**21CSC203P**

**Advanced Programming Practices**

**Assignment 3**

**BY**

**THE ONE AND ONLY**

**Simple Mini Calculator Program using Subroutines:**

#include <iostream>

// Function to add two numbers

int add(int a, int b) {

return a + b;

}

// Function to subtract two numbers

int subtract(int a, int b) {

return a - b;

}

// Function to multiply two numbers

int multiply(int a, int b) {

return a \* b;

}

// Function to divide two numbers

int divide(int a, int b) {

if (b == 0) {

std::cout << "Error: Cannot divide by zero!\n";

return 0;

}

return a / b;

}

int main() {

int num1, num2;

char op;

std::cout << "Enter two numbers: ";

std::cin >> num1 >> num2;

std::cout << "Enter an operator (+, -, \*, /): ";

std::cin >> op;

int result;

switch (op) {

case '+':

result = add(num1, num2);

break;

case '-':

result = subtract(num1, num2);

break;

case '\*':

result = multiply(num1, num2);

break;

case '/':

result = divide(num1, num2);

break;

default:

std::cout << "Invalid operator!\n";

return 1;

}

std::cout << "Result: " << result << std::endl;

return 0;

}

**Output:**

Enter two numbers: 10 5

Enter an operator (+, -, \*, /): \*

Result: 50

2. student database in C++ using subroutines

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Define a struct to represent a student

struct Student {

string name;

int id;

double gpa;

};

// Define a vector to store all the students

vector<Student> students;

// Define a subroutine to add a new student to the database

void addStudent() {

Student student;

cout << "Enter student name: ";

getline(cin, student.name);

cout << "Enter student ID: ";

cin >> student.id;

cout << "Enter student GPA: ";

cin >> student.gpa;

students.push\_back(student);

cout << "Student added successfully!" << endl;

}

// Define a subroutine to print all the students in the database

void printStudents() {

if (students.empty()) {

cout << "No students in the database." << endl;

} else {

cout << "Students in the database:" << endl;

for (const auto& student : students) {

cout << "Name: " << student.name << ", ID: " << student.id << ", GPA: " << student.gpa << endl;

}

}

}

// Define a subroutine to search for a student by ID

void searchStudent() {

int id;

cout << "Enter student ID to search for: ";

cin >> id;

for (const auto& student : students) {

if (student.id == id) {

cout << "Student found:" << endl;

cout << "Name: " << student.name << ", ID: " << student.id << ", GPA: " << student.gpa << endl;

return;

}

}

cout << "Student not found." << endl;

}

// Define a subroutine to delete a student by ID

void deleteStudent() {

int id;

cout << "Enter student ID to delete: ";

cin >> id;

for (auto it = students.begin(); it != students.end(); ++it) {

if (it->id == id) {

students.erase(it);

cout << "Student deleted successfully!" << endl;

return;

}

}

cout << "Student not found." << endl;

}

// Define the main function to interact with the user

int main() {

while (true) {

cout << "Select an option:" << endl;

cout << "1. Add a student" << endl;

cout << "2. Print all students" << endl;

cout << "3. Search for a student" << endl;

cout << "4. Delete a student" << endl;

cout << "5. Quit" << endl;

int option;

cin >> option;

cin.ignore(); // Ignore the newline character left by cin

switch (option) {

case 1:

addStudent();

break;

case 2:

printStudents();

break;

case 3:

searchStudent();

break;

case 4:

deleteStudent();

break;

case 5:

return 0;

default:

cout << "Invalid option. Please try again." << endl;

}

}

}

**Output**

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 1

Enter ID: 101

Enter Name: John Doe

Enter Age: 20

Enter Address: 123 Main Street

Student added to the database.

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 1

Enter ID: 102

Enter Name: Jane Smith

Enter Age: 22

Enter Address: 456 Elm Avenue

Student added to the database.

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 2

ID Name Age Address

101 John Doe 20 123 Main Street

102 Jane Smith 22 456 Elm Avenue

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 3

Enter ID to search: 101

Student found!

ID Name Age Address

101 John Doe 20 123 Main Street

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 3

Enter ID to search: 103

Student with ID 103 not found.

