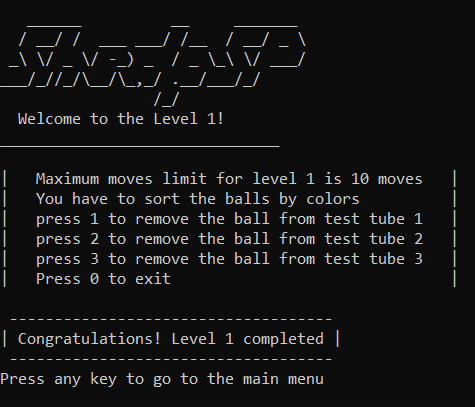
* 1. **Output Screenshots:**





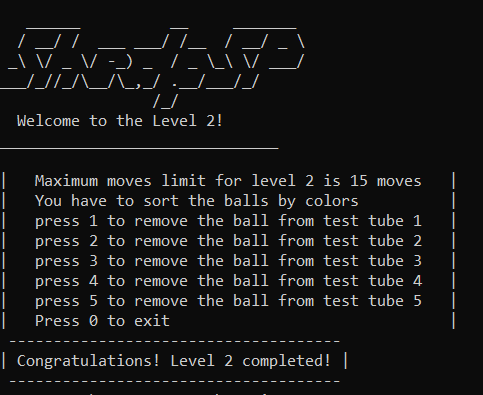
A screenshot of a computer

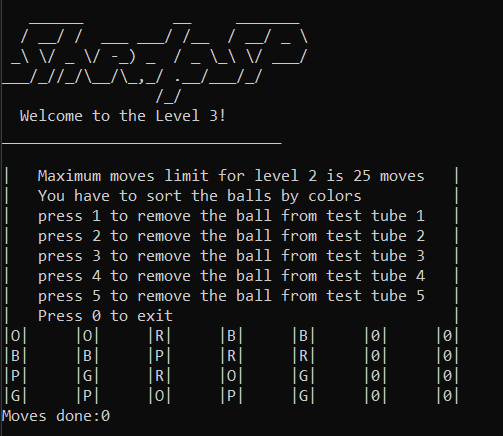
Description automatically generated with medium confidence



A screenshot of a computer

Description automatically generated with medium confidence





A screenshot of a computer

Description automatically generated with medium confidence

**CHAPTER 4: CONCLUSION**

In conclusion, the color sorter program utilizing arrays and test tubes offers an efficient and reliable solution for sorting objects based on their colors. By leveraging an array-based data structure and applying sorting algorithms, the program simplifies the sorting process, grouping objects of the same color together. This automated system streamlines color-based sortingoperations and provides a customizable approach for various applications. With its ability to accurately categorize and arrange objects, the color sorter program enhances productivity, minimizes errors