2. TO DO

#include <iostream>

#include <stack>

using namespace std;

class todo {

private:

stack<string> s;

public:

void addingtask() {

string t;

cout << "Enter your task: ";

cin >> t;

s.push(t);

cout << "\*\*\*\*\*\*Task is added to the list\*\*\*\*\*\*\*\*" << endl;

}

bool isCompleted(const string& task) {

return task.find(" [Completed]") != string::npos;

}

void view() {

if (s.empty()) {

cout << "No tasks available." << endl;

return;

}

cout << "Tasks:" << endl;

stack<string> tempnames = s;

while (!tempnames.empty()) {

string currentTask = tempnames.top();

cout << "- " << currentTask << " - " << (isCompleted(currentTask) ? "Completed" : "Pending") << endl;

tempnames.pop();

}

}

void Completed() {

string completedtask;

cout << "Enter the task that has been completed: ";

cin >> completedtask;

if (s.empty()) {

cout << "No tasks available to mark as completed." << endl;

return;

}

stack<string> tempStack;

while (!s.empty()) {

string currentTask = s.top();

if (currentTask == completedtask) {

currentTask += " [Completed]";

s.pop();

s.push(currentTask);

cout << currentTask << " Task marked as completed!" << endl;

while (!tempStack.empty()) {

s.push(tempStack.top());

tempStack.pop();

}

return;

} else {

tempStack.push(currentTask);

s.pop();

}

}

cout << "The specified task was not found in the to-do list." << endl;

while (!tempStack.empty()) {

s.push(tempStack.top());

tempStack.pop();

}

}

void remove() {

string removedtask;

cout << "Enter the task that has to be removed: ";

cin >> removedtask;

if (s.empty()) {

cout << "No tasks available to remove." << endl;

return;

}

string currentTask = s.top();

if (currentTask == removedtask) {

s.pop();

cout << currentTask << " Task removed!" << endl;

} else {

cout << "The specified task is not at the top of the stack." << endl;

}

}

};

int main() {

todo t;

while (true) {

cout << "To-Do List:" << endl;

cout << "1. Add Task" << endl;

cout << "2. View Tasks" << endl;

cout << "3. Task Completed" << endl;

cout << "4. Remove Task" << endl;

cout << "5. Exit" << endl;

int choice;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

t.addingtask();

break;

case 2:

t.view();

break;

case 3:

t.Completed();

break;

case 4:

t.remove();

break;

case 5:

cout << "Exiting program. Have a nice day!" << endl;

return 0;

default:

cout << "Invalid choice. Please enter a valid option." << endl;

break;

}

}

return 0;

}

3)TIC TAC TOE GAME

#include <iostream>

using namespace std;

class tictac {

private:

int X = 1;

int O = -1;

int EMPTY = 0;

int tictactoe[3][3];

int player;

public:

tictac() {

player = X;

}

void starting() {

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

tictactoe[i][j] = EMPTY;

}

}

}

void playing(int i, int j) {

tictactoe[i][j] = player;

player = -player;

}

bool winning(int move) {

int win = 3 \* move;

return ((tictactoe[0][0] + tictactoe[0][1] + tictactoe[0][2] == win) ||

(tictactoe[1][0] + tictactoe[1][1] + tictactoe[1][2] == win) ||

(tictactoe[2][0] + tictactoe[2][1] + tictactoe[2][2] == win) ||

(tictactoe[0][0] + tictactoe[1][0] + tictactoe[2][0] == win) ||

(tictactoe[0][1] + tictactoe[1][1] + tictactoe[2][1] == win) ||

(tictactoe[0][2] + tictactoe[1][2] + tictactoe[2][2] == win) ||

(tictactoe[0][0] + tictactoe[1][1] + tictactoe[2][2] == win) ||

(tictactoe[2][0] + tictactoe[1][1] + tictactoe[0][2] == win));

}

void winner() {

if (winning(X)) {

cout << "X has won the game" << endl;

} else if (winning(O)) {

cout << "O has won the game" << endl;

} else {

cout << "No one has won the game" << endl;

}

}

void layout() {

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

switch (tictactoe[i][j]) {

case 1: cout << "X"; break;

case -1: cout << "O"; break;

case 0: cout << " "; break;

}

if (j < 2) cout << "|";

}

if (i < 2) cout << "\n---------\n";

}

cout << endl;