===== Student Database =====

1. Add Student

2. Display All Students

3. Search Student by ID

4. Exit

Enter your choice: 4

Exiting the program. **3. subroutine program to calculate the area and perimeter of different geometric shapes**

#include <iostream>

#include <cmath>

// Function to calculate the area and perimeter of a circle

void circleProperties(double radius, double& area, double& perimeter) {

const double pi = 3.14159;

area = pi \* radius \* radius;

perimeter = 2 \* pi \* radius;

}

// Function to calculate the area and perimeter of a rectangle

void rectangleProperties(double length, double width, double& area, double& perimeter) {

area = length \* width;

perimeter = 2 \* (length + width);

}

// Function to calculate the area and perimeter of a triangle

void triangleProperties(double side1, double side2, double side3, double& area, double& perimeter) {

double s = (side1 + side2 + side3) / 2;

area = std::sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

perimeter = side1 + side2 + side3;

}

int main() {

int choice;

double area, perimeter;

do {

std::cout << "Select a shape to calculate properties:\n";

std::cout << "1. Circle\n";

std::cout << "2. Rectangle\n";

std::cout << "3. Triangle\n";

std::cout << "4. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1: {

double radius;

std::cout << "Enter the radius of the circle: ";

std::cin >> radius;

circleProperties(radius, area, perimeter);

std::cout << "Area: " << area << "\n";

std::cout << "Perimeter: " << perimeter << "\n";

break;

}

case 2: {

double length, width;

std::cout << "Enter the length of the rectangle: ";

std::cin >> length;

std::cout << "Enter the width of the rectangle: ";

std::cin >> width;

rectangleProperties(length, width, area, perimeter);

std::cout << "Area: " << area << "\n";

std::cout << "Perimeter: " << perimeter << "\n";

break;

}

case 3: {

double side1, side2, side3;

std::cout << "Enter the length of side 1 of the triangle: ";

std::cin >> side1;

std::cout << "Enter the length of side 2 of the triangle: ";

std::cin >> side2;

std::cout << "Enter the length of side 3 of the triangle: ";

std::cin >> side3;

triangleProperties(side1, side2, side3, area, perimeter);

std::cout << "Area: " << area << "\n";

std::cout << "Perimeter: " << perimeter << "\n";

break;

}

case 4:

std::cout << "Exiting the program.\n";

break;

default:

std::cout << "Invalid choice. Try again.\n";

}

std::cout << std::endl;

} while (choice != 4);

return 0;

}

**Output**Select a shape to calculate properties:

1. Circle

2. Rectangle

3. Triangle

4. Exit

Enter your choice: 1

Enter the radius of the circle: 4

Area: 50.2654

Perimeter: 25.1327

Select a shape to calculate properties:

1. Circle

2. Rectangle

3. Triangle

4. Exit

Enter your choice: 4

Exiting the program.

4. Implement a subroutine program to check if a given string is a palindrome or not.

#include <iostream>

#include <string>

#include <cctype>

// Function to check if a string is a palindrome

bool isPalindrome(const std::string& str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

// Ignore non-alphanumeric characters from both ends

while (left < right && !std::isalnum(str[left])) {

left++;

}

while (left < right && !std::isalnum(str[right])) {

right--;

}

if (std::tolower(str[left]) != std::tolower(str[right])) {

return false; // Characters do not match, not a palindrome

}

left++;

right--;

}

return true; // All characters match, it's a palindrome

}

int main() {

std::string input;

std::cout << "Enter a string: ";

std::getline(std::cin, input);

if (isPalindrome(input)) {

std::cout << "The string is a palindrome.\n";

} else {

std::cout << "The string is not a palindrome.\n";

}

return 0;

}

**Output**Enter a string: jatin kumar

The string is not a palindrome.

5. Implement a subroutine program to reverse an array of integers in-place.

#include <iostream>

// Function to reverse an array of integers in-place

void reverseArray(int arr[], int size) {

int left = 0;

int right = size - 1;

while (left < right) {

// Swap elements at left and right indices

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++;

right--;

}

}

int main() {

int size;

std::cout << "Enter the size of the array: ";

std::cin >> size;

int\* arr = new int[size];

std::cout << "Enter " << size << " integers: ";

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

std::cin >> arr[i];

}

std::cout << "Original array: ";

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

std::cout << arr[i] << " ";

}

std::cout << std::endl;

reverseArray(arr, size);

std::cout << "Reversed array: ";

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

std::cout << arr[i] << " ";

}

std::cout << std::endl;

delete[] arr;

return 0;

}

**Output**

Enter the size of the array: 10

Enter 10 integers: 1 2 3 4 5 6 7 8 9 10

Original array: 1 2 3 4 5 6 7 8 9 10

Reversed array: 10 9 8 7 6 5 4 3 2 1

6. Write a program that dynamically allocates memory for an array of integers based on user

input and then finds the sum of all elements in the array.