}

};

int main() {

tictac t;

t.starting();

t.playing(0, 0);

t.playing(1, 1);

t.playing(0, 1);

t.playing(1, 2);

t.playing(0, 2);

t.playing(1, 2);

t.playing(1, 0);

t.playing(2, 0);

t.layout();

t.winner();

return 0;

}

2) BASIC CALCULATOR

#include <iostream>

#include <cmath>

using namespace std;

class calculator

{

public:

void addition(float a, float b)

{

float result;

result=a+b;

cout<<"the result after addition is "<<result<<endl;

}

void subtraction(float a, float b)

{

float result;

result=a-b;

cout<<"the result after addition is "<<result<<endl;

}

void multiplication(float a, float b)

{

float result;

result=a\*b;

cout<<"the result after addition is "<<result<<endl;

}

void division(float a, float b)

{

float result;

result=a/b;

cout<<"the result after addition is "<<result<<endl;

}

void exponent(float a ,float b)

{

float result;

result=pow(a,b);

cout<<result;

}

};

int main()

{

float a;

float b;

char operation;

calculator c;

cout<<"enter your first number:"<<endl;

cin>>a;

cout<<"enter your second number:"<<endl;

cin>>b;

cout<<"enter your operation: +,-,\*,/,^"<<endl;

cin>>operation;

switch(operation)

{

case '+':

c.addition(a,b);

break;

case '-':

c.subtraction(a,b);

break;

case '\*':

c.multiplication(a,b);

break;

case '/':

c.division(a,b);

break;

case '^':

c.exponent(a,b);

break;

default:

cout<<"invalid"<<endl;

break;

}

}

2) #include <iostream>

#include <stack>

#include <string>

using namespace std;

class Todo {

private:

stack<string> tasks;

public:

void addTask() {

string task;

cout << "Enter your task: ";

cin.ignore();

getline(cin, task);

tasks.push(task);

cout << "\*\*\*\*\*\*\*\* Task has been added \*\*\*\*\*\*\*\*\*\*\*" << endl;

}

bool isComplete(const string& task) {

return task.find("[completed]") != string::npos;

}

void viewTasks() {

if (tasks.empty()) {

cout << "No tasks" << endl;

return;

}

cout << "Tasks:" << endl;

stack<string> temp = tasks;

while (!temp.empty()) {

string current = temp.top();

cout << "-" << current << "-" << (isComplete(current) ? "completed" : "pending") << endl;

temp.pop();

}

}

void markCompleted() {

string completedTask;

cout << "Enter the task that you have completed: ";

cin.ignore();

getline(cin, completedTask);

if (tasks.empty()) {

cout << "No tasks remaining" << endl;

return;

}

stack<string> temp;

while (!tasks.empty()) {

string current = tasks.top();

tasks.pop();

if (current == completedTask) {

current += "[completed]";

tasks.push(current);

cout << current << " has been marked as completed!" << endl;

while (!temp.empty()) {

tasks.push(temp.top());

temp.pop();

}

return;

}

temp.push(current);

}

cout << "Task not found in the list" << endl;

while (!temp.empty()) {

tasks.push(temp.top());

temp.pop();

}

}

void removeTask() {

string removingTask;

cout << "Enter the task you wish to remove: ";

cin.ignore();

getline(cin, removingTask);

if (tasks.empty()) {

cout << "No task available to remove" << endl;

return;

}

stack<string> temp;

while (!tasks.empty()) {

string current = tasks.top();

tasks.pop();

if (current == removingTask) {

cout << current << " has been removed" << endl;

while (!temp.empty()) {

tasks.push(temp.top());

temp.pop();

}

return;

}

temp.push(current);

}

cout << "Task not found in the list" << endl;

while (!temp.empty()) {

tasks.push(temp.top());

temp.pop();

}

}

};

int main() {

Todo todoList;

while (true) {

cout << "TO-DO LIST" << endl;

cout << "1) Add task" << endl;

cout << "2) View tasks" << endl;

cout << "3) Mark task as completed" << endl;

cout << "4) Remove task" << endl;

cout << "5) Exit" << endl;

int choice;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

todoList.addTask();

break;

case 2:

todoList.viewTasks();

break;

case 3:

todoList.markCompleted();

break;

case 4:

todoList.removeTask();

break;

case 5:

cout << "Exiting program" << endl;

return 0;

default:

cout << "Invalid choice" << endl;

break;

}

}

return 0;

}