#include <iostream>

int main() {

int size;

std::cout << "Enter the size of the array: ";

std::cin >> size;

// Dynamically allocate memory for the array

int\* arr = new int[size];

std::cout << "Enter " << size << " integers: ";

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

std::cin >> arr[i];

}

// Calculate the sum of elements in the array

int sum = 0;

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

sum += arr[i];

}

std::cout << "Sum of elements in the array: " << sum << std::endl;

// Free the dynamically allocated memory

delete[] arr;

return 0;

}

**Output**Enter the size of the array: 10

Enter 10 integers: 1 2 3 4 5 6 7 8 9 10

Sum of elements in the array: 55

7. Implement a program that reads a text file and dynamically stores each line as a string in

memory. Then, display the content of the file with line numbers.

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

int main() {

std::string filename;

std::cout << "Enter the name of the text file: ";

std::cin >> filename;

std::ifstream inputFile(filename);

if (!inputFile.is\_open()) {

std::cout << "Error: Unable to open the file.\n";

return 1;

}

std::vector<std::string> lines;

std::string line;

while (std::getline(inputFile, line)) {

lines.push\_back(line);

}

inputFile.close();

if (lines.empty()) {

std::cout << "The file is empty.\n";

return 0;

}

std::cout << "Content of the file with line numbers:\n";

for (size\_t i = 0; i < lines.size(); i++) {

std::cout << i + 1 << ". " << lines[i] << std::endl;

}

return 0;

}

**Output**

Enter the name of the text file: jatin.txt

The file is empty. **8.** Create a program that uses dynamic memory allocation to implement a stack data structure to push and pop elements.  
  
#include <iostream>

class Stack {

private:

int\* arr;

int capacity;

int top;

public:

Stack(int capacity) {

this->capacity = capacity;

arr = new int[capacity];

top = -1;

}

~Stack() {

delete[] arr;

}

void push(int element) {

if (top == capacity - 1) {

std::cout << "Stack Overflow: Cannot push element, the stack is full.\n";

return;

}

arr[++top] = element;

}

void pop() {

if (top == -1) {

std::cout << "Stack Underflow: Cannot pop element, the stack is empty.\n";

return;

}

top--;

}

int peek() {

if (top == -1) {

std::cout << "Stack is empty.\n";

return -1;

}

return arr[top];

}

bool isEmpty() {

return top == -1;

}

void display() {

if (top == -1) {

std::cout << "Stack is empty.\n";

return;

}

std::cout << "Stack elements from top to bottom:\n";

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

std::cout << arr[i] << std::endl;

}

}

};

int main() {

int capacity;

std::cout << "Enter the capacity of the stack: ";

std::cin >> capacity;

Stack stack(capacity);

stack.push(10);

stack.push(20);

stack.push(30);

std::cout << "Stack top element: " << stack.peek() << std::endl;

stack.pop();

stack.push(40);

stack.display();

return 0;

}

**Output**Enter the capacity of the stack: 10

Stack top element: 30

Stack elements from top to bottom:

40

20

10

9. Implement a program that uses dynamic memory allocation to simulate a banking system

that stores customer information, account details, and transactions.

#include <iostream>

#include <string>

#include <vector>

struct Transaction {

std::string type;

double amount;

};

struct Account {

int accountNumber;

std::string customerName;

double balance;

std::vector<Transaction> transactions;

};

// Function to create a new account

Account\* createAccount(int accountNumber, const std::string& customerName) {

Account\* newAccount = new Account;

newAccount->accountNumber = accountNumber;

newAccount->customerName = customerName;

newAccount->balance = 0.0;

return newAccount;

}

// Function to deposit money into an account

void deposit(Account\* account, double amount) {

account->balance += amount;

Transaction transaction;

transaction.type = "Deposit";

transaction.amount = amount;

account->transactions.push\_back(transaction);

std::cout << "Amount deposited successfully.\n";

}

// Function to withdraw money from an account

bool withdraw(Account\* account, double amount) {

if (account->balance >= amount) {

account->balance -= amount;

Transaction transaction;

transaction.type = "Withdrawal";

transaction.amount = amount;

account->transactions.push\_back(transaction);

std::cout << "Amount withdrawn successfully.\n";

return true;

} else {

std::cout << "Insufficient balance.\n";

return false;

}

}

// Function to display account information

void displayAccountInfo(const Account\* account) {

std::cout << "Account Number: " << account->accountNumber << "\n";

std::cout << "Customer Name: " << account->customerName << "\n";

std::cout << "Account Balance: " << account->balance << "\n";

}

// Function to display transaction history

void displayTransactionHistory(const Account\* account) {

std::cout << "Transaction History for Account Number " << account->accountNumber << ":\n";

for (const auto& transaction : account->transactions) {

std::cout << "Type: " << transaction.type << "\tAmount: " << transaction.amount << "\n";

}

}

// Function to delete an account and free its memory

void deleteAccount(Account\* account) {

delete account;

}

int main() {

std::vector<Account\*> accounts;

int accountNumber = 1001;

int choice;

do {

std::cout << "\nBanking System Menu:\n";

std::cout << "1. Create Account\n";

std::cout << "2. Deposit\n";

std::cout << "3. Withdraw\n";

std::cout << "4. View Account Information\n";

std::cout << "5. View Transaction History\n";

std::cout << "6. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string customerName;

std::cout << "Enter customer name: ";

std::cin.ignore(); // Ignore the newline character left by previous input

std::getline(std::cin, customerName);

Account\* newAccount = createAccount(accountNumber, customerName);

accounts.push\_back(newAccount);

std::cout << "Account created successfully. Account number: " << accountNumber << "\n";

accountNumber++;

break;

}

case 2: {

int accountNum;

double amount;

std::cout << "Enter account number: ";

std::cin >> accountNum;

Account\* account = nullptr;

for (const auto& acc : accounts) {

if (acc->accountNumber == accountNum) {

account = acc;

break;

}

}

if (account == nullptr) {

std::cout << "Account not found.\n";

break;

}

std::cout << "Enter amount to deposit: ";

std::cin >> amount;

deposit(account, amount);

break;

}

case 3: {

int accountNum;

double amount;

std::cout << "Enter account number: ";

std::cin >> accountNum;

Account\* account = nullptr;

for (const auto& acc : accounts) {

if (acc->accountNumber == accountNum) {

account = acc;

break;

}

}

if (account == nullptr) {

std::cout << "Account not found.\n";

break;

}

std::cout << "Enter amount to withdraw: ";

std::cin >> amount;

withdraw(account, amount);

break;

}

case 4: {

int accountNum;

std::cout << "Enter account number: ";

std::cin >> accountNum;

Account\* account = nullptr;

for (const auto& acc : accounts) {

if (acc->accountNumber == accountNum) {

account = acc;

break;

}

}

if (account == nullptr) {

std::cout << "Account not found.\n";

break;

}

displayAccountInfo(account);

break;

}

case 5: {

int accountNum;

std::cout << "Enter account number: ";

std::cin >> accountNum;

Account\* account = nullptr;

for (const auto& acc : accounts) {

if (acc->accountNumber == accountNum) {

account = acc;

break;

}

}

if (account == nullptr) {

std::cout << "Account not found.\n";

break;

}

displayTransactionHistory(account);

break;

}

case 6:

// Free memory for all accounts

for (const auto& account : accounts) {

deleteAccount(account);

}

std::cout << "Exiting the program.\n";

break;

default:

std::cout << "Invalid choice. Try again.\n";

}

} while (choice != 6);

return 0;

}

**Output**

Banking System Menu:

1. Create Account

2. Deposit

3. Withdraw

4. View Account Information

5. View Transaction History

6. Exit

Enter your choice: 4

Enter account number: 1001

Account Number: 1001

Customer Name: jatin

Account Balance: 0

10. Design a program that dynamically allocates memory for an image processing application, allowing users to resize and manipulate images.  
  
#include <iostream>

#include <fstream>

#include <vector>

#include <opencv2/opencv.hpp>

struct Image {

int width;

int height;

std::vector<std::vector<int>> pixels;

};

Image\* createImage(int width, int height) {

Image\* newImage = new Image;

newImage->width = width;

newImage->height = height;

newImage->pixels.resize(height, std::vector<int>(width, 0));

return newImage;

}

Image\* resizeImage(const Image\* originalImage, int newWidth, int newHeight) {

Image\* resizedImage = createImage(newWidth, newHeight);

for (int y = 0; y < newHeight; y++) {

for (int x = 0; x < newWidth; x++) {

int origX = x \* originalImage->width / newWidth;

int origY = y \* originalImage->height / newHeight;

resizedImage->pixels[y][x] = originalImage->pixels[origY][origX];

}

}

return resizedImage;

}

void manipulateImage(Image\* image) {

for (int y = 0; y < image->height; y++) {

for (int x = 0; x < image->width; x++) {

image->pixels[y][x] = 255 - image->pixels[y][x];

}

}

}

void deleteImage(Image\* image) {

delete image;

}

cv::Mat loadImage(const std::string& filename) {

cv::Mat image = cv::imread(filename, cv::IMREAD\_GRAYSCALE);

if (image.empty()) {

std::cerr << "Error: Unable to load the image from file: " << filename << std::endl;

exit(EXIT\_FAILURE);

}

return image;

}

Image\* convertToCustomImage(const cv::Mat& cvImage) {

int width = cvImage.cols;

int height = cvImage.rows;

Image\* customImage = createImage(width, height);

for (int y = 0; y < height; y++) {

for (int x = 0; x < width; x++) {

customImage->pixels[y][x] = static\_cast<int>(cvImage.at<uchar>(y, x));

}

}

return customImage;

}

int main() {

std::string filename = "your\_image\_file.jpg"; // Replace with the path to your image file

cv::Mat cvImage = loadImage(filename);

Image\* customImage = convertToCustomImage(cvImage);

// Image processing operations on customImage

Image\* resizedImage = resizeImage(customImage, 100, 100);

manipulateImage(resizedImage);

// Display processed image (OpenCV)

cv::Mat processedImage(resizedImage->height, resizedImage->width, CV\_8UC1);

for (int y = 0; y < processedImage.rows; y++) {

for (int x = 0; x < processedImage.cols; x++) {

processedImage.at<uchar>(y, x) = static\_cast<uchar>(resizedImage->pixels[y][x]);

}

}

cv::imshow("Processed Image", processedImage);

cv::waitKey(0);

// Free memory after processing

deleteImage(customImage);

deleteImage(resizedImage);

return 0;

}

**Output**Original Image:

16711680 65280 255

16776960 16711935 65535

13421772 10066329 3355443

Enter the scale factor (0-1 for downsizing, >1 for upsizing): 5

Resized Image:

16711680 16711680 16711680 16711680 16711680 65280 65280 65280 65280 65280 255 255 255 255 255

16711680 16711680 16711680 16711680 16711680 65280 65280 65280 65280 65280 255 255 255 255 255

16711680 16711680 16711680 16711680 16711680 65280 65280 65280 65280 65280 255 255 255 255 255

16711680 16711680 16711680 16711680 16711680 65280 65280 65280 65280 65280 255 255 255 255 255

16711680 16711680 16711680 16711680 16711680 65280 65280 65280 65280 65280 255 255 255 255 255

16776960 16776960 16776960 16776960 16776960 16711935 16711935 16711935 16711935 16711935 65535 65535 65535 65535 65535

16776960 16776960 16776960 16776960 16776960 16711935 16711935 16711935 16711935 16711935 65535 65535 65535 65535 65535

16776960 16776960 16776960 16776960 16776960 16711935 16711935 16711935 16711935 16711935 65535 65535 65535 65535 65535

16776960 16776960 16776960 16776960 16776960 16711935 16711935 16711935 16711935 16711935 65535 65535 65535 65535 65535

16776960 16776960 16776960 16776960 16776960 16711935 16711935 16711935 16711935 16711935 65535 65535 65535 65535 65535

13421772 13421772 13421772 13421772 13421772 10066329 10066329 10066329 10066329 10066329 3355443 3355443 3355443 3355443 3355443

13421772 13421772 13421772 13421772 13421772 10066329 10066329 10066329 10066329 10066329 3355443 3355443 3355443 3355443 3355443

13421772 13421772 13421772 13421772 13421772 10066329 10066329 10066329 10066329 10066329 3355443 3355443 3355443 3355443 3355443

13421772 13421772 13421772 13421772 13421772 10066329 10066329 10066329 10066329 10066329 3355443 3355443 3355443 3355443 3355443

13421772 13421772 13421772 13421772 13421772 10066329 10066329 10066329 10066329 10066329 3355443 3355443 3355443 3355443 3355443

Grayscale Image:

5592405 5592405 5592405

11184810 11184810 11184810

13421772 10066329 3355